

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

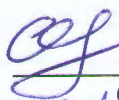
БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММНЫЙ КОМПОНЕНТ ЭМИССИИ И УЧЕТА КОДОВ

МАРКИРОВКИ

Студент



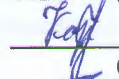
31.05.2022

(подпись, дата)

Свищев Данил Сергеевич

(ФИО полностью)

Руководитель



31.05.2022

(подпись, дата)

Карамышева Н.С.

(фамилия, инициалы)

Нормоконтролёр



31.05.2022

(подпись, дата)

Бычков А.С.

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой


(подпись)

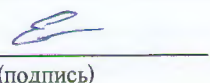
Митрохин М.А.

(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

07.06.2022 № 1)

Секретарь ГЭК


(подпись)

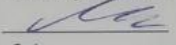
Е.И. Гурин
(фамилия, инициалы)

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«26» апреля 2022г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Свищеву Данилу Сергеевичу гр. 18BB1 Факультета вычислительной техники

2. Тема работы «Программный компонент эмиссии и учета кодов маркировки»

Тема утверждена приказом ПГУ № 357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы Карамышева Н.С.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Требуется разработать систему компонентов (специализированное ПО), реализующую программный (REST API) и графический (UI) интерфейсы для обеспечения участников товарооборота автоматизированной подсистемой управления процессами эмиссии и нанесения КМ, верификацией и агрегаций.

5. Перечень подлежащих разработке вопросов
Требуется:

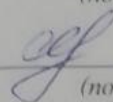
1. Произвести анализ предметной области
2. Выбрать среду, язык и стек технологий, на основе которых будет проектироваться система.
3. Произвести сравнительный анализ систем, являющихся аналогами.
4. Разработка компонент эмиссии и учета кодов маркировки.
5. Произвести тестирование системы на основе создания заказа, получения кодов маркировки и нанесения на продукцию.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	18	26.04.2022 – 28.04.2022	
2	Анализ, выбор и настройка инструментов разработки	64	29.04.2022 – 04.05.2022	
3	Анализ, выбор и настройка окружения для развертывания приложения	30	05.05.2022 – 08.05.2022	
4	Реализация программного интерфейса системы	72	09.05.2022 – 16.05.2022	
5	Реализация пользовательского интерфейса системы	56	17.05.2022 – 22.05.2022	
6	Отладка системы и тестирование функционала	48	23.05.2022 – 27.05.2022	
7	Оформление выпускной работы	36	28.05.2022 - 31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы 26.04.2022  Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент  26.04.2022 Свищев Д.С.
(подпись, дата)

Работу к защите допустить
Декан факультета  31.05.2022 Фионова Л.Р.
(подпись, дата)

Реферат

Пояснительная записка содержит 200 листов, 42 рисунка, 3 приложения.

ПРОГРАММНЫЙ КОМПОНЕНТ ЭМИССИИ И УЧЕТА КОДОВ
МАРКИРОВКИ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, СЕРВИС-
ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА, БАЗА ДАННЫХ.

Цель работы – разработать систему специализированного ПО, реализующую программный и графический интерфейсы для обеспечения участников оборота товаров автоматизированной подсистемой управления процессами эмиссии и нанесения кодов маркировки, верификацией и агрегацией.

В процессе выполнения работы проводилось:

- изучение предметной области;
- сравнительный анализ аналогичных систем специализированного ПО;
- разработка компонента эмиссии и учета кодов маркировки на основе клиент-серверной, микросервисной архитектуры;
- отладка, развертывание и тестирование системы.

Результатом проведенной работы является реализованное приложение, удовлетворяющее поставленной задаче, а именно, разработана система, представляющая компонент эмиссии и учета кодов маркировки.

Разработка проводилась в среде программирования *IntelliJ IDEA 2020*.
Использованные языки программирования: *Java, JavaScript*.

Созданное приложение внедрено в товарооборот, осуществляемый на территории Российской Федерации, а также ряда стран СНГ, установлено на крупных производственных фабриках и заводах и развернуто в распределенной облачной системе.

					ПГУ 1.090301.36.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Сеишее Д.С.			Программный компонент эмиссии и учета кодов маркировки Пояснительная записка	Лит.	Лист
Пров.		Карамышева Н.С.					2
Н. Контр.		Бычков А.С.					200
Уте.		Митрохин М.А.				ФВТ, каф. ВТ гр. 18ВВ1	

Содержание

Введение	8
Перечень сокращений и обозначений	10
Перечень терминов и определений	12
1 Программный компонент эмиссии и учета кодов маркировки	16
1.1 Анализ предметной области	16
1.1.1 Анализ проблемы	16
1.1.2 Способы решения	16
1.2 Обоснование актуальности разработки	20
1.2.1 Компоненты эмиссии и учета кодов маркировки	24
1.3 Обзор аналогов	25
1.3.1 Единая государственная автоматизированная система	26
1.3.2 Национальная система маркировки	27
1.4 Постановка задачи	29
1.5 Выбор инструментов разработки	31
1.5.1 Языки программирования	31
1.5.1.1 Язык программирования сервера	31
1.5.1.2 Библиотека сервера	32
1.5.1.3 Язык программирования клиента	33
1.5.2 Интегрированная среда разработки	34
1.5.3 Сборка программного обеспечения	35
1.5.4 Система развертывания и управления компонентами	36
1.5.5 Системы управления базами данных	36
1.5.5.1 Система управления реляционными базами данных	37
1.5.5.2 Система управления нереляционными базами данных	38
1.5.6 Система управления версиями	39
1.5.7 Система визуального проектирования отчетов	40
1.6 Архитектура приложения	41
1.6.1 Концепция архитектуры	41
1.6.2 Подсистемы и составные части программного компонента	43
1.6.3 Сервисы программного компонента	44
1.6.4 Модель системы	46

1.6.5	Взаимодействие с внешними системами	47
1.7	Проектирование и разработка.....	50
1.7.1	Проектирование программного компонента.....	50
1.7.1.1	Определение назначения.....	50
1.7.1.2	Проектирование основных функций	51
1.7.1.3	Определение показателей назначения.....	51
1.7.1.4	Проектирование компонентов	52
1.7.1.5	Доступность	52
1.7.1.6	Масштабируемость	53
1.7.1.7	Единый источник данных	53
1.7.2	Разработка программного компонента.....	54
1.7.2.1	Паттерны разработки.....	54
1.7.2.2	Уровень контроллеров.....	55
1.7.2.3	Уровень сервисов.....	56
1.7.2.4	Уровень базы данных	57
1.7.2.5	Обработка ошибок.....	59
1.7.2.6	Особенности реализации.....	61
1.8	Описание алгоритма.....	62
1.8.1	Последовательность процессов	62
1.8.2	Ключевые этапы процессов	64
1.8.3	Создание заказа на эмиссию кодов маркировки.....	65
1.8.4	Получение статуса массива кодов маркировки	67
1.8.5	Получение кодов маркировки.....	69
1.8.6	Отправка отчета о нанесении кодов маркировки	71
1.9	Описание программы	73
1.9.1	Метод создания заказа на эмиссию кодов маркировки.....	74
1.9.2	Метод создания отчета о нанесении КМ.....	77
1.9.3	Метод создания отчета об отбраковке КМ	79
1.9.4	Метод создания отчета об агрегации КМ	80
1.10	Условия применения программы	83
1.10.1	Используемые технические средства	83
1.10.1.1	Характеристики программного компонента	83

1.10.1.2	Характеристики интерфейса программного компонента	83
1.10.2	Необходимое программное обеспечение	84
1.10.2.1	Веб-интерфейс программного компонента	84
1.10.3	Информационное обеспечение	84
1.10.3.1	Требования по обработке JSON формата	84
1.11	Отладка и тестирование	85
1.11.1	Отладка системы.....	85
1.11.2	Тестирование системы	86
1.11.2.1	Функциональное тестирование	87
1.11.2.2	Ручное тестирование	87
1.11.2.3	Интеграционное тестирование.....	89
2	Безопасность жизнедеятельности.....	94
2.1	Анализ негативных факторов и способы их устранения	94
2.2	Физическое и психофизическое влияние	95
2.3	Оборудование рабочих мест	96
2.4	Освещенность рабочего места	97
	Заключение	99
	Список использованных источников	100
	Приложение А – UML-диаграммы.....	101
	Приложение Б – Листинг программы	106
	Приложение В - Презентация	200

Введение

В настоящее время большое внимание уделяется экономической отрасли, а именно: товарообороту Российской Федерации, качеству и оригинальности производимой продукции. Важным направлением данной сферы деятельности является поиск, определение и устранение источников производства контрафактной продукции. Данное направление получило большое внимание и активное участие потребителей товаров в нем. Ведь качество продукции и товаров может непосредственно сказываться на здоровье, самочувствии и состоянии покупателей.

Однако развитие и эволюционирование информационных технологий сказалось и на появлении новых и усовершенствованных методов по созданию низкокачественной контрафактной продукции, схожей с оригинальной, но способной нанести существенный ущерб здоровью. Существующие меры борьбы по обнаружению и устранению контрафактной продукции показывают себя все менее и менее эффективными, в связи с этим развитие национальной маркировки является весьма актуальным и перспективным направлением.

Актуальность проекта обосновывается огромными объемами производства и потребления промышленной и пищевой продукции, а также других товаров, использующихся в повседневной жизнедеятельности. В связи с чем фальсифицированная продукция может быть не замечена и доставлена конечным получателям, не подозревающим всей опасности.

Внедрение процесса маркировки в производство товаров позволит:

- обеспечить индивидуальную прослеживаемость каждой единицы товара (за счет уникальной идентификации кодов маркировки, передаваемых по защищенным каналам связи), продающейся в России, от фабрики или от границы страны до конкретного покупателя;
- сделать процесс производства товаров и их перемещения с завода-изготовителя до пунктов продажи максимально прозрачным и честным для пользователей;

					ПГУ 1.090301.36.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

- обеспечить повышение качества продукции;
- обеспечить ситуацию, в которой нелегальная торговля станет просто невыгодной и фактически невозможной;
- обеспечить повышение собираемости налогов и иных платежей
- обеспечить повышение качества контроля и управления во всех отраслях экономики;
- обеспечить защиту потребителя от некачественных товаров;
- осуществлять сбор и накопление статистических данных по наиболее востребованной продукции в конкретном регионе страны.

Таким образом, разработка компонента эмиссии и учета кодов маркировки выступит непосредственно входной точкой по старту внедрения, реализации и запуска процесса маркировки для каждой единицы произведенного товара, в связи с чем может считаться крайне актуальной и востребованной в современном мире.

Новизна разработки данной системы заключается в реализации уникального решения по объемам обрабатываемых и хранимых данных, охвату пользователей товарооборота, параллельности и нагрузке системы, что позволяет разработать приложение, удовлетворяющее критериям и требованиям, предъявляемых к современным системам аналогичного уровня.

Перечень сокращений и обозначений

GS1 – международная организация, ведающая вопросами стандартизации учета и штрихового кодирования логистических единиц.

GTIN – глобальный номер товарной продукции в единой международной

APM – автоматизированное рабочее место

АС – автоматизированная система

АСУТП – автоматизированная система управления технологическими процессами

базе товаров GS1, код товара

ГИС МТ – государственная информационная система маркировки товаров

ЕАЭС – евразийский экономический союз - Международная организация региональной экономической интеграции, обладающая международной правосубъектностью и учрежденная Договором о Евразийском экономическом союзе [1]

КМ – код маркировки

КП – код проверки

Криптохвост – ключ проверки и электронной подписи в составе кода маркировки.

НК – национальный каталог

ОПО – общее программное обеспечение

ОС – операционная система

ПК – программный комплекс

ПО – программное обеспечение

ПТК – программно-технический комплекс

РЭ – регистратор эмиссии

СИ – средство идентификации

СКЗКМ – система криптографической защиты кодов маркировки

СПО – специальное программное обеспечение

СУБД – система управления базами данных

СУЗ – станция управления заказами

СЭ – сервер эмиссии

ТГ – товарная группа

УКЭП – усиленная квалифицированная электронная подпись

УОТ – участник оборота товаров

ЦЭМ – центр этикетирования и маркировки

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

Перечень терминов и определений

Ниже приведен перечень терминов, использующихся в ВКР:

- средство вычислительной техники (СВТ) — ПЭВМ (персональная электронно-вычислительная машина) либо другое вычислительное оборудование (мэйнфрейм, мини-ЭВМ, микро-ЭВМ, КПК (карманный персональный компьютер), компьютерный терминал) [1];

- общее программное обеспечение — совокупность программных компонентов, обеспечивающая минимум функциональности СВТ:

1) среду для запуска и работы остальных программных средств (операционная система) [1];

2) средства для работы со структурированными наборами данных [1];

3) средства для доступа к ресурсам сети Интернет (веб-браузер) [1];

4) средства для публикации ресурсов СВТ в сети Интернет [1];

- специальное программное обеспечение — совокупность программных компонентов, специально разрабатываемых для данного конкретного средства вычислительной техники [1];

- агрегирование — процесс объединения товаров в транспортную упаковку первого уровня, а также транспортных упаковок первого уровня в транспортную упаковку последующих уровней с нанесением на создаваемую транспортную упаковку кода идентификации транспортной упаковки, содержащего информацию о взаимосвязи кодов идентификации каждого вложенного товара, потребительской упаковки, транспортной упаковки [1];

- государственная информационная система мониторинга за оборотом товаров, подлежащих обязательной маркировке средствами идентификации — государственная информационная система, созданная в целях автоматизации процессов сбора и обработки информации об обороте товаров, подлежащих обязательной маркировке средствами идентификации, хранения такой информации, обеспечения доступа к ней, ее предоставления и распространения, повышения эффективности обмена такой информацией и обеспечения

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

прослеживаемости указанных товаров, а также в иных целях, предусмотренных федеральными законами;

- код маркировки (КМ) — уникальная последовательность символов, состоящая из кода идентификации и кода проверки, формируемая оператором для целей идентификации товаров, в том числе в потребительской упаковке [1];

- код маркировки включает в себя 4 группы данных, из которых первая и вторая группы образуют код идентификации, третья и четвертая группа образуют код проверки:

- 1) первая группа состоит из 14 цифр и содержит код товара, которому предшествует идентификатор применения (01), генерируется оператором на основании сведений, поданных участником оборота товаров при регистрации товаров в информационной системе мониторинга [1];

- 2) вторая группа состоит из 13 символов (цифр, строчных и прописных букв латинского алфавита, а также специальных символов) и содержит индивидуальный серийный номер единицы товара, которому предшествует идентификатор применения (21) и который генерируется оператором или участником оборота товаров. Завершающим символом для данной группы данных необходимо использовать разделитель, имеющий код 29 в таблице символов ASCII [1];

- 3) третья группа состоит из 4 символов и содержит ключ проверки, которому предшествует идентификатор применения (91) и который генерируется оператором. Завершающим символом для данной группы данных необходимо использовать специальный символ-разделитель, имеющий код 29 в таблице символов ASCII [1];

- 4) четвертая группа состоит из 88 символов (цифр, строчных и прописных букв латинского алфавита, а также специальных символов) и содержит код проверки, которому предшествует идентификатор применения (92) и который генерируется оператором. Завершающим

символом для данной группы данных необходимо использовать специальный символ-разделитель, имеющий код 29 в таблице символов ASCII [1];

- код проверки (КП) — последовательность символов, сформированная в результате криптографического преобразования кода идентификации и позволяющая выявить фальсификацию кода идентификации при его проверке с использованием фискального накопителя и (или) технических средств проверки кода проверки, формируемая оператором [2];

- код товара (КТ, GTIN) — уникальный код, присваиваемый группе товаров при их описании в информационном ресурсе, обеспечивающем учет и хранение достоверных данных о товарах по соответствующим кодам единой Товарной номенклатуры внешнеэкономической деятельности Евразийского экономического союза (товарная номенклатура) [2];

- маркированные товары — товары, на которые нанесены средства идентификации и достоверные сведения о которых (в том числе сведения о нанесенных на них средствах идентификации и (или) материальных носителях, содержащих средства идентификации товаров) содержатся в информационной системе мониторинга [2];

- производитель товаров (производитель) — юридическое лицо или физическое лицо, зарегистрированное в качестве индивидуального предпринимателя, являющиеся налоговыми резидентами Российской Федерации, которые осуществляют производство и реализацию товаров [2];

- устройство регистрации эмиссии (РЭ) — программно-аппаратное шифровальное (криптографическое) техническое средство, включающее в себя техническое средство проверки кода проверки либо соответствующее требованиям к техническому средству проверки кода проверки, используемое участниками оборота товаров для обмена информацией с информационной системой мониторинга в части получения участниками оборота товаров кодов маркировки и передачи в информационную систему мониторинга сведений о

маркировке товаров средствами идентификации, в отношении которого Федеральной службой безопасности Российской Федерации выдан документ о его соответствии установленным требованиям к шифровальным (криптографическим) средствам защиты кодов маркировки [3];

- участники оборота товаров - юридические лица и физические лица, зарегистрированные в качестве индивидуальных предпринимателей, являющиеся налоговыми резидентами Российской Федерации, осуществляющие ввод товаров в оборот, оборот и (или) вывод из оборота товаров, за исключением юридических лиц и физических лиц, зарегистрированных в качестве индивидуальных предпринимателей, приобретающих товары для использования в целях, не связанных с их последующей реализацией (продажей) [4].

					ПГУ 1.090301.36.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

1 Программный компонент эмиссии и учета кодов маркировки

1.1 Анализ предметной области

1.1.1 Анализ проблемы

В современном мире весьма актуальна проблема «контрафактной продукции». Регулярно в новостях сообщают о перехвате контрафакта при ввозе на территорию РФ, о производстве “серых товаров” в подвальных помещениях, о недобросовестном применении современных информационных технологий, используемых в целях создания поддельной продукции. Такое пристальное внимание приковано к “серому рынку товаров” неспроста. Фальсификат, прежде всего, несет в себе опасность для жизни и здоровья каждого человека, потери доходов бизнеса, подрыв доверия населения к власти и производителям товаров. Производство и распространение контрафактной продукции ежегодно наносит ущерб, который измеряется миллиардами рублей. Большое количество людей страдает от использования поддельных товаров и продукции.

Применение мер для борьбы с контрафактной продукцией является очень актуальным, своевременным и просто необходимым в современном мире процессом. Это обусловлено катастрофическими цифрами по выявленной “серой продукции”. Так, по данным ФТС, за весь 2019 год в РФ было выявлено более 7,6 млн единиц такой продукции, а за три квартала 2020 г. – еще 8,2 млн единиц. И это без учета того, что часть контрафакта осталась в тени.

1.1.2 Способы решения

Для решения данной проблемы необходимо создание комплексной системы маркировки, которая обеспечит мониторинг и прослеживаемость товаров, а также оценку ситуации в сфере борьбы с контрафактом.

Казалось бы, уже давно, для этого был придуман всеми известный штрих-код, который используется на каждом товаре. Штрих-код (черно-белые полосы) — это международная система кодирования товаров с довольно ограниченными

возможностями. Структура штрих-кода ограничивается тринадцатью цифрами, которые несет за собой обширную информацию о товаре. Возможности штрих-код довольно ограничены:

- отсутствует возможность проверки, производителя и/или импортера товара;
- отсутствует возможность контроля количество привезенного или произведенного товара;
- отсутствует возможность получения информации о материалах, использующихся при производстве товара, о сертификатах и другие подтверждающих безопасность и качество документах.
- отсутствует возможность определения легальности товара, а также учета оплаты налогов и таможенных пошлин, условий доставки товара конечным пользователям и информации о продаже продукции.

Исходя из вышеописанных нюансов, штрих-код не представляет возможным подтверждение законности происхождения товара и ведения его учета. Штрих-код не является уникальным идентификационным знаком для одного и того же типа продукции, реализована возможность создания десятка тысяч одинаковых штрих-кодов. Соответственно, такой код является уязвимым знаком, подвергающимся легкой подделке и копированию с помощью современных средств и информационных технологий. Целью штрих-кода является классификация продукции на минимальном уровне.

Система цифровой маркировки обязана решать принципиально иную задачу, находящуюся на другом уровне сложности, а именно: обеспечивать индивидуальную прослеживаемость каждой единицы товара (за счет уникальной идентификации кодов маркировки, передаваемых по защищенным каналам связи), продающейся в России, от фабрики или от границы страны до конкретного покупателя (Рисунок 1).

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

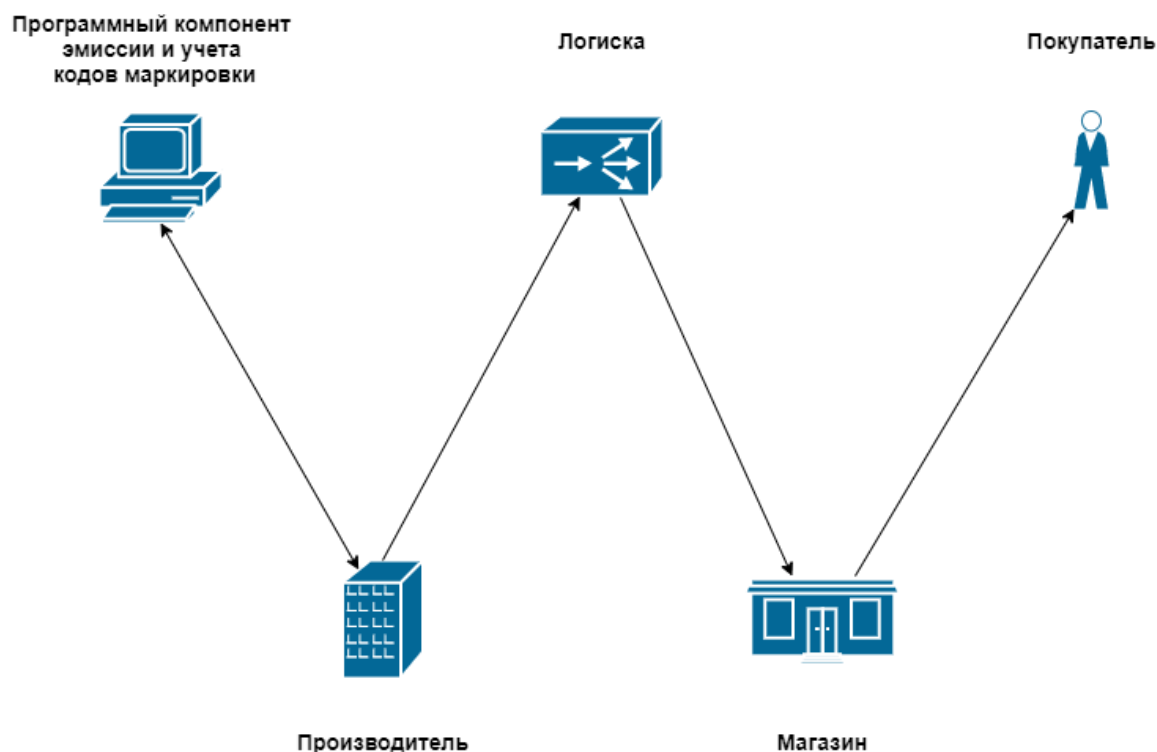


Рисунок 1 - Цепь производитель-покупатель

Помимо своей основной цели, которой является борьба с незаконным оборотом товаров, система преследует следующие цели:

- повышение собираемости налогов и иных платежей;
- повышение качества контроля и управления во всех отраслях экономики;
- защиту потребителя от некачественных товаров.

Как известно, кроссовки, духи или электронику можно купить в фирменном магазине, в известной сети, на официальном сайте или немного дешевле. Сомнения в подлинности товара могут возникнуть во многих ситуациях, поэтому кто-то изучает в сети признаки оригинальной вещи и сам проверяет товар, кто-то закрывает глаза на «серую» гарантию, а кто-то просто надеется на лучшее.

Несмотря на широкое распространение техники и гаджетов, у покупателя, чаще всего, отсутствует возможность проверить качество и легальность товара. Несомненно, последствия покупок такого рода могут быть самыми разными и иметь неприятный характер:

- некачественная продукция или товар;
- отказ в ремонте или возврате электроники или техники;
- аллергические и болезненные реакции на некачественную косметику.

Система маркировки и прослеживания товаров призвана защитить потребителя от приобретения нелегальной продукции.

Идея реализации маркировки не заиклена лишь на подтверждении законности происхождения товара, уникальность ее состоит в том, что она предусматривает тесное взаимодействие с потребителями и их участие в общественном контроле в торговле за счет современных средств вычислительной техники и информационных технологий.

Данная система обуславливается ее простотой и доступностью для непосредственных потребителей, то есть, граждан Российской Федерации. С помощью современных гаджетов и программного обеспечения идентификация товара может производиться в один клик, это позволяет моментально проверить информацию в единой базе о данной продукции и подтвердить/опровергнуть ее подлинность, а также предоставить потребителю необходимую информацию, которая могла бы быть полезна при покупке, например: наименование компании-производителя или импортера, дата производства, наименование компании-продавца, описание товара, а также дополнительная информация, указываемая в зависимости от категории товара (срок годности, максимальная розничная цена и т.п.).

Отсутствие товара в единой базе и невозможность его идентификации может свидетельствовать о том, что перед покупателем представлена фальсифицированная продукция, не являющаяся гарантом качества и безопасности. Данное нарушение может быть оформлено в виде жалобы (с использованием предоставляемого программного обеспечения) и передано в контролирующие и надзорные органы.

Предполагается, что маркировка и процессы прослеживания товаров обеспечат ситуацию, в которой нелегальная торговля станет просто невыгодной

и фактически невозможной, а процесс производства товаров и их перемещения с завода-изготовителя до пунктов продажи станет максимально прозрачным и честным для потребителей.

1.2 Обоснование актуальности разработки

Система цифровой маркировки товаров является действительно уникальным проектом, обеспечивающим поштучный учет и прослеживаемость продукции на территории России. Каждый товар с уникальным кодом может быть произведен или ввезен в страну и продан на кассе в магазине всего один раз. С помощью различных реализаций потребители товаров могут получать полную информацию о них.

В системе маркировки эмиссия КМ возложена на специальную систему - “Регистраторы эмиссии” (РЭ). Регистратор эмиссии в сфере маркировки представляет собой утвержденную правительством систему криптографической защиты кодов Государственной Информационной Системы Маркировки Товаров - ГИС МТ. РЭ представляет собой специальное оборудование, которое реализует криптографическую защиту кодов маркировки (КМ) для достижения необходимой информационной безопасности. РЭ предназначен для заказа кодов маркировки (уникальная последовательность символов, состоящая из нескольких частей и наносимая на товар или товарную упаковку в виде кода DataMatrix), чтобы в последующем исключить сбыт фальсификата, регистрации сведений о выпуске маркированных товаров и хранении кодов маркировки. РЭ представляет собой защищенное хранилище кодов маркировки, способное хранить до 200 миллионов кодов.

У самых истоков развития национальной маркировки в РФ стоят, так называемые, “железные РЭ”. Они представляют собой программно-аппаратные устройства, которые устанавливаются непосредственно на производственные линии, фабрики и заводы, и посредством специфичного бинарного протокола

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

обеспечивают вышеописанный функционал, что явилось первым и самым большим неудобством при внедрении данной системы.

В процессе эволюционирования национальной системы маркировки, было представлено несколько реализаций РЭ (Рисунок 2), а именно:

- облачные РЭ (“HTTP based”, в одном заказе до 150.000 КМ);
- безлимитные РЭ (в одном заказе до 2.000.000 КМ);
- программно-аппаратные РЭ.



Рисунок 2 - Виды регистраторов эмиссии

Промышленность РФ для целей маркировки подразделяется на отрасли, называемыми товарными группами (ТГ). Правительство РФ формирует логические объединения товаров в соответствии со спецификой каждой отрасли и бизнес-требованиями, определяемыми к каждой из них.

Необходимость реализации компонентов эмиссии объясняется рядом причин:

- неудобство и, в некоторых случаях, отсутствие возможности прямой интеграции АСУТП с РЭ, как с программно-аппаратными сервисами, реализующими эмиссию КМ. Исходя из разделения производства промышленности на товарные группы, каждому заводу или промышленной фабрике в разных отраслях необходимо создавать собственные интеграции с РЭ, предоставляющими сложный бинарный, HTTP протоколы и асинхронную систему эмиссии КМ;

- управление использованием РЭ. РЭ являются интернет-зависимыми устройствами и, в случае сбоев, регистраторы эмиссии будут способны самостоятельно производить резервацию кодовых данных, которых

ориентировочно должно хватить на 10-14 дней. Конечно, данный кейс является возможной уязвимостью, которая устраняется СУЗ за счет обеспечения конфигурируемости регистраторов для каждого УОТ и, в случае отказа одного из РЭ, переключения на другой доступный УОТ программно-аппаратный РЭ, причем не обязательно физический регистратор;

- ограничения программно-аппаратных РЭ. Аппаратный РЭ позволяет заказывать эмиссию КМ только по одной определенной товарной позиции (GTIN) в одном запросе, что усложняет построение интеграционного решения в таких отраслях, как Легкая промышленность (обувь, одежда и т.п.);

- необходимость процесса агрегирования. В рамках реализации логистических передвижений, товары перемещаются в транспортных упаковках, обеспечивающих защиту от повреждений и образующих самостоятельную транспортную единицу. Процесс агрегирования обеспечивает объединение соответствующих им товаров (товаров в потребительской упаковке) в транспортную упаковку товаров первого уровня, а также транспортных упаковок первого уровня в транспортную упаковку последующих уровней с нанесением на создаваемую транспортную упаковку кода идентификации транспортной упаковки содержащего информацию о взаимосвязи кодов идентификации каждого вложенного товара, потребительской упаковки, транспортной упаковки;

- необходимость формирования отчетности о нанесении КМ. Для реализации и обеспечения прозрачного и честного жизненного цикла товара, необходимо учитывать факт нанесения DataMatrix на произведенную продукцию. Обеспечением данной функциональности являются отчеты о нанесении КМ, которые в зависимости от отрасли продукции подразделяются на несколько видов: ручные (УОТ товарной группы самостоятельно должен формировать и подавать отчет) и автоматические (при получении КМ, на основании данных из заказа, формируется отчет). Данные отчеты направляются в РЭ по специальному протоколу, предоставляемому программно-аппаратными средствами. Функциональность формирования отчетов в РЭ, естественно, не

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

поддерживается и передача данных в РЭ происходит в виде, отличном от того, что предоставляет УОТ (протокол определяет РЭ);

- необходимость учета срока годности КМ и отбраковки выведенной продукции. В соответствии с логическим разделением промышленности на товарные группы, для каждой из них определяется индивидуальный срок годности поставляемых КМ, который должен контролироваться системой учета. В случае, если заказ был создан ошибочно и коды маркировки не будут реализованы, или же срок годности эммитированных КМ истек, необходимо формирование специального документа, отчета об отбраковке, который передается в РЭ за счет специального протокола и обеспечивает аннулирование ранее полученных КМ.

- необходимость выполнения биллинговых операций. При формировании заказов на эмиссию КМ и получении КМ появляется необходимость резервирования денежных средств (ДС) УОТ и их списания, в случае, успешного осуществления одной из описанных операций, или же отмены резерва - в случае неуспеха выполнения одного из действий.

- необходимость учета карточек товаров и реализации проверки заказов. Для старта маркировки конкретной продукции необходимо на этапе создания заказа осуществлять проверку соответствия данных о товаре, подаваемых пользователем и хранящихся в едином национальном каталоге, за счет которых, в том числе, определяется допустимость УОТ в реализации конкретной продукции. Для реализации данных проверок, необходимо вести учет и индексацию большого количества данных о товарах, так называемых карточках товара (КТ);

- разработка множественных интеграций и поддержка дополнительного программного обеспечения для маркировки продукции требует постоянного наличия в штате команд высококвалифицированных ИТ-специалистов, что является дорогостоящим удовольствием и ставит участников товарооборота в

положение неготовности выпуска продукции на первых этапах внедрения обязательной маркировки.

Все вышеописанные причины еще раз подтверждают необходимость создания высоконагруженной, отказоустойчивой и горизонтально масштабируемой системы, обеспечивающей удобство в использовании национальной системы маркировки производителями товаров и упрощающей их работу с необходимыми для эмиссии процессами.

1.2.1 Компоненты эмиссии и учета кодов маркировки

Разработка компонентов эмиссии и учета кодов маркировки призвана решить задачи упрощения взаимодействия пользователя (т.е. производителя товаров) со специальными криптографическими программно-аппаратными средствами, которые необходимы для получения по зашифрованным каналам цифровых кодов маркировки (данные программно-аппаратные средства утверждены указами правительства РФ), а также реализации бизнес-логики процессов эмиссии, устанавливаемой правительством для каждой товарной группы.

Данная необходимость обуславливает реализацию системы, которая представляет собой программное обеспечение, отвечающее за управление процессами обработки заказов на эмиссию и эмиссией кодов маркировки.

Непосредственно сама эмиссия осуществляется в сервере эмиссии - СЭ, клиентским устройством которого, установленным на предприятии, является регистратор эмиссии кодов маркировки. Таким образом, компоненты эмиссии, называемые станцией управления заказами (СУЗ), отвечают за взаимодействие с оборудованием, размещенным в инфраструктуре участника оборота товаров (УОТ), которое предназначено для:

- осуществления заказов на эмиссию КМ;
- отправки отчетов о нанесенных КМ;
- верификации КМ;
- агрегации готовой, упакованной продукции с нанесенными СИ.

Данное программное обеспечение должно стать уникальным решением, обеспечивающим взаимодействие между РЭ и производственными линиями (АСУТП), станциями технического зрения (верификации), принтерами этикеток, учетной системой предприятия и станциями агрегации, а также координирующим работу оборудования на производстве в части заказов на эмиссию КМ их распределения по принтерам и производственным линиям, принимающим списки КМ, прошедших верификацию, принимающим файлы агрегации со станций агрегации и осуществляющим их передачу в ГИС МТ, таким образом, представляя собой подсистему ведущую полный учет эмиссии и использования КМ на производствах.

Цель разработки данного специализированного ПО - обеспечение участников рынка автоматизированной подсистемой управления процессами эмиссии, нанесения КМ, верификации, агрегации, ведения транзакционных (с кодами маркировки) и системных журналов, мониторинга РЭ за счет имплементированного REST-интерфейса (REST API) и графического интерфейса участника оборота.

1.3 Обзор аналогов

Система маркировки и прослеживаемости товаров в России является специализированным программным обеспечением, предназначенным для осуществления контроля прохождения товарами цепочки начиная с производителей и импортеров и завершая потребителями. Компонент эмиссии и учета кодов маркировки, предоставляя собой начало процесса маркировки, представляет собой уникальный процесс, по своим масштабам, своему функционалу и возможностям не имеющий конкурентов ни на Российском, ни на всемирном рынке.

На текущий момент, российский рынок может наблюдать существование ряда подобных систем, реализующих небольшую часть аналогичного функционала. Ряд проектов работал и в предшествующие годы, но был закрыт

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

по разным причинам. Среди работающих на текущий момент систем можно выделить следующие:

- ЕГАИС – представляет единую государственную автоматизированную информационную систему учет объема производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции, которая создана в целях учета объема производства, анализа информации и контроля за производством;

- ЕГАИС «Маркировка» - представляет собой единую государственную систему учета оборота продукции изготовленной из натурального меха и является тестовым проектом;

- ФГИС «Меркурий» - представляет собой систему обеспечения оборота продукции, которая подлежит ветеринарному контролю.

Наиболее конкурентоспособным из представленных выше решений является система ЕГАИС, которая нацелена исключительно на оборот алкогольной продукции.

1.3.1 Единая государственная автоматизированная система

Одним из конкурирующих решений в сфере маркировки является федеральная информационная система ЕГАИС, целью которой является обеспечение информационно-технологической поддержки задача, установленных в ФЗ № 171-ФЗ «О государственном регулировании производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции [5].

С учетом направленности системы на обеспечение полноты и достоверности учета производства и оборота этилового спирта, алкогольной и спиртосодержащей продукции, определяется узкая специализация системы на работу исключительно с алкогольной продукции, что является большим неудобством для мультитоварных производителей товаров, которым придется разрываться между несколькими систем маркировки.

Нельзя не выделить тот факт, что система является слабоориентированной на потребителей продукции, так как процессы построены на учете федеральных

и акцизных марок (Рисунок 3), начислении акциза, учета оборота и производства алкогольной продукции.

Одним из самых главных и явных уязвимостей системы является отсутствие контроля акцизных марок в большинстве розничных магазинов. Произведенная марка может быть подделана и являться контрафактной, но система ЕГАИС все равно позволит осуществить продажу, что не позволяет полностью преследовать установленную решением цепь: производитель-оптовик-розница-покупатель.

Исходя из вышесказанного, декларируемые цели данной системы о поштучном учете и снижении контрафакта в рознице не достигнуты.



Рисунок 3 - Акцизная марка

1.3.2 Национальная система маркировки

Компонент эмиссии и учета кодов маркировки под эгидой системы национальной маркировки «Честный знак» представляет собой специализированное программное обеспечение с развитым пользователь-ориентированным интерфейсом. Данная система реализует широкую

специализацию в сфере экономике Российской Федерации по работе с различными отраслями по производству продукции, а функционал системы компонента эмиссии учета кодов маркировки разработан с учетом исправления совокупности недостатков аналогичных систем и наполнения информационного продукта уникальными решениями в сфере IT-технологий.

Так, система подразумевает идентификацию каждой единицы товара с помощью присвоения цифровых кодов, каждый из которых является уникальным и защищенным современными средствами криптографии, что позволяет непосредственным заказчикам (государству), бизнесу (производителям товаров) и конечным потребителям и пользователям (гражданам Российской Федерации) осуществлять контроль любого товара на его пути от производителя до потребителя, противодействуя и предотвращая незаконный оборот товарооборота.

Модель функционирования системы включает в себя ряд процессов, перечисленных далее:

- уникальная идентификация товара;
- создание и развитие информационной технологической системы маркировки, которая обеспечивает хранение всей информации о товаре и его движении;
- формирование единого каталога товаров России.

Наиболее выделяемым преимуществом данной системы выступают криптографические способы шифрования кодов маркировки и их преобразования в цифровой код, который преобразуется в двухмерный штрих-код стандарта Datamatrix (Рисунок 4). Данный штрих-код содержит включает большое количество закодированной информации и считывается даже при повреждениях кода.



Data Matrix штрих код

Рисунок 4 - Цифровой код Datamatrix

1.4 Постановка задачи

Требуется разработать систему компонентов (специализированное ПО), реализующую программный (REST API) и графический (UI) интерфейсы для обеспечения участников товарооборота автоматизированной подсистемой управления процессами эмиссии и нанесения КМ, верификацией и агрегаций.

Система компонентов должна реализовывать:

- осуществление заказов на эмиссию кодов маркировки;
- предоставление эмитированных кодов маркировки (посредством API в формате json и UI в форматах PDF, CSV, EPS) для нанесения на продукцию;
- верификацию нанесения КМ;
- агрегацию готовой, упакованной продукции с нанесенными средствами идентификации (СИ);
- списание бракованной продукции;

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

- взаимодействие с системой формирования биллинговых резервов и списания денежных средств;
- взаимодействие с системой “Национальный каталог” в части регистрации и учета карточек товаров;
- регистрацию участника оборота товаров в системе “СУЗ” и реестре УОТ при заключении договора обслуживания;
- единый механизм авторизации пользователей в системе;
- реализация механизма квитирования событий системы;
- поддержку сети дистрибуции кодов маркировки.

Общие требования к системе:

- микросервисная архитектура взаимодействия компонентов;
- обеспечение отказоустойчивости системы;
- многопоточная и асинхронная обработка данных;
- использование протоколов HTTP и HTTPS;
- формат передаваемых данных JSON;
- операционные системы Microsoft Windows, Linux, Mac OS;
- система программирования IntelliJ IDEA;
- языки программирования Java, JavaScript;
- библиотеки Spring 5, Spring Boot 2, Protobbeans;
- реляционная СУБД PostgreSQL;
- нереляционная СУБД CouchDb;
- HTTP-клиент Feign;
- библиотека для создания и использования отчетов JasperReports
- система развертывания и поддержания жизненного цикла компонентов Imagenarium;
- интеграция компонентов системы с государственной информационной системой маркировки товаров (ГИС МТ).

1.5 Выбор инструментов разработки

Анализ и выбор инструментов разработки является неотъемлемой частью процесса реализации и тестирования программного обеспечения и оказывает непосредственное влияние на качество продукта, стройность и масштабируемость архитектуры, корректность работы и соответствие поставленным требованиям.

1.5.1 Языки программирования

На начальном этапе проектирования и создания любой программы перед разработчиком системы становится важный вопрос - вопрос выбора языка программирования. Данный этап является наиболее важным и ответственным, так как технологии, выбранные в рамках данного процесса, будут использоваться во время всего жизненного цикла продукта и определять скорость и трудозатраты разработки программы, быстродействие программы, скорость тестирования, гибкость использования программы на разных платформах и так далее.

На данный момент существует огромное количество языков программирования, классифицируемых по типу решаемых задач, уровню, организации или парадигме языка. Выбор языков программирования в рамках данной работы был основан на ряде следующих критериев:

- скорость работы конечного продукта;
- объем занимаемой оперативной памяти;
- скорость разработки программы;
- кроссплатформенность;
- скорость внесения изменений и тестирования;
- объемы и быстрота разработки.

1.5.1.1 Язык программирования сервера

Северные языки программирования применяются для реализации алгоритмов, бизнес-логики, описания сценариев использования сайта или приложения.

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		31

Существует большое количество языков программирования, используемых при разработке серверной части приложения. На российском рынке наиболее популярными являются Java, C#, Python, PHP.

В ходе анализа языка программирования, подходящего для данной системы, был выбран строго типизированный объектно-ориентированный язык программирования от компании Sun Microsystems – Java. Данный выбор был основан на следующих факторах:

- гибкость - в отличие от C, C# Java является кроссплатформенным языком программирования и за счет наличия Java-машины позволяет код на разных платформах и операционных системах;

- скорость – как правило, Java имеет преимущество в скорости выполнения программного кода среднего или большого объема, в отличие от Python, интерпретируемого языка программирования, выполнения больших скриптов которого может занять продолжительное время;

- тип языка – Java является компилируемым языком программирования;

- порог входа и синтаксические особенности – языки C, C# содержат функции, которые считаются довольно-таки сложными для входа в программирование, а также излишний «синтаксический сахар» языка C# оказывает отрицательное влияние на читабельность кода;

- использование – как известно, большое количество высоконагруженных и параллельных систем написано на языке программирования Java. Данная технология является более открытой и поддерживает огромное количество open-source проектов, доступных для пользователей на просторах интернета.

1.5.1.2 Библиотека сервера

В качестве основной библиотеки для backend-приложения была выбрана Spring Framework – универсальная библиотека с открытым исходным кодом для Java-платформы, предназначенный для решения большого количества и объема задач, с которыми сталкиваются java backend-разработчики [6]. Spring Framework представляет собой целую экосистему (Рисунок 5) в виде коллекции внутренних

библиотек, которые, в большинстве своем, работают не зависимо друг от друга, обеспечивая большую функциональность при их совместном использовании. Данная библиотека обеспечивает:

- гибкую настройку приложения за счет функциональности конфигурирования и управления объектами;
- собственную MVC-платформу веб-приложений;
- предоставляет свой слой доступа к базам данных.



Рисунок 5 - Экосистема Spring

1.5.1.3 Язык программирования клиента

В качестве языка разработки клиентской части приложения был выбран язык программирования JavaScript на основании своих несомненных достоинств:

- веб-разработка – в данной сфере язык JavaScript является незаменимым, так как поддерживает скрипты со всеми популярными браузерами и обеспечивает полную интеграцию верстки страниц с серверной частью;
- экосистема – обеспечивает наличие большого количества готовых решений в своей копилке за счет большого опыта применения на рынке;
- простота применения – благодаря синтаксическим особенностям языка и его встроенным функциям решение простых задач обеспечивается в кратчайшие сроки;
- легкость освоения – низкий порог входа для написания базового функционала с использованием данного языка.

1.5.2 Интегрированная среда разработки

Производя анализ и сравнение, существующих инструментов, используемых для разработки программного обеспечения на языке Java, была выбрана интегрированная среда разработки от компании JetBrains IntelliJ IDEA.

IntelliJ IDEA — интегрированная среда разработки программного обеспечения, разработанная компанией JetBrains и предоставляющая свои возможности для многих языков программирования, таких как: Java, JavaScript, Python. Данная среда разработки сильно выделяется на фоне своих непосредственных конкурентов обширным набором инструментов для проектирования, рефакторинга, отладки, тестирования и оптимизации кода.

Инфраструктура и дизайн данной среды разработки направлены в первую очередь на продуктивность работы программистов в написании программного кода, позволяя сконцентрироваться именно на функциональных задачах, не отвлекаясь на выполнение рутинных операций, которые IntelliJ IDEA берет на себя.

В зависимости от потребностей использования среды разработки и ее функциональных возможностей, выделяют две редакции: Community Edition и Ultimate Edition, отличающиеся функциональными возможностями и ценой.

В отличие от своих прямых конкурентов (Eclipse и Netbeans) IntelliJ IDEA обеспечивает:

- богатую функциональность системы, где осуществлена поддержка многих языков и фреймворков, которые используются в веб разработке: Java, Kotlin, Groovy, JavaScript, Spring, Spring Boot и другие;
- большой выбор инструментов для работы с кодом — система обеспечивает разработчиков интеллектуальной помощью во время написания кода, выполняет гибкий анализ и создается виртуальную карту кода, статический анализ для обнаружения ошибок, автоматического дополнения с учетом контекста доработки, огромный инструментарий по рефакторизации кода.

дополнения с учетом контекста доработки, огромный инструментарий по рефакторингу кода и так далее;

- инструменты для совместной и удаленной работы – режимы парного программирования, интеграции внутри платформы и режим удаленной работы позволяют осуществлять командную разработку программного обеспечения;

Среди прочих возможностей, интегрированная среда разработки хорошо совместима со многими популярными свободными инструментами разработчиков, такими как Apache Maven, Ant, JUnit.

1.5.3 Сборка программного обеспечения

В рамках данной работы в качестве инструмента для управления и сборки проектов используется библиотека Apache Maven. Apache Maven — библиотека для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, который является подмножеством языка описания XML.

Непосредственным и наиболее известным конкурентом является система сборки Gradle. Можно выделить следующие преимущества системы сборки Maven, который повлияли на выбор именно данной системы:

- независимость от операционной системы. На входе – сборка системы осуществляется в любой операционной системе, на выходе – один и тот же файл проекта;

- Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. Различие заключается в том, что файлы описания проекта содержат спецификацию самого проекта, а не его отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов;

- свою инфраструктуру – в отличие от Gradle, Maven имеет свой репозиторий, из которого загружаются зависимости, отсюда удобство использования и добавления своих библиотек в качестве maven-зависимостей.

1.5.4 Система развертывания и управления компонентами

Одним из самых важных и необходимых инструментов разработчика является система развертывания, поддержания жизненного цикла и управления компонентами.

Исходя потребности в системе, которая должна сочетать простоту и удобство в использовании, легкость в освоении и отсутствие необходимости глубоких знаний в сфере DevOps процессов, была выбрана контейнерная платформа Imagenarium, реализованная на базе классического Docker и предназначенная для упрощения и автоматизации многих рутинных процессов при разработке программного обеспечения.

Данная система позволяет с легкостью перейти от традиционных методов разработки ПО к полностью контейнеризированным решениям, предоставляя широкий выбор инструментов для решения трудоемких задач по упаковке, конфигурированию, тестированию и развертыванию разрабатываемых программных продуктов.

В отличие от своего непосредственного конкурента Kubernetes, для развертывания компонента в контейнере необходимо лишь описать шаблон развертывания и с помощью визуального интерфейса произвести деплой сервиса в контейнер простым заполнением значений параметров, описанных в шаблоне, в то время как Kubernetes требует приложить не мало усилий и опыта квалифицированного сотрудника для развертывания даже небольшого сервиса.

Одной из самых важных особенностей – направленность системы, в первую очередь на разработчиков и тестировщиков, что позволяет производить значительные манипуляции по развертыванию компонентов нажатием нескольких кнопок интерфейса.

1.5.5 Системы управления базами данных

При разработке программного обеспечения, задействующего в своем функционале логику работы с базами данных, немаловажным инструментом является система управления базами данных.

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36

Система управления базами данных обеспечивает механизм хранения и поиска данных. Существует три основных типа систем управления базами данных:

- системы управления реляционными базами данных;
- онлайн-системы аналитической обработки;
- системы управления нереляционными базами данных.

1.5.5.1 Система управления реляционными базами данных

Система управления реляционными базами данных является основой для SQL и для всех современных систем баз данных, таких как PostgreSQL, MS SQL Server, Oracle, MySQL и Microsoft Access.

Система управления реляционными базами данных представляет собой систему управления базами данных, которая основанная на реляционной модели.

В рамках данной работы в качестве системы управления реляционными базами данных используется платформа pgAdmin (Рисунок 6 - Интерфейс PgAdmin), предназначенная для администрирования и разработки PostgreSQL и связанных с ней систем управления базами данных. Платформа написана на Python и jQuery и поддерживает все функции PostgreSQL. PgAdmin предназначен для любых операций, начиная с записи базовых SQL-запросов и заканчивая осуществлением мониторинга ваших баз данных и настройки продвинутых архитектур баз данных.

Основными преимуществами данной СУБД относительно своих аналогов являются:

- PostgreSQL представляется собой не просто реляционную базу данных, а объектно-реляционную, что дает ряд преимуществ над другими БД с открытым исходным кодом, например: поддержка пользовательский объектов, а также их поведения, включая типы данных, функции, операции, домены и индексы [7];
- PostgreSQL поддерживает самый обширный список типов данных, начиная с числовых и чисел с плавающей точкой и заканчивая UUID, json, битовыми строками и другими [7];

- PostgreSQL может обрабатывать большие объемы данных, о чем говорит официальная документация (существует ряд ограничений, который так же указаны). Для сравнения, другие реляционные БД, такие как MySQL и MariaDb печально известны своими ограничениями по размеру строк [7].

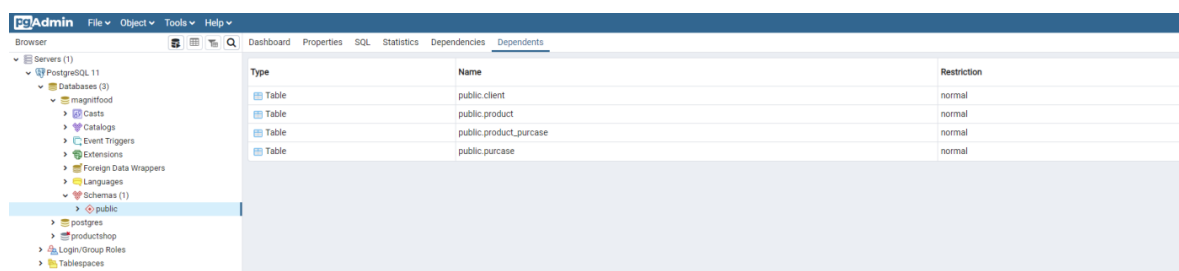


Рисунок 6 - Интерфейс PgAdmin

1.5.5.2 Система управления нереляционными базами данных

Нереляционная база данных — это база данных, которая предоставляет механизм для хранения и извлечения данных, отличных от табличных отношений, используемых в реляционных базах данных. Эти базы данных не содержат схем, поддерживают простую репликацию, имеют простой API, в конечном итоге непротиворечивы и могут обрабатывать (большие данные).

Исходя из архитектурных потребностей в осуществлении репликаций баз данных, а также хранения большого количества данных, имеющих нестандартную вложенную структуру, возникла потребность в использовании нереляционной БД, так, в рамках данной дипломной работы в качестве системы управления нереляционными базами данных была выбрана CouchDB (Рисунок 7) - база данных NoSQL для хранения документов, которая представляет возможность хранения документов с уникальными именами, REST API для выполнения базовых операций с документами БД, например: чтение, обновление, добавление, редактирование и удаление.

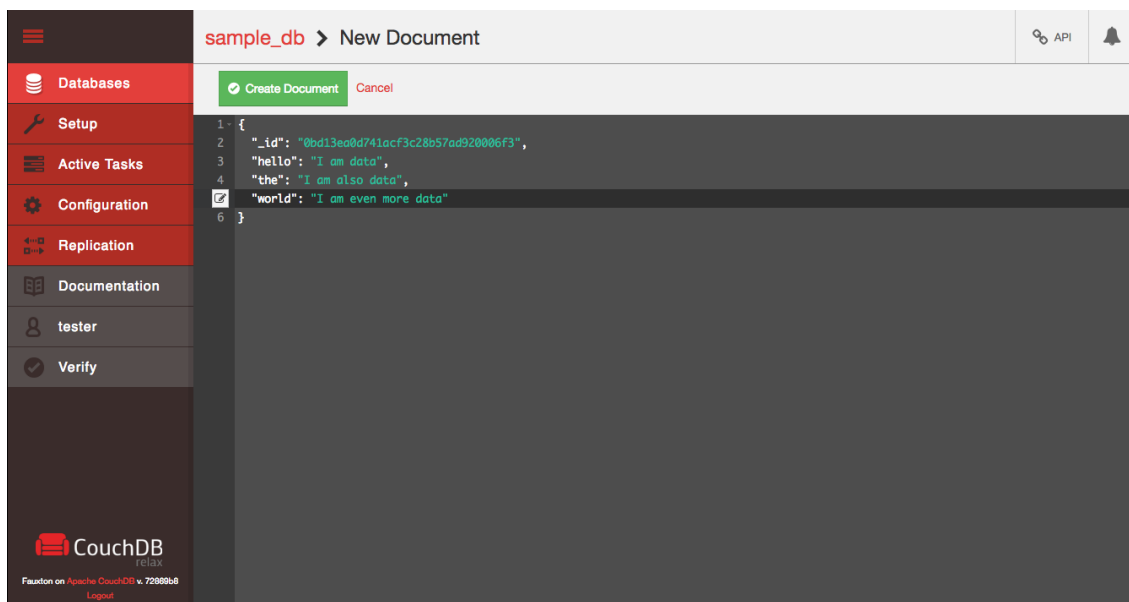


Рисунок 7 - Интерфейс CouchDb

Главные преимущества данной БД:

- документо-образная структура данных;
- REST-интерфейс для всех операций;
- множественные репликации;
- фильтры репликации;
- удобный в использовании графический интерфейс.

1.5.6 Система управления версиями

Для хранения программного кода и поддержки возможности его совместной разработки используется веб-приложение и система управления репозиториями программного кода Git - Gitlab, включающий в себя систему контроля версий для размещения различных цепочек разработки и веток, позволяя разработчикам проверять код и откатываться к стабильной версии программного продукта в случае непредвиденных проблем.

GitLab является конкурентом GitHub, в котором среди многих других проектов размещается разработка ядра Linux Линуса Торвальдса.

Использование данного ПО обуславливается удобством управления разрабатываемым кодом и возможностью подключения множества интеграций, таких как - Jira (решение для отслеживания задач и дефектов ПО), процессы CI

(continuous integration), представленный на Рисунок 8, и CD (continuous delivery), мониторинг, Gitlab Registries - безопасный приватный реестр для образов (images) Docker.

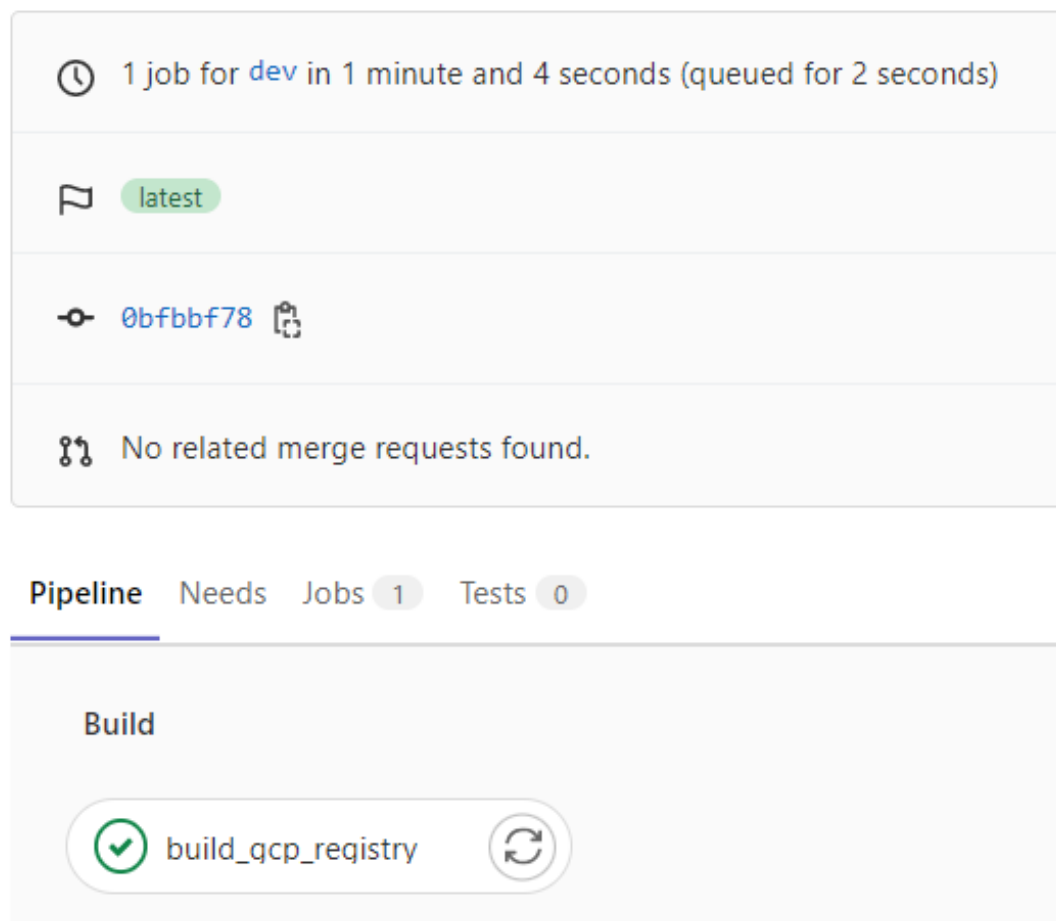


Рисунок 8 - Процесс continuous integration

1.5.7 Система визуального проектирования отчетов

Для работы с шаблонами этикеток, применяемых при печати кодов маркировки, в рамках данной дипломной работы используется программное обеспечение Jaspersoft Studio - конструктор отчетов на основе Eclipse для библиотеки JasperReports и сервера JasperReports, который можно использовать как плагин Eclipse или как отдельное приложение [8]. Jaspersoft Studio позволяет:

- создавать сложные макеты, включая диаграммы, изображения, подотчеты, кросс-таблицы;

- получить доступ к данным через различные источники, такие как JDBC, TableModels, JavaBeans, XML, Hibernate, большие данные (например, Hive), CSV, XML / А и пользовательские источники;

- опубликовать отчет в формате PDF, RTF, XML, XLS, CSV. , HTML, XHTML, текст, DOCX или OpenOffice.

Jaspersoft Studio является инструментом визуального проектирования отчетов, который облегчает визуальное создание отчетов. Результатом разработки является файл XML в формате .jrxml, а файл .jrxml может быть скомпилирован в файл формата .jasper для удобного анализа с помощью механизма отчетов JasperReport [8].

1.6 Архитектура приложения

Архитектура программного обеспечения представляет собой совокупность важнейших решений об организации программной системы.

Архитектура каждого приложения включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;

- соединение выбранных элементов структуры и поведения во всё более крупные системы;

- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

1.6.1 Концепция архитектуры

Исходя из потребностей и нужд участников оборота товаров в реализации автоматизированной системы управления процессами эмиссии, нанесения КМ, верификации, агрегации, ведения транзакционных (с кодами маркировки) и системных журналов, мониторинга РЭ за счет унифицированного интерфейса, архитектурным решением компонента эмиссии учета кодов маркировки является

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

АС, представляющая совокупность технически объединенного специального программного обеспечения, реализованного в виде микросервисов.

За основу положена микросервисная архитектура, представляющая собой распространенный подход к разработке ПО, при котором большое и объемное приложение, в процессе декомпозиции, разбивается на небольшие автономные компоненты, называемые микросервисами, с четко определенными интерфейсами. Данная архитектура служит еще одним вариантом сервис-ориентированной архитектуры программного обеспечения, направленным на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей.

Каждый микросервис представляет собой независимую единицу, имеющую четкие функциональные и физические границы, что позволяет разбивать монолитные процессы и обеспечивать выполнение простых функций каждым микросервисом. Данная особенность архитектуры обеспечивает ее масштабирование для дополнения и расширения сервисов, а также гибкость внесения изменений, внедрения новых технологий и высокую отказоустойчивость при наличии неисправности или неполадки конкретного сервиса.

Таким образом, архитектура компонента эмиссии и учета кодов маркировки представляется в виде совокупности информационных сервисов и комплекса технических средств, которые включают в себя, в том числе, средства вычислительной техники, программного и информационного обеспечения, предназначенной для выполнения целевых функций системы. На Рисунке 9 приведены составные части и подсистемы компонента эмиссии и учета кодов маркировки.

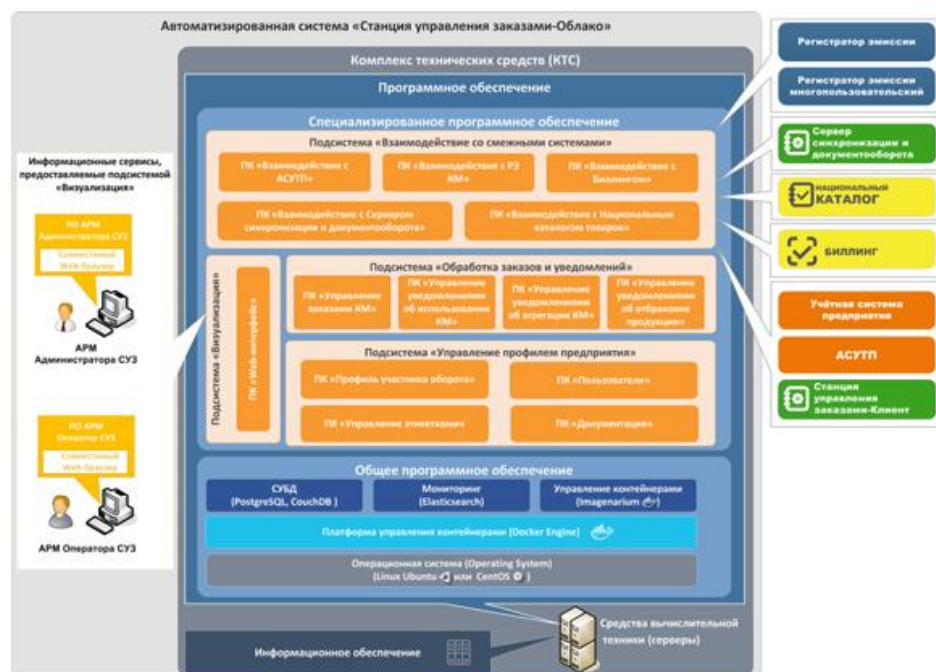


Рисунок 9 - Архитектура компонента эмиссии и учета кодов маркировки

1.6.2 Подсистемы и составные части программного компонента

Компонент эмиссии учета кодов маркировки включает в себя следующие подсистемы:

- подсистема, обеспечивающая визуализацию и графический интерфейс
- подсистема, обеспечивающая обработку заказов и уведомлений;
- подсистема, реализующая управление профилем предприятия;
- подсистема, осуществляющая взаимодействие со смежными системами.

Также, специальное программное обеспечение, представленное в виде автоматизированной системы включает в себя следующие составные части:

- в составе подсистемы «Визуализация»:
 - 1) программный комплекс «Web-интерфейс»;
- в составе подсистемы «Управление профилем предприятия»:
 - 1) программный комплекс «Профиль участника оборота»;
 - 2) программный комплекс «Пользователи»;
 - 3) программный комплекс «Управление этикетками»;
 - 4) программный комплекс «Документация».
- в составе подсистемы «Обработка заказов и уведомлений»:

- 1) программный комплекс «Управление заказами КМ»;
- 2) программный комплекс «Управление уведомлениями об агрегации КМ»;
- 3) программный комплекс «Управление уведомлениями об отбраковке продукции»;
- 4) программный комплекс «Управление уведомлениями об использовании КМ»;

- в составе подсистемы «Взаимодействие со смежными системами»:

- 1) программный комплекс «Взаимодействие с АСУТП»;
- 2) программный комплекс «Взаимодействие с системами товарных групп»;
- 3) программный комплекс «Взаимодействие с РЭ КМ»;
- 4) программный комплекс «Взаимодействие с Биллингом»;
- 5) программный комплекс «Взаимодействие с Национальным каталогом товаров»;
- 6) программный комплекс «Взаимодействие с Центральным сервером синхронизации и документооборота».

1.6.3 Сервисы программного компонента

Исходя из основных паттернов разработки программного обеспечения на основе микросервисной архитектуры, как описано выше, реализован принцип логического объединения функциональности в атомарную единицу, которая представляет собой микросервис. В соответствии с этим, можно выделить логические группы и представить следующее разделение на микросервисы (Рисунок 10):

- микросервис заказов - обеспечивает обработку, учет, хранение и процессинг заказов;
- микросервис отчетов - обеспечивает обработку, учет, хранение и процессинг отчетов;

- микросервис биллинговых операций - обеспечивает взаимодействие с биллинговыми системами;
- микросервис кодов маркировки – обеспечивает заказ, учет, хранение и обработку уникальных кодов идентификации;
- микросервис карточек товара – обеспечивает обработку, учет, хранение и процессинг данных о товарах;
- микросервис дистрибуции КМ – обеспечивает реализацию процесса дистрибуции кодов маркировки;
- микросервис квитирования – обеспечивает формирование событий в соответствии с действиями, производимыми пользователями или внешними системами, и сборку квитанций на их основе;
- микросервис менеджмента пользователей – обеспечивает регистрацию, учет, обработку, хранение и процессинг профилей пользователей;
- микросервис роутинга – обеспечивает роутинг запросов между объединенными в контейнеры микросервисами.

Рисунок 10 - Микросервисы компонента эмиссии и учета кодов маркировки

1.6.4 Модель системы

Если в традиционной разработке с применением сервис-ориентированной архитектуры модули могут иметь достаточно сложную структуру программной системы, взаимодействие внутри которой зачастую осуществляется с использованием стандартизованных тяжеловесных протоколов таких, как: SOAP, XML-RPC, то в микросервисной архитектуре специального программного обеспечения, представленного в данной работе, системы выстраиваются из компонентов, выполняющих элементарные функции и взаимодействующих на основе простых в использовании и экономичных сетевых коммуникационных протоколов в стиле REST при взаимодействии с JSON. Непосредственно за счет повышения гранулярности модулей архитектура нацелена на уменьшение степени зацепления и увеличение связности, что позволяет проще добавлять и изменять функции в системе на каждом этапе жизненного цикла программного обеспечения.

Каждый микросервис предоставляет интерфейс для взаимодействия основанный на REST-архитектуре (Рисунок 11), то есть на передаче репрезентативного состояния [9], и представляет набор согласованных ограничений распределенной системы. Данный архитектурный стиль обеспечивает повышение производительности компонентов системы и упрощает модель системы в целом.

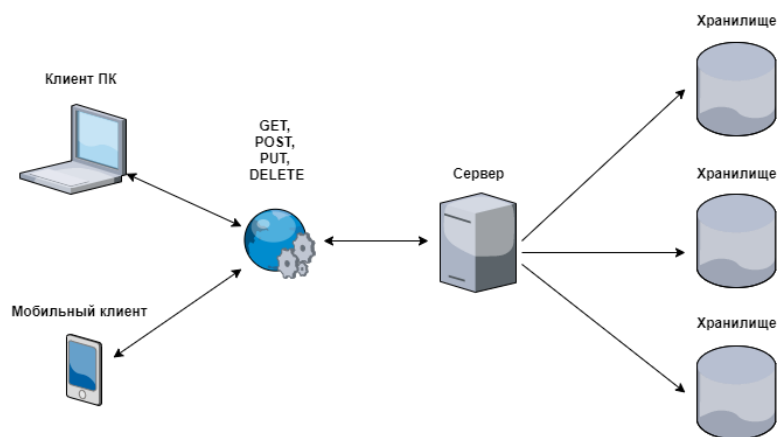


Рисунок 11 - REST архитектура

1.6.5 Взаимодействие с внешними системами

При развёртывании программный компонент эмиссии и учета кодов маркировки встраивается в технологическую инфраструктуру участника товарооборота в соответствии с Рисунок 12, отображающим промышленное применение автоматизированной системы.

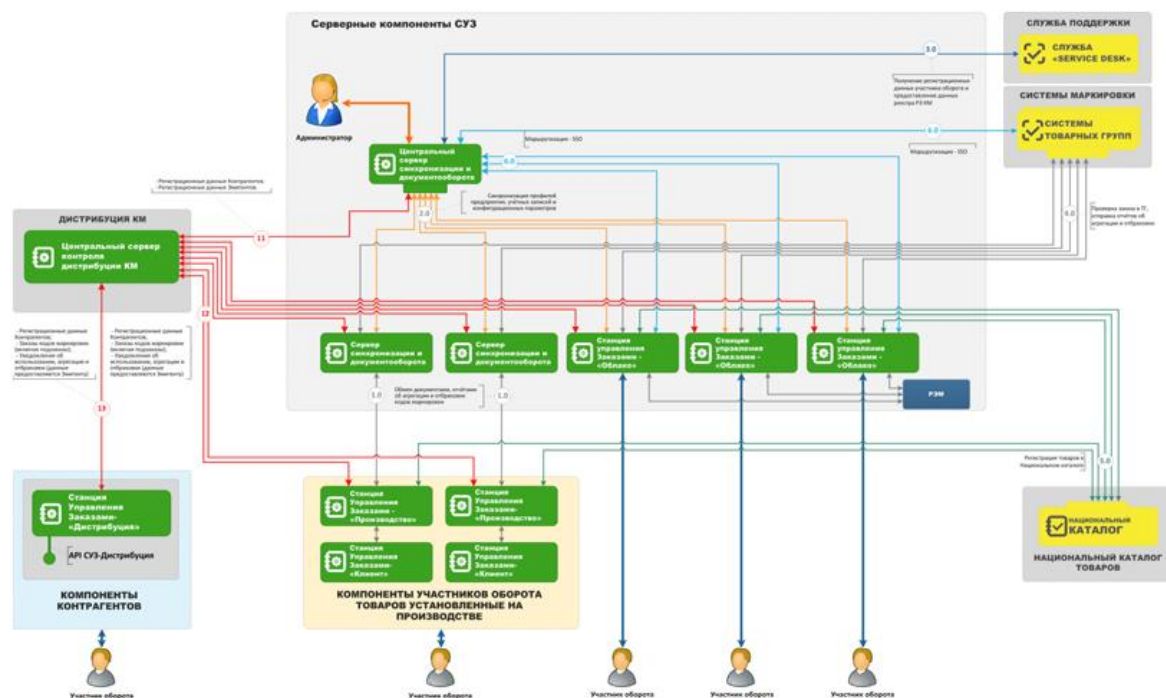


Рисунок 12 - Взаимодействие программного компонента с внешними системами

В ходе эксплуатации автоматизированной системы специализированного программного обеспечения в рамках инфо-коммуникационного и функционального взаимодействия осуществляет обмен данными со смежными системами, перечень и описание которых отображает

Таблица 2 - Список смежных систем

Условное обозначение	Описание
Центральный сервер синхронизации и документооборота	Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с Центральным сервером синхронизации и документооборота осуществляется в рамках получения информации о профиле предприятия и сведений регистраторов эмиссии
АСУТП	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с АСУТП:</p> <ul style="list-style-type: none"> – взаимодействие между системами осуществляется в рамках процесса создания и обработки заказов на эмиссию кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса предоставления эмитированных кодов маркировки для нанесения на продукцию; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов об использовании кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов об агрегации; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов о выбытии/отбраковке кодов маркировки.

.

Таблица 2 - Список смежных систем

Условное обозначение	Описание
Центральный сервер синхронизации и документооборота	Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с Центральным сервером синхронизации и документооборота осуществляется в рамках получения информации о профиле предприятия и сведений регистраторов эмиссии
АСУТП	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с АСУТП:</p> <ul style="list-style-type: none"> – взаимодействие между системами осуществляется в рамках процесса создания и обработки заказов на эмиссию кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса предоставления эмитированных кодов маркировки для нанесения на продукцию; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов об использовании кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов об агрегации; – взаимодействие между системами осуществляется в рамках процесса отправки и обработки отчетов о выбытии/отбраковке кодов маркировки.

Условное обозначение	Описание
АС «Сервер Эмиссии» (СЭ КМ)	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с АС «Сервер Эмиссии»;</p> <ul style="list-style-type: none"> – взаимодействие между системами осуществляется в рамках процесса получения эмитированных кодов маркировки от АС «Сервер Эмиссии»; – взаимодействие между системами осуществляется в рамках процесса предоставления информации об использовании кодов маркировки в АС «Сервер Эмиссии»
АС «Регистратор эмиссии» (РЭ КМ)	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с АС «Регистратор эмиссии»:</p> <ul style="list-style-type: none"> – взаимодействие между системами осуществляется в рамках процесса создания и закрытия заказов на эмиссию кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса отправки уведомлений об использовании кодов маркировки; – взаимодействие реализовано с использованием бинарного протокола, который определен производителем АС «Регистратор эмиссии».
Национальный каталог товаров	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с Национальным каталогом товаров в рамках процесса учета и обработки сведений о внесении или обновлении данных каталога.</p>

Условное обозначение	Описание
Системы товарных групп	<p>Подсистема «Взаимодействие со смежными системами» автоматизированной системы управления обеспечивает взаимодействие с системами товарных групп:</p> <ul style="list-style-type: none"> – взаимодействие между системами осуществляется в рамках процесса предоставления сведений об агрегации и выбытии/отбраковке кодов маркировки; – взаимодействие между системами осуществляется в рамках процесса проверки заказа на эмиссию кодов маркировки.

1.7 Проектирование и разработка

1.7.1 Проектирование программного компонента

Процесс проектирования программы является одним из наиболее важных этапов жизненного цикла создаваемого продукта. На данном этапе определяется поиск способа, который будет удовлетворять требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений, а также формируются цели проекта.

1.7.1.1 Определение назначения

В ходе проектирования программного компонента были определены цели работы автоматизированной системы в составе компонента эмиссии и учета кодов маркировки для обеспечения выполнения целевых функций специализированного программного обеспечения:

- процесс осуществление заказов на эмиссию кодов маркировки в разрезе товарных групп;
- процесс предоставления эмитированных кодов маркировки для нанесения на продукцию;
- процесс верификации нанесения КМ, формирования отчета о нанесении;
- процесс агрегации готовой и упакованной продукции с нанесенными средствами идентификации (СИ);

- процесс списания бракованной продукции с указанием причины отбраковки.

1.7.1.2 Проектирование основных функций

В рамках проектирования выпускной квалификационной работы, были определены основные функции программы, выполнение которых обеспечит реализацию специализированного программного обеспечения автоматизированной системы в соответствии с установленными целями:

- функция получения заказов;
- функция предоставления эмитированных кодов маркировки;
- функция получения информации об использовании (нанесении) кодов маркировки и регистрации статуса кодов маркировки в Системе товарной группы;
- функция получения информации об агрегации кодов маркировки и регистрации агрегации упакованной продукции в Системе товарной группы;
- функция получения информации об отбраковке продукции и регистрации отбракованной продукции в Системе товарной группы.

1.7.1.3 Определение показателей назначения

Программный компонент эмиссии и учета кодов маркировки предназначен для обеспечения реализации целевых функций в соответствии с показателями назначения системы, определенными в ходе проектирования и отраженными в таблице 3.

Таблица 3 - Перечень показателей назначения, которым должна удовлетворять система

Показатели назначения	Значение показателей
Количество товарных позиций в одном заказе КМ	Не более 10
Количество КМ одной товарной позиции (КТ, GTIN) в заказе	Не более 150000 КМ
Количество КМ в отчете о нанесении кодов маркировки	Не более 30000 КМ
Количество КМ в отчете о выбытии/отбраковке кодов маркировки	Не более 30000 КМ

1.7.1.4 Проектирование компонентов

На основе технического задания, установленного для выпускной квалификационной работы, программный компонент должен выполнять бесперебойную работу при высокой нагрузке и интенсивности запросов, обработки больших данных и выполнении тяжелых операции, и соответствовать современным многопользовательским асинхронным приложениям.

Проектирование компонентов осуществлено таким образом, чтобы программный компонент соответствовал следующим принципам:

- доступность;
- масштабируемость;
- наличие единого источника данных.

1.7.1.5 Доступность

Современные и информационные технологии позволяют создавать интеграционные решения, порождающие высокую интенсивность запросов, передачу больших объемов данных, создавая таким образом сильную нагрузку на работу системы в целом. Более отягощающим обстоятельством является факт количества таких интеграций, ведь с их увеличением требуется соизмеримые ресурсы для выполнения получаемых запросов.

Решением данной проблемы является наличие многопоточных масштабируемых на множество экземпляров сервисов, которые могут быть распределены по отдельным серверам, обеспечивая высокую доступность. Запросы, поступающие на конкретный сервис маршрутизируются и равномерно распределяются по существующим работающим экземплярам. В случае недоступности одного из сервисов, он может быть подменен на новый экземпляр.

Таким образом, программный компонент включает в себя совокупность экземпляров сервисов, работающих одновременно и обеспечивающих поддержание состояния, в котором как минимум один сервис является активным, работоспособным и производящим обработку поступающих запросов.

1.7.1.6 Масштабируемость

Масштабируемость решает задачу, аналогичную той, которая стоит перед доступностью. В отличие от предыдущего принципа, масштабируемость ориентирована на производительность системы. При наличии экземпляра сервиса, который находится в перегруженном состоянии, создаются новые экземпляры, позволяющие снизить возросшую интенсивность запросов для конкретного экземпляра сервиса. Таким образом производится балансировка нагрузки и поддержание быстродействия отклика запросов к сервисам.

Для программного компонента установлено определенное время ожидания запроса в зависимости от специфики процесса и выполняемых функций, поэтому данный принцип является подходящим решением для формирования границ времени отклика на запрос участника товарооборота.

1.7.1.7 Единый источник данных

Исходя из потребности в высоконагруженной параллельной системе, для поддержания быстродействия системы и распределении нагрузки нельзя забывать про такую неотъемлемую часть программного компонента, как база данных, которая тоже должна быть спроектирована с учетом интенсивности и количества запросов к ней.

Если для монолитных реализаций данные могут находиться в одной, как правило, реляционной, базе данных, то для микросервисных реализаций данное решение не подойдет, так как увеличение обращений к базе данных по операциям записи или чтения приводит к потере производительности, а так же не исключаются проблемы касающиеся изменения схемы базы данных.

При проектировании программного компонента данная проблема была решена за счет выделения индивидуальной базы данных для каждого микросервиса, что обеспечивает использование распределенного источника данных и поддерживает чистую архитектуру.

1.7.2 Разработка программного компонента

Неотъемлемой частью жизненного цикла каждого программного обеспечения является этап разработки, представляющий совокупность взаимосвязанных между собой процессов проектирования, программирования, документирования, отладки и тестирования, целью которых является создание и обслуживание программных продуктов.

1.7.2.1 Паттерны разработки

Разработка программного компонента эмиссии и учета кодов маркировки осуществляется на базе Spring Framework, предоставляющего целую экосистему, включающую в себя различные функциональные возможности. В рамках реализации автоматизированной системы управления одной из использованных подсистем данной библиотеки является Spring MVC, представляющий собой интегрированный в Spring веб-библиотеку и позволяющий создавать веб-сайты и RESTful-сервисы с использованием форматов данных JSON, XML.

Данная библиотека обеспечивает архитектуру паттерна MVC (Model-View-Controller), отраженную на Рисунке 13, при помощи слабо взаимосвязанных компонентов и разделяет такие аспекты приложения, как: предоставляемый интерфейс (логика ввода), бизнес-логику (алгоритмы) и логику отображения.

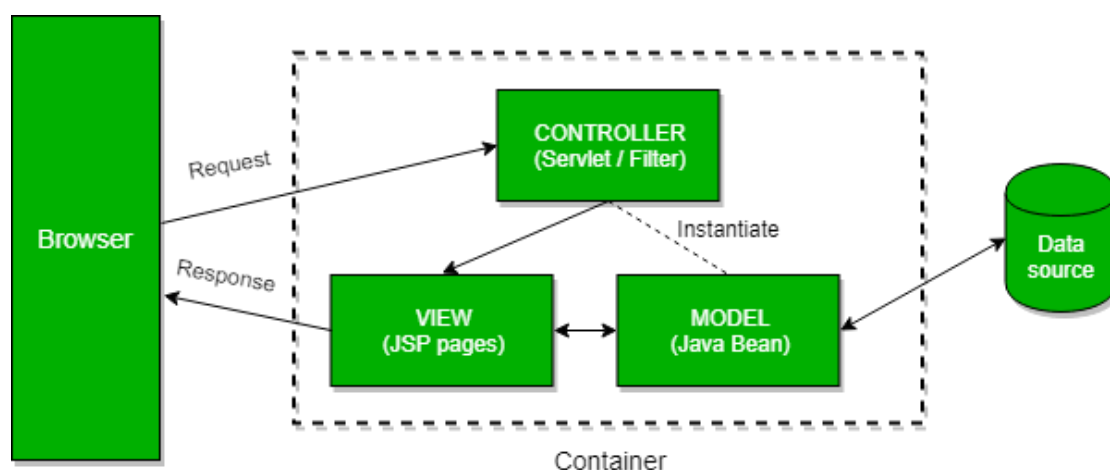


Рисунок 13 - Архитектура Spring MVC

В рамках данной выпускной квалификационной работы паттерн MVC использован в одной из своих многих реализаций, а именно на реализации функционального разделения приложения на 3 слоя:

- слой контроллеров – предоставляет пользователю конечные точки приложения (энд-поинты), описывает интерфейс взаимодействия с пользователем и передает получаемые на входе данные в слой процессинга;
- слой сервисов – представляет собой промежуточный слой, реализующий бизнес-логику конкретного процесса и взаимосвязан с контроллерами и репозиториями;
- слой репозитория (слой базы данных) – представляет собой конечный слой, отвечающий за хранение, выборку и обновление поступающих, запрашиваемых и обрабатываемых данных соответственно.

1.7.2.2 Уровень контроллеров

Контроллеры являются входными точками для работы пользователя с системой, предоставляя ее конечные точки (энд-поинты).

На уровне контроллеров осуществляется:

- регистрация относительного пути;
- определение типа запроса;
- определение формата, количества и структуры входных параметров;
- определение формата и структуры результата выполнения запроса;

- формируется описание метода;
- производится ряд базовых контрольных валидаций входящих параметров для обеспечения работы сервисов впоследствии с консистентными и корректными данными;

- формируется исходящий от сервера ответ.

В рамках разработки программного компонента эмиссии и учета кодов маркировки реализованы контроллеры, указанные в приложении.

В ходе разработки контроллеров автоматизированной системы управления были применены такие принципы объектно-ориентированного программирования, как: наследование, инкапсуляция, полиморфизм.

Так, в соответствии со спецификой разделения правил и состава атрибутов каждой отрасли производства под каждую товарную группу разработаны реализации контроллеров, наследующихся от базового и дополняя родительский функционал индивидуальными процессам.

1.7.2.3 Уровень сервисов

Сервисы выступают промежуточным звеном между входными данными и сохраняемыми данными, обеспечивая контроллеры обработкой, процессингом и сохранением данных.

Сервисы представляют собой логически сгруппированную совокупность функций, каждый из которых реализует внутри себя установленную бизнес-логику. В компоненте эмиссии и учета кодов маркировки реализованы такие сервисы как:

- сервис валидации заказов;
- сервис валидации отчетов;
- сервис работы с кодами маркировки;
- сервис работы с заказами;
- сервис работы с сервис работы с отчетами;
- сервис печати кодов маркировки;
- сервис работы с документами эмиссии и учета кодов маркировки;

- сервис работы с профилями участников товарооборота;
- сервис работы с биллинговыми системами;
- сервис работы с квити́рованием;
- сервис работы с этикетками;
- сервис работы с карточками товаров.

1.7.2.4 Уровень базы данных

Соединение с базой данных является одной из важнейших составляющих приложения. При реализации программных продуктов, работающих с хранилищами данных, выделяют модули, отвечающие за передачу запросов в хранилище и обработку полученных от него ответов.

В рамках разработки выпускной квалификационной работы доступ к слою базы данных реализован с использованием технологии Spring Data (для реляционной базы данных postgresQL) и CouchDb Client (для нереляционной базы данных couchDb).

Spring Data представляет удобный и простой в понимании механизм для взаимодействия с сущностями реляционной базы данных, организации их в репозитории, а также выполнении таких базовых CRUD-операций, как: сохранение данных, обновление данных, удаление данных, извлечение данных. Встроенные функциональные возможности фреймворка позволяют реализовать интерфейс, наделенный базовым функционалом (Рисунок 14):

```
public interface SystemParamRepository extends JpaRepository<SystemParam, String> {
}
```

Рисунок 14 - Пример пустого репозитория

В программном компоненте эмиссии и учета кодов маркировки реализованы для каждого микросервиса реализованы два основных репозитория: работа с функциональными данными и работа с данными событийной модели, то есть с эвентами. Так, можно выделить следующие репозитории системы:

- репозиторий заказов;

- репозиторий событий заказов;
- репозиторий кодов маркировки;
- репозиторий регистраторов участников;
- репозиторий пулов;
- репозиторий событий кодов маркировки;
- репозиторий отчетов;
- репозиторий событий отчетов;
- репозиторий квитанций.

CouchDb Client, для работы с базами данных, предоставляет прослойку между сервисами и самой базой данных в виде абстрактного объекта базы данных, включающий совокупность базовых методов и представлений для работы с нереляционной базой данных и документно-образным содержимым записей в ней. Клиент нереляционной базы данных оперирует записями в документнообразном формате Рисунок 14, представляющими собой пары ключ-значение, где первым является наименование поля объекта, а вторым – значение данного. Поиск и выборка данных по непроиндексированным полям осуществляет написанием представлений и построением индексов по данному представлению (view).

id	key	value
00101586-c420-42ff-971d-394d4a740e94	00101586-c420-42ff-971d-394d4a740e94	{ "rev": "1-a8423019e8a7f98d0d0ecd3..." }
122836ad-e096-4abc-8ffd-fbaae2f29db0	122836ad-e096-4abc-8ffd-fbaae2f29db0	{ "rev": "4-feff529b7712134defd7d8d5d..." }
150b6aec-a1b1-43ec-893e-a2cdcbc21...	150b6aec-a1b1-43ec-893e-a2cdcbc21...	{ "rev": "21-70a00b71db36d337fd71faa..." }
15eba548-9139-4466-82cd-cf91a5b6d...	15eba548-9139-4466-82cd-cf91a5b6d...	{ "rev": "9-8a486f37ec466eb911812601..." }
17ee597b-33ad-4551-a03a-9995870a4...	17ee597b-33ad-4551-a03a-9995870a4...	{ "rev": "2-aa1bba7ad3aba41c7682965..." }
341ca2da-bd32-499f-a295-699b956a9...	341ca2da-bd32-499f-a295-699b956a9...	{ "rev": "1-60e6d923a40a5e4c487d212..." }
7a890ac1-994e-42d3-a5f5-e5757f49b9c1	7a890ac1-994e-42d3-a5f5-e5757f49b9c1	{ "rev": "3-ef95c6f166de87c3f9e46b07a..." }
7b59d7b8-2b9c-4db4-877c-5a421a2fd...	7b59d7b8-2b9c-4db4-877c-5a421a2fd...	{ "rev": "2-13009857084f7e27b5579d2..." }
802d94f0-a238-430e-b412-2ec003d63...	802d94f0-a238-430e-b412-2ec003d63...	{ "rev": "9-264c2dcd54c85cc7503f9c68..." }
80b966ab-3b70-4880-b323-f3975adcd...	80b966ab-3b70-4880-b323-f3975adcd...	{ "rev": "7-12f12c6500709459da339bc9..." }
8abac633-9f59-42f7-bf68-592cb12a0c38	8abac633-9f59-42f7-bf68-592cb12a0c38	{ "rev": "5-683a873954ca6e8552122f0e..." }

Рисунок 15 - База данных couchDb

1.7.2.5 Обработка ошибок

Важным преимуществом программного компонента эмиссии и учета кодов маркировки является предоставление user-friendly пользовательского интерфейса, обеспечивающего не только простую и понятную функциональность процессов эмиссии, но и пользователе-ориентированную систему обработки и отображения ошибки.

Данная функциональная особенность реализована благодаря возможностям Spring Framework по перехвату и обработке базовых и определенных видов ошибок, возникающих во время исполнения программы.

Физически это реализуется за счет наследования функционала класса-обработчика ошибок `ResponseEntityExceptionHandler` библиотеки Spring [7] и реализации собственных обработчиков в зависимости от потребностей приложения. Так, для каждого обработчика указывается конкретный тип или типы исключений, которые должны быть перехвачены и обработаны в соответствии с описанием, предоставленном внутри функции-обработчика (Рисунок 16).

```
@ExceptionHandler(Exception.class)
public ResponseEntity<Object> handleServerException(Exception ex) {
    log.error("Unidentified error: {}", ex.getMessage());

    return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
        .header(HttpHeaders.CONTENT_TYPE,
            MediaType.APPLICATION_JSON_VALUE)
        .body(new GcpRestResult(new
            GcpApiGlobalError(HttpStatus.INTERNAL_SERVER_ERROR.value(),
                ex.getMessage())));
}
```

Рисунок 16 - Пример перехвата и обработки ошибок

Использование ресурсных переводов для каждого кода ошибки позволяет и учет локали запроса позволяют осуществить формирование ошибки в понятном развернутом виде на языке пользователя. Для этого программный компонент определяет собственный формат исключений, которые могут быть выброшены во время выполнения программы.

Процесс регистрации собственного типа исключения заключается в реализации класса, как правило, унаследованного от существующих в Java типов ошибок. Структура ошибки и вариант ее обработки представлен ниже:

Формат ответа с ошибкой на запрос отображает **Error! Reference source not found..**

Таблица 3 - Формат ответа с ошибкой

Поле	Описание	Тип
fieldErrors	Ошибка	List<FieldError>
globalErrors	Описание глобальных ошибок	List<GlobalError>
success	Результат выполнения запроса	Boolean

Описание формата объекта «FieldError» отображает **Error! Reference source not found.**

Таблица 4 - Формат объекта «FieldError»

Поле	Описание	Тип
fieldError	Описание ошибки	String
fieldName	Наименование поля	String
errorCode	Код ошибки	int

Описание формата объекта «GlobalError» отображает **Error! Reference source not found..**

Таблица 5 - Формат объекта « GlobalError»

Поле	Описание	Тип
error	Описание ошибки	String
errorCode	Код ошибки	String

Пример JSON ответа с ошибкой отображает Рисунок 17.

```
{
  "fieldErrors": [
    {
      "fieldError": "Не указано значение параметра «Номер производственного заказа»",
      "fieldName": "orderDto.productionOrderId",
      "errorCode": 6010
    }
  ],
  "globalErrors": [
    {
      "error": "Указано недопустимое значение параметра «Способ выпуска товара в оборот», значение должно соответствовать перечню допустимых",
      "errorCode": 6020
    }
  ],
  "success": false
}
```

Рисунок 17 - Пример ответа с ошибкой

Коды ошибок в ответе на запрос отображает **Error! Reference source not found..**

Таблица 6 - Коды ошибок отправки сведений

Код ошибки	Описание
400	Операция не выполнена. Неверные входные параметры
404	Операция не выполнен
500	Операция не выполнена. Внутренняя ошибка сервера

1.7.2.6 Особенности реализации

Контроллер API REST аутентифицирует клиентов посредством клиентского токена, который передается в заголовке HTTP-запроса. Клиентский токен представлен в виде маркера безопасности, передаваемого в заголовке HTTP клиентского токена - «clientToken».

В случаях взаимодействия с интерфейсом посредством POST-запросов необходимо указание дополнительного HTTP-заголовка – «Content-Type: application/json».

В качестве уникальной идентификации участников товарооборота, методы программного компонента эмиссии и учета кодов маркировки в качестве параметров используют уникальный идентификатор участника в системе «omsId», который доступен в настройках автоматизированной системы и представляет собой идентификатор формата UUID.

Символы, допустимые при использовании кодов маркировки в ходе работы с системой, отображает таблица 7.

Данные символы используются в следующих группах данных кодов маркировки:

- серийный номер;
- идентификатор ключа;
- код проверки.

Таблица 7 - Допустимые символы КМ

Допустимые символы КМ
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!"#\$%&'()*+,-./_`{;}=<>?

1.8 Описание алгоритма

1.8.1 Последовательность процессов

В данном разделе представлен алгоритм процессов эмиссии и учета кодов маркировки на основе последовательности вызова методов, предоставляемых программным интерфейсом компонента, при создании нового заказа на эмиссию кодов маркировки:

- проверка доступности программного компонента;

- создание заказа на эмиссию КМ в разрезе товарной группы с учетом специфики данных;
- получение статуса массива КМ из заказа и ожидание наполнения буферов заказа;
- получение КМ из заказа по заданным параметрам;
- отправка отчёта нанесения КМ;
- отправка отчёта об агрегации КМ;
- отправка отчета о выбытии/отбраковке КМ.

Рисунок 18 представляется диаграмму, которая отражает последовательность вызова методов программного интерфейса автоматизированной системы управления.

Программный интерфейс компонента эмиссии и учета кодов маркировки также предоставляет вспомогательные методы, которые доступны в API и также являются используемыми в процессах эмиссии кодов маркировки:

- получение статуса обработки отчета;
- отправка лог файлы АСУТП;
- получение статуса подзаказов и заказов;
- получение квитанций по отправленным заказам и отчетам;
- получение атрибутов товаров из созданного заказа на эмиссию КМ;
- получение маркера безопасности по имени пользователя и паролю;
- закрытие заказа или подзаказа по заданному GTIN;
- получение повторно коды маркировки из заказа кодов маркировки;
- отправка отчет об использовании и об автоматическом вводе в оборот;
- получение списка идентификаторов пакетов или блоков кодов маркировки;
- получение информации о номере документа отчета по вводу в оборот;
- получение информацию об агрегации КМ;
- получение версию СУЗ и API.

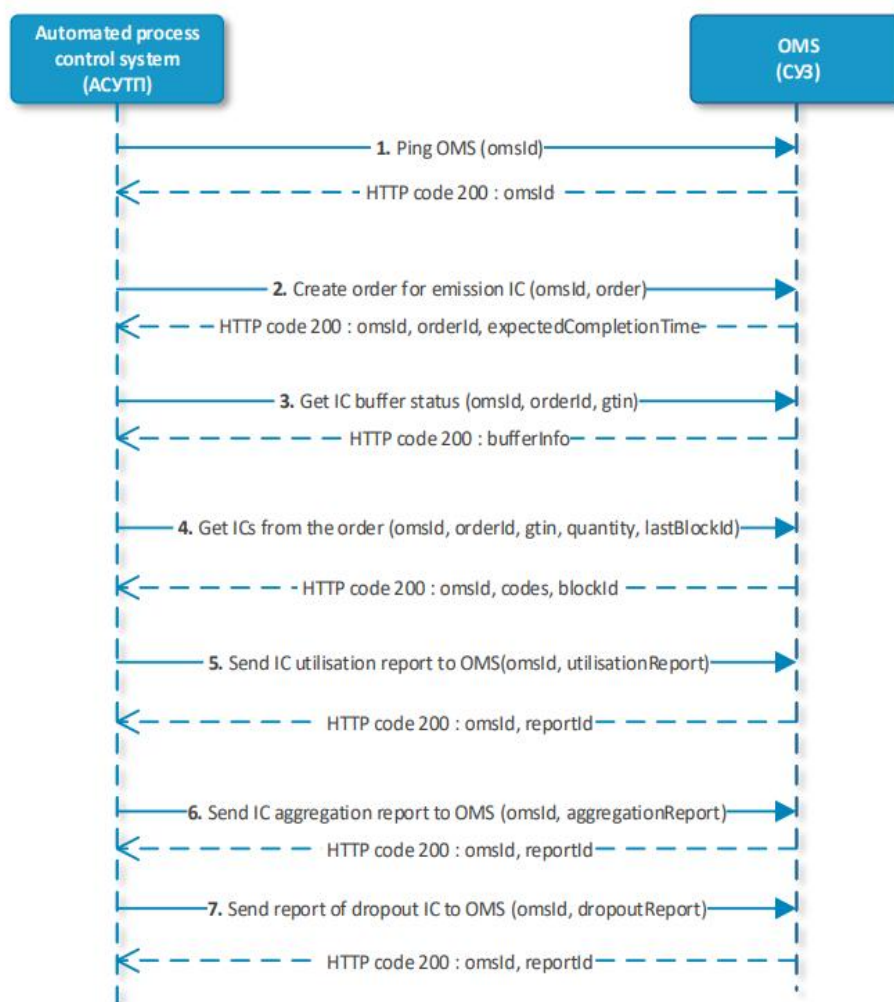


Рисунок 18 - Последовательность вызова методов СУЗ

1.8.2 Ключевые этапы процессов

Общий процесс эмиссии кодов маркировки включает в себя четыре ключевых этапа:

- создание бизнес-заказа на эмиссию КМ;
- получение статуса массива КМ из бизнес-заказа для определения наполнения буферов и готовности к использованию;
- получение КМ из бизнес-заказа, печать или запрос на формирование CSV- или EPS-документа;
- отправка отчета об использовании КМ (также возможно включение пятого и шестого этапа - отправки отчетов об агрегации КМ и отчетов об отбраковке или выбытии соответственно).

1.8.3 Создание заказа на эмиссию кодов маркировки

Одной из основных, наиболее используемых и важной функций программного компонента эмиссии и учета кодов маркировки является создание заказа.

На Рисунок 19 представлена диаграмма процесса создания заказа на эмиссию КМ.

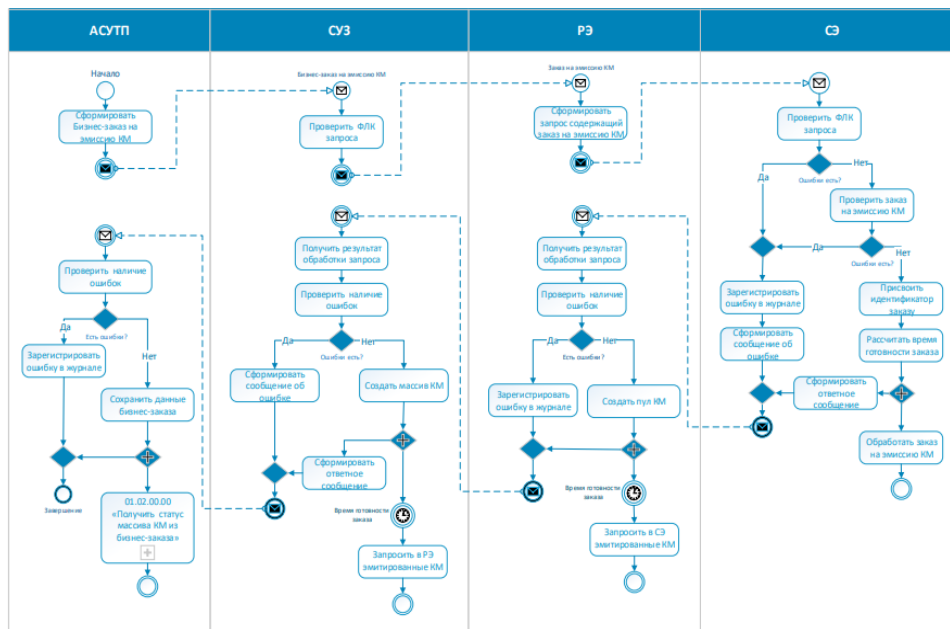


Рисунок 19 - Создание заказа на эмиссию КМ

Далее представлена последовательность выполнения функции и организация процесса, в соответствии с представленной диаграммой последовательности:

- участник товарооборота или интеграционные решения, представленные в виде автоматизированной системы управления товаропроизводством формирует заказ и отправляет его в программный компонент эмиссии и учета кодов маркировки;
- автоматизированная система программного компонента производит синхронную и асинхронную проверку запроса, после чего, в случае успешной проверки, отправляет заказ в Регистратор эмиссии;

- регистратор эмиссии формирует запрос, содержащий заказ на эмиссию КМ и отправляет его в Сервер эмиссии;

- сервер эмиссии при получении запроса, который содержит заказ на эмиссию КМ, на своей стороне производит проверку корректности запроса:

1) в случае если проверка запроса не пройдена, Сервер эмиссии регистрирует ошибку в журнале, формирует сообщение об ошибке и отправляет в Регистратор эмиссии, а затем осуществляется переход на шаг получения результата обработки заказа;

- сервер эмиссии при отсутствии ошибок в заказе на эмиссию КМ, генерирует идентификатор заказа и рассчитывает время готовности заказа:

1) сервер эмиссии, отправляет заказ на обработку (действие выполняется асинхронно);

- сервер эмиссии формирует исходящее сообщение, которое отправляется в регистратор эмиссии;

- регистратор эмиссии получает результат обработки запроса;

- регистратор эмиссии проверяет наличие ошибок:

1) в случае наличия ошибок, Регистратор эмиссии регистрирует ошибку в журнале учета;

2) на основе полученных данных, регистратор эмиссии определяет исходящее сообщение об ошибке и отправляет его в программный компонент эмиссии и учет кодов маркировки;

- регистратор эмиссии при отсутствии ошибок формирует пустой пул КМ:

1) регистратор эмиссии ожидает время готовности заказа, установленное Сервером эмиссии и запрашивает эмитированные КМ;

- регистратор эмиссии отправляет ответное сообщение в программный компонент эмиссии и учета кодов маркировки;

- программный компонент ожидает получения результата создания заказа от Регистратора эмиссии;

- осуществляется проверка наличия ошибок:

1) в случае если произошла ошибка, происходит регистрация записи в системном журнале;

2) система формирует сообщение об ошибке и отправляет в участнику товарооборота или АСУТП;

- при отсутствии ошибок осуществляется формирование массива КМ:

1) устанавливается ожидание времени готовности заказа и периодически запрашиваются эмитированные КМ в Регистраторе эмиссии;

- система формирует ответное сообщение и отправляет его УОТ или АСУТП;

- УОТ или АСУТП получают результат обработки запроса на создание заказа;

- при отсутствии ошибок осуществляется сохранение данных заказа и инициируется выполнение процесса по получению статуса массива кодов маркировки из заказа.

1) АСУТП инициирует выполнение процесса «Получение статус массива КМ из заказа».

1.8.4 Получение статуса массива кодов маркировки

На Рисунок 20 представлена диаграмма процесса получения статуса массива КМ.

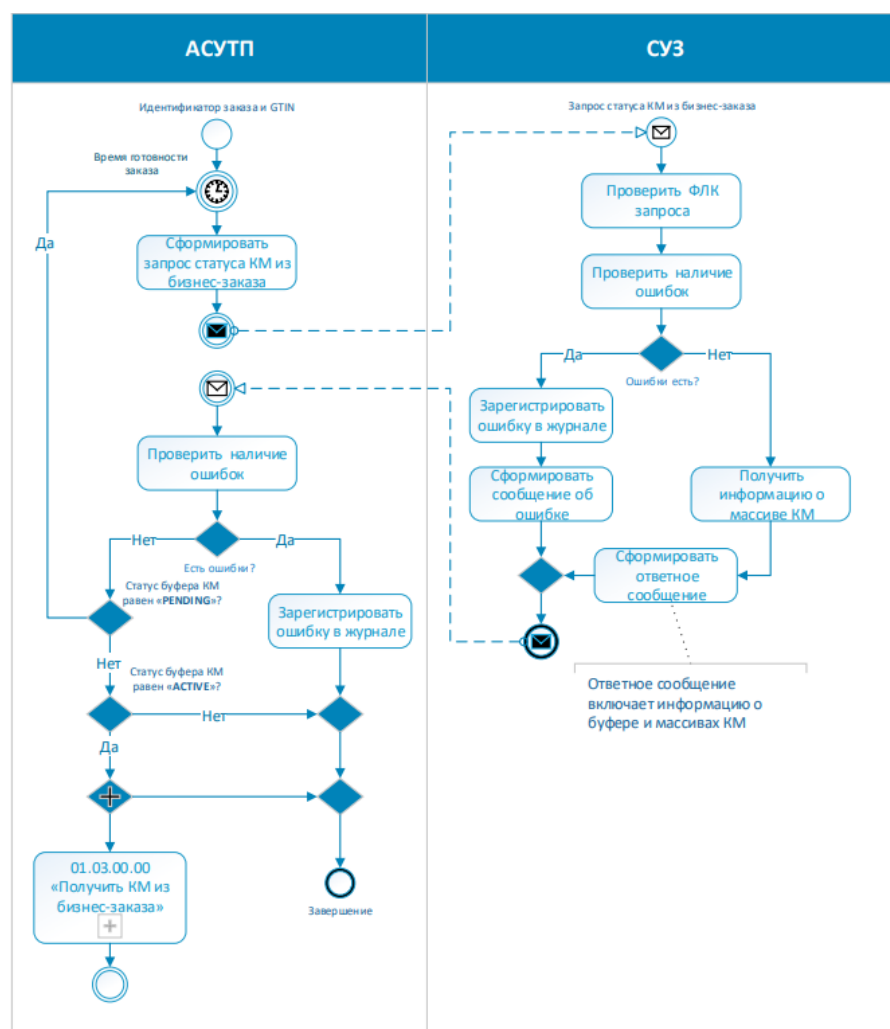


Рисунок 20 - Получение статуса массива КМ

Далее представлена последовательность выполнения функции и организация процесса, в соответствии с представленной диаграммой последовательности:

- УОТ или автоматизированная система управления товаропроизводством ожидают получение данных времени готовности заказа;
- УОТ или формируют запрос получения статуса массива КМ и отправляет его в программный компонент эмиссии и учета кодов маркировки;
- программный компонент осуществляет верификацию запроса:
 - 1) в случае если верификация не пройдена, система регистрирует ошибку в системном журнале;

2) затем формируется сообщение об ошибке и осуществляется отправка данных УОТ или АСУТП;

- программный компонент получает информацию о массиве КМ;
- программный компонент определяет формат ответного сообщения и отправляет данные УОТ или АСУТП;
- УОТ или АСУТП получают ответное от программного компонента сообщение;
- УОТ или АСУТП при отсутствии ошибок ответа, осуществляет проверку статуса буфера КМ на соответствие значению «PENDING»:

1) в случае если статус буфера соответствует значению «PENDING», АСУТП инициирует повторный запрос статуса массива КМ;

- при отсутствии ошибок, осуществляется проверка статус буфера КМ на соответствие значению «ACTIVE»:

1) в случае если статус буфера КМ не соответствует значению «ACTIVE» процесс завершается;

- в случае если статус буфера КМ соответствует значению «ACTIVE», УОТ или АСУТП инициируют осуществление запроса по получению КМ из заказа;
- процесс завершается.

1.8.5 Получение кодов маркировки

На Рисунок 21 представлена диаграмма процесса получения КМ из бизнес-заказа.

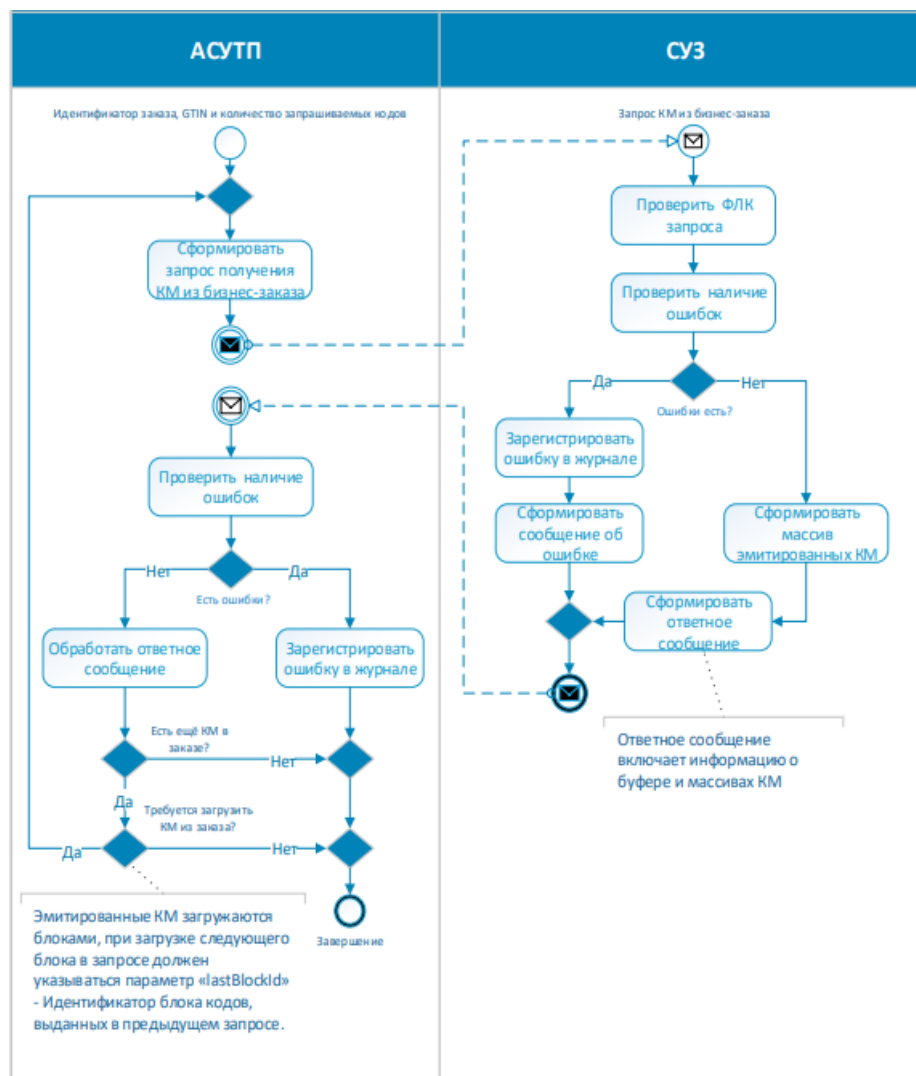


Рисунок 21 - Получение КМ из бизнес-заказа

Далее представлена последовательность выполнения функции и организация процесса, в соответствии с представленной диаграммой последовательности:

- УОТ или АСУТП генерируют запрос получения КМ из заказа и отправляет его в программный компонент эмиссии и учета кодов маркировки;
- программный компонент осуществляет верификацию запроса:
 - 1) при наличии ошибки, производится ее регистрация в системном журнале;
 - 2) затем формируется сообщение об ошибке и осуществляется отправка УОТ или АСУТП;

- система осуществляет формирование массива эмитированных КМ;
- система осуществляет формирование ответного сообщения и отправляет его УОТ или АСУТП;

- УОТ или АСУТП, при получении ответного сообщения, осуществляют проверку наличия ошибок:

- 1) при наличии ошибки, АСУТП осуществляет ее регистрацию в системном журнале;

- 2) процесс завершается;

- УОТ или АСУТП обрабатывают исходящее сообщение и осуществляют проверку наличия КМ в заказе:

- 1) в случае если КМ в заказе отсутствуют, процесс завершается;

- при наличии КМ в заказе, УОТ или АСУТП осуществляют проверку необходимости загрузки оставшихся КМ:

- 1) при необходимости загрузки оставшихся КМ в заказе, УОТ или АСУТП инициируют процесс повторное выполнение функции получение КМ из заказа;

- в случае если не требуется загрузка оставшихся КМ в заказе, то процесс завершается.

1.8.6 Отправка отчета о нанесении кодов маркировки

Далее представлена последовательность выполнения функции и организация процесса, в соответствии с диаграммой последовательности, указанной на Рисунок 22 и описывающей процесс отправки отчета об использовании КМ:

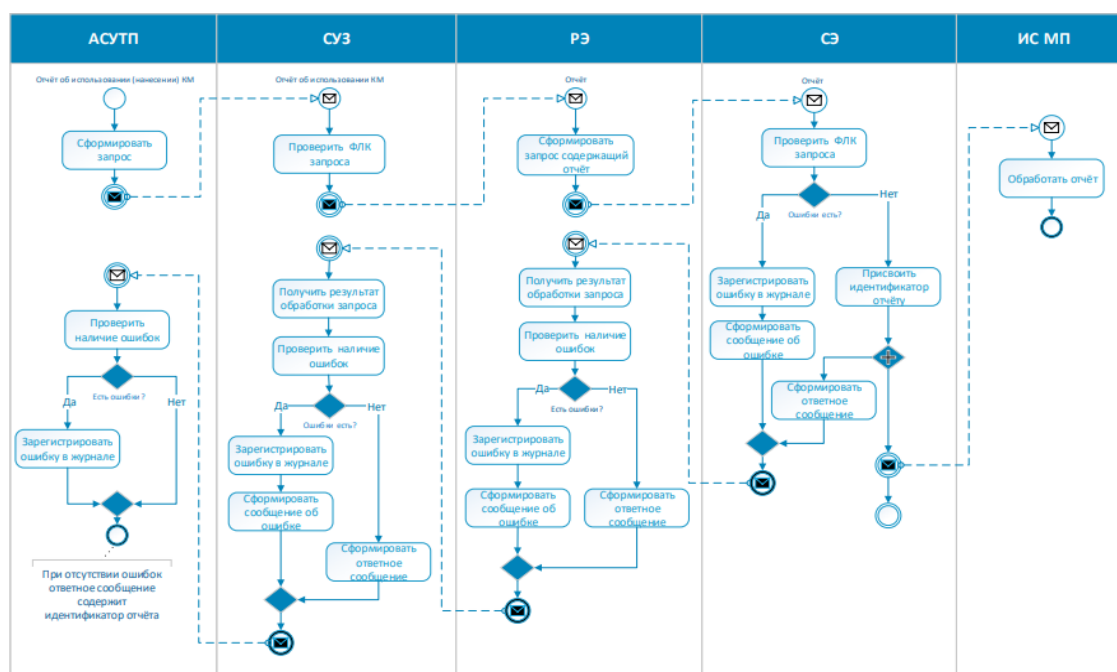


Рисунок 22 - Отправка отчета об использовании КМ

- УОТ или АСУТП генерируют запрос на отправку отчета об использовании КМ и направляет его в программный компонент эмиссии и учета кодов маркировки;

- программный компонент осуществляет верификацию запроса и отправляет запрос с отчётом о нанесении КМ в Регистратор эмиссии;

- регистратор эмиссии формирует запрос содержащий отчёт о нанесении КМ и отправляет его в Сервер эмиссии;

- сервер эмиссии, получив запрос с отчётом о нанесении КМ, осуществляет верификацию запроса:

1) при наличии ошибки, Сервер эмиссии осуществляет ее регистрацию в журнале;

2) сервер эмиссии формирует сообщение об ошибке и осуществляет отправку данных в Регистратор эмиссии;

- при отсутствии ошибок Сервер эмиссии, формирует идентификатор отчета:

1) Сервер эмиссии, осуществляет отправку отчёта в обработку в ИС МП;

- Сервер эмиссии формирует ответное сообщение и отправляет в регистратор эмиссии;

- Регистратор эмиссии получает результат обработки запроса и проверяет наличие ошибок:

1) при наличии ошибки, Регистратор эмиссии осуществляется запись в журнале;

2) регистратор эмиссии формирует сообщение об ошибке и осуществляет его отправку в программный компонент эмиссии и учета кодов маркировки;

- при отсутствии ошибок Регистратор эмиссии формирует и отправляет ответное сообщение в программный компонент эмиссии и учета кодов маркировки;

- программный компонент при получении результат обработки запроса проверяет наличие ошибок:

1) при наличии ошибок, осуществляется ее регистрация в журнале;

2) система осуществляет формирование сообщения об ошибке и его отправку УОТ или АСУТП;

- при отсутствии ошибок программный компонент формирует ответное сообщение и отправляет его УОТ или АСУТП;

- АСУТП или УОТ получают результат ответа от программного компонента и осуществляют проверку наличия ошибок:

1) при наличии ошибок, осуществляется ее регистрация в журнале;

- УОТ или АСУТП при отсутствии ошибок осуществляют сохранение идентификатора отчета, процесс завершается.

1.9 Описание программы

Программный интерфейс компонента эмиссии и учета кодов маркировки предоставляют расширения в разрезе товарных групп и предоставляет конечные точки для:

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

- фармацевтической промышленности;
- табачной промышленности;
- легкой промышленности.

Определена следующая структура конечный точек программы
<base_url>/api/v2/{productGroup}:

- базовый URL стенда;
- расширение в разрезе товарной группы.

Список допустимых товарных групп представлен в таблице 7.

Таблица 7 - Товарные группы системы

Товарная группа	Описание
lp	Предметы одежды, белье постельное, столовое, туалетное и кухонное
shoes	Обувные товары
pharma	Фармацевтическая промышленность
tobacco	Табачная промышленность
tires	Шины
photo	Фототовары
perfum	Парфюмерная продукция
milk	Молочная продукция
bicycle	Велосипеды и велосипедные рамы
wheelchairs	Медицинские изделия
otp	Альтернативная табачная продукция
water	Упакованная вода
beer	Пиво, напитки, изготавливаемых на основе пива и слабоалкогольных напитков
ncp	Никотиносодержащая продукция
bio	Биологически активные добавки
antiseptic	Антисептики и дезинфицирующие средства
nabeer	Безалкогольное пиво
softdrinks	Соковая продукция и безалкогольные напитки

1.9.1 Метод создания заказа на эмиссию кодов маркировки

Этот метод используется для создания и отправки заказа на эмиссию КМ.
Описание параметров представлено в таблице 8.

Таблица 8 - Данные запроса

Параметр	Значение
Тип запроса	POST
Конечная точка	/api/v2/{productGroup}/orders?omsId={omsId}
Заголовок запроса	clientToken (маркер безопасности)
Параметр строки запроса	omsId (идентификатор участника)
Тело запроса	Объект OrderDto

Описание структуры тела запроса, объект OrderDto представлено в таблице 9.

Таблица 9 - Структура объекта OrderDto

Параметр	Описание
List<OrderProduct> products	Список товаров
String serviceProviderId	Идентификатор сервис-провайдера

Описание вложенных параметров, а именно объекта OrderProduct представлено в таблице 10.

Таблица 10 - Структура объекта OrderProduct

Параметр	Описание
String gtin	Список товаров
int quantity	Идентификатор сервис-провайдера
String serialNumberType	Способ генерации серийных номеров
List<String> serialNumbers	Серийные номера
int templateId	Идентификатор шаблона КМ

Для каждой товарной группы, приведенные выше объекты, имеют расширения, приведенные в приложении.

Пример запроса на создание заказа изображен на Рисунк 23.

```

POST http://localhost:8080/api/v2/tobacco/orders?omsId=a2a16a41-42b0-4309-
9ae1-c19d53cc544f
accept: application/json
clientToken: 1cecc8fb-fb47-4c8a-af3d-d34c1ead8c4f
Content-Type: application/json

{
  "expectedStartDate": "2019-03-01",
  "factoryAddress": "ул. Пervaya Tabachnaya, 7",
  "factoryCountry": "RU",
  "factoryId": "ftf_001",
  "factoryName": "First Tobacco Factory",
  "poNumber": "123456",
  "productCode": "ABCD123456",
  "productDescription": "Cigarettes Friend",
  "productionLineId": "123",
  "products": [
    {
      "gtin": "04805555555555",
      "quantity": 11,
      "serialNumberType": "OPERATOR",
      "templateId": 4
    }
  ],
  "releaseMethodType": "IMPORT"
}

```

Рисунок 23- Структура запроса

Структура ответа представлена в таблице 11

Таблица 11 - Структура ответа на запрос

Параметр	Описание
String omsId	Идентификатор участника
String orderId	Идентификатор заказа
int expectedCompleteTimestamp	Планируемое время выполнения заказа

Пример ответа представлен на Рисунок 24.

```

HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Content-Length: 111
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId": "a2a16a41-42b0-4309-9ae1-c19d53cc544f",
  "orderId": "ec535302-8f2e-4b15-9b4f-dd9e7857a2d5",
  "expectedCompleteTimestamp": 120000
}

```

Рисунок 24 - Структура ответа на запрос

1.9.2 Метод создания отчета о нанесении КМ

Этот метод используется для создания и отправки отчета о нанесении КМ. Описание параметров представлено в таблице 12.

Таблица 12 - Данные запроса

Параметр	Значение
Тип запроса	POST
Конечная точка	/api/v2/{productGroup}/utilisation?omsId={omsId}
Заголовок запроса	clientToken (маркер безопасности)
Параметр строки запроса	omsId (идентификатор участника)
Тело запроса	Объект UtilisationReportDto

Описание структуры тела запроса, объект UtilisationReportDto представлено в таблице 13.

Таблица 13 - Структура объекта UtilisationReportDto

Параметр	Описание
List<String> sntins	Список кодов маркировки
String usageType	Тип использования

Для каждой товарной группы, приведенный выше объект, имеет расширения, приведенные в приложении.

Пример запроса на создание отчета о нанесении КМ изображен на Рисунок 25.

```
POST https://localhost:8080/api/v2/tobacco/utilisation?omsId=123baaa7-ff9d-4a42-b814-34527b7231cb
clientToken: 3132846f-20ac-6ba1-c273-9b46579bd786
Content-Type: application/json
Accept: application/json

{
  "productionLineId": "3",
  "sntins": [
    "010491361293960821EB2x3Tq\u001D93MUP0",
    "0104913612939608212qmcB8u\u001D93KE1e"
  ],
  "sourceReportId": "7a9eb56a-6d44-41b5-8a97-fc91bf9f89fa",
  "usageType": "USED_FOR_PRODUCTION"
}
```

Рисунок 25 - Структура запроса

Структура ответа представлена в таблице 14

Таблица 14 - Структура ответа на запрос

Параметр	Описание
String omsId	Идентификатор участника
String reportId	Идентификатор отчета

Пример ответа представлен на Рисунок 26.

```
HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "123baaa7-ff9d-4a42-b814-34527b7231cb",
  "reportId" : "3179f5d2-2bf5-47d1-8df0-9452b257d851"
}
```

Рисунок 26 - Структура ответа на запрос

1.9.3 Метод создания отчета об отбраковке КМ

Этот метод используется для создания и отправки отчета об отбраковке КМ. Описание параметров представлено в таблице 15.

Таблица 15 - Данные запроса

Параметр	Значение
Тип запроса	POST
Конечная точка	/api/v2/{productGroup}/dropout?omsId={omsId}
Заголовок запроса	clientToken (маркер безопасности)
Параметр строки запроса	omsId (идентификатор участника)
Тело запроса	Объект DropoutReportDto

Описание структуры тела запроса, объект DropoutReportDto представлено в таблице 16.

Таблица 16 - Структура объекта DropoutReportDto

Параметр	Описание
List<String> sntins	Список кодов маркировки
String dropoutReason	Причина выбытия

Для каждой товарной группы, приведенный выше объект, имеет расширения, приведенные в приложении.

Пример запроса на создание отчета об отбраковке КМ изображен на Рисунок 27.

```

POST http://localhost:8080/api/v2/tobacco/dropout?omsId=15ecde9e-a686-4934-a337-ae319c944242
clientToken: 15544d88-dea7-4274-b7ac-0f1f2cc12c11
Content-Type: application/json
Accept: application/json

{
  "dropoutReason": "DEFECT",
  "sntins": [
    "010460026600186321nnnn100"
  ],
  "address": "440052 Vsevolod T01",
  "withChild": false,
  "participantId": "7736673512"
}

```

Рисунок 27 - Структура запроса

Структура ответа представлена в таблице 17.

Таблица 17 - Структура ответа на запрос

Параметр	Описание
String omsId	Идентификатор участника
String reportId	Идентификатор отчета

Пример ответа представлен на Рисунок 28.

```

HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json; charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "15ecde9e-a686-4934-a337-ae319c944242",
  "reportId" : "3179f5d2-2bf5-47d1-8df0-9452b257d851"
}

```

Рисунок 28 - Структура ответа на запрос

1.9.4 Метод создания отчета об агрегации КМ

Этот метод используется для создания и отправки отчета об агрегации КМ.

Описание параметров представлено в таблице 18.

Таблица 18 - Данные запроса

Параметр	Значение
Тип запроса	POST
Конечная точка	/api/v2/{productGroup}/aggregation?omsId={omsId}
Заголовок запроса	clientToken (маркер безопасности)
Параметр строки запроса	omsId (идентификатор участника)
Параметр строки запроса	userName (имя пользователя)
Тело запроса	Объект AggregationReportDto

Описание структуры тела запроса, объект AggregationReportDto представлено в таблице 19.

Таблица 19 - Структура объекта AggregationReportDto

Параметр	Описание
List<AggregationUnit> aggregatinUnits	Массив единицы агрегации
String participantId	Идентификационный номер налогоплательщика

Описание вложенных параметров, а именно объекта AggregationUnit представлено в таблице 20.

Таблица 20 - Структура объекта AggregationReportDto

Параметр	Описание
int aggregationItemsCount	Фактическое количество штук в единице агрегации
String aggregationType	Тип агрегации
int aggregationUnitCapacity	Емкость упаковки
List<String> sntins	Массив агрегированных КМ
String unitSerialNumber	Идентификационный код единицы агрегации

Для каждой товарной группы, приведенный выше объект, имеет расширения, приведенные в приложении.

Пример запроса на создание отчета об отбраковке КМ изображен на Рисунок 29.

```
POST http://localhost:8080/api/v2/tobacco/aggregation?omsId=beb021a4-bfed-498f-bf3d-84e8b8e4502f
clientToken: 994ea10d-1eda-4fed-8d1c-dee6a367dcaa
userName: API
Content-Type: application/json

{
  "aggregationUnits": [
    {
      "aggregatedItemsCount": "10",
      "aggregationType": "AGGREGATION",
      "aggregationUnitCapacity": "10",
      "sntins": [
        "04710138110189U:WZw&N",
        "04710138110189zR!uJqv"
      ],
      "unitSerialNumber": "010118018511356621)hMPrY5"
    }
  ],
  "participantId": "306535627",
  "productionLineId": "3"
}
```

Рисунок 29 - Структура запроса

Структура ответа представлена в таблице 21.

Таблица 21 - Структура ответа на запрос

Параметр	Описание
String omsId	Идентификатор участника
String reportId	Идентификатор отчета

Пример ответа представлен на Рисунок 30.

```

HTTP/1.1 200 OK
Content-Length: 74
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Options: nosniff
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "omsId" : "beb021a4-bfed-498f-bf3d-84e8b8e4502f",
  "reportId" : "3179f5d2-2bf5-47d1-8df0-9452b257d851"
}

```

Рисунок 30 - Структура ответа на запрос

1.10 Условия применения программы

1.10.1 Используемые технические средства

1.10.1.1 Характеристики программного компонента

Для эксплуатации программного компонента эмиссии и учета кодов маркировки, должны быть использованы следующие средства вычислительной техники:

- процессор CPU:

- 1) архитектура Intel x86;
- 2) тактовая частота 3 GHz;

- платформа: 64-х разрядная;

- объем оперативной памяти: 16 Gb;

- жесткий диск: 100GB;

- сетевой интерфейс: Ethernet 100 Mbit/s.

1.10.1.2 Характеристики интерфейса программного компонента

Для эксплуатации веб-интерфейса программного компонента эмиссии и учета кодов маркировки, должны быть использованы следующие средства вычислительной техники:

- средства вычислительной техники индивидуального пользования:

- 1) АРМ пользователя с одной или несколькими установленными функциональными ролями: «Оператор», «Администратор», «Руководитель организации», «Пользователь организации»,

«Пользователь организации с возможностью подписи», представляющее собой ПЭВМ с характеристиками, соответствующими рекомендуемыми требованиям операционной системы Microsoft Windows версии 8.1 и выше.

1.10.2 Необходимое программное обеспечение

1.10.2.1 Веб-интерфейс программного компонента

Для эксплуатации веб-интерфейса программного компонента эмиссии и учета кодов маркировки необходимы следующие компоненты общего программного обеспечения:

- операционная система Microsoft Windows версии 8.1 и выше;
- веб-браузер:

- 1) Mozilla Firefox версии 40 и выше;
- 2) Internet Explorer версии 9 и выше;
- 3) Google Chrome версии 37 и выше.

1.10.3 Информационное обеспечение

1.10.3.1 Требования по обработке JSON формата

Так как код маркировки содержит специальные символы, то Участник оборота, интегрируя свои решения, должен корректно обрабатывать формат JSON средствами, которые соответствуют RFC 8259, а не обрабатывать его как «Plain Text». Применение в своих решениях средств, которые соответствуют RFC 8259 обеспечивает корректную передачу и получение кодов маркировки, содержащих специальные символы (специальные символы экранируются).

Аналогично если в своих решениях участники оборота используют формат XML, то специальные символы, содержащиеся в строке, должны конвертироваться в соответствии со спецификациями XML.

1.11 Отладка и тестирование

1.11.1 Отладка системы

Процесс отладки является составной частью процесса разработки программного обеспечения и осуществляет пошаговую отладку разработанного функционала, локализацию обнаруженных в ходе процесса ошибок и их устранение.

Анализ текущего состояния программы осуществлялся за счет расположения в критических точках программы операторов останова (Рисунок 31) и современных средств интеграционной среды разработки, а именно: консоль отладки, переменные выполнения, выполнение функции на лету и анализ прохождения точек останова (Рисунок 31 и Рисунок 32).

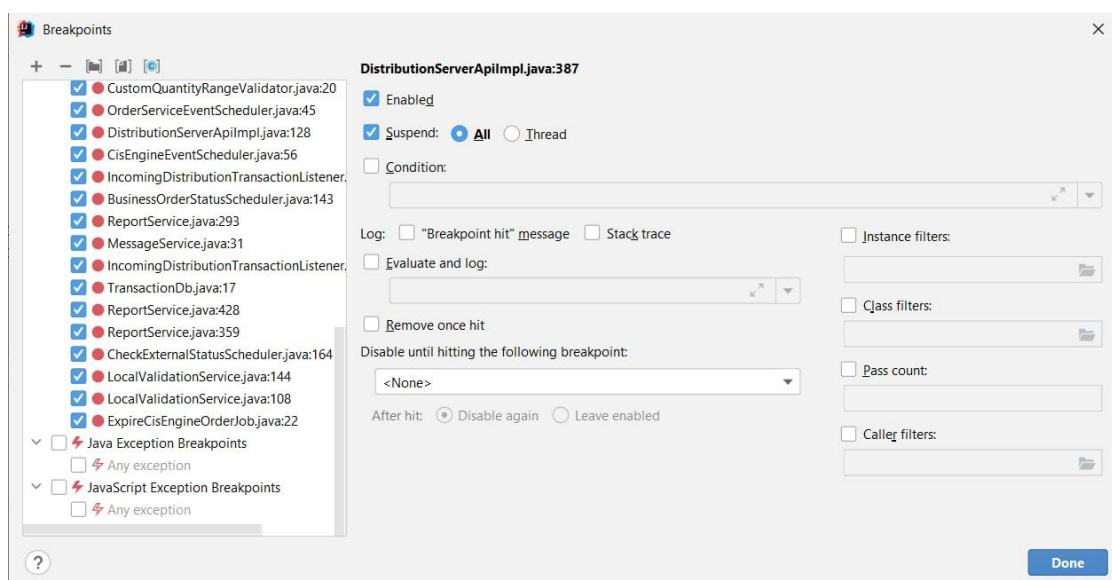


Рисунок 31 - Точки останова

В ходе отладки были выявлены и устранены следующие дефекты:

- ошибка при проверке подписи участника товара оборота;
- ошибка наименования параметра с использованием буквы «С» русского алфавита;
- ошибка десериализации объекта нереляционной базы данных;

- ошибка выполнения нативного HQL-запроса репозитория Spring Data JPA;
- ошибка написания шаблона развертывания;
- ошибка установки параметров окружения;
- ошибка при передаче тегов Регистратор эмиссии;
- ошибка обрезки КМ при формировании отчета через веб-интерфейс;
- ошибка построения представления нереляционной базы данных.

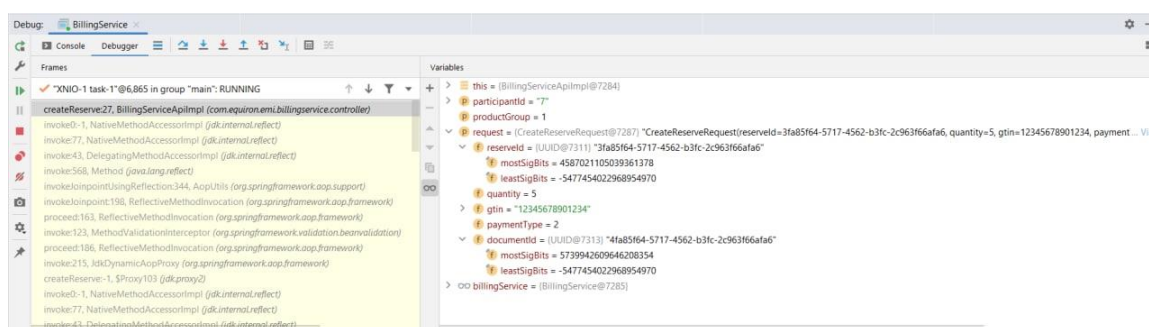


Рисунок 32 - Консоль отладки

1.11.2 Тестирование системы

Под тестированием программного обеспечения понимаются процессы испытания и исследования реализованного функционала в рамках программного продукта. Целью данного процесса является проверка ожидаемой и описанной в документации реализации и поведения с ее фактическим или реальным состоянием выполнения процессов.

Важной частью задачи тестирования является декомпозиция объема задач, подлежащих проверке, а также разбиение процесса тестирования в соответствии с его различными видами. В рамках тестирования функционала программного компонента эмиссии и учета кодов маркировки, были проведены следующие виды тестирования:

- функциональное тестирование;
- ручное тестирование;
- интеграционное тестирование.

Рассмотрим подробнее каждый из описанных видов тестирования и его применение в рамках данной выпускной квалификационной работы.

1.11.2.1 Функциональное тестирование

Как правило, функциональное тестирование основное внимание уделяет бизнес-требованиям реализуемой системы, осуществляет проверку конкретный действий без учета верификации промежуточного состояния системы.

В рамках данной работы вид функционального тестирования представлен в разрезе unit-тестов, написанных на языке программирования Java и находящихся непосредственно в кодовой базе проекта в директории /test.

Выполнение функционального теста осуществляется посредством выполнения в интеграционной среде разработки (Рисунок 33) или за счет встроенных средств системы сборки проектов.

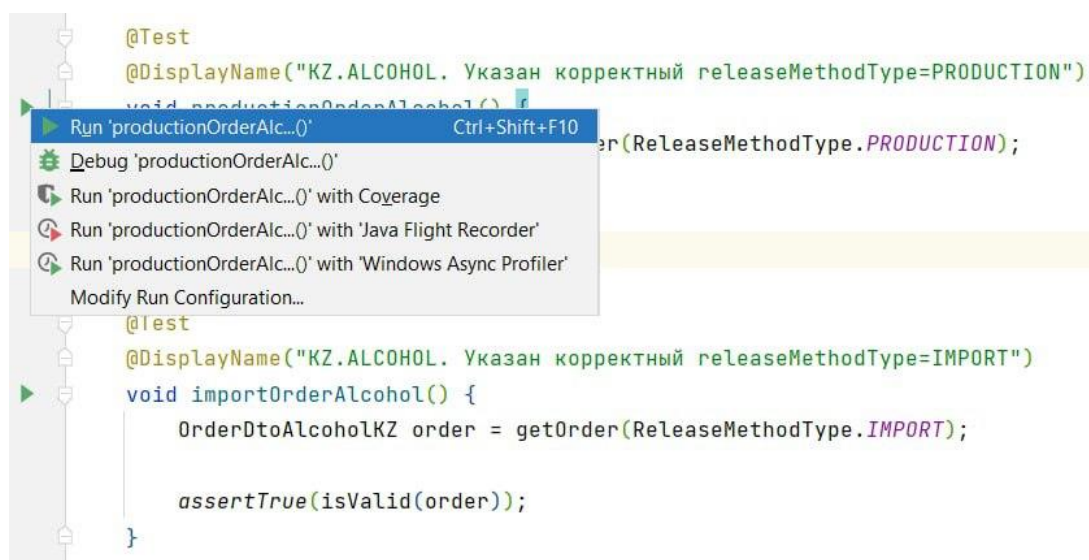


Рисунок 33 - Пример функционального теста

1.11.2.2 Ручное тестирование

Ручное тестирование системы является наиболее простым типом тестирования, в рамках которого тест-кейсы выполняются вручную, без использования средств автоматизации, и определяются дефекты приложения. Наиболее важным является факт невозможности полной автоматизации

тестирования программного продукта, поэтому ручное тестирование присуще каждому программному обеспечению.

В рамках тестирования специализированного программного обеспечения выпускной квалификационной работы были созданы тестовые кейсы и тестовые коллекции в инструменте тестирования Postman (Рисунок 34) и в встроенных в интеграционную среду разработки средств осуществления базовых запросов HTTP-requests (Рисунок 35 и Рисунок 36).

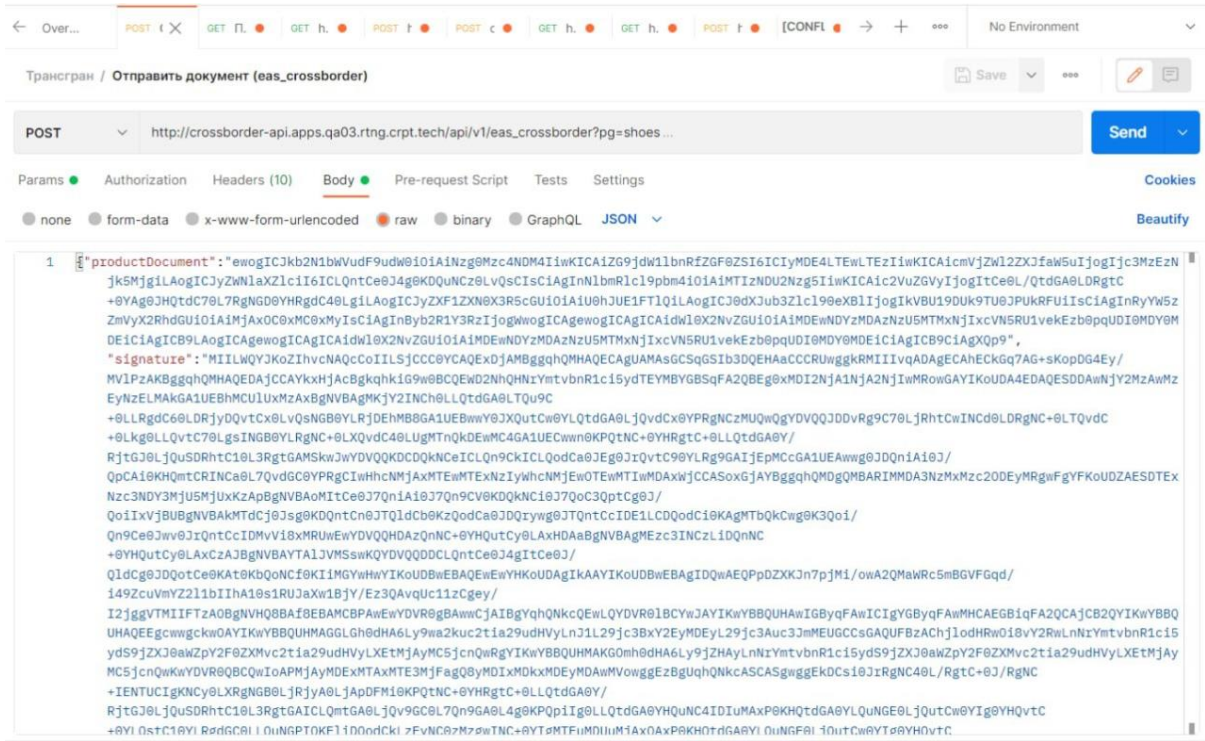


Рисунок 34 - Ручное тестирование Postman-коллекциями

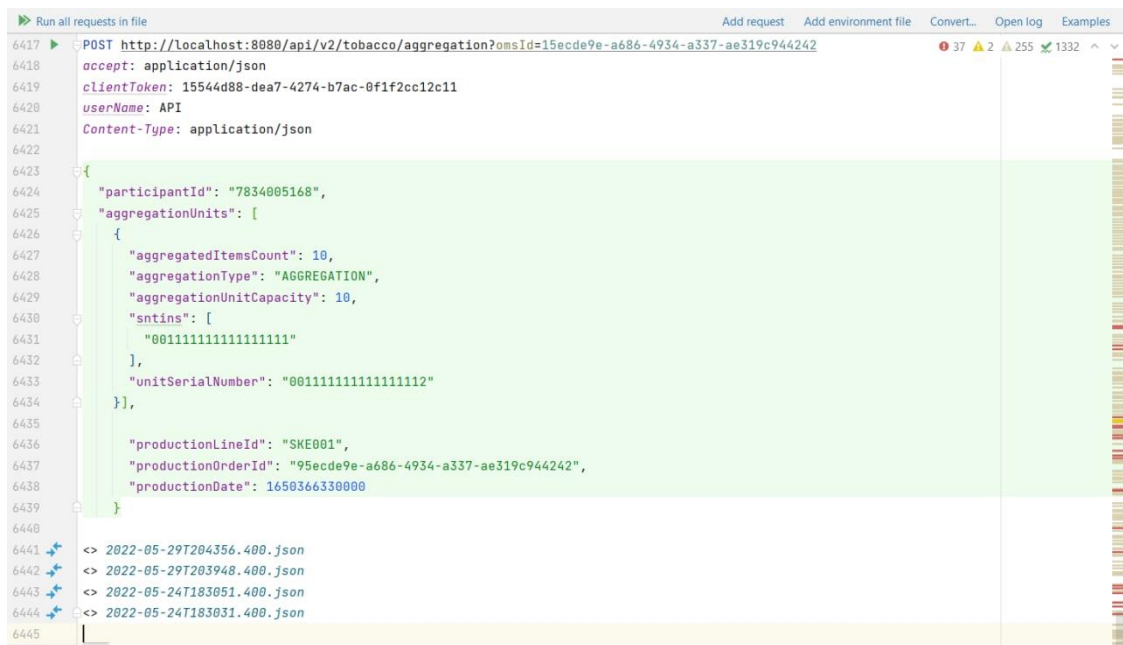


Рисунок 35 - Ручное тестирование в IDE

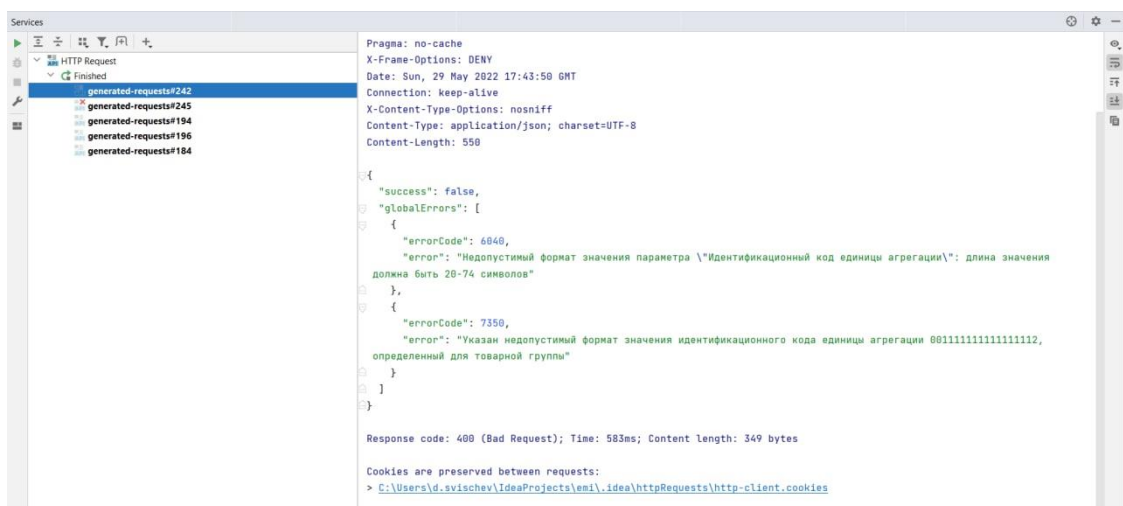


Рисунок 36 - Результат выполнения ручного тест-кейса

1.11.2.3 Интеграционное тестирование

Под интеграционным тестированием понимается совокупная проверка работы компонентов системы (сервисы, база данных), а также их взаимодействие с внешними системами. Данный вид тестирования является наиболее затратным, так как для его реализации необходимо полное развертывание системы, подготовка полного объема тестовых данных и тестового стенда.

Интеграционное тестирование программного компонента эмиссии и учета кодов маркировки производилось посредством написания авто-тестов на языке программирования Java с использованием библиотек JUnit5, Jupiter, Allure и системы сборки проекта Maven.

Создание автоматизированных интеграционных тестов производилось в соответствии со спецификой программного продукта. В разрезе каждой товарной группы были созданы:

- тест планы, позволяющие описать структуру проверяемых кейсов;
- описаны тестовые кейс с логическим разделением по основным процессам программного компонента эмиссии и учета кодов маркировки, а именно:

- 1) создание заказа, ожидание готовности заказа, получение кодов маркировки, закрытие заказа;

- 2) подготовка кодов маркировки для агрегации, формирования кода маркировки агрегата, создание отчета об агрегации кодов маркировки, ожидание перехода отчета в статус «Обработан»;

- 3) подготовка кодов маркировки для формирования отчета об отбраковке кодов маркировки, создание отчета об отбраковке кодов маркировки, ожидание перехода отчета в статус «Обработан»;

- 4) подготовка кодов маркировки для формирования отчета о нанесении кодов маркировки, создание отчета о нанесении кодов маркировки, ожидание перехода отчета в статус «Обработан»;

- 5) создание полного интеграционного теста, включающего процессы по созданию и обработке заказа, созданию и обработке отчета об агрегации кодов маркировки, создание и обработке отчета о нанесении кодов маркировки.

Описание интеграционных тестов построено на основе концепции Given-When-Then (Рисунок 37), которая представляет собой последовательности выполнения тестов, а именно:

- given – секция, в которой осуществляется подготовка тестовых данных для каждого кейса;

- when – секция, в которой производятся манипуляции с данными, подготовленными на предыдущем этапе, осуществляются запросы к сервисам и формируются результаты их ответов;

- then – секция, в которой осуществляется анализ полученных на предыдущем этапе данных, а также выполняются проверки корректности поведения сервиса в определенных условиях, на основе заданных на входе значений.

```
@Test
@Severity(SeverityLevel.NORMAL)
@DisplayName(USE_CASE_I_006_011)
@Description(USE_CASE_I_006_011)
void createOrder_whenPackAndOperator_thenOk() {
    // Given
    Env env = enviromentService.getEnvironment(TOBACCO, FELICHE);
    env.setCaseId(USE_CASE_I_006_011);

    env.setOrderDto(tobaccoStepService.getOrderFromFile("order_operator_unit",
    env));

    // When
    OrderResult order = tobaccoStepService.createOrder(env);

    // Then
    assertAll("checkOrderResult",      AssertionUtils.orderAssertions(env,
    order));

    try {
        env.setOrderId(order.getOrderId());
        tobaccoStepService.waitBuffers(env);

        List<CodesDto> orderCodes = tobaccoStepService.getOrderCodes(env);
        orderCodes.forEach(codes -> assertAll("checkCodesResult " +
        codes.getBlockId(), AssertionUtils.codesAssertions(env, codes)));
    } finally {
        OMSIdentifier closeBuffer = tobaccoStepService.closeBuffer(env);

        assertAll("checkCloseBufferResult",
        AssertionUtils.omsIdentifierAssertions(env, closeBuffer));
    }
}
```

Рисунок 37 - Пример интеграционного теста

Разработка интеграционных тестов осуществлялась в среде разработки IntelliJ IDEA IDE, которая позволяет осуществлять запуск тестов на основе встроенных средств (Рисунок 38) или с помощью плагинов системы сборки проекта Maven (Рисунок 39).



Рисунок 38 - Пример запуска интеграционного теста средствами IDE

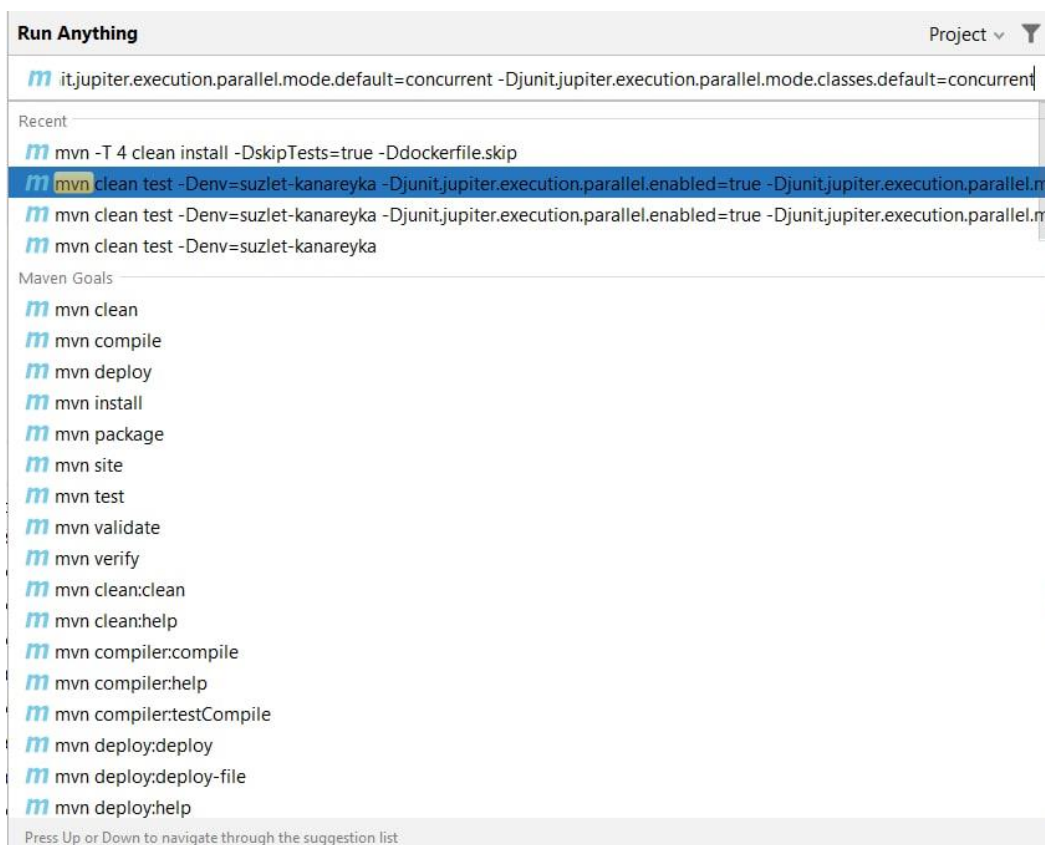


Рисунок 39 - Пример запуска интеграционных тестов средствами Maven

Для анализа результатов тестирования и формирования отчетности по процессу тестирования разработанного функционала используется библиотека Allure, которая при прохождении тестов генерирует JSON-файлы, формируемые впоследствии в полноценный отчет прохождения тестов (Рисунок 40 и Рисунок 41).

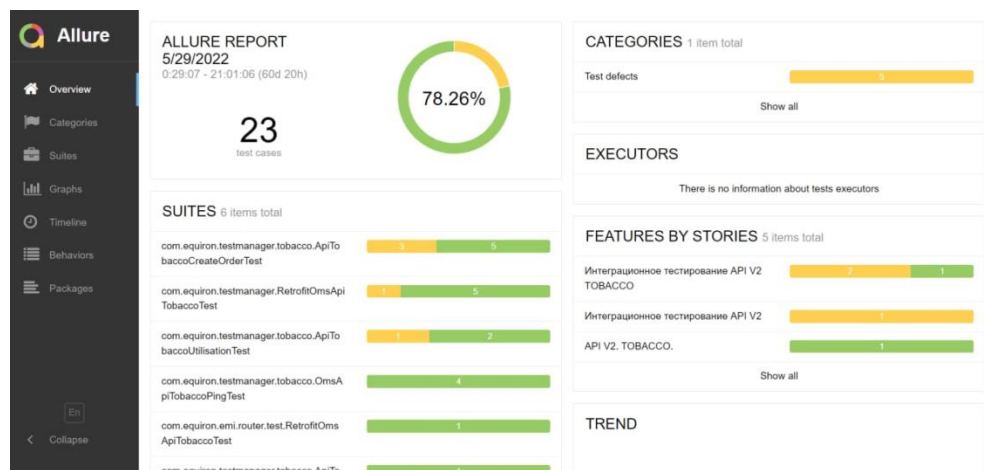


Рисунок 40 - Отчет прохождения интеграционных тестов

Результатом создания интеграционных тестов явилась разработка полноценного тестового движка, позволяющего создание не только интеграционные, но и модульные и сквозные тесты, производить приемочное и smoke-тестирование.

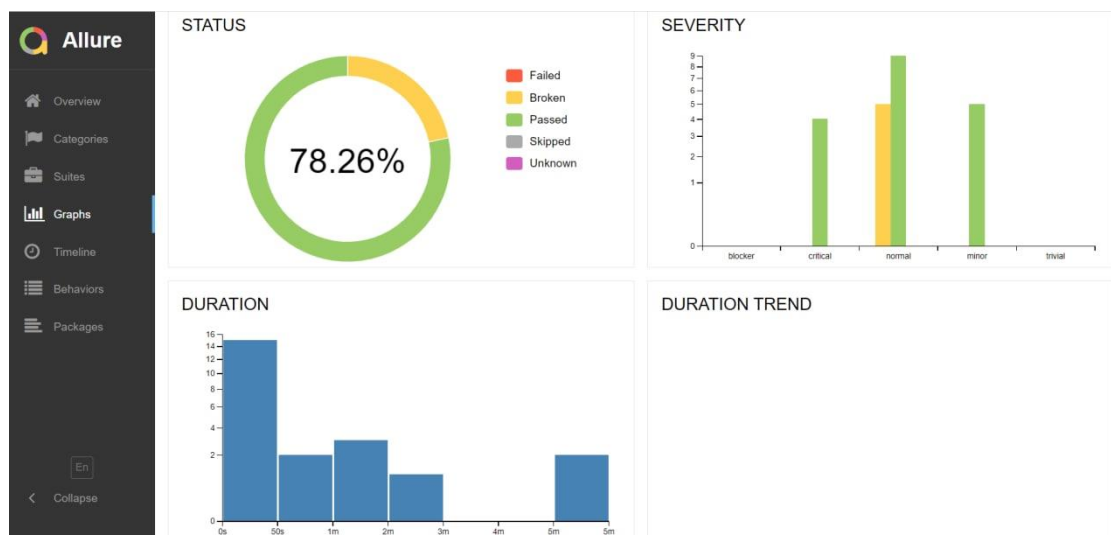


Рисунок 41 - Статистика прохождения интеграционных тестов

2 Безопасность жизнедеятельности

2.1 Анализ негативных факторов и способы их устранения

В век активного развития информационных технологий, средства вычислительной техники, в состав которых входят ПЭВМ, стали неотъемлемой частью в различных сферах деятельности и используются для обеспечения автоматизации рабочих процессов операторов ПК.

Несмотря на современную тенденцию по применению инновационных и наиболее безопасных технологий при проектировании ПЭВМ, невозможно полностью избежать процесса отрицательного воздействия ПЭВМ на использующего его оператора. Важной задачей современно мира информационных технологий является минимизация факторов отрицательного воздействия.

Для минимизации отрицательного воздействия, необходимо соблюдать ряд правил пользования при взаимодействии с ПЭВМ. Основными правилами использования являются:

- правильное расположение техники при работе с ПЭВМ;
- правильное расположение рабочих мест операторов ПЭВМ в помещениях;
- освещённость рабочего места;
- соблюдение правил безопасности;

В данном разделе выпускной квалификационной работы рассмотрены вопросы следующие вопросы:

- организации рабочего пространства для операторов ПЭВМ, взаимодействующих с автоматизированной системой управления специализированного программного обеспечения;
- методы уменьшения воздействия вредных факторов.

2.2 Физическое и психофизическое влияние

При работе операторов с ПЭВМ, средства информационных технологий могут оказывать неблагоприятное воздействие, включая опасные и вредные производственные факторы:

- физические:

- 1) повышенный уровень электромагнитных излучений;
- 2) повышенный уровень статического электричества;
- 3) повышенная напряженность электростатического поля;
- 4) повышенная яркость света;
- 5) прямая и отраженная блескость;

- психофизиологические:

- 1) статические перегрузки костно-мышечного аппарата и динамические локальные перегрузки мышц кистей рук;
- 2) перенапряжение зрительного аппарата;
- 3) умственное перенапряжение;
- 4) эмоциональные перегрузки;
- 5) монотонность труда.

Важным процессом при организации труда оператора для взаимодействия с ПЭВМ является минимизация влияния описанных выше факторов за счет применения оптимального режима труда и соблюдения отдыха в течение рабочего дня, а именно:

- использование жидкокристаллических дисплеев и мониторов с встроенной защитой, высоким разрешением и комфортной диагональю экрана;
- организация 10 минутных ежечасовых перерывов в течение интенсивного рабочего дня;
- проведение оздоровительных зарядок и разминок глаз, кистей, спины.

2.3 Оборудование рабочих мест

Рисунок 42 отражает требования к организации рабочего места оператора ПЭВМ в соответствии с ГОСТ 12.2.032-78.

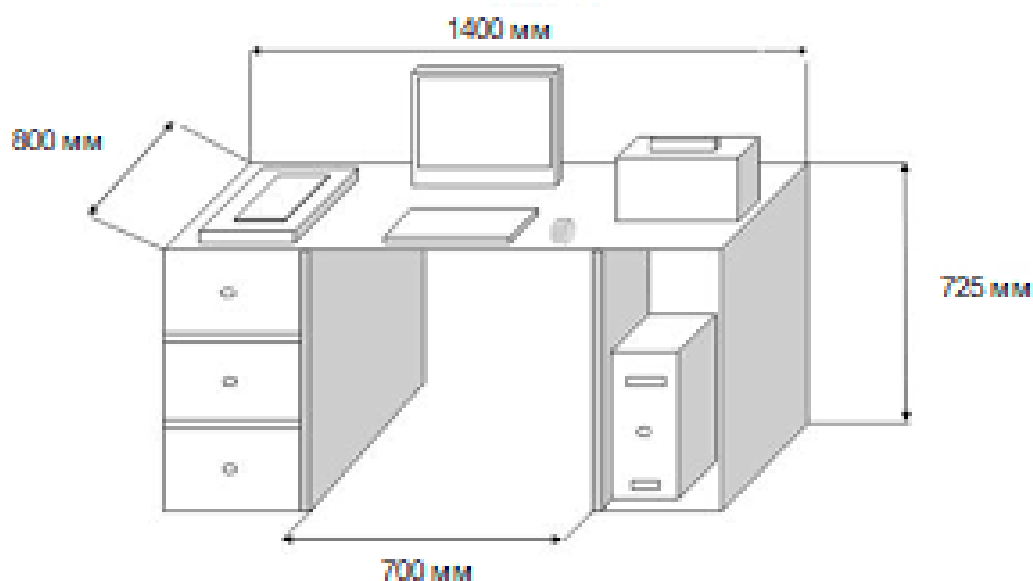


Рисунок 42 - Требования к организации рабочего места

Рабочее место оператора ПЭВМ должно соответствовать следующим требованиям:

- высота рабочей поверхности стола должна быть регулируемой и соответствовать 680-800 мм, а нерегулируемая высота – составлять 725 мм;
- модульные размеры рабочей поверхности стола для ПЭВМ– следует считать:

1) ширина 800, 1200, 1400 мм;

2) глубина 800, 1000 мм при нерегулируемой высоте 725 мм.

- рабочий стул должен быть подъемно-поворотным и– регулируемым по высоте, углам наклона, а также расстоянию спинки до сидения;
- расположение клавиатуры на рабочем столе от края 100-300 мм, или на специальной регулируемой по высоте рабочей поверхности.

Нельзя забывать про такую неотъемлемую часть минимизации вредных воздействий на оператора при работе с ПЭВМ, как обеспечение правильной посадки пользователя ПК, которая отображена на Рисунок 43.

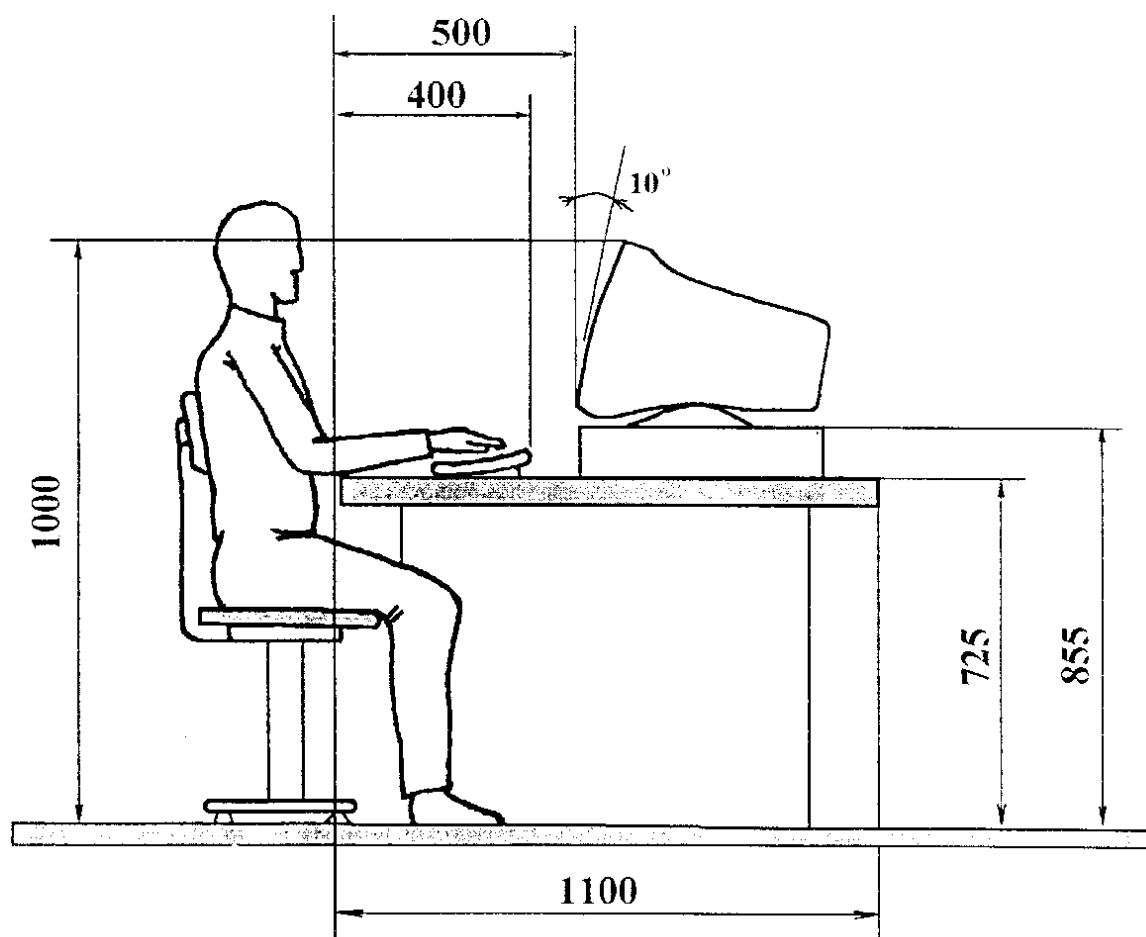


Рисунок 43 - Правильная посадка оператора при работе с ПЭВМ

2.4 Освещенность рабочего места

Соблюдение правил по организации освещенности рабочего места является важной задачей уменьшения влияния вредных факторов при работе с ПЭВМ.

В соответствии с ГОСТ Р 50923-96 освещение рабочих мест внутри зданий должно определять по следующим правилам:

- освещенность рабочего места оператора на рабочем столе в горизонтальной плоскости от общего искусственного освещения должна быть от 300 до 500 лк;

- отношение яркостей в зоне наблюдения (экран, документ, поверхность стола) должно быть не более 10:1;

- необходимо обеспечение отсутствия блескости за счет оборудования светопроемов солнцезащитными устройствами, использования дисплеев с антибликовой поверхностью экрана, специализированным размещением рабочих столов операторов ПЭВМ, при котором оконный проем находится сбоку (слева или справа);

- использование для местного освещения специализированных светильников с непросвечивающим отражателем и защитным углом не менее 30 градусов.

					ПГУ 1.090301.36.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения выпускной квалификационной работы было разработан клиент-серверный программный компонент эмиссии и учета кодов маркировки, основанный на сервис-ориентированной архитектуре. Созданный программный компонент полностью соответствует поставленному техническому заданию и реализует следующие основные функции:

- осуществление заказов на эмиссию кодов маркировки;
- предоставления эмитированных кодов маркировки для нанесения на продукцию;
- верификация нанесения КМ, формирования отчета о нанесении;
- агрегация готовой и упакованной продукции с нанесенными средствами идентификации;
- списание бракованной продукции с указанием причины отбраковки.

Разработанный программный компонент решает актуальные проблемы по борьбе с производством, распространением и продажей контрафактной продукции и товаров, предоставляя обширный функционал для работы с системой и пользователе-ориентированный интерфейс. Автоматизированная система специализированного программного обеспечения внедрена в процессы национальной маркировки продукции Российской Федерации и является входной точкой для осуществления процесса маркировки.

В ходе выполнения работы были получены навыки по проектированию и разработке программного обеспечения, основанного на клиент-серверной и сервис-ориентированной архитектуре, разработки высоконагруженных и многопоточных программ. А также, усовершенствованы умения анализа предметной области, постановки технических требований и декомпозиции задач и процессов.

Список использованных источников

1. Д.В. Мантуров, «Методические рекомендации для проведения эксперимента по маркировке табачной продукции средствами идентификации и мониторингу оборота табачной продукции на территории Российской Федерации». Утверждено Минпромторгом России 25.12.2017.
2. Федеральный закон "О внесении изменений в Федеральный закон "Об основах государственного регулирования торговой деятельности в Российской Федерации" и статьи 4.4 и 4.5 Федерального закона "О применении контрольно-кассовой техники при осуществлении расчетов в Российской Федерации" от 25.12.2018 N 488-ФЗ.
3. Д.В. Мантуров, «Методические рекомендации для участников эксперимента по маркировке средствами идентификации и мониторингу оборота фотокамер (кроме кинокамер), фотовспышек и ламп-вспышек в Российской Федерации». Утверждено Минпромторгом России 28.10.2019.
4. Г.М. Кадырова, «Методические рекомендации для участников эксперимента по маркировке средствами идентификации и мониторингу оборота духов и туалетной воды в Российской Федерации». Утверждено Минпромторгом России 31.10.2019.
5. Электронная библиотека, свободная энциклопедия Википедия. «Система маркировки и прослеживаемости товаров (Россия)» [Электронный ресурс]. URL : https://ru.wikipedia.org/wiki/Система_маркировки_и_прослеживаемости_товаров.
6. Юлиана Козмина, Роб Харроп, Крис Шефер, Кларенс Хо. «Spring 5 для профессионалов». 2019 год, 1120 стр.
7. Саймон Ригс, Ханну Кросинг. «Администрирование PostgreSQL9. Книга рецептов». 2013 год, 368 стр.
8. David Heffelfinger. «JasperReports 3.5 for Java Developers». 2009 год, 368 стр.
9. Ричардсон Крис. «Микросервисы. Паттерны разработки и рефакторинга». 2022 год, 544 стр.

Приложение А

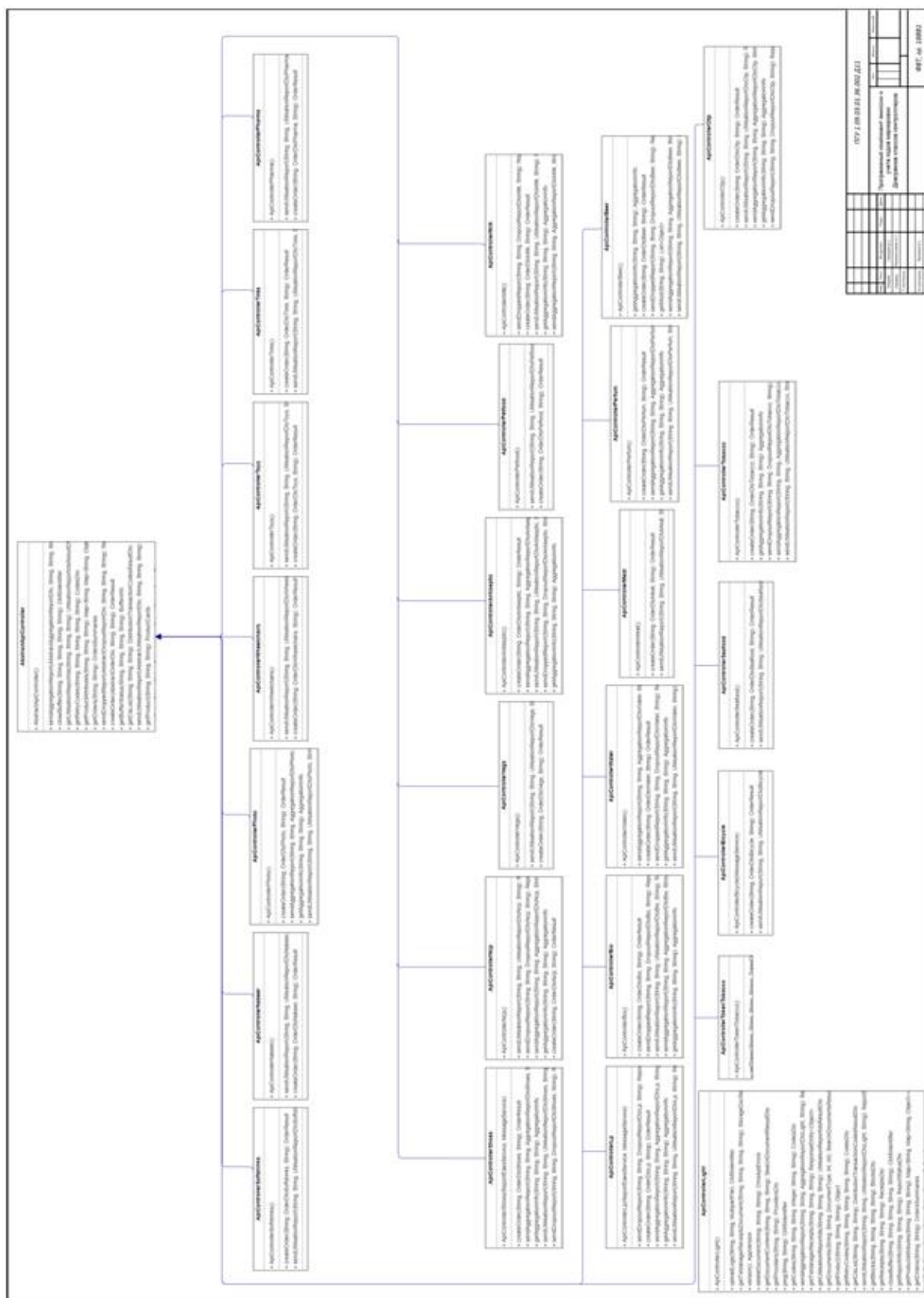
(обязательное)

Программный компонент эмиссии и учета кодов маркировки

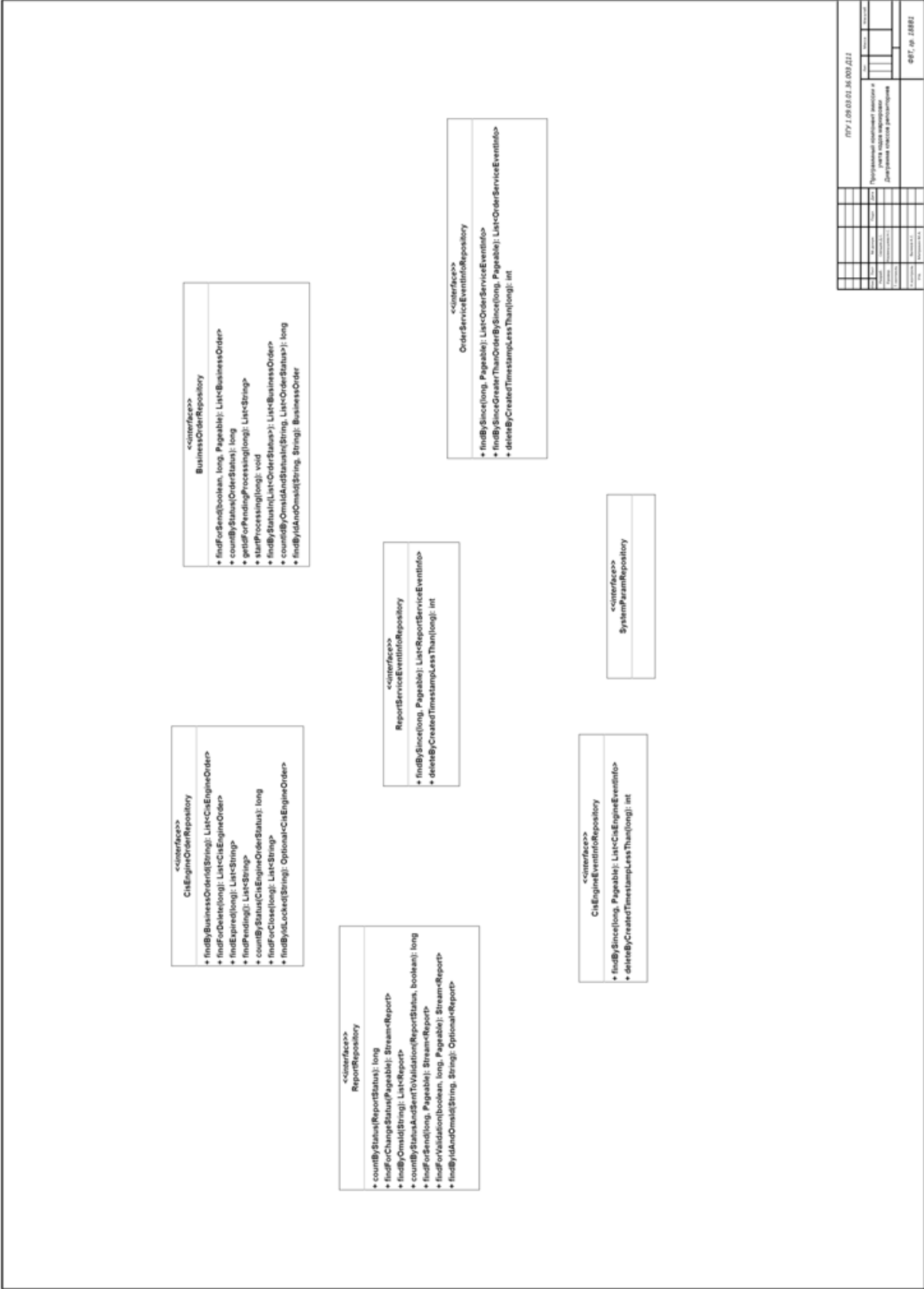
UML-диаграммы

					ПГУ 1.090301.36.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 - UML-диаграмма классов контроллеров

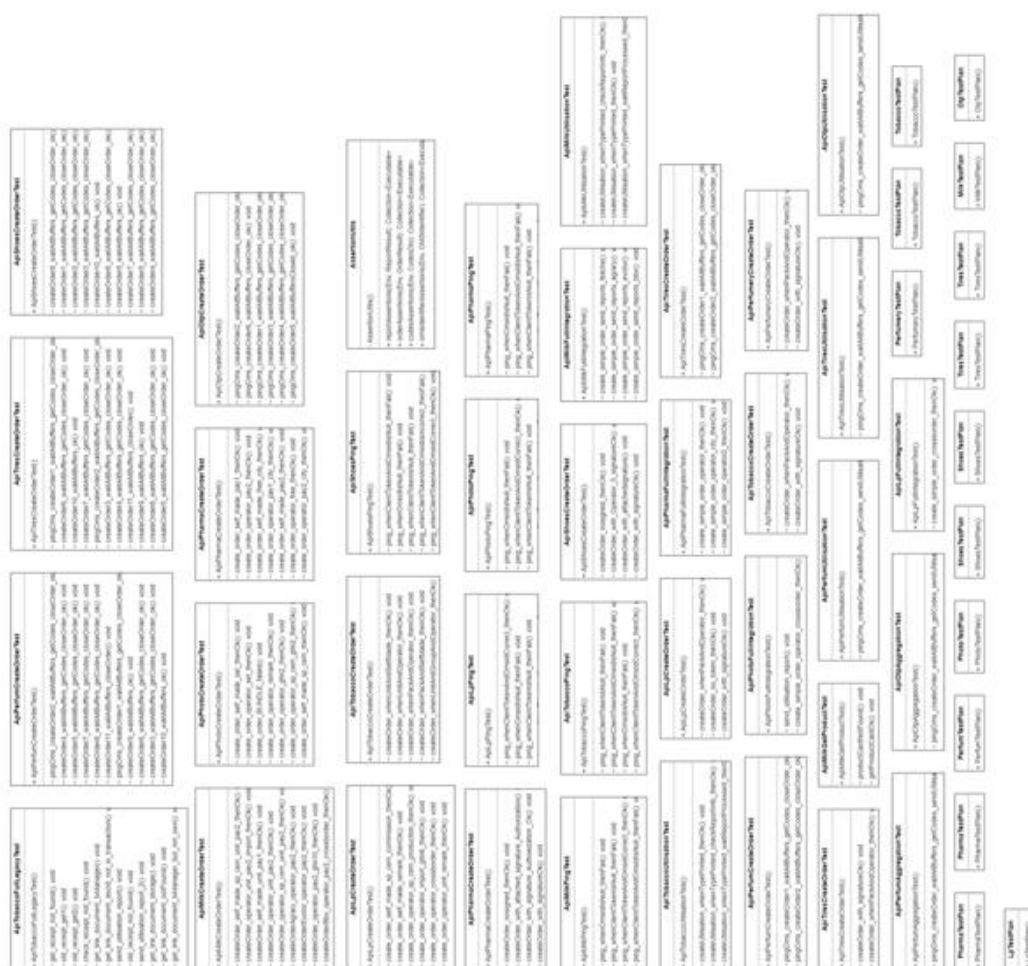


Приложение А.2 - UML-диаграмма классов репозиториев

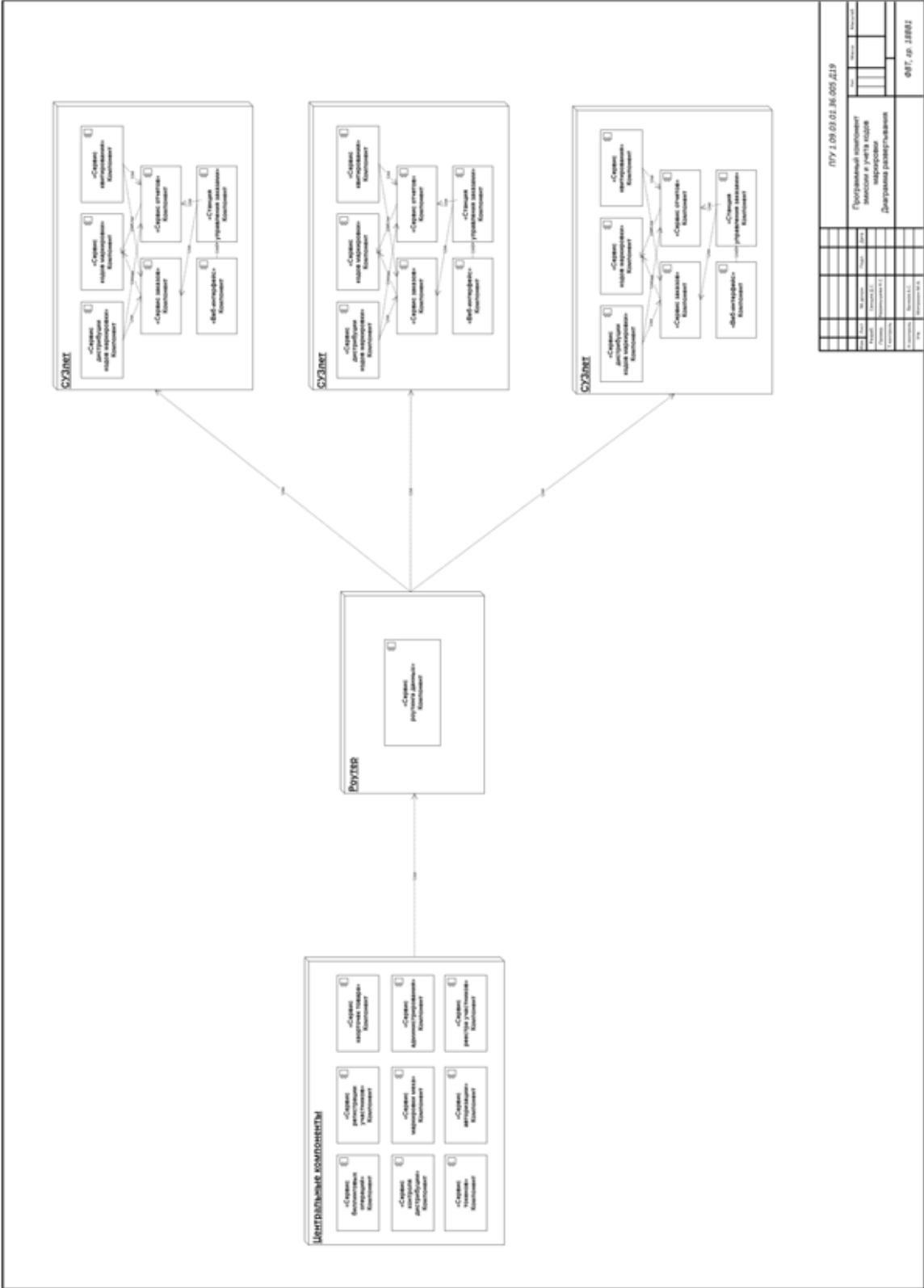


ПГУ 1.09.03.01.36.001.012		ПГУ 1.09.03.01.36.001.012	
№ докум.	Лист	№ докум.	Лист
Изм.	Лист	Изм.	Лист
Подпись	Дата	Подпись	Дата
Итого: 105		Итого: 105	

Приложение А.3 - UML-диаграмма классов сервисов

[illegible]

Приложение А.4 - UML-диаграмма развертывания



ПГУ 1.09.03.01.36.001.019		Планирование мероприятий по обеспечению безопасности жизни и имущества граждан		ФОРТ, 09.10.2012	
№	Имя	Дата	Статус	Исполнитель	Проверенный
1	Иванов И.И.	09.10.2012	В работе	Иванов И.И.	Иванов И.И.
2	Петров П.П.	09.10.2012	Завершено	Петров П.П.	Петров П.П.
3	Сидоров С.С.	09.10.2012	В работе	Сидоров С.С.	Сидоров С.С.
4	Климов К.К.	09.10.2012	Завершено	Климов К.К.	Климов К.К.
5	Лебедев Л.Л.	09.10.2012	В работе	Лебедев Л.Л.	Лебедев Л.Л.
6	Новиков Н.Н.	09.10.2012	Завершено	Новиков Н.Н.	Новиков Н.Н.
7	Осипов О.О.	09.10.2012	В работе	Осипов О.О.	Осипов О.О.
8	Попов П.П.	09.10.2012	Завершено	Попов П.П.	Попов П.П.
9	Рябенко Р.Р.	09.10.2012	В работе	Рябенко Р.Р.	Рябенко Р.Р.
10	Смирнов С.С.	09.10.2012	Завершено	Смирнов С.С.	Смирнов С.С.
11	Тихонов Т.Т.	09.10.2012	В работе	Тихонов Т.Т.	Тихонов Т.Т.
12	Усачев У.У.	09.10.2012	Завершено	Усачев У.У.	Усачев У.У.
13	Фролов Ф.Ф.	09.10.2012	В работе	Фролов Ф.Ф.	Фролов Ф.Ф.
14	Харин Х.Х.	09.10.2012	Завершено	Харин Х.Х.	Харин Х.Х.
15	Цыганов Ц.Ц.	09.10.2012	В работе	Цыганов Ц.Ц.	Цыганов Ц.Ц.
16	Чайков Ч.Ч.	09.10.2012	Завершено	Чайков Ч.Ч.	Чайков Ч.Ч.
17	Шевченко Ш.Ш.	09.10.2012	В работе	Шевченко Ш.Ш.	Шевченко Ш.Ш.
18	Щербак Щ.Щ.	09.10.2012	Завершено	Щербак Щ.Щ.	Щербак Щ.Щ.
19	Юрьев Ю.Ю.	09.10.2012	В работе	Юрьев Ю.Ю.	Юрьев Ю.Ю.
20	Яковлев Я.Я.	09.10.2012	Завершено	Яковлев Я.Я.	Яковлев Я.Я.

Приложение Б

(обязательное)

Программный компонент эмиссии и учета кодов маркировки

Листинг программы

					ПГУ 1.090301.36.001 ПЗ	Лист
						108
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «OrderProductAntiseptic»

```
package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;

import io.swagger.annotations.ApiModel;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductAntiseptic extends OrderProductRu implements
OrderProductWithCisType {
    private CisType cisType;

    @Override

    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.ANTISEPTIC ||
getProductGroup() == ProductGroupRu.ANTISEPTIC_SHORT) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.ANTISEPTIC;
    }
}
```

Файл «OrderProductBeer»

```
package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		109

```

import io.swagger.annotations.ApiModel;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductBeer extends OrderProductRu implements
OrderProductWithCisType {
    private CisType cisType;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.BEER) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.BEER;
    }
}

```

Файл «OrderProductBicycle»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
public class OrderProductBicycle extends OrderProductRu {
    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.BICYCLE) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		110

```

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.BICYCLE;
    }
}

```

Файл «OrderProductBio»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import lombok.Getter;
import lombok.Setter;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductBio extends OrderProductRu implements
OrderProductWithCisType {
    private CisType cisType;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.BIO) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }

        if (getProductGroup() == ProductGroupRu.BIO_SHORT) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.BIO;
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		111

Файл «OrderProductFashion»

```
package com.equiron.sitemanager.api.v2.model.order.product;

import java.util.Map;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.enums.IdentificationType;
import com.equiron.sitemanager.api.v2.enums.MarkingType;
import com.equiron.sitemanager.api.v2.enums.ReleaseMethodType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductFashion extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    @Override
    public ProductGroupType getProductGroupType() {
        if(getProductGroup() == null) return null;

        if (getProductGroup() == ProductGroupRu.SHOES) return
ProductGroupType.SHOES;

        if (getProductGroup() == ProductGroupRu.LIGHT || getProductGroup()
== ProductGroupRu.REAPPLY) {
            return ProductGroupType.LIGHT;
        }

        return null;
    }

    @Override
    public CisParam getCisParam() {
```

					ПГУ 1.090301.36.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (getProductGroup() == ProductGroupRu.SHOES || getProductGroup()
== ProductGroupRu.LIGHT) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }
}

```

Файл «OrderProductMeat»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductMeat extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.MEAT) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		113

```

        return ProductGroupType.MEAT;
    }
}

```

Файл «OrderProductMilk»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import com.equiron.sitemanager.api.v2.model.validator.order.MilkTemplateId;
import com.equiron.sitemanager.api.v2.model.validator.report.ValidMilkDate;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@MilkTemplateId
@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductMilk extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "\"191231\"", value = "Expiry date of the
product (expiration date more than 72 hours) (Дата окончания срока годности
продукции (срок хранения более 72 часов))", hidden = true)
    @ValidMilkDate(message = "{6050.validator.milk.expDate.format}", format
= "uuMMdd")
    private String expDate;

    @ApiModelProperty(example = "\"1912312359\"", value = "Expiration date
of the product (shelf life less than 72 hours) Дата окончания срока годности
продукции (срок хранения менее 72 часов)", hidden = true)
    @ValidMilkDate(message = "{6050.validator.milk.expDate72.format}",
format = "uuMMddHHmm")
    private String expDate72;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		114

```

        if (getProductGroup() == ProductGroupRu.MILK) {
            if (getExpDate() != null && !getExpDate().isBlank()) {
                return new CisParam("01" + getGtin() + "21#####]17"
+
                getExpDate(),
                getProductGroup().getCisType(),
getProductGroup().getSerialLength());
            }

            if (getExpDate72() != null && !getExpDate72().isBlank()) {
                return new CisParam("01" + getGtin() +
"21#####]7003" +
                getExpDate72(),
                getProductGroup().getCisType(),
getProductGroup().getSerialLength());
            }

            return new CisParam("01" + getGtin() + "21#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
        } else if (getProductGroup() == ProductGroupRu.MILK_V2) {
            return new CisParam("01" + getGtin() + "215####",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
        }

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.MILK;
    }
}

```

Файл «OrderProductNabeer»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductNabeer extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		115

```

        @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
        private String exporterTaxpayerId;

        @Override
        public CisParam getCisParam() {
            if (getProductGroup() == ProductGroupRu.NABEER) {
                return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
            }
            throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
        }

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.NABEER;
        }
    }
}

```

Файл «OrderProductNcp»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import com.equiron.sitemanager.api.v2.model.validator.order.NcpTemplateId;

import io.swagger.annotations.ApiModel;
import lombok.Getter;
import lombok.Setter;

@NcpTemplateId
@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductNcp extends OrderProductRu implements
OrderProductWithCisType {
    private CisType cisType;

    public String getMrp() {
        if (getTemplateId() == ProductGroupRu.NCP_PACK.getId()) {
            return "000000";
        }
        if (getTemplateId() == ProductGroupRu.NCP.getId()) {
            return "AAAA";
        }
        throw new IllegalStateException("TemplateId " + getTemplateId() + "
is incorrect for NCP");
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		116

```

@Override
public CisParam getCisParam() {
    int serialLength = getProductGroup().getSerialLength();

    if (getProductGroup() == ProductGroupRu.NCP_PACK) {
        return new CisParam("01" + getGtin() + "21" +
"#".repeat(serialLength) + "]8005" + getMrp(), getProductGroup().getCisType(),
serialLength);
    }

    if (getProductGroup() == ProductGroupRu.NCP) {
        return new CisParam(getGtin() + "#".repeat(serialLength) +
getMrp(), getProductGroup().getCisType(), serialLength);
    }

    throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
}

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.NCP;
}
}

```

Файл «OrderProductOtp»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductOtp extends OrderProductRu implements
OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234", value = "Maximum retail price
(Максимальная розничная цена)", required = false)
    private String mrp;

    @Override

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		117

```

        public CisParam getCisParam() {
            if (getProductGroup() == ProductGroupRu.OTP) {
                return new CisParam("01" + getGtin() + "21#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
            }

            if (getProductGroup() == ProductGroupRu.OTP_MRP) {
                return new CisParam(getGtin() + "#####AAAA",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
            }

            throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
        }

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.OTP;
        }
    }
}

```

Файл «OrderProductPerfum»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductPerfum extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		118

```

        if (getProductGroup() == ProductGroupRu.PERFUM) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PERFUM;
    }
}

```

Файл «OrderProductPetfood»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductPetfood extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        //Корма для домашних животных (кроме сельскохозяйственных)
        if (getProductGroup() == ProductGroupRu.PETFOOD) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		119

```

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PETFOOD;
    }
}

```

Файл «OrderProductPharma»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import com.equiron.sitemanager.api.v2.model.validator.order.PharmaTemplateId;
import io.swagger.annotations.ApiModel;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@ApiModel
@PharmaTemplateId
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
public class OrderProductPharma extends OrderProductRu {
    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.PHARM) return new
CisParam("01" + getGtin() + "21#####", getProductGroup().getCisType(),
getProductGroup().getSerialLength());
        if (getProductGroup() == ProductGroupRu.PHARM_SHORT) return new
CisParam("01" + getGtin() + "21#####", getProductGroup().getCisType(),
getProductGroup().getSerialLength());

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PHARMA;
    }
}

```

Файл «OrderProductPhoto»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		120

```

import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductPhoto extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.PHOTO) {
            return new CisParam("01" + getGtin() +
"215#####",
getProductGroup().getSerialLength() -1);
        }
        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PHOTO;
    }
}

```

Файл «OrderProductRu»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProduct;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;

import javax.xml.bind.annotation.XmlTransient;

@XmlTransient
public abstract class OrderProductRu extends OrderProduct {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						121
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public ProductGroupRu getProductGroup() {
        return ProductGroupRu.fromId(getTemplateId());
    }

    @Override
    public CisParam getCisParam() {
        return null;
    }
}

```

Файл «OrderProductSeafood»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductSeafood extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        //Морепродукты
        if (getProductGroup() == ProductGroupRu.SEAFOOD) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.SEAFOOD;
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		122

Файл «OrderProductShoes»

```
package com.equiron.sitemanager.api.v2.model.order.product;

import java.util.Map;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import org.springframework.web.servlet.mvc.annotation.annotation.BusinessException;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.enums.IdentificationType;
import com.equiron.sitemanager.api.v2.enums.MarkingType;
import com.equiron.sitemanager.api.v2.enums.ReleaseMethodType;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductShoes extends OrderProductRu implements
OrderProductWithTaxpayerId {
    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    public OrderProductShoes(OrderProductFashion orderProductFashion) {
        if (orderProductFashion.getTemplateId() == 10) throw new
BusinessException("Incorrect templateId");

        this.productAttributes =
orderProductFashion.getProductAttributes();
        this.releaseMethodType =
orderProductFashion.getReleaseMethodType();
        this.identificationType =
orderProductFashion.getIdentificationType();
        this.markingType = orderProductFashion.getMarkingType();
        this.productionLine = orderProductFashion.getProductionLine();
        this.codeTnVed = orderProductFashion.getCodeTnVed();
        this.exporterTaxpayerId =
orderProductFashion.getExporterTaxpayerId();
    }

    public OrderProductShoes() {
        //empty
    }
}
```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		123

```

@Override
public CisParam getCisParam() {
    if (getProductGroup() == ProductGroupRu.SHOES) {
        return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
    }

    if (getProductGroup() == ProductGroupRu.REAPPLY) {
        return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
    }

    throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
}

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.SHOES;
}
}

```

Файл «OrderProductSoftdrinks»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductSoftdrinks extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        //Соковая продукция и безалкогольные напитки
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		124

```

        if (getProductGroup() == ProductGroupRu.SOFTDRINKS) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.SOFTDRINKS;
    }
}

```

Файл «OrderProductTires»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductTires extends OrderProductRu implements
OrderProductWithTaxpayerId {
    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    @Size(min = 1, max = 20, message =
"{6040.validator.invalid.length.exporterTaxpayerId}")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.TIRES) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }

        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

```

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.TIRES;
    }
}

```

Файл «OrderProductTobacco»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.validation.constraints.NotNull;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.emi.util.CisUtils;
import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import com.equiron.sitemanager.api.v2.model.validator.order.MRPTobacco;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType (XmlAccessType.FIELD)
@MRPTobacco
@Getter
@Setter
public class OrderProductTobacco extends OrderProductRu {
    @ApiModelProperty(example = "1234", value = "Maximum retail price
(Максимальная розничная цена)", required = true)
    @NotNull(message = "{6010.validator.OrderProduct.mrp.empty}")
    private String mrp;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.TOBACCO_PACK) {
            if (getMrp() == null || getMrp().isBlank()) {
                return new CisParam("01" + getGtin() + "21#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
            }

            return new CisParam("01" + getGtin() + "21#####]8005" +
CisUtils.prependingZerosMrp(mrp), getProductGroup().getCisType(),
getProductGroup().getSerialLength());
        }

        if (getProductGroup() == ProductGroupRu.TOBACCO) {
            if (getMrp() == null || getMrp().isBlank()) {
                return new CisParam(getGtin() + "#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength());
            }
        }
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						126
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }

        return new CisParam(getGtin() + "#####" +
CisUtils.codingMrp(mrp), getProductGroup().getCisType(),
getProductGroup().getSerialLength());
    }

    throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
}

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.TOBACCO;
}
}

```

Файл «OrderProductToys»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductToys extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        //Соковая продукция и безалкогольные напитки
        if (getProductGroup() == ProductGroupRu.TOYS) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() -1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						127
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.TOYS;
}
}

```

Файл «OrderProductVegs»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductVegs extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.VEGS) {
            return new CisParam("01" + getGtin() + "215####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }
        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.VEGS;
    }
}

```

Файл «OrderProductVegs»

```

package com.equiron.sitemanager.api.v2.model.order.product;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		128

```

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.CisParam;
import com.equiron.sitemanager.api.v2.OrderProductWithCisType;
import com.equiron.sitemanager.api.v2.OrderProductWithTaxpayerId;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.CisType;

import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderProductWater extends OrderProductRu implements
OrderProductWithTaxpayerId, OrderProductWithCisType {
    private CisType cisType;

    @ApiModelProperty(example = "1234567890", value = "ИНН/УНБ (или аналог)
экспортера")
    private String exporterTaxpayerId;

    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.WATER) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }

        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.WATER;
    }
}

```

Файл «OrderProductWheelchairs»

```

package com.equiron.sitemanager.api.v2.model.order.product;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		129

```

import com.equiron.sitemanager.api.v2.CisParam;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.ProductGroupRu;
import io.swagger.annotations.ApiModel;

@ApiModel
@XmlRootElement(name = "product")
@XmlAccessorType(XmlAccessType.FIELD)
public class OrderProductWheelchairs extends OrderProductRu {
    @Override
    public CisParam getCisParam() {
        if (getProductGroup() == ProductGroupRu.WHEELCHAIRS) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }

        if (getProductGroup() == ProductGroupRu.REAPPLY) {
            return new CisParam("01" + getGtin() + "215#####",
getProductGroup().getCisType(), getProductGroup().getSerialLength() - 1);
        }

        throw new IllegalStateException("Incorrect template id for " +
this.getClass().getName() + ": " + getProductGroup().getId());
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.WHEELCHAIRS;
    }
}

```

Файл «LightIndustryOrderDto»

```

package com.equiron.sitemanager.api.v2.model.order;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlTransient;

import com.equiron.sitemanager.api.v2.enums.CreateMethodType;
import com.equiron.sitemanager.api.v2.enums.ReleaseMethodType;

import com.equiron.sitemanager.api.v2.model.validator.order.CheckReapply;
import
com.equiron.sitemanager.api.v2.model.validator.order.ContractorOrder;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@XmlTransient
@Getter
@Setter
@ContractorOrder
@CheckReapply
public abstract class LightIndustryOrderDto extends OrderDtoRu {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		130

```

        @ApiModelProperty(example = "John Smith (Иванов Петр Сидорович)", value
= "Contact Person (Контактное лицо)", required = false)
        @Size(min = 1, max = 128, message =
"{6040.validator.invalid.length.contactPerson}")
        private String contactPerson;

        @ApiModelProperty(example = "IMPORT", value = "Product Release Type
(Способ выпуска товаров в оборот)", required = true)
        @NotNull(message = "{6010.validator.orderDto.releaseMethodType.empty}")
        private ReleaseMethodType releaseMethodType;

        @ApiModelProperty(example = "SELF_MADE", value = "Marking Manufacturing
Type (Способ изготовления)", required = true)
        @NotNull(message = "{6010.validator.order.createMethodType.empty}")
        private CreateMethodType createMethodType;

        @ApiModelProperty(example = "63b5d1cd-919b-4cef-b117-2ce59059bc33",
value = "Production Order ID (Идентификатор производственного заказа)")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productionOrderId}")
        @Pattern(regexp = "^([\\x00-\\x1F]{1,256})$",
message =
"{6050.validator.productionOrderId.incorrect.format}")
        private String productionOrderId;

        @ApiModelProperty(example = "БФ0000001", value = "Contract number with
operator (Номер договора с оператором)", hidden = true)
        private String contractNumber;

        @ApiModelProperty(example = "2019-12-31", value = "Date of contract
with operator (Дата договора с оператором)", hidden = true)
        @Pattern(regexp = "^\\d{4}-(0[1-9]|1[012])-(0[1-9]|12)[0-9]|3[01])$")
        private String contractDate;
    }

```

Файл «OrderDtoAntiseptic»

```

package com.equiron.sitemanager.api.v2.model.order;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import java.util.ArrayList;
import java.util.List;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductAntiseptic;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						131
Изм.	Лист	№ докум.	Подпись	Дата		

```

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoAntiseptic.class)
@XmlRootElement(name = "antiseptic")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoAntiseptic extends LightIndustryOrderDto {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductAntiseptic> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.ANTISEPTIC;
    }
}

```

Файл «OrderDtoBeer»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductBeer;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoBeer.class)
@XmlRootElement(name = "beer")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoBeer extends LightIndustryOrderDto {

```

```

        @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
        @NotNull
        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElements(@XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductBeer> products = new ArrayList<>();

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.BEER;
        }
    }
}

```

Файл «OrderDtoBicycle»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductBicycle;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@JsonTypeInfo(use = Id.CLASS, defaultImpl = OrderDtoBicycle.class)
@XmlRootElement(name = "bicycle")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoBicycle extends LightIndustryOrderDto {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductBicycle> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.BICYCLE;
    }
}

```

Файл «OrderDtoBio»

```
package com.equiron.sitemanager.api.v2.model.order;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import java.util.ArrayList;
import java.util.List;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductBio;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl = OrderDtoBio.class)
@XmlRootElement(name = "bio")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoBio extends LightIndustryOrderDto {
    @ApiModelProperty(example = "", value = "Product list (Список товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message = "{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductBio> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.BIO;
    }
}
```

Файл «OrderDtoLight»

```
package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
```

					ПГУ 1.090301.36.001 ПЗ	Лист
						134
Изм.	Лист	№ докум.	Подпись	Дата		

```

import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.WithRemainsAvailable;
import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.AbstractOrderDto;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductFashion;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import
com.equiron.sitemanager.api.v2.model.validator.order.RemainsAvailable;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@RemainsAvailable(
    remainsRequiredFrom = "2021-02-01",
    remainsAvailableEmptyMessage = "{3600.validator.remains.empty.lp}",
    remainsAvailableFalseMessage =
"{3610.validator.remains.false.light}",
    remainsUnavailableFrom = "2021-05-01",
    remainsUnavailableShoesFrom = "2021-06-01",
    remainsUnavailableMessage = "{3620.validator.remains.unavailable}")
@OrderCisTypeComposition
@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoLight.class)
@XmlRootElement(name = "lp")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoLight extends LightIndustryOrderDto implements
WithRemainsAvailable, ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductFashion> products = new ArrayList<>();

    @ApiModelProperty(example = "true", notes = "Признак того, что товар
произведен/приобретен до даты запрета оборота немаркированных товаров по данной
ТГ")
    private Boolean remainsAvailable;
}

```

Файл «OrderDtoLp»

```
package com.equiron.sitemanager.api.v2.model.order;
```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		135

```

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.model.validator.order.RemainsFinalDate;
import org.springframework.beans.BeanUtils;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductFashion;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@RemainsFinalDate(value = "2021-06-01", message =
"{3030.order.product.group.releaseMethodType.notAllowed.lp}")
@OrderCisTypeComposition
@JsonTypeInfo(use=Id.CLASS, defaultImpl= OrderDtoLp.class)
@XmlRootElement(name = "lp")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoLp extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductFashion> products = new ArrayList<>();

    public OrderDtoLp(OrderDtoLight orderDtoLight) {
        BeanUtils.copyProperties(orderDtoLight, this);
        this.products = orderDtoLight.getProducts();
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		136

```

        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.LIGHT;
    }
}

```

Файл «OrderDtoMeat»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductMeat;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoMeat.class)
@XmlRootElement(name = "meat")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoMeat extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductMeat> products = new ArrayList<>();
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						137
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.MEAT;
}

@JsonIgnore
@Override
public boolean checkExporterTaxpayerId() {
    return OrderDtoUtils.checkExporterTaxpayerId(this);
}

@JsonIgnore
@Override
public boolean checkExporterTaxpayerIdForEmpty() {
    return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
}
}

```

Файл «OrderDtoMeat»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductMeat;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoMeat.class)
@XmlRootElement(name = "meat")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoMeat extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull

```

```

        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductMeat> products = new ArrayList<>();

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.MEAT;
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerId() {
            return OrderDtoUtils.checkExporterTaxpayerId(this);
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerIdForEmpty() {
            return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
        }
    }
}

```

Файл «OrderDtoMilk»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductMilk;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.equiron.sitemanager.api.v2.model.validator.order.PaymentTypeRange;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@OrderCisTypeComposition
@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoMilk.class)
@XmlRootElement(name = "milk")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						139
Изм.	Лист	№ докум.	Подпись	Дата		

```

@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoMilk extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElement(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductMilk> products = new ArrayList<>();

    @ApiModelProperty(example = "1", allowableValues = "1, 2", value =
"Payment Type (Тип оплаты: 1 - Оплата по эмиссии, 2 - Оплата по нанесению
(значение по умолчанию)")
    @PaymentTypeRange(min = 1, max = 2)
    private Integer paymentType = 2;

    public Integer getPaymentType() {
        return paymentType;
    }

    public void setPaymentType(Integer paymentType) {
        this.paymentType = paymentType;
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.MILK;
    }
}

```

Файл «OrderDtoNabeer»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		140

```

import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductNabeer;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoNabeer.class)
@XmlRootElement(name = "nabeer")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoNabeer extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductNabeer> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.NABEER;
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }
}

```

Файл «OrderDtoNcp»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotBlank;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		141

```

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.ReleaseMethodType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductNcp;
import com.equiron.sitemanager.api.v2.model.validator.report.ValidateDate;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl = OrderDtoNcp.class)
@XmlRootElement(name = "ncp")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoNcp extends OrderDtoRu {
    @ApiModelProperty(example = "", value = "Product list (Список товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message = "{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductNcp> products = new ArrayList<>();

    @ApiModelProperty(example = "ftf_001", value = "Factory Identifier (GLN) Идентификатор производства. (Глобальный номер места нахождения)", required = true)
    @NotBlank(message = "{6010.validator.order.factoryId.empty}")
    @Size(min = 2, max = 128, message = "{6040.validator.invalid.length.factoryId}")
    private String factoryId;

    @ApiModelProperty(example = "First Tobacco Factory", value = "Factory Name (Наименование производства)")
    @Size(min = 2, max = 256, message = "{6040.validator.invalid.length.factoryName}")
    private String factoryName;

    @ApiModelProperty(example = "ul. Pervaya Tabachnaya, 7", value = "Factory Address (Адрес производства)")
    @Size(min = 2, max = 256, message = "{6040.validator.invalid.length.factoryAddress}")
    private String factoryAddress;

    @ApiModelProperty(example = "RU", value = "Factory Country (Страна производителя)", required = true)
    @NotBlank(message = "{6010.validator.order.factoryCountry.empty}")
    @Size(min = 2, max = 128, message = "{6040.validator.invalid.length.factoryCountry}")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		142

```

        private String factoryCountry;

        @ApiModelProperty(example = "\"123\"", value = "Line Identifier
(Идентификатор производственной линии)", required = true)
        @NotBlank(message =
"{6010.validator.reportDto.productionLineId.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productionLineId}")
        private String productionLineId;

        @ApiModelProperty(example = "ABCD123456", value = "Product Code SKU
(Код продукта)", required = true)
        @NotBlank(message = "{6010.validator.order.productCode.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productCode}")
        private String productCode;

        @ApiModelProperty(example = "Cigarettes Friend", value = "Product
Description (Описание продукта)", required = true)
        @NotBlank(message = "{6010.validator.order.productDescription.empty}")
        @Size(min = 1, max = 512, message =
"{6040.validator.invalid.length.productDescription}")
        private String productDescription;

        @ApiModelProperty(example = "\"123456\"", value = "PO Number (Номер
производственного заказа)")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.poNumber}")
        private String poNumber;

        @ApiModelProperty(example = "2019-12-31", value = "Expected Start Date
(Дата начала производства продукции по данному заказу)")
        @ValidDate(format = "uuuu-MM-dd", parameterName = "expectedStartDate")
        private String expectedStartDate;

        @ApiModelProperty(example = "PRODUCTION", value = "Product Release Type
(Способ выпуска товаров в оборот)", required = true)
        @NotNull(message = "{6010.validator.orderDto.releaseMethodType.empty}")
        private ReleaseMethodType releaseMethodType;

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.NCP;
        }
    }

```

Файл «OrderDtoOtp»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		143

```

import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductOtp;
import
com.equiron.sitemanager.api.v2.model.validator.order.CheckReleaseMethodType;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@CheckReleaseMethodType
@OrderCisTypeComposition
@JsonTypeInfo(use= JsonTypeInfo.Id.CLASS, defaultImpl=OrderDtoOtp.class)
@XmlRootElement(name = "otp")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoOtp extends LightIndustryOrderDto {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductOtp> products = new ArrayList<>();

    @ApiModelProperty(example = "123456", value = "Production identifier
(Global Location Number) (Идентификатор производства (Глобальный номер места
нахождения))", required = true)
    @NotBlank(message = "{6010.validator.order.factoryId.empty}")
    @Size(min = 2, max = 128, message =
"{6040.validator.invalid.length.factoryId}")
    private String factoryId;

    @ApiModelProperty(example = "Some production", value = "Name of
production (Наименование производства)", required = false)
    @Size(min = 2, max = 256, message =
"{6040.validator.invalid.length.factoryName}")
    private String factoryName;

    @ApiModelProperty(example = "Uryupinsk", value = "Manufacturer's
address (Адрес производства)", required = false)
    @Size(min = 2, max = 256, message =
"{6040.validator.invalid.length.factoryAddress}")
    private String factoryAddress;

    @ApiModelProperty(example = "Uruguay", value = "Country of origin
(Страна производителя)", required = true)
    @NotBlank(message = "{6010.validator.order.factoryCountry.empty}")
    @Size(min = 2, max = 128, message =
"{6040.validator.invalid.length.factoryCountry}")
    private String factoryCountry;

```

```

        @ApiModelProperty(example = "123456", value = "Production line
identifier (Идентификатор производственной линии)", required = true)
        @NotBlank(message =
"{6010.validator.reportDto.productionLineId.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productionLineId}")
        private String productionLineId;

        @ApiModelProperty(example = "12-34-56", value = "Product code (SKU)
(Код продукта (SKU))", required = true)
        @NotBlank(message = "{6010.validator.order.productCode.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productCode}")
        private String productCode;

        @ApiModelProperty(example = "Description", value = "Product description
(Описание продукта)", required = true)
        @NotBlank(message = "{6010.validator.order.productDescription.empty}")
        @Size(min = 1, max = 512, message =
"{6040.validator.invalid.length.productDescription}")
        private String productDescription;

        @ApiModelProperty(example = "123456", value = "Production order number
(Номер производственного заказа)", required = false)
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.poNumber}")
        private String poNumber;

        @ApiModelProperty(example = "2020-01-01", value = "Start date for the
production of this order (Дата начала производства продукции по данному заказу)",
required = false)
        private String expectedStartDate;

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.ОТП;
        }
    }
}

```

Файл «OrderDtoPerfum»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		145

```

import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductPerfum;
import
com.equiron.sitemanager.api.v2.model.validator.order.CloseSetCisTypeForRemains;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@CloseSetCisTypeForRemains
@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoPerfum.class)
@XmlRootElement(name = "perfumery")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
public class OrderDtoPerfum extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElement(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductPerfum> products = new ArrayList<>();

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PERFUM;
    }
}

```

Файл «OrderDtoPetfood»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		146

```

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductPetfood;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoPetfood.class)
@XmlRootElement(name = "petfood")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoPetfood extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductPetfood> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PETFOOD;
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }
}

```

Файл «OrderDtoPharma»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						147
Изм.	Лист	№ докум.	Подпись	Дата		

```

import java.util.Optional;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductPharma;
import com.equiron.sitemanager.api.v2.model.validator.ValidUUID;
import
com.equiron.sitemanager.api.v2.model.validator.order.PaymentTypeRange;
import com.fasterxml.jackson.annotation.JsonGetter;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoPharma.class)
@XmlRootElement(name = "pharma")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoPharma extends OrderDtoRu {
    public static final boolean DEFAULT_FREE_CODE_VALUE = false;

    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 1, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductPharma> products = new ArrayList<>();

    @ApiModelProperty(example = "subjectId", value = "Subject ID (Субъект
обращения)", required = true)
    @NotNull(message = "{6010.validator.order.subjectId.empty}")
    @ValidUUID(message =
"{6050.validator.orderDto.subjectID.format.farma}")
    private String subjectId;

    @ApiModelProperty(example = "true", value = "Is code emission free
(Признак оплаты эмиссии КМ: true - не подлежит оплате, false - подлежит оплате
(значение по умолчанию))")
    private Boolean freeCode = null;

    // Legacy поле с русской буквой "С" оставлено для совместимости
    @ApiModelProperty(hidden = true)
    private Boolean freeCode = null;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		148

```

        @ApiModelProperty(example = "1", allowableValues = "1, 2", value =
"Payment Type (Тип оплаты: 1 - Оплата по эмиссии, 2 - Оплата по нанесению
(значение по умолчанию)")
        @PaymentTypeRange(min = 1, max = 2)
        private Integer paymentType = 2;

        private boolean getRuFreeCode() {
            return
Optional.ofNullable(freeCode).orElse(DEFAULT_FREE_CODE_VALUE);
        }

        private Boolean getEnFreeCode() {
            return freeCode;
        }

        @JsonGetter("freeCode")
        public boolean isFreeCode() {
            return
Optional.ofNullable(getEnFreeCode()).orElse(getRuFreeCode());
        }

        // Метод позволяет исключить поле с русской буквой при сериализации
        @JsonGetter("freeCode")
        public Boolean nullable() {
            return null;
        }

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.PHARMA;
        }
    }
}

```

Файл «OrderDtoPhoto»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductPhoto;
import
com.equiron.sitemanager.api.v2.model.validator.order.CheckPhotoCisTypeAndReleaseM
ethodType;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;

```

```

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@CheckPhotoCisTypeAndReleaseMethodType
@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoPhoto.class)
@XmlRootElement(name = "electronics")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
public class OrderDtoPhoto extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElement(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductPhoto> products = new ArrayList<>();

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.PHOTO;
    }
}

```

Файл «OrderDtoSeafood»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		150

```

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductSeafood;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoSeafood.class)
@XmlRootElement(name = "seafood")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoSeafood extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElement(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductSeafood> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.SEAFOOD;
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }
}

```

Файл «OrderDtoShoes»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		151

```

import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import
com.equiron.sitemanager.api.v2.model.validator.order.RemainsFinalDate;
import org.springframework.beans.BeanUtils;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductFashion;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductShoes;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@RemainsFinalDate(value = "2021-06-01", message =
"{3030.order.product.group.releaseMethodType.notAllowed.shoes}")
@JsonTypeInfo(use=Id.CLASS, defaultImpl= OrderDtoShoes.class)
@XmlRootElement(name = "shoes")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoShoes extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductShoes> products = new ArrayList<>();

    public OrderDtoShoes(OrderDtoLight orderDtoLight) {
        BeanUtils.copyProperties(orderDtoLight, this);

        for (OrderProductFashion f : orderDtoLight.getProducts()) {
            products.add(new OrderProductShoes(f));
        }
    }

    public OrderDtoShoes() {
        //empty
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }
}

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		152

```

@JsonIgnore
@Override
public boolean checkExporterTaxpayerIdForEmpty() {
    return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
}

@Override
public ProductGroupType getProductGroupType() {
    return ProductGroupType.SHOES;
}
}

```

Файл «OrderDtoSoftdrinks»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductSoftdrinks;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoSoftdrinks.class)
@XmlRootElement(name = "softdrinks")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoSoftdrinks extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
    @XmlElements(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductSoftdrinks> products = new ArrayList<>();

    @Override

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		153

```

    public ProductGroupType getProductGroupType() {
        return ProductGroupType.SOFTDRINKS;
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerId() {
        return OrderDtoUtils.checkExporterTaxpayerId(this);
    }

    @JsonIgnore
    @Override
    public boolean checkExporterTaxpayerIdForEmpty() {
        return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
    }
}

```

Файл «OrderDtoTires»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductTires;
import com.equiron.sitemanager.api.v2.model.validator.order.RemainsFinalDate;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@RemainsFinalDate(value = "2021-03-01", message =
"{3030.order.product.group.releaseMethodType.notAllowed.tires}")
@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoTires.class)
@XmlRootElement(name = "tires")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
public class OrderDtoTires extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
    @NotNull

```

```

        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductTires> products = new ArrayList<>();

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerId() {
            return OrderDtoUtils.checkExporterTaxpayerId(this);
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerIdForEmpty() {
            return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
        }

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.TIRES;
        }
    }
}

```

Файл «OrderDtoTobacco»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.enums.ReleaseMethodType;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductTobacco;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;
import lombok.Setter;

@JsonTypeInfo(use=Id.CLASS, defaultImpl=OrderDtoTobacco.class)
@XmlRootElement(name = "tobacco")
@XmlAccessorType (XmlAccessType.FIELD)
@Getter
@Setter
public class OrderDtoTobacco extends OrderDtoRu {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						155
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
        @NotNull
        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElement(@XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductTobacco> products = new ArrayList<>();

        @ApiModelProperty(example = "ftf_001", value = "Factory Identifier
(GLN) Идентификатор производства. (Глобальный номер места нахождения)", required
= true)
        @NotBlank(message = "{6010.validator.order.factoryId.empty}")
        @Size(min = 2, max = 128, message =
"{6040.validator.invalid.length.factoryId}")
        private String factoryId;

        @ApiModelProperty(example = "First Tobacco Factory", value = "Factory
Name (Наименование производства)")
        @Size(min = 2, max = 256, message =
"{6040.validator.invalid.length.factoryName}")
        private String factoryName;

        @ApiModelProperty(example = "ul. Pervaya Tabachnaya, 7", value =
"Factory Address (Адрес производства)")
        @Size(min = 2, max = 256, message =
"{6040.validator.invalid.length.factoryAddress}")
        private String factoryAddress;

        @ApiModelProperty(example = "RU", value = "Factory Country (Страна
производителя)", required = true)
        @NotBlank(message = "{6010.validator.order.factoryCountry.empty}")
        @Size(min = 2, max = 128, message =
"{6040.validator.invalid.length.factoryCountry}")
        private String factoryCountry;

        @ApiModelProperty(example = "\\123\\", value = "Line Identifier
(Идентификатор производственной линии)", required = true)
        @NotBlank(message =
"{6010.validator.reportDto.productionLineId.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productionLineId}")
        private String productionLineId;

        @ApiModelProperty(example = "ABCD123456", value = "Product Code (Код
продукта)", required = true)
        @NotBlank(message = "{6010.validator.order.productCode.empty}")
        @Size(min = 1, max = 256, message =
"{6040.validator.invalid.length.productCode}")
        private String productCode;

        @ApiModelProperty(example = "Cigarettes Friend", value = "Product
Description (Описание продукта)", required = true)
        @NotBlank(message = "{6010.validator.order.productDescription.empty}")
        @Size(min = 1, max = 512, message =
"{6040.validator.invalid.length.productDescription}")
        private String productDescription;

```

```

        @ApiModelProperty(example = "\"123456\"", value = "PO Number (Номер заказа)")
        @Size(min = 1, max = 256, message = "{6040.validator.invalid.length.poNumber}")
        private String poNumber;

        @ApiModelProperty(example = "2019-03-01", value = "Expected Start Date (Ожидаемая дата начала)")
        @Pattern(regex = "^\\d{4}\\-(0[1-9]|1[012])\\-(0[1-9]|[12][0-9]|3[01])$")
        private String expectedStartDate;

        @ApiModelProperty(example = "PRODUCTION", value = "Product Release Type (Способ выпуска товаров в оборот)", required = false)
        private ReleaseMethodType releaseMethodType;

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.TOBACCO;
        }
    }
}

```

Файл «OrderDtoToys»

```

package com.equiron.sitemanager.api.v2.model.order;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductToys;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;
import java.util.ArrayList;
import java.util.List;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl = OrderDtoToys.class)
@XmlRootElement(name = "toys")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoToys extends LightIndustryOrderDto implements ValidateExporterTaxpayerId {
    @ApiModelProperty(example = "", value = "Product list (Список товаров)", required = true)
    @NotNull

```

```

        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductToys> products = new ArrayList<>();

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.TOYS;
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerId() {
            return OrderDtoUtils.checkExporterTaxpayerId(this);
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerIdForEmpty() {
            return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
        }
    }
}

```

Файл «OrderDtoVegs»

```

package com.equiron.sitemanager.api.v2.model.order;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductVegs;
import
com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;
import java.util.ArrayList;
import java.util.List;

@OrderCisTypeComposition
@JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoVegs.class)
@XmlRootElement(name = "vegs")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoVegs extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		158

```

        @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
        @NotNull
        @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
        @XmlElement(name = "product", required = false))
        @XmlElementWrapper(required = true)
        private List<OrderProductVegs> products = new ArrayList<>();

        @Override
        public ProductGroupType getProductGroupType() {
            return ProductGroupType.VEGS;
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerId() {
            return OrderDtoUtils.checkExporterTaxpayerId(this);
        }

        @JsonIgnore
        @Override
        public boolean checkExporterTaxpayerIdForEmpty() {
            return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
        }
    }
}

```

Файл «OrderDtoWater»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.util.OrderDtoUtils;
import com.equiron.sitemanager.api.v2.ProductGroupType;
import com.equiron.sitemanager.api.v2.ValidateExporterTaxpayerId;
import com.equiron.sitemanager.api.v2.model.order.product.OrderProductWater;
import com.equiron.sitemanager.api.v2.model.validator.order.OrderCisTypeComposition;
import com.equiron.sitemanager.api.v2.model.validator.order.PaymentTypeRange;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@OrderCisTypeComposition

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		159

```

        @JsonTypeInfo(use = JsonTypeInfo.Id.CLASS, defaultImpl =
OrderDtoWater.class)
        @XmlRootElement(name = "water")
        @XmlAccessorType(XmlAccessType.FIELD)
        @Getter
        public class OrderDtoWater extends LightIndustryOrderDto implements
ValidateExporterTaxpayerId {
            @ApiModelProperty(example = "", value = "Product list (Список
товаров)", required = true)
            @NotNull
            @Size(min = 1, max = 10, message =
"{3100.validator.orderDto.products.count}")
            @XmlElement(@XmlElement(name = "product", required = false))
            @XmlElementWrapper(required = true)
            private List<OrderProductWater> products = new ArrayList<>();

            @ApiModelProperty(example = "1", allowableValues = "1, 2", value =
"Payment Type (Тип оплаты: 1 - Оплата по эмиссии, 2 - Оплата по нанесению
(значение по умолчанию)")
            @PaymentTypeRange(min = 1, max = 2)
            private Integer paymentType = 2;

            @Override
            public ProductGroupType getProductGroupType() {
                return ProductGroupType.WATER;
            }

            @JsonIgnore
            @Override
            public boolean checkExporterTaxpayerId() {
                return OrderDtoUtils.checkExporterTaxpayerId(this);
            }

            @JsonIgnore
            @Override
            public boolean checkExporterTaxpayerIdForEmpty() {
                return OrderDtoUtils.checkExporterTaxpayerIdForEmpty(this);
            }
        }

```

Файл «OrderDtoWheelchairs»

```

package com.equiron.sitemanager.api.v2.model.order;

import java.util.ArrayList;
import java.util.List;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlElements;
import javax.xml.bind.annotation.XmlRootElement;

import com.equiron.sitemanager.api.v2.ProductGroupType;
import
com.equiron.sitemanager.api.v2.model.order.product.OrderProductWheelchairs;
import com.fasterxml.jackson.annotation.JsonTypeInfo;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						160
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.fasterxml.jackson.annotation.JsonTypeInfo.Id;

import io.swagger.annotations.ApiModelProperty;
import lombok.Getter;

@JsonTypeInfo(use = Id.CLASS, defaultImpl = OrderDtoWheelchairs.class)
@XmlRootElement(name = "wheelchairs")
@XmlAccessorType(XmlAccessType.FIELD)
@Getter
public class OrderDtoWheelchairs extends LightIndustryOrderDto {
    @ApiModelProperty(example = "", value = "Product list (Список товаров)", required = true)
    @NotNull
    @Size(min = 1, max = 10, message = "{3100.validator.orderDto.products.count}")
    @XmlElement(@XmlElement(name = "product", required = false))
    @XmlElementWrapper(required = true)
    private List<OrderProductWheelchairs> products = new ArrayList<>();

    @Override
    public ProductGroupType getProductGroupType() {
        return ProductGroupType.WHEELCHAIRS;
    }
}

```

Файл «OmsApiAntiseptic»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoAntiseptic;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoAntiseptic;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoAntiseptic;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoAntiseptic;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		161

```

    @Api(tags = {"API.v2.ANTISEPTIC"}, description = "Order Management Station
API v.2 for ANTISEPTIC group")
    @RequestMapping("/api/v2/antiseptic")
    @RestController
    @Validated
    public interface OmsApiAntiseptic extends OmsApi {
        @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = {"/orders"}, produces = "application/json")
        OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoAntiseptic orderDto,
            @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

        @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = {"/dropout"}, produces = "application/json")
        ReportResult sendDroppedReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
            @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoAntiseptic
dropoutReportDto,
            @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

        @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
        @ApiResponses({

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		162

```

        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoAntiseptic aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (КМ агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						163
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoAntiseptic
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
}

```

Файл «OmsApiBeer»

```

package com.equiron.sitemanager.api.v2;

import java.util.List;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.ProvidersDto;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoBeer;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import
com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoBeer;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoBeer;
import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoBeer;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		164

```

import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.BEER"}, description = "Order Management Station API
v.2 for BEER group")
@RequestMapping("/api/v2/beer")
@RestController
@Validated
public interface OmsApiBeer extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoBeer orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/dropout"}, produces = "application/json")
    ReportResult sendDroppedReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
        @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoBeer dropoutReportDto,
        @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.36.001 ПЗ

Лист

165

```

        @ApiResponse({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = "/aggregation", produces = "application/json")
        ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
            @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoBeer aggregationReportDto,
            @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

        @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @GetMapping(value = "/aggregation/info", produces = "application/json")
        AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Aggregation unit
number (КМ агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
            @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

        @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		166

```

        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoBeer
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Получение информации из профиля УОТ о местах
осуществления деятельности")
    @ApiResponses({
        @ApiResponse(code = 200, response = ProvidersDto.class, message
= "Information on UOT service providers (Информация о местах осуществления
деятельности УОТ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "A technical error has occurred, please try again later (Возникла
техническая ошибка, необходимо повторить попытку позже)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping("/mod")
    public List<Object> getMod(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception ;
}

```

Файл «OmsApiBicycle»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		167

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoBicycle;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoBicycle;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.BICYCLE"},description = "Order Management Station API
v.2 for BICYCLE group")
@RequestMapping("/api/v2/bicycle")
@RestController
@Validated
public interface OmsApiBicycle extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoBicycle orderDtoBicycle,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)", hidden = true)
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })

```

```

    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                     @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
                                     @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoBicycle
utilisationReportDto,
                                     @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

}

```

Файл «OmsApiBio»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoBio;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoBio;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoBio;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoBio;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.BIO"}, description = "Order Management Station API v.2
for BIO group")
@RequestMapping("/api/v2/bio")
@RestController
@Validated
public interface OmsApiBio extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		169

```

        @ApiResponse({
            @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = {"/orders"}, produces = "application/json")
        OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoBio orderDto,
            @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

        @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
        @ApiResponse({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = {"/dropout"}, produces = "application/json")
        ReportResult sendDroppedReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
            @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoBio dropoutReportDto,
            @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

        @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
        @ApiResponse({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		170

```

        @PostMapping(value = "/aggregation", produces = "application/json")
        ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                           @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
                                           @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoBio aggregationReportDto,
                                           @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

        @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @GetMapping(value = "/aggregation/info", produces = "application/json")
        AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                           @ApiParam(value = "Aggregation unit
number (КМ агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
                                           @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

        @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
        @ApiResponses({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = "/utilisation", produces = "application/json")
        ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		171

```

        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,

        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}")
UtilisationReportDtoBio
utilisationReportDto,

        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    }

```

Файл «OmsApiLp»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoLp;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoLp;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoLp;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoLp;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import springfox.documentation.annotations.ApiIgnore;

@Api(tags = {"API.v2.LP"},description = "Order Management Station API v.2
for LP group")
@RequestMapping("/api/v2/lp")
@RestController
@Validated
public interface OmsApiLp extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						172
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoLp orderDtoLp,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoLp aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)" ),
    })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						173
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (KM arperata)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Отчет о
нанесении KM) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о нанесении)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoLp
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiIgnore
    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке KM) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/dropout", produces = "application/json")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						174
Изм.	Лист	№ докум.	Подпись	Дата		

```

        ReportResult sendDropoutReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                     @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
                                     @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoLp dropoutReportDto,
                                     @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

}

```

Файл «OmsApiMeat»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoMeat;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoMeat;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.MEAT"}, description = "Order Management Station API
v.2 for MEAT group")
@RequestMapping("/api/v2/meat")
@RestController
@Validated
public interface OmsApiMeat extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)" ),
    })

```

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoMeat orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoMeat
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

}

```

Файл «OmsApiMilk»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		176

```

import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import
com.equiron.sitemanager.api.v2.model.DistributionTransactionCodesResultDto;
import com.equiron.sitemanager.api.v2.model.UtilisationReportIdsResultDto;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoMilk;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import
com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoMilk;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoMilk;
import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoMilk;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.MILK"},description = "Order Management Station API v.2
for MILK group")
@RequestMapping("/api/v2/milk")
@RestController
@Validated
public interface OmsApiMilk extends OmsApi {

    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)" ),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)" )
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoMilk orderDtoMilk,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)" ),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						177
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = {"/dropout"}, produces = "application/json")
    ReportResult sendDroppedReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
        @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoMilk dropoutReportDto,
        @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoMilk aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),

```

```

        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                     @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
                                     @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoMilk
utilisationReportDto,
                                     @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации KM)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)" ),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)" ),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                     @ApiParam(value = "Aggregation unit
number (KM агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
                                     @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

}

```

Файл «OmsApiNabeer»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						179
Изм.	Лист	№ докум.	Подпись	Дата		

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoNabeer;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoNabeer;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.NABEER"}, description = "Order Management Station API
v.2 for NABEER group")
@RequestMapping("/api/v2/nabeer")
@RestController
@Validated
public interface OmsApiNabeer extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoNabeer
orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
    })

```

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoNabeer
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

}

```

Файл «OmsApiNcp»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoNcp;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoNcp;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoNcp;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoNcp;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.NCP"},description = "Order Management Station API v.2
for NCP group")
@RequestMapping("/api/v2/ncp")
@RestController

```

					ПГУ 1.090301.36.001 ПЗ	Лист 181
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Validated
public interface OmsApiNcp extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoNcp orderDtoNcp,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/dropout"}, produces = "application/json")
    ReportResult sendDropoutReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
        @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoNcp dropoutReportDto,
        @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		182

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoNcp aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)" ),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}")
UtilisationReportDtoNcp
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)" ),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						183
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (KM агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
    }

```

Файл «OmsApiOtp»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoOtp;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoOtp;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoOtp;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoOtp;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.OTP"}, description = "Order Management Station API v.2
for OTP group")
@RequestMapping("/api/v2/otp")
@RestController
@Validated
public interface OmsApiOtp extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		184

```

        @ApiResponse({
            @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = {"/orders"}, produces = "application/json")
        OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoOtp orderDto,
            @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

        @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации KM)")
        @ApiResponse({
            @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
            @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
        })
        @PostMapping(value = "/aggregation", produces = "application/json")
        ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
            @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
            @ApiParam(value = "Aggregation Data
(Сведения об агрегации KM)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoOtp aggregationReportDto,
            @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

        @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации KM)")
        @ApiResponse({
            @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
            @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),

```

```

        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (KM агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoOtp
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		186

```

    })
    @PostMapping(value = {"/dropout"}, produces = "application/json")
    ReportResult sendDropoutReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
                                @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoOtp dropoutReportDto,
                                @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;
    }

```

Файл «OmsApiPerfum»

```

package com.equiron.sitemanager.api.v2;

import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoPerfum;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoPerfum;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import
com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoPerfum;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.PERFUM"},description = "Order Management Station API
v.2 for PERFUM group")
@RequestMapping("/api/v2/perfum")
@RestController
@Validated
public interface OmsApiPerfum extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
    })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						187
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoPerfum
orderDtoPerfum,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации KM)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации KM)", required = true) @RequestBody(required = false)
@Validated @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoPerfum aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации KM)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)")
    })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						188
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (KM arperata)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Отчет о
нанесении KM) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о нанесении)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoPerfum
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
}

```

Файл «OmsApiPetfood»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		189

```

import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoPetfood;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoPetfood;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.PETFOOD"}, description = "Order Management Station API
v.2 for PETFOOD group")
@RequestMapping("/api/v2/petfood")
@RestController
@Validated
public interface OmsApiPetfood extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoPetfood
orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		190

```

        ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                           @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
                                           @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoPetfood
utilisationReportDto,
                                           @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
    }

```

Файл «OmsApiPharma»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoPharma;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoPharma;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.PHARMA"},description = "Order Management Station API
v.2 for PHARMA group")
@RequestMapping("/api/v2/pharma")
@RestController
@Validated
public interface OmsApiPharma extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
    })

```

```

        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoPharma orderDtoPharma,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoPharma
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
    }
}

```

Файл «OmsApiPhoto»

```

package com.equiron.sitemanager.api.v2;

import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoPhoto;
import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		192

```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoPhoto;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import
com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoPhoto;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.PHOTO"},description = "Order Management Station API
v.2 for PHOTO group")
@RequestMapping("/api/v2/photo")
@RestController
@Validated
public interface OmsApiPhoto extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)" ),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)" )
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoPhoto orderDtoPhoto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)" ),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)" ),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		193

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoPhoto aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции) "),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры) "),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден) "),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера) ")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (КМ агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Отчет о
нанесении КМ) ")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о нанесении) "),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры) "),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера) ")
    })

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		194

```

        @PostMapping(value = "/utilisation", produces = "application/json")
        ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
                                           @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
                                           @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoPhoto
utilisationReportDto,
                                           @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;
    }

```

Файл «OmsApiSeafood»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoSeafood;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoSeafood;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.SEAFOOD"}, description = "Order Management Station API
v.2 for SEAFOOD group")
@RequestMapping("/api/v2/seafood")
@RestController
@Validated
public interface OmsApiSeafood extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
    })

```

```

        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoSeafood
orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoSeafood
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

}

```

Файл «OmsApiShoes»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import org.springframework.validation.annotation.Validated;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		196

```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderDtoShoes;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;
import com.equiron.sitemanager.api.v2.model.report.AggregationInfo;
import
com.equiron.sitemanager.api.v2.model.report.AggregationReportDtoShoes;
import com.equiron.sitemanager.api.v2.model.report.DropoutReportDtoShoes;
import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoShoes;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import springfox.documentation.annotations.ApiIgnore;

@Api(tags = {"API.v2.SHOES"},description = "Order Management Station API
v.2 for SHOES group")
@RequestMapping("/api/v2/shoes")
@RestController
@Validated
public interface OmsApiShoes extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class, message =
"Request failed. Incorrect parameters. (Операция не выполнена. Неверные входные
параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class, message =
"Request failed. Internal Server Error. (Операция не выполнена. Внутренняя ошибка
сервера)"))
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") OrderDtoShoes orderDtoShoes,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Aggregation Report (Отчет об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об агрегации)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		197

```

        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/aggregation", produces = "application/json")
    ReportResult sendAggregationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "Aggregation Data
(Сведения об агрегации КМ)", required = true) @RequestBody(required = false)
@Valid @NotNull(message = "{6240.validation.request.boby.empty}")
AggregationReportDtoShoes aggregationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Get aggregation info (Получение информации об
агрегации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Operation Status (Статус
операции)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 404, response = OmsRestResult.class,
message = "Request failed. Aggregation unit not found. (Операция не выполнена.
Агрегат не найден)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @GetMapping(value = "/aggregation/info", produces = "application/json")
    AggregationInfo getAggregationInfo(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Aggregation unit
number (КМ агрегата)", required = true) @RequestParam(required = true) @NotBlank
String unitSerialNumber,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Отчет о
нанесении КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о нанесении)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
    })

```

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные KM)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoShoes
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    @ApiIgnore
    @ApiOperation(value = "Identification Code Dropout Report (Передача
отчета о выбытии/отбраковке KM)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета о выбытии/отбраковке в СУЗ)",
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)",
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/dropout", produces = "application/json")
    ReportResult sendDropoutReport(@ApiParam(value = "Authorization token
(Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя пользователя",
required = false) @RequestHeader(defaultValue = "API") String userName,
        @ApiParam(value = "Identification code
dropout report (Отчет о выбытии/отбраковке)", required = true)
@RequestBody(required = false) @Valid @NotNull(message =
"{6240.validation.request.boby.empty}") DropoutReportDtoShoes dropoutReportDto,
        @ApiParam(value = "OMS Id (Идентификатор
СУЗ)", required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

}

```

Файл «OmsApiSoftdrinks»

```

package com.equiron.sitemanager.api.v2;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import com.equiron.sitemanager.api.v2.model.order.OrderDtoSoftdrinks;

```

					ПГУ 1.090301.36.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		199

```

import
com.equiron.sitemanager.api.v2.model.report.UtilisationReportDtoSoftdrinks;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.equiron.emi.common.controller.advice.OmsRestResult;
import com.equiron.emi.model.ReportResult;
import com.equiron.sitemanager.api.v2.model.order.OrderResult;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@Api(tags = {"API.v2.SOFTDRINKS"}, description = "Order Management Station
API v.2 for SOFTDRINKS group")
@RequestMapping("/api/v2/softdrinks")
@RestController
@Validated
public interface OmsApiSoftdrinks extends OmsApi {
    @ApiOperation(value = "Create Order (Создание бизнес-заказа)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Request result (Результат
создания бизнес-заказа)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),
        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = {"/orders"}, produces = "application/json")
    OrderResult createOrder(@ApiParam(value = "Authorization token (Токен
авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Order Description (Бизнес-
заказ)", required = true) @RequestBody(required = false) @Validated
@NotNull(message = "{6240.validation.request.boby.empty}") OrderDtoSoftdrinks
orderDto,
        @ApiParam(value = "OMS Id (Идентификатор СУЗ)",
required = true) @RequestParam(required = false) @NotBlank(message =
"{1110.enterprise.profile.empty.omsId}") String omsId) throws Exception;

    @ApiOperation(value = "Identification Code Utilisation Report (Передача
отчета об утилизации КМ)")
    @ApiResponses({
        @ApiResponse(code = 200, message = "Unique OMS Report ID
(Идентификатор отчета об нанесении КМ)"),
        @ApiResponse(code = 400, response = OmsRestResult.class,
message = "Request failed. Incorrect parameters. (Операция не выполнена. Неверные
входные параметры)"),

```

					ПГУ 1.090301.36.001 ПЗ	Лист
						200
Изм.	Лист	№ докум.	Подпись	Дата		

```

        @ApiResponse(code = 500, response = OmsRestResult.class,
message = "Request failed. Internal Server Error. (Операция не выполнена.
Внутренняя ошибка сервера)")
    })
    @PostMapping(value = "/utilisation", produces = "application/json")
    ReportResult sendUtilisationReport(@ApiParam(value = "Authorization
token (Токен авторизации)", required = true) @RequestHeader @NotBlank(message =
"{1090.clientToken.isEmpty}") String clientToken,
        @ApiParam(value = "Имя
пользователя", required = false) @RequestHeader(defaultValue = "API") String
userName,
        @ApiParam(value = "List of the
utilized Identification Codes (Нанесенные КМ)", required = true)
@RequestBody(required = false) @Validated @NotNull(message =
"{6240.validation.request.boby.empty}") UtilisationReportDtoSoftdrinks
utilisationReportDto,
        @ApiParam(value = "OMS Id
(Идентификатор СУЗ)", required = true) @RequestParam(required = false)
@NotBlank(message = "{1110.enterprise.profile.empty.omsId}") String omsId) throws
Exception;

    }

```

Приложение В

(обязательное)

Программный компонент эмиссии и учета кодов маркировки

Презентация

					ПГУ 1.090301.36.001 ПЗ	Лист
						202
Изм.	Лист	№ докум.	Подпись	Дата		



Программный компонент эмиссии и учета кодов маркировки



центр развития
перспективных технологий

Автор работы:
студент гр. 18ВВ1 Свищев Данил Сергеевич
Научный руководитель:
к.т.н., доцент Карамышева Надежда Сергеевна

Содержание

1. Цель и объект работы
2. Актуальность работы
3. Техническое задание
4. Стек-технологий
5. Проектирование и разработка
6. Результаты работы
7. Подведение итогов

1

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.36.001 ПЗ

Лист

203

Цели и задачи создания программного компонента эмиссии и учета кодов маркировки



Цифровая платформа

- **Сервис и услуги** (лизинг, факторинг, страхование и т.д.)
- **Открытые данные и интерфейсы** (оценка рисков, проверка контрагентов, мониторинг рынка и т.д.)
- **Новые возможности для бизнеса** (подключение финансовых, транспортных, страховых, сервисных компаний и их услуг)

Бизнес

Повышение конкурентоспособности "белых" участников рынка за счет сокращения доли незаконного оборота

Доступ к данным о движении продукции по логистической цепи в режиме реального времени



Потребитель

- Защита жизни и здоровья граждан
- Общественный контроль
- Защита прав потребителей

Государство

Повышение собираемости таможенных и налоговых, в т.ч. акцизных, платежей

Экономия затрат бюджета на контроль товарных рынков

Статистика в масштабе всей страны в режиме реального времени

3

Актуальность работы



Внедрение программного компонента в систему маркировки и прослеживаемости товаров создает предпосылки для экономического роста в России и в ЕАЭС



- Позволяет избежать "арбитража" регулирования между странами и ликвидирует большую часть "серого" оборота
- Повышает прозрачность торговых потоков внутри ЕАЭС
- Позволяет координировать деятельность надзорных органов стран-участниц ЕАЭС
- Способствует росту товарооборота за счет унификации регулирования и информации о товарах в едином каталоге

- **Для участников рынка:** позволяет сократить издержки на персонал, управление запасами, логистику, обмен и возврат товаров на 6-7% в каждой отрасли
- **Для государства:** позволяет кардинально повысить эффективность контрольно-надзорной деятельности

Сокращение доли контрафактной продукции на 50-60%, что приводит к росту выручки легальных производителей:

- от 50-100 млрд руб. на отрасль (при доле контрафакта 5-10%)
- до 200-250 млрд руб. (при доле контрафакта 45-50%)

При маркировке >10 товарных групп суммарный эффект > 1 трлн руб.

Рост собираемости акцизов на 20-40% за 1-2 года после внедрения (26-40 млрд руб. только по табачным изделиям)

5

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.36.001 ПЗ

Лист

204



Спасибо за внимание!



центр развития
перспективных технологий