МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Информационная автоматизированная система обеспечения логистики торгового предприятия

Студент 20.06.19 (подпись, дата) А.А.Чуваев (ФИО полностью)

Руководитель 21.06.19 (подпись, дата) С.А. Зинкин (фамилия, инициалы)

Нормоконтролёр 21.06.19 (подпись, дата) А.В. Кучин (фамилия, инициалы)

Рецензент ведущий специалист ООО «Бизнес-программа» 21.06.19 (должность, место работы) (подпись, дата) А.А. Севастьянов (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.2019 г. № 38-09-14)

Заведующий кафедрой (подпись) М.А. Митрохин (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 05.07.2019 г. № 1)

Секретарь ГЭК Е.И. Гурин

Пенза 2019

ОТЗЫВ

на выпускную квалификационную работу специалиста

Пензенского государственного университета

Чуваева Александра Андреевича

Тема работы: "Информационная автоматизированная система обеспечения логистики торгового предприятия "

Дипломная работа посвящена разработке информационной автоматизированной системы обеспечения логистики торгового предприятия.

Актуальность темы заключается в том, что на данный момент большинство предприятий старается автоматизировать свой рабочий процесс. Программное решение проблемы позволяет упростить предприятиям работу с логистической частью, затрачивать меньше времени на обработку заказов, и таким способом увеличить товарооборот предприятия.

В процессе проектирования проведено моделирование базы данных в системе управления базами данных MySQL HeidiSQL, проведена разработка серверного и клиентского приложений на языке PHP в среде NetBeans.

Чуваев А.А. справился с поставленной задачей. Он проявил самостоятельность при выполнении дипломной работы. Дипломный проект полностью соответствует заданию на проектирование. В процессе работы Чуваев А.А. показал хорошие знания по специальным дисциплинам, проявил себя сформировавшимся специалистом, способным самостоятельно решать инженерные задачи. Он хорошо владеет современными средствами разработки программного обеспечения.

Считаю, что работа соответствует требованиям к дипломным работам специалистов, заслуживает оценки "отлично", а Чуваев Александр Андреевич - присвоения квалификации специалиста по направлению 09.05.01 "Применение и эксплуатация автоматизированных систем специального назначения."

Руководитель проекта

д.т.н., профессор каф. ВТ С.А. Зинкин

21.06.19

Р Е Ц Е Н З И Я

на дипломный проект студента

Пензенского государственного университета

Чуваева Александра Андреевича

Тема проекта: **“ Информационная автоматизированная система обеспечения логистики торгового предприятия “**

Проект Чуваева Александра Андреевича посвящен разработке информационной автоматизированной системы для обеспечения логистики торгового предприятия.

Актуальность данного проекта заключается в том, что на данный момент большинство предприятий старается автоматизировать свой рабочий процесс. Программное решение проблемы позволяет упростить предприятиям работу с логистической частью, затрачивать меньше времени на обработку заказов, и таким способом увеличить товарооборот предприятия.

Дипломный проект полностью соответствуют заданию на дипломное проектирование. Пояснительная записка и графическая часть выполнены грамотно и с высоким качеством.

По материалам дипломного проекта, очевидно, что Чуваев А.А. умеет решать инженерные задачи, владеет реальными практическими навыками разработки ПО на языке PHP.

Материалы дипломного проекта имеют практическую значимость и, могут быть использованы в различных торговых предприятиях.

Считаю, что работа соответствует требованиям к дипломным работам специалистов, установленных государственным образовательным стандартом, заслуживает оценки "отлично", а Чуваев Александр Андреевич - присвоения квалификации специалиста по направлению 09.05.01 "Применение и эксплуатация автоматизированных систем специального назначения."

Рецензент

К.т.н., гл. специалист

ООО «Бизнес-программа» _____ Севастьянов А.В.

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«25» апреля 2019 г

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Чуваев А.А. гр. 14ВО2 факультета Вычислительной техники

2.Тема работы: Разработка информационной автоматизированной системы обеспечия логистики торгового предприятия

Тема утверждена приказом ПГУ № 554/0 от "29" 04 2019 г.

3. Руководитель работы Зинкин Сергей Александрович

4. Задание на работу (назначение разработки, исходные данные и т.п.) В данной работе поставлена задача разработать програмное обеспечение, позволяющее автоматизировать логистику торгового предприятия. Программа должена предоставлять пользователям интуитивно понятный интерфейс, чтобы поиск информации был максимально удобным. Для администратора должны быть реализованы возможности добавления новых пользователей на странице программы, а также вносить изменения в базу данных из внешних источников и создвать бэкап базы данных. Также необходимо защитить эти страницы от доступа со стороны простых пользователей. Для менеджеров предприятия должна быть создана специальная страница, которая будет позволять им осуществлять учёт поставок, учёт хранения товара, учёт отправки, а также учёт всех операций, связанных с логистикой торгового предприятия.

5. Перечень подлежащих разработке вопросов Разработка базы данных онлайн сервиса, разработка логической модели базы данных, разработка физической модели базы данных, разработка програмного обеспечения, разработка и анализ позьвательского интерфейса, разработка програмного обеспечения клиентской части, разработка програмного обеспечения серверной части.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Выбор темы и разработка индивидуального плана проведения работ	10	11.02.19-12.02.19	
2	Подбор и изучение материала по теме работы	464	12.02.19-17.04.19	
3	Получение и анализ результатов	300	17.04.19-25.05.19	
4	Оформление отчёта	90	02.06.19	
	Общий объём часов	864		

Дата выдачи задания "12" 02 2019 г.

Руководитель работы _____ / Зинкин С.А. / 12.02.19
(подпись, дата)

Задание к исполнению принял студент _____ / Чуваев 12.02.19
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 28.06.2019
(подпись, дата)

Заведующему кафедрой

"Вычислительная техника"

от студента группы 14ВО2

Чуваева Александра Андреевича

Заявление

Прошу утвердить в качестве руководителя ВКР: Зинкина С.А., доктора технических наук профессора кафедры вычислительная техника, Пензенского государственного университета.

Руководитель - тема ВКР: Информационная автоматизированная система обеспечения логистики торгового предприятия.

Дата: 23.04.2019 г.

Руководитель_____
(подпись)

Студент_____
(подпись)

Ректору Пензенского
государственного университета
А. Д. Гулякову
студента 5 курса ФВТ очной
формы обучения
Чуваева Александра
Андреевича

ЗАЯВЛЕНИЕ

Я, Чуваев Александр Андреевич, студент 5 курса очной формы обучения направления подготовки 09.05.01 Применение и эксплуатация автоматизированных систем специального назначения, прошу провести проверку с использованием системы «Антиплагиат» работы специалиста на тему «Информационная автоматизированная система обеспечения логистики торгового предприятия», выполненной мной самостоятельно, на содержание элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

Подпись _____ «4» 06 2019 г.

ВКР представлена на проверку «4» 06 2019 г.

Руководитель ВКР _____ С.А. Зинкин

Протокол

проверки на оригинальность в системе «АНТИПЛАГИАТ» (https://www.antiplagiat.ru) выпускной квалификационной работы на тему «Информационная автоматизированная система обеспечения логистики торгового предприятия» студента 5 курса Чуваева А.А. направления подготовки/специальности 09.05.01, Применение и эксплуатация автоматизированных систем специального назначения.

Руководитель ВКР д.т.н., профессор кафедры «ВТ» Зинкин С.А.

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: Зинкин Сергей Александрович frn_penza@mail.ru / ID: 253
Проверяющий: Зинкин Сергей Александрович (frn_penza@mail.ru / ID: 253)
Организация: Пензенский государственный университет

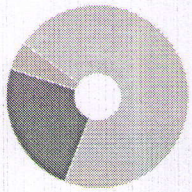
Отчет предоставлен сервисом «Антиплагиат» - http://pnzgu.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 15
Начало загрузки: 06.06.2019 10:47:50
Длительность загрузки: 00:00:36
Имя исходного файла:
pz_chuvaev_zinkin_2019
Размер текста: 2154 кБ
Тип документа: Выпускная квалификационная работа
Символов в тексте: 176288
Слов в тексте: 20408
Число предложений: 1187

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 06.06.2019 10:48:29
Длительность проверки: 00:00:16
Корректировка от 06.06.2019 10:50:04
Комментарии: [Автосохраненная версия]
Модули поиска: Модуль поиска ИПС "Адилет", Модуль выделения библиографических записей, Сводная коллекция ЭБС, Коллекция РГБ, Цитирование, Модуль поиска переводных заимствований, Коллекция eLIBRARY.RU, Коллекция ГАРАНТ, Модуль поиска Интернет, Коллекция Медицина, Модуль поиска перефразирований eLIBRARY.RU, Модуль поиска перефразирований Интернет, Коллекция Патенты, Модуль поиска "ПГУ", Модуль поиска общеупотребительных выражений, Кольцо вузов



ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
24,31%	5,35%	70,34%

Все заимствования представляют собой цитаты из литературных источников, все ссылки сделаны корректно.

«6» 06 2019 г. _____ Зинкин С.А.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Информационная автоматизированная система обеспечения логистики торгового предприятия

Студент _____ (подпись, дата) А.А.Чуваев (ФИО полностью)

Руководитель _____ (подпись, дата) С.А. Зинкин (фамилия, инициалы)

Нормоконтролёр _____ (подпись, дата) А.В. Кучин (фамилия, инициалы)

Рецензент _____ (должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от _____г. № _____)

Заведующий кафедрой _____ (подпись) М.А. Митрохин (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№_____)

Секретарь ГЭК _____ Е.И. Гурин

Пенза 2019

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«____»_____20____г
.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент__Чуваев А.А.___гр.__14ВО2_____факультета_____

Вычислительной техники____

2.Тема работы: Разработка информационной автоматизированной системы обеспечия логистики торгового предприятия_____

Тема утверждена приказом ПГУ №_____от "___"_____20_____г.

3. Руководитель работы_____Зинкин Сергей Александрович_____

4. Задание на работу (назначение разработки, исходные данные и т.п.) В данной работе поставлена задача разработать програмное обеспечение, позволяющее автоматизировать логистику торгового предприятия. Программа должена предоставлять пользователям интуитивно понятный интерфейс, чтобы поиск информации был максимально удобным. Для администратора должны быть реализованы возможности добавления новых пользователей на странице программы, а также вносить изменения в базу данных из внешних источников и создвать бэкап базы данных. Также необходимо защитить эти страницы от доступа со стороны простых пользователей. Для менеджеров предприятия должна быть создана специальная страница, которая будет позволять им осуществлять учёт поставок, учёт хранения товара, учёт отправки, а также учёт всех операций, связанных с логистикой торгового предприятия.

5. Перечень подлежащих разработке вопросов Разработка базы данных онлайн сервиса, разработка логической модели базы данных, разработка физической модели базы данных, разработка програмного обеспечения, разработка и анализ позьвательского интерфейса, разработка програмного обеспечения клиентской части, разработка програмного обеспечения серверной части.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Выбор темы и разработка индивидуального плана проведения работ	10	11.02.19-12.02.19	
2	Подбор и изучение материала по теме работы	464	12.02.19-17.04.19	
3	Получение и анализ результатов	300	17.04.19-25.05.19	
4	Оформление отчёта	90	02.06.19	
	Общий объём часов	864		

Дата выдачи задания "_____"_____20_____г.

Руководитель работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 172 листа, 41 рисунок, 23 таблиц, 6 источников, 3 приложений.

СИСТЕМА АВТОМАТИЗАЦИИ ТОРГОВЛИ, БАЗА ДАННЫХ, REACT.

Объектом исследования является система автоматизации логистики торгового предприятия.

Цель работы – разработка программного обеспечения, позволяющего автоматизировать логистику торгового предприятия.

Разработка велась на языке PHP.

В процессе разработки был проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов, анализ аналогичных систем, изучены способы реализации WEB проектов.

В результате выполнения дипломного проекта было разработано программное обеспечение, осуществляющее учёт поставок, хранения товара, отправки и всех операций, связанных с логистикой торгового предприятия. Осуществлено функциональное тестирование разработанного приложения, которое показало корректность его работы.

В проектной части описаны выполненные функции приложения.

В технологической части описана технология разработки и отладки приложения.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ, а также рассмотрено воздействие работы с ПК на зрение человека.

Графическая часть содержит схемы всей системы, данных, структуры базы данных, алгоритма работы программных модулей.

Содержание

Введение 7

1 Предпроектный анализ 10

1.2 Стандарты оценки качества программного обеспечения 14

1.5. Анализ существующих автоматизированных систем для обеспечения логистики торгового предприятия 23

АСТ «Парус» 24

АСТ «1С Торговля» 26

АСТ «АВЕРС» 28

АСТ «Террасофт» 30

АСТ «Галактика» 31

2 Разработка автоматизированной системы для обеспечения логистики предприятия 36

2.1 Формализация задачи 36

2.2 Архитектура программно-аппаратного комплекса 43

2.3 Разработка структуры данных 44

2.4 Разработка программной части 51

2.4 Разработка модели 58

2.5 Формирование шаблонов страниц пользовательского интерфейса на основе модели GOMS 62

2.6 Описание работы приложения 69

3 Тестирование и развертка программного обеспечения 76

3.1 Создание проекта в NetBeans 76

3.2 Отладка кода с помощью XDebug 80

3.2.1 Как работает отладка 81

4 Технико-экономическое обоснование разработки программного обеспечения дипломного проекта 84
4.1 Организация создания программного обеспечения 84
4.2 Расчет цены программного обеспечения 84
4.2.1 Определение затрат на разработку программного обеспечения 85
4.2.2 Установление цены программного обеспечения 88
4.2.3 Установление нижнего предела цены с учетом тиражирования 89
5 Безопасность жизнедеятельности 90
Заключение 108
Список использованных источников 109
КОД ПРОГРАММЫ (СЕРВЕРНАЯ ЧАСТЬ) 111
КОД ПРОГРАММЫ (КЛИЕНТСКАЯ ЧАСТЬ) 123

Введение

За последние двадцать лет значительно возрос объём и оборот информации во всех сферах жизнедеятельности человека: экономической, финансовой, политической, социальной. И процесс накопления, обработки и использования знаний постоянно ускоряется. Учёные утверждают, что каждые десять лет количество информации увеличивается вдвое. В связи с этим возникает необходимость использования автоматических средств, позволяющих эффективно хранить, обрабатывать и распределять накопленные данные.

Развитие средств вычислительной техники, ее доступность и наличие доступных информационных технологий, позволяют автоматизировать сферы управления (бухгалтеров, плановиков, специалистов кредитно-банковской системы и т.д.). Организация и реализация управленческих функций требует радикального изменения, как самой технологии управления, так и технических средств обработки информации, среди которых важное место занимают информационные системы. Они все более превращаются из систем автоматической переработки входной информации в средства накопления опыта управленческих работников, анализа, оценки и выработки более эффективных экономических решений [1].

Информационные системы, применяемые в торговле, имеют, свои особенности и радикально отличается от других информационных систем. Компьютер не только облегчает учет, сокращая время, требующееся на оформление документов и обобщение накопленных данных для анализа хода торговой деятельности, необходимого для управления ею. Отчеты о положение в торговле, получаемые с помощью компьютера, можно получить и без него – никакой особой математики в компьютере не содержится – но на расчеты уйдет столько времени, что они уже ни на что не будут нужны; или ими придется занять такое количество расчетчиков, что на их зарплату уйдет значительно больше, чем будет получено прибыли в результате их расчетов.

Таким образом, при применении компьютера “количество переходит в качество”: увеличение скорости расчетов делает возможным качественное улучшение самой схемы построения торговли.

Создание автоматизированных систем предполагает, что основные операции по накоплению, хранению и переработке информации возлагается на вычислительную технику, а менеджер выполняет часть ручных операций и операций, требующих творческого подхода при подготовке управленческих решений. Персональная техника применяется пользователем для контроля производственно-хозяйственной деятельности, изменения значений отдельных параметров в ходе решения задач, а также ввода исходных данных в автоматизированную систему для решения текущих задач и анализа функций управления.

В условиях реализации торгового процесса целью внедрения автоматизированной системы является усиление интеграции управленческих функций, поэтому каждое рабочее место должно обеспечивать работу в многофункциональном режиме.

Одной из задач автоматизации торгового процесса является обработка заказов торгового предприятия. При этом требуется подготовка различных отчетных документов, формируемых различными специалистами (бухгалтерами, кладовщиками, другими материально-ответственными лицами)

В связи с большой номенклатурой товарно-материальных ценностей, большим количеством поставщиков, покупателей и материально ответственных лиц, целесообразно использовать базы данных для хранения справочной и текущей информации о поступлении, фактическом наличии, плановом и фактическом расходовании товарно-материальных ценностей.

Большое количество расчетных операций делают целесообразным использование компьютеров для организации автоматизированной системы по учету движения товарно-материальных ценностей организации с применением базы данных.

Использование автоматизированных систем обработки экономической информации позволяет повысить производительность труда сотрудников организаций, занятых подготовкой отчетных документов.

Темой дипломного проекта является разработка системы обработки заказов товара для торгового предприятия. Эта система ориентируется на автоматизацию процесса осуществления покупок.

Система должна использовать современную СУБД для обеспечения хранения всего спектра данных, и система должна иметь возможность размещения на информационных серверах в компьютерных сетях различной конфигурации и для ее работы использовать унифицированные средства взаимодействия вычислительной системы пользователя с удаленным сервером. Однако, при отсутствии компьютерной сети, должна существовать возможность и локального использования автоматизированной системы. Поэтому в рамках данного дипломного проекта приводиться разработка возможности, как сетевого использования системы, так и ее локальное применение. Отличительной чертой этих версий является различие в методиках установки в зависимости от используемого комплекса вычислительных средств.

Информационная система позволит автоматизировать процесс товарооборота от закупок товара и отправки его заказчику. Так как раньше вся документация велась вручную, при использовании большого перечня номенклатуры – это занимало много времени. Автоматизация позволит значительно сократить время. Менеджеры, работающие с системой, смогут контролировать наличия товара, вести учет товара по поставщикам, иметь полную информацию по поставщикам товара и контроль процесса осуществления заказа с полной информацией о заказываемом товаре, ее цене, получаемых скидках, способе доставки и информации о заказчике.

Эффект от использования подобного средства автоматизации товарооборота будет достигнут за счет сокращения времени осуществлении выписки всей необходимой торговой документации.

1 Предпроектный анализ

Тема дипломного проекта – разработка программного обеспечения обработки заказов торгового предприятия. Программное обеспечение представляет автоматизированную информационную систему менеджеров торгового предприятия ответственных за работу с заказами, товарами и клиентами.

Разработка программного комплекса ведется на основании задания на дипломное проектирование, и утверждено кафедрой "Вычислительная техника" ФГБОУ ВО ПГУ.

Решение задач по учету движения товаров и заказов торговой организации связано с участием различных специалистов (бухгалтеров, сотрудников коммерческого отдела, маркетологов, кладовщиков и других материально ответственных лиц) и характеризуется значительной трудоемкостью. При подготовке отчетных документов следует выполнять:

- формирование достоверных данных по движению товарно-материальных ценностей для торговой организации;
- проверку исходных документов (приход, расходу и др.) на правильность заполнения для дальнейшего бухгалтерского учета в целях правильного налогообложения;
- учет фактического прихода и расхода товарно-материальных ценностей;
- оформление документов по движению товарно-материальных ценностей;
- учет заказов товарно-материальных ценностей;
- учет поставщиков и покупателей.

Задачи учета торгового оборота включает в себя:

- учет материальных ценностей на складах, у материально ответственных лиц (формирование прихода и расхода материалов,

определения текущего наличия, выполнение инвентаризации, оформление отчетных документов и т.д.);

- учет материальных ценностей в бухгалтерии (формирование приходных, расходных и оборотных ведомостей, с учетом распределения материалов по бухгалтерским счетам, передача материальных ценностей на объекты строительства и в подотчет материально ответственным лицам, вычисление стоимости материальных ценностей с учетом износа, подготовка ведомостей движения материалов)

Исходными (первичными) документами для учета движения товарно-материальных ценностей служат:

- накладная и счет-фактура на поставку товарно-материальных ценностей от поставщика;
- накладная и счет-фактура на поставку товарно-материальных ценностей покупателю;
- счет на поставку;
- счет на продажу;
- договор поставки или купли-продажи на товарно-материальные ценности от поставщика;
- договор поставки или купли-продажи на товарно-материальные ценности с покупателем;
- доверенность на получение товарно-материальных ценностей от поставщика;

Выходные документы должны формироваться по каждому поставщику и покупателю, а также по отдельным наименованиям товарно-материальных ценностей:

- ведомость поступления товарно-материальных ценностей на склад с группировкой по поставщикам, приходным документам и основным счетам;

- ведомость расхода товарно-материальных ценностей со склада с группировкой по получателям и расходным документам; в расход по складу включаются все отпущенные со склада материалы;
- оборотная ведомость движения материалов по складу;
- сводная ведомость движения материалов (в суммовом выражении);
- книга покупок и книга продаж с указанием сумму поступивших товарно-материальных ценностей по установленной форме;
- книга учета доверенностей на получение товарно-материальных ценностей от поставщиков;
- кассовая книга по установленной форме (учет наличных денежных средств от поступивших и проданных товарно-материальных ценностей).

Формы документов представлены в приложении.

Для автоматизации вышеперечисленных работ и повышения производительности труда менеджера по продажам, можно использовать ПК, которые обладают характеристиками, позволяющими их эксплуатировать без участия специалистов по вычислительной техники (исключая техническое и системное обслуживание)

При разработке программного обеспечения для решения перечисленных задач следует учесть:

- необходимость работы со справочниками данных;
- возможность редактирования справочных и отчетных данных;
- обеспечение удобного интерфейса;
- необходимость контроля действий пользователя;
- возможность сохранения данных на внешних носителях;
- особенности технологии обработки данных, принятых в конкретной организации.
- возможность работы с современными телекоммуникационными технологиями.

Разрабатываемая система предназначена для автоматизации обработки заказов торгового предприятия и предназначена для контроля и учета товаров в торговой организации на рабочем месте менеджера:

- обеспечение автоматизированного формирования документов по оформлению движения товаров, сопроводительных расчетно-платежных документов;
- ведение справочников по поставщикам:
- ведение справочников по покупателям:
- сокращение сроков подготовки документов, отражающих перемещение товарно-материальных ценностей и сопроводительных расчетно-платежных документов оптовой организации;
- повышение производительности труда менеджера по учету товарно-материальных ценностей в оптовой организации.

Система должна функционировать в кросплатформенном операционном пространстве. Для работы с программой обслуживающему персоналу достаточно знаний персонального компьютера на уровне умения работы с операционной системой (ОС) Windows 8 и приложениями написанных под данную ОС.

В связи с тем что система автоматизации торговли является программным бизнес инструментом, к ее разработке применяются очень серьезные требования. Ниже представлены характеристики, по которым можно судить о качестве программного продукта.

1.1 Стандарты оценки качества программного обеспечения

Одной из особенностей программных систем является невозможность выделения одного единственного критерия качества, который бы полностью характеризовал систему, ее конструктивные и функциональные особенности.

Существуют отечественные и зарубежные стандарты [2, 6, 8] по проблемам качества программных продуктов. Им посвящен ряд трудов и практических методик [5, 6].

Например, в [5] предложена иерархическая древовидная структура характеристик качества программного обеспечения, верхние уровни которой показывают сферы приложения оценок качества программных средств, а нижние представляют собой множество элементарных свойств, таких как, например, завершенность, точность, согласованность, структурированность, открытость, расширяемость и т.п.

В стандарте качества ISO 9000 [6], под качеством программного продукта понимается совокупность его характеристик, относящихся к способности удовлетворять установленные и предполагаемые потребности клиентов, а основными параметрами качества считаются:

- функциональная полнота,
- соответствие требованиям законодательства стран СНГ,
- безопасность информации,
- минимизация затрат на эксплуатацию,
- развитие и модернизацию,
- эргономичность пользовательского интерфейса,
- безопасность информации,
- простота эксплуатации, не требующая специальных знаний в области информационных технологий.

При этом под надежностью программного обеспечения (ПО) понимается, прежде всего, безотказная работа и отсутствие ошибок, препятствующих нормальному функционированию предприятия.

Надежность и качество программного продукта обеспечивают его высокие потребительские свойства [8]. Стандарт ISO 9000 был одной из первых моделей качества, первая версия которого была выпущена в 1987 году и с тех пор сохраняет неизменную популярность и признается во всем мире.

Стандарт ISO 9001 рассматривает модель обеспечения качества при проектировании, разработке, производстве, монтаже и обслуживании [9].

ISO 9000-3 помимо всех положений общего стандарта ISO 9001, включает в себя необходимые дополнения к ним, относящиеся к разработке, поставке и обслуживанию программного обеспечения, Но методики, положенные в основу стандартов серии ISO 9000, постепенно устаревают. В качестве их наиболее существенных недостатков можно отметить следующие [9]:

- недостаточная подробность стандарта, возможность самых различных его толкований в зависимости от представлений аудитора;
- неточность оценки качества процессов, задействованных при создании и внедрении программного обеспечения;
- отсутствие в стандарте механизмов, способствующих улучшению существующих процессов.

Перечисленные выше проблемы заставили экспертов разрабатывать более совершенные решения в области обеспечения качества. В начале 90-х годов прошлого века появился целый ряд новых стандартов и методологий.

Одними из наиболее удачных и содержательных стандартов можно назвать два стандарта: Capability Maturity Model (CMM) и ISO/IEC 15504 (SPICE)

Изначальной целью разработки стандарта Capability Maturity Model было создание методики, позволяющей крупным правительственным организациям США выбирать наилучших поставщиков программного обеспечения, для чего предполагалось создать исчерпывающее описание способов оценки процессов разработки программного обеспечения и методики их дальнейшего усовершенствования [10].

Авторам удалось добиться такой степени подробности и детализации, что стандарт оказался пригодным и для обычных компаний-разработчиков, желающих улучшить существующие процессы. Согласно [7], «главным понятием стандарта является зрелость организации.

Незрелой считается организация, в которой процесс разработки программного обеспечения зависит только от конкретных исполнителей и менеджеров, и решения зачастую просто импровизируются "на ходу". В этом случае велика вероятность превышении бюджета или увеличение сроков сдачи проекта, и потому менеджеры вынуждены заниматься только разрешением ближайших проблем.

С другой стороны, в зрелой организации имеются четко определенные процедуры создания программных продуктов и управления проектами. Эти процедуры по мере необходимости уточняются и совершенствуются в пилотных проектах или с помощью анализа стоимость/прибыль. Оценки времени и стоимости выполнения работ основываются на накопленном опыте и достаточно точны. Наконец, в компании существуют стандарты на процессы разработки, тестирования и внедрения ПО, правила оформления конечного программного кода, компонент, интерфейсов и т.д. Все это составляет инфраструктуру и корпоративную культуру, поддерживающую процесс разработки программного обеспечения» В модели CMM определено пять уровней зрелости организаций, один из которых присваивается компании в результате аттестации в дальнейшем этот уровень может повышаться или понижаться. Каждый следующий уровень включает в себя все ключевые характеристики предыдущих уровней. На рисунке 1.1 приведены некоторые технологии, внедрение которых необходимо для достижения различных уровней зрелости организации [7].

Начальному уровню организации (initial level) соответствуют предприятия, на которых не существует стабильных условий для создания качественного программного обеспечения.

Это приводит к тому, что результат любого проекта целиком и полностью зависит от личных качеств менеджера и опыта программистов. При этом успех в одном проекте может быть повторен только в случае назначения на следующий проект тех же менеджеров и программистов. Если же такие менеджеры или программисты уходят с предприятия, то резко падает качество производимых программных продуктов. Процесс разработки в стрессовых ситуациях сводится к написанию кода и его минимальному тестированию [3].

Рисунок. 1.1 Пять уровней зрелости в модели CMM [1]

Повторяемый уровень организации (repeatable level) достигается внедрением на предприятии технологии управления проектами.

Планирование и управление проектами основывается на накопленном опыте.

Существует специальная группа, которая обеспечивает следование разрабатываемого программного обеспечения определенным стандартам.

При необходимости организация взаимодействует с субподрядчиками. При этом в критических условиях процесс имеет тенденцию скатываться на начальный уровень.

Определенный уровень (defined level) характеризуется тем, что стандартный процесс создания и сопровождения программного обеспечения документирован (включая и разработку ПО, и управление проектами). В процессе стандартизации происходит переход на наиболее эффективные практики и технологии [7], В организации должна быть создана специальная группа для создания и поддержания подобного стандарта.

Кроме того, обязательным условием для достижения этого уровня является наличие на предприятии программы постоянного повышения квалификации и обучения сотрудников, с достижением этого уровня организация перестает зависеть от качеств конкретных разработчиков и не имеет тенденции скатываться на уровень ниже в стрессовых ситуациях.

Управляемого уровня (maraged level) организация достигает тогда, когда В ней устанавливаются количественные показатели качества — как на программные продукты, так и на процесс в целом, Более совершенное управление проектами достигается за счет уменьшения отклонений различных показателей проекта, а осмысленные вариации в производительности процесса можно отличить от случайных вариаций (шума), особенно в хорошо освоенных областях.

Основной характеристикой последнего, оптимизирующего уровня (optimizing level) является то, что мероприятия по улучшению применяются не только к существующим процессам, но и для оценки эффективности ввода новых технологий. Улучшение процессов в идеале должно помогать предупреждать возможные Ошибки или дефекты. Кроме того, должны вестись работы ПО уменьшению стоимости разработки программного обеспечения, например с помощью создания и повторного использования компонент [7].

Согласно стандарту CMM при сертификации проводится оценка соответствия всех ключевых областей по 10-балльной шкале, при этом для успешной квалификации данной ключевой области необходимо набрать не менее б баллов. Показатели, по которым производится оценка ключевой области, следующие [7]:

- заинтересованность руководства, а данной области (планируется пи практическое внедрение данной ключевой области, существует ли понимание у руководства необходимости данной области и т.д.);
- насколько широко данная область применяется в организации (например, оценке в 4 балла соответствует фрагментарное применение);
- успешность использования данной области на практике (например, оценке в 0 баллов соответствует полное отсутствие какого-либо эффекта, а оценка в 8 баллов выставляется при наличии систематического и измеримого положительного результата практически во всей организации).

Как отмечается в [7], «можно сертифицировать только один процесс или подразделение организации, например, подразделение разработки программного обеспечения компании IBM сертифицировано на пятый уровень. Кстати, в мире существует немного компаний, которые могут похвастаться наличием у них пятого уровня CMM хотя бы в одном из подразделений - таких всего около 50.

С другой стороны, насчитывается несколько тысяч компаний, сертифицированных по 3 или 4 уровню, то есть существует колоссальный разрыв между оптимизированным уровнем зрелости и предыдущими уровнями».

Согласно статистическим данным, приведенным в [7], свыше 70 % всех компаний-разработчиков находится на первом уровне CMM.

Интересен вопрос о том, на каком уровне CMM должна находиться организация, чтобы получить сертификат соответствия ISO 9001? Как отмечается в [7], «на первый взгляд, для этого необходимо иметь как минимум

3 или 4 уровень CMM, Тем не менее, существуют примеры организаций 1-го уровня, имеющих сертификат ISO 9001. Иногда, как это ни странно, аудиторам не хватает элементарного знакомства с реальной практикой разработки программ». Более подробно вопрос о соотношении ISO 9001 и CMM рассмотрен в [16], следует заметить, что ряд важных вопросов, например, таких как отбор, повышение квалификации и сохранение компетентных сотрудников, остались за рамками CMM.

Однако эти вопросы исключительно важны, ибо, как замечено в [5] «и 20-30 лет назад было известно, что два программиста, сидящих за соседними столами и получающих примерно одинаковую зарплату, могут писать программы, отличающиеся по скорости счета, скажем, в 10 раз».

Особенно важен подбор сотрудников для организаций первого уровня, для которых сотрудники являются единственной гарантией качества.

«Человеческий фактор» сохраняет свою значимость и на более высоких уровнях зрелости. В 1995 году был опубликован стандарт People CMM, являющийся дополнением к Software CMM и внедрение которого обеспечивает организацию целым набором процедур по оценке и развитию всей системы найма, обучения и сохранения квалифицированных сотрудников.

Использование CMM, к сожалению, затруднено рядом проблем [76], одной из которых является то, что стандарт CMM является собственностью Software Engineering Institute и не является общедоступным (в частности, дальнейшая разработка стандарта ведется самим институтом, без заметного влияния остальной части программистского сообщества). Кроме того, оценка качества процессов организаций может проводиться только специалистами, прошедшими специальное обучение и аккредитованными SEL. Наконец, стандарт ориентирован на применение в относительно крупных компаниях.

В 1991 году Международной организацией по стандартизации была инициирована работа по созданию единого стандарта оценки программных процессов. Основной задачей стандарта SPICE (Software Process Improvement

and Capability determination, официальное название «ISO/IEC 15504: Information Technology - Software Process Assessment»), было создание международного стандарта, учитывающего весь накопленный опыт в области разработки ПО, Таким образом, стандарт SPICE унаследовал многие черты более ранних стандартов (рисунок 1.2) [7].

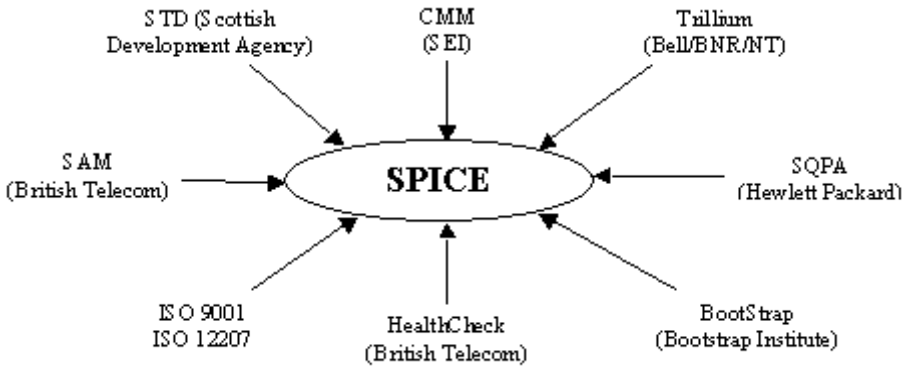


Рисунок 1.2 Стандарты, идеи которых были использованы при создании стандарта SPICE

Основной задачей организации по стандарту SPICE является постоянное улучшение процесса разработки ПО, так же, как и в стандарте CMM. Используется схема с различными уровнями возможностей, но эти уровни применяются не только к организации в целом, но и к отдельно взятым процессам,

В таблице 1.1 представлен список уровней способностей модели SPICE и характерные для них процедуры управления [15].

Таблица 1.1 Уровни способностей процесса по стандарту SPICE

Уровни	Названия
Уровень 0	Процесс не выполняется
Уровень 1	Выполняемый процесс
Уровень 1.1	Измерение производительности процесса
Уровень 2	Управляемый процесс

Уровень 2.1.	Управление производительностью
Уровень 2.2	Управление созданием продуктов
Уровень 3	Установленный процесс
Уровень 3.1.	Документирование процесса
Уровень 3.2	Отслеживание ресурсов процесса
Уровень 4	Предсказуемый процесс
Уровень 4.1	Измерение процесса
Уровень 4.2	Управление процессом
Уровень 5	Оптимизирующий процесс
Уровень 5.1.	Изменение процесса
Уровень 5.2	Постоянное совершенствование

Задачи, которые решаются во время оценки и улучшения качества процессов разработки ПО, следующие [7]:

- оценка соответствия процесса разработки ПО, существующего в данной организации, модели, описанной в стандарте. Анализ полученных на этом этапе результатов дает возможность определить сильные и слабые стороны процесса, выявить внутренние риски, присущие данному процессу, оценить эффективность процессов н определить причины ухудшения качества и связанные с этим издержки во времени или стоимости;
- определение возможностей процесса, позволяющее оценить способы улучшения данного процесса. Очень часто эта задача выполняется компанией-поставщиком, чтобы убедить существующих или потенциальных заказчиков в своей способности достичь заданных показателей;
- улучшение того или иного процесса, понимание необходимости которого может появиться в организации в результате предыдущих и шагов. Цели совершенствования процесса к этому моменту уже четко

сформулированы и остается только техническая реализация поставленных задач, после чего весь цикл работ начинается сначала.

Стандартом SPICE предоставляется более полный набор средств обеспечения качества и улучшения процессов, чем ISO 9001. Использование этого стандарта поможет организации заметно улучшить существующие процессы, а затем при необходимости получить и сертификат ISO 9001.

Таким образом, стандарты CMM и SPICE на сегодняшний день представляют собой наиболее развитые модели качества, прошедшие применение на практике.

В данной дипломной работе использован подход SPICE, для постоянного совершенствования разрабатываемой системы, в связи с тем, что требования к системе со стороны персонала постоянно меняются.

1.2 Анализ существующих автоматизированных систем для обеспечения логистики торгового предприятия

Специфика счетов, форм квартальной и годовой отчетностей, структуре баланса, методика ведения бухгалтерского учета, систем проводок хозяйственных операций и жесткие требования по соблюдению нормативных документов, регулирующих ведение торговли, все эти особенности необходимо учитывать при выборе и разработке программного обеспечения автоматизированной системы торговли и/или логистики. Еще несколько лет назад, разработчики программных продуктов для автоматизации, презентуя на выставках свои продукты для хозрасчетных предприятий, добавляли, что программу можно настроить под нужды конкретного предприятия.

Однако практика показала, что один и тот же программный продукт не может соответствовать полностью требованиям разных организаций и требует не только общей но и «тонкой» настройки.

В настоящее время на рынке программных продуктов представлено множество различных программных комплексов, предназначенных для

автоматизации логистики и торговли [2, 4, 5]. К сожалению, среди всего этого разнообразия, лишь небольшое число разработок посвящено автоматизации с «тонкой» настройкой.

Ниже приведен обзор известных на сегодняшний день автоматизированных систем для торговых организаций (АСТ).

АСТ «Парус»

Одним из основных фирм-разработчиков программного обеспечения для автоматизации является корпорация «ПАРУС» (рисунок 1.3).

Рисунок 1.3 Общая структура системы «Парус»

Основанная в 1990 г, компания с самого начала провозгласила основным направлением своей деятельности разработку и внедрение комплексных систем автоматизации финансово-хозяйственной деятельности организаций.

И хотя спектр разработок этой фирмы в настоящее время включает также множество различных разработок для предприятий малого, среднего и даже крупного бизнеса, она остается заслуженным лидером в сфере автоматизации бюджетных организаций.

Комплексная система автоматизации управления учреждениями и организациями «ПАРУС-предприятие» состоит из нескольких модулей: «Реализация и склад», «Кадровый учет и зарплата», «бухгалтерия» и т.д. Бухгалтерский комплекс является к системе объединяющим звеном всех остальных функциональных моделей.

Основными функциональными особенностями «Бухгалтерского комплекса» являются:

- возможность глубокого анализа учетных данных, благодаря ведению расширенного аналитического учета по счетам (уровень аналитики до пятого порядка);
- возможность вести учет денежных средств на счетах, товарно-материальных ценностей, основных средств и нематериальных активов, расчетов с дебиторами-кредиторами в рублях, иностранной валюте и рублевом эквиваленте (мультивалютность);
- широкая интеграция системы с пакетами Microsoft Office - возможность редактирования и печати документов, подготовленных с помощью системы «ПАРУС», массового ввода информации через электронные таблицы, создания и редактирования с помощью Microsoft Excel шаблонов документов;
- разграничение прав доступа пользователей, как к разделам информации, так и к функциям системы;
- возможность пакетной обработки однотипных документов (например, кассовых или банковских) в журнале хозяйственного учета и т.д.

Одной из последних разработок компании является корпоративный вариант бюджетной версии системы «ПАРУС», созданный на базе Oracle (рисунок 1.4).

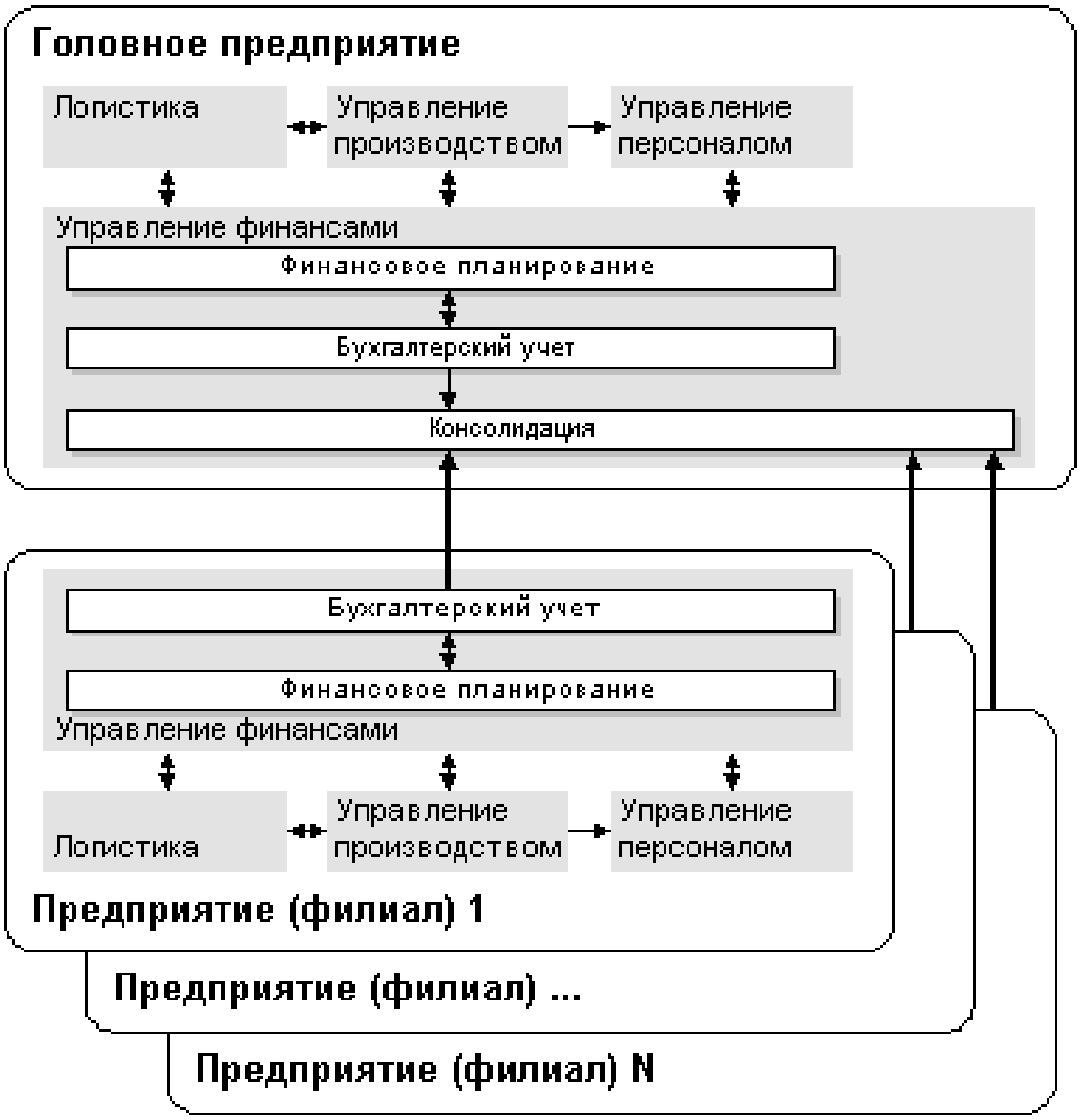


Рисунок. 1.4 Корпоративный вариант системы «Парус»

АСТ «1С Торговля»

Компанией «1С» выпущена конфигурация «Бухгалтерия для бюджетных организаций» системы «1С Предприятие» (рисунок 1.5). Данная версия программного продукта реализует наиболее стандартную методику учета для бюджетных учреждений в соответствии с текущим законодательством. В качестве особенностей данной разработки можно отметить следующие [7, 8, 10, 17]:

- возможность гибкого конфигурирования вариантов обработки учетной информации. Это позволяет квалифицированному пользователю

самостоятельно, без помощи разработчиков, изменять алгоритмы обработки и выходные данные;

- наличие мощных и в то же время единообразных средств создания отчетов любой сложности;
- интеграция со справочно-информационными системами «Гарант» и «Кодекс»;
- удобный и продуманный интерфейс;
- технологическая гибкость, достигаемая за счет наличия нескольких способов ввода исходной информации - операциями вручную, типовыми операциями иди документами;
- возможность интеграции и разделения доступа пользователей к информации при работе с системой в сети. Конфигуратор системы позволяет установить для каждого пользователя права на работу с информацией, обрабатываемой системой, которые могут быть заданы в достаточно широких пределах (от возможности только просмотра некоторых видов документов до полного набора прав по вводу, просмотру, корректировке и удалению любых видов данных).

Рисунок 1.5 Структура пакета 1С Предприятие

АСТ «АВЕРС»

Информационным центром «АВЕРС» выпущен типовой комплекс средств автоматизации торгового предприятия «АВЕРС» (рисунок 1.6). Данный программный продукт полностью автоматизирует работу на всех участках, позволяет вести полный учет, а также многоуровневый аналитический учет, легко конфигурируясь под перечень задачорганизации [12].

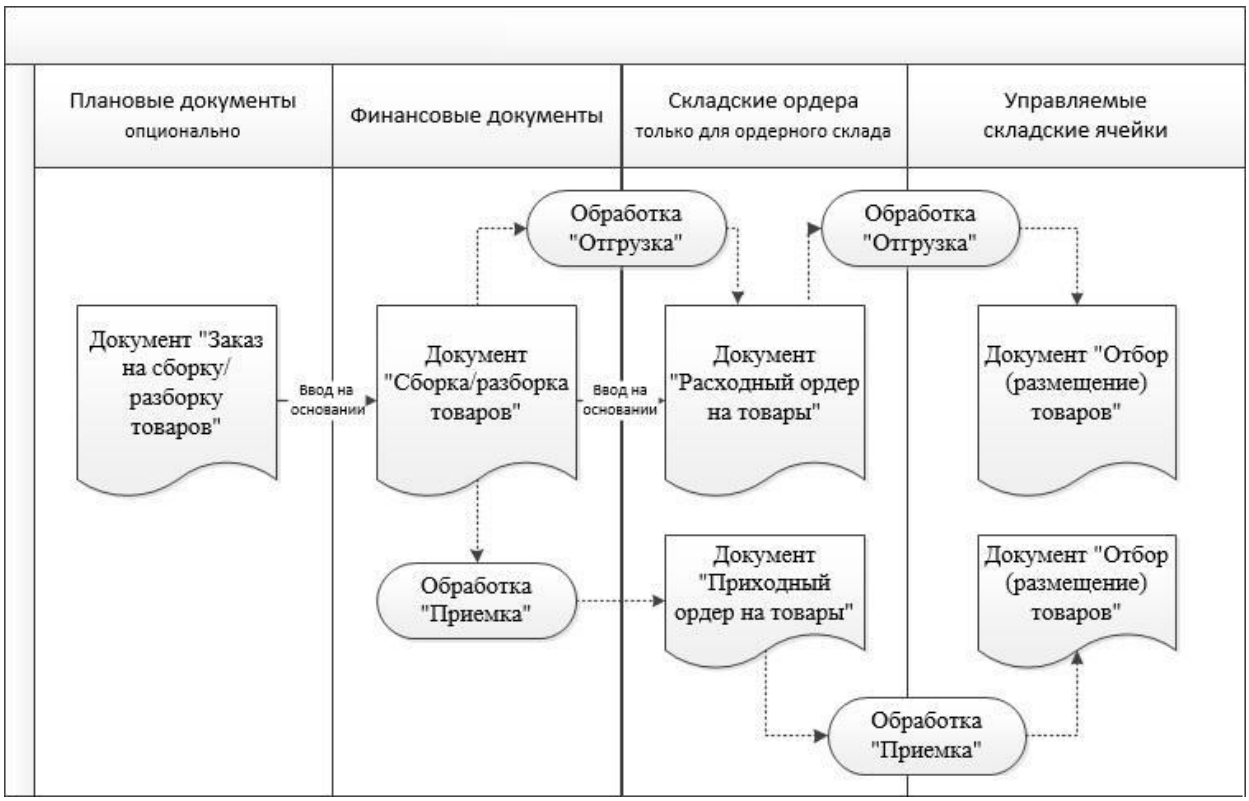


Рисунок 1.6 АСТ «АВЕРС»

Комплекс построен на основе использования файл-серверной архитектуры и структурно состоит из совокупности функционально полных моделей, каждый из которых соответствует определенному разделу бухгалтерского учета, модули объединены единой базой данных, могут функционировать как совместно под управлением операционной системы, так и автономно. Основными компонентами, входящими в состав комплекса являются следующие:

- «Заказ»;
- «Приемка»;
- «Расходный ордер»;
- «Отбор товаров»;
- «Отгрузка товара»;
- «Расчеты с поставщиками»;

Данную программную разработку отличает высокая степень адаптации к специфическим требованиям крупных организаций, достаточно низкие

требования к технической платформе функционирования, в связи с чем - широкая аппаратная совместимость, легкость в освоении, единство интерфейса для всех модулей, входящих в комплекс.

АСТ «Террасофт»

Разработчики программного продукта «Террасофт» (фирма «FMCG») изначально ставили перед собой задачу создания программы, которая бы быстро осваивалась пользователями различного уровня квалификации и при этом была бы довольно универсальной [4].

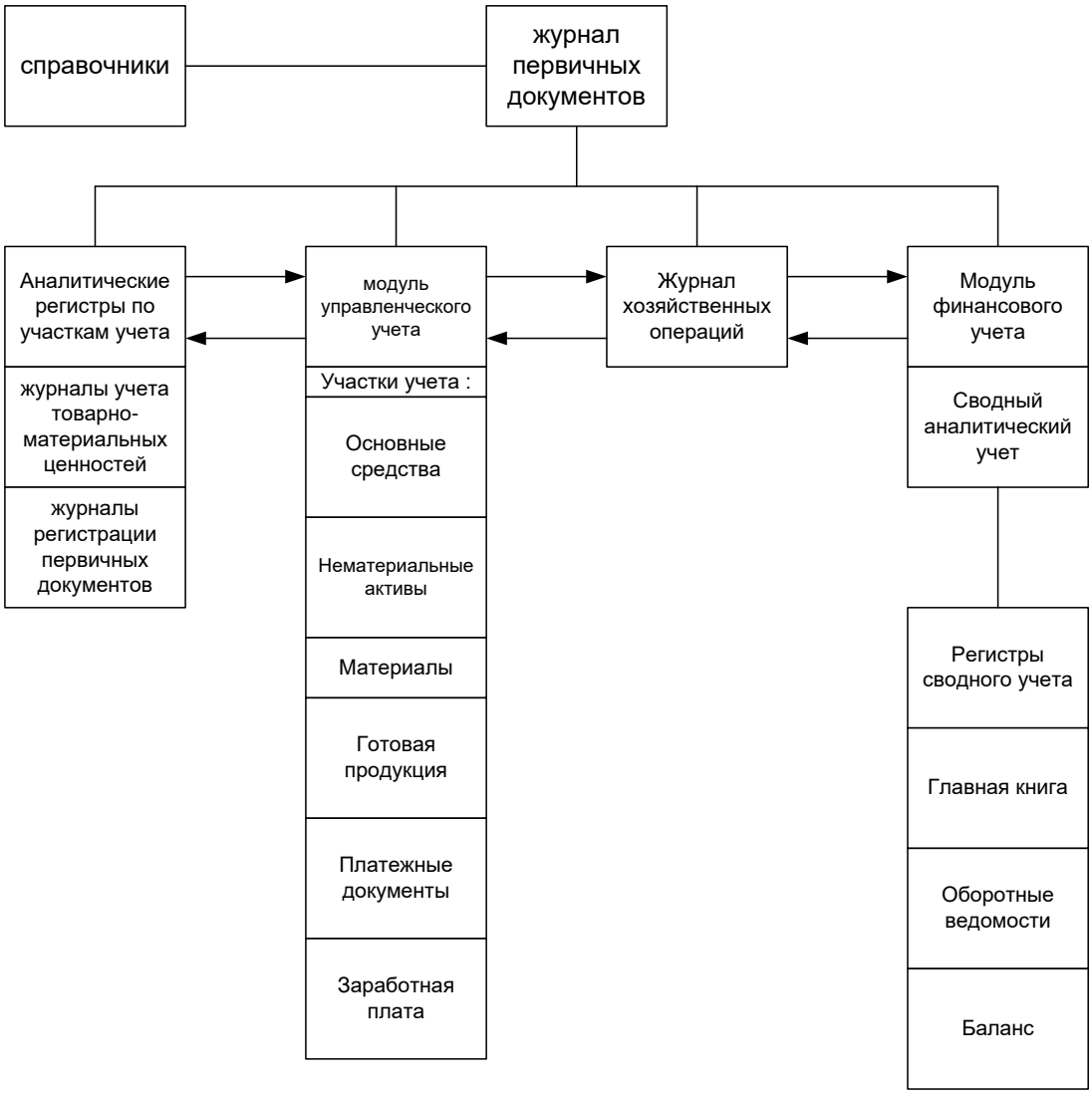


Рисунок 1.7 Схема АСТ «Террасофт»

В бюджетном варианте «Террасофт» предусмотрены следующие возможности:

- учет денежных средств (в частности, ведение кассовой книги);
- формирование заказов;
- формирование списков поставщиков.

В качестве положительных особенностей данной разработки следует отметить:

- наглядность и простоту интерфейса программы, благодаря которым основные операции по формированию первичных документов и созданию отчетов выполняются достаточно легко пользователями с различным уровнем подготовки;
- возможность представления информации в графическом виде облегчает процесс анализа торговой деятельности;
- возможность заполнения журнала операций как вручную с помощью диалогового окна или путем копирования аналогичной и ее коррекции, так и автоматически, в результате ввода первичных документов;

АСТ «Галактика»

В настоящее время практически все ведущие производители финансово-экономического программного обеспечения активно работают над созданием комплексных информационных систем масштаба предприятия и корпорации, в которых возрастает интеграция отдельных программных модулей. При создании таких систем на первый план выходят проблемы правильной организации единой информационной среды управления предприятием: уровней иерархии, агрегирования информации, ее оперативности и достоверности, консолидации данных и т.д.К системам данного класса относится система «Галактика», разработанная корпорацией «Галактика», одним из крупнейших разработчиков комплексных решений в области автоматизации управления производственно-хозяйственной и финансовой

деятельностью предприятия (5, 6, 12, 14). Система «Галактика» ориентирована на автоматизацию решения задач, возникающих на всех стадиях управленческого цикла: планирование, учет и контроль реализации планов, анализ результатов.

Основные принципы системы «Галактика»:

- Корпоративность. Охват всего спектра типовых производственно-экономических функций, модульность и работа в информационном пространстве единой базы данных, обеспечение гибкой настройки на специфику и сферу деятельности конкретного предприятия, масштабируемость системы позволяют отнести систему «Галактика» к классу корпоративных систем.
- Интеллектуальность. Отличительной особенностью системы является направленность на решение задач управления предприятием, в отличие от традиционного подхода, основанного на регистрации и накоплении информации,
- Интегрированность и модульность. Из набора интегрированных модулей заказчику может быть составлена необходимая конфигурация, что дает возможность поэтапного построения нужной функциональности и внедрения системы.
- Многоплатформенность. В базовом варианте система основывается на собственной СУБД, которая использует недорогой менеджер записей Btrieve. В тех случаях, когда требуется применение более производительных средств система «Галактика» легко перенастраивается для работы с мощными СУБД Oracle, Interbase, Sybase, При этом клиентская часть может функционировать в различных операционных средах: Windows 8, Windows Server 2008 и т.д.

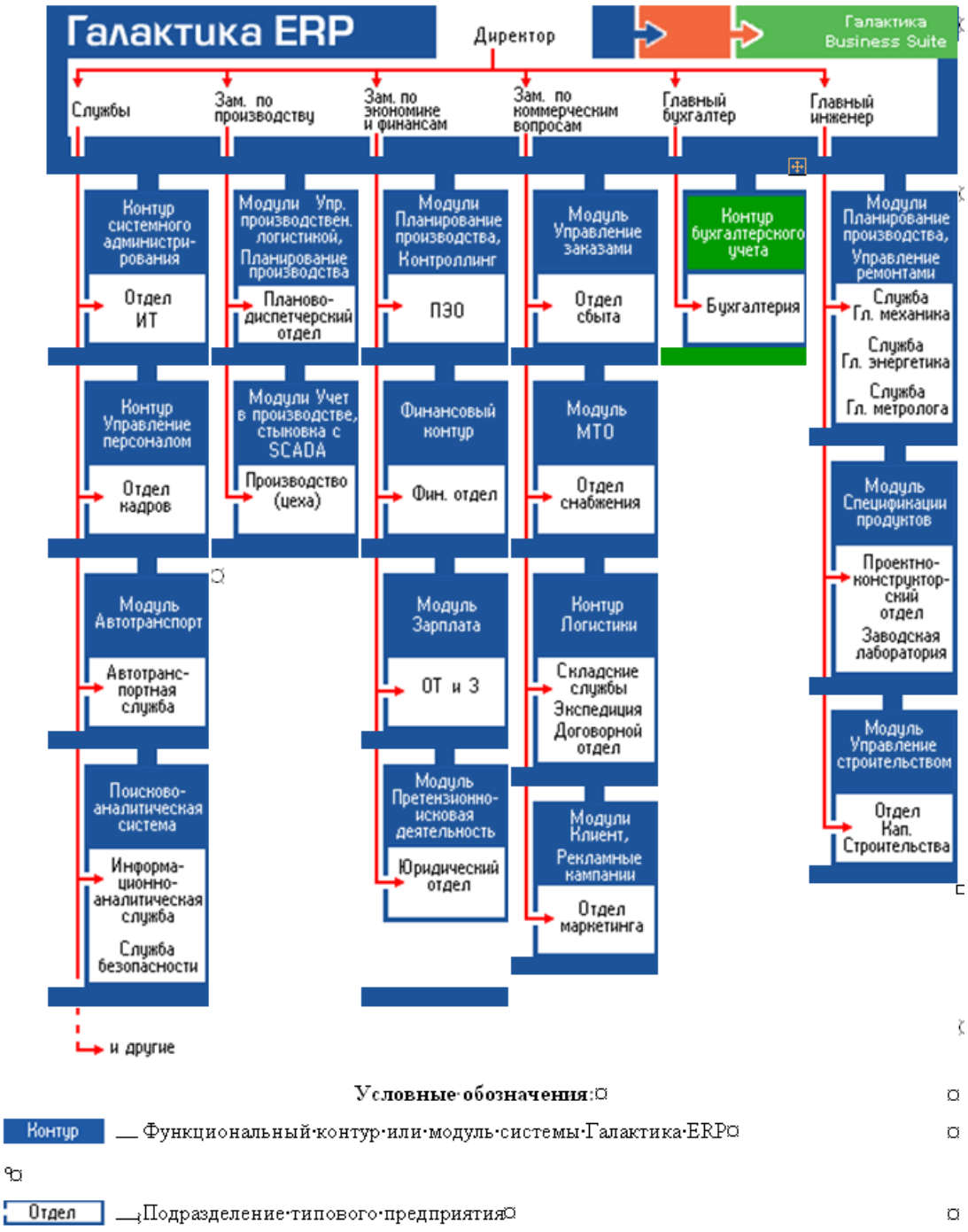


Рисунок 1.8 Модульная структура системы Галактика ERP

Система построена по модульной структуре, допускающей как изолированное использование отдельных ее составляющих, так и их произвольные комбинации. Составные модули условно объединены в функциональные контуры, основными из которых являются следующие:

- контур управления персоналом, предназначенный для автоматизированного учета кадров и выполнения вычислительных процедур, связанных с оплатой труда персонала предприятий;
- контур управления финансами, обеспечивающий решение задач финансового менеджмента, предоставляет набор средств, для управления бюджетом, ведения платежного календаря и финансового анализа;
- контур бухгалтерского учета, представляющий собой функционально полную систему ведения бухгалтерского учета на предприятиях любой формы собственности и видов деятельности. Единое информационное пространство системы обеспечивает автоматическое отражение в бухгалтерском контуре всех хозяйственных операций;
- контур логистики, охватывающий разнообразные задачи, связанные с организацией и управлением производственной и коммерческой деятельностью предприятия;
- контур администрирования, предоставляющий набор сервисных средств для квалифицированного пользователя и программиста, обеспечивающих администрирование базы данных, корпоративным обмен данными, обмен документами с внешними информационными системами, а также проектирование пользовательского интерфейса и отчетов.

Отличительными особенностями системы являются универсальность и адаптивность. На базе системы «Галактика» предлагаются специализированные решения для предприятий нефтегазовой отрасли, энергетики, торговли, химической промышленности, металлургии и горнодобывающей промышленности, пищевой промышленности, предприятий сняли и телекоммуникаций, предприятий лесопромышленного комплекса и др. Для бюджетных организаций отсутствует специализированное решение, но наличие мощного механизма настроек, обеспечивающего различные схемы эксплуатации системы, позволяет

реализовать на бале ядра системы «Галактика» версию системы для бюджетных организаций,

Полная стоимость вышеописанных систем (с учетом затрат на обследование, поставку, внедрение и обучение персонала) составляет около 150-200 тыс. долл. Такие расходы может себе позволить далеко не каждое даже крупное предприятие, не говоря уже о компаниях среднего и малого масштаба.

Таким образом, рассмотрены известные на рынке программных продуктов разработки для автоматизации логистики и торговли. Следующими этапами являются выделение перечня функций, реализуемых АСТ для малых предприятий их реализация.

2 Разработка автоматизированной системы для обеспечения логистики предприятия

2.1 Формализация задачи

В дипломном проекте разработано программное обеспечение для автоматизации работы торгового предприятия, в части создания автоматизированного цикла товарооборота и складского учета. Программное обеспечение может быть включено в состав автоматизированной системы по товарообороту и электронным средствам коммерции.

При разработке программного обеспечения необходимо учитывать процесс движения товарно-материальных ценностей в торговом предприятии. При его анализе необходимо изучить формы входных и выходных документов, влияющих на разработку подсистемы хранения данных.

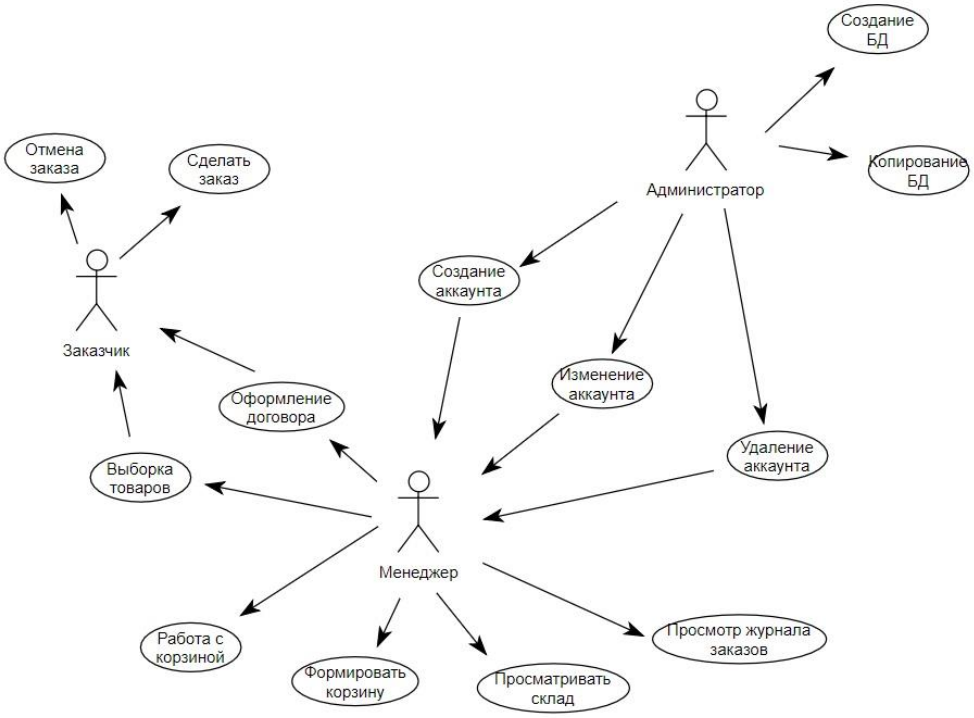


Рисунок 2.1 Схема движения товаров и ролей в организации

Исходными или входными (первичными) документами для учета движения товарно-материальных ценностей служат:

- накладная и счет-фактура на поставку товарно-материальных ценностей от поставщика;
- накладная и счет-фактура на поставку товарно-материальных ценностей покупателю;
- счет на поставку;
- счет на продажу;
- договор поставки или купли-продажи на товарно-материальные ценности от поставщика;
- договор поставки или купли-продажи на товарно-материальные ценности с покупателем;
- доверенность на получение товарно-материальных ценностей от поставщика;

Выходные документы должны формироваться по каждому поставщику и покупателю, а также по отдельным наименованиям (или группам, при большом ассортименте) товарно-материальных ценностей. К ним относятся:

- ведомость поступления товарно-материальных ценностей на склад с группировкой по поставщикам, приходным документам и основным счетам;
- ведомость расхода товарно-материальных ценностей со склада с группировкой по получателям и расходным документам; в расход по складу включаются все отпущенные со склада материалы;
- оборотная ведомость движения материалов по складу;
- сводная ведомость движения материалов (в суммовом выражении);
- книга покупок и книга продаж с указанием сумму поступивших товарно-материальных ценностей по установленной форме;
- книга учета доверенностей на получение товарно-материальных ценностей от поставщиков;

- кассовая книга по установленной форме (учет наличных денежных средств от поступивших и проданных товарно-материальных ценностей)

Все программное обеспечение можно разделить на несколько частей: база данных, где хранятся все необходимые данные, программные модули для выполнения расчетов и визуальный интерфейс взаимодействия с пользователем. От правильной реализации каждого их компонентов программного комплекса зависит качество работы программного комплекса в целом. Основой построения такой программной системы является база данных. Для ее управления используют системы управления базами данных (СУБД). При проектировании используется СУБД MySQL.

СУБД — совокупность информационных массивов, технических, программных и языковых средств, предназначенных для сбора, хранения, поиска, обработки и выдачи данных по запросам конечных пользователей. Конечным называется пользователь, обращающийся к системе для получения необходимых ему данных. Необходимость использования СУБД обусловлена целью, упорядочить механизм хранения данных и исключить их потери.

В разрабатываемом программном обеспечении СУБД выполняет следующие функции:

- ввод данных, используя формы ввода;
- формирование отчетных документов;
- вывод документов на экран и печать сформированных документов и отчетов;
- поиск и отбор данных по запросам пользователя.

Проектирование базы данных является основной задачей при разработке программной системы. Основные компоненты базы данных являются связующими частями между функциональными частями системы, так как входные данные одной из частей системы являются входными данными для другой. Проектирование охватывает три основные области:

- Проектирование конкретных объектов, которые будут реализованы в базе данных. Такими объектами являются таблицы, представления, индексы и хранимые процедуры.
- Проектирование конкретных экранных форм, отчетов и программ, которые будут сопровождать данные в базе данных, и обеспечивать выполнение запросов к этим данным.
- При определенных обстоятельствах в процессе проектирования также необходимо учитывать конкретную среду или технологию – например, топологию сети, конфигурацию аппаратных средств, использование реляционной СУБД, параллельной обработки и распределенной архитектуры базы данных.

При проектировании базы данных одной из важнейших является задача выделения различных уровней представления и описания данных. Известно, что одна и та же информация может быть представлена различными способами. Существуют несколько уровней представления данных:

1. Внешнее представление – это совокупность информационных моделей, каждая из которых описывает ту часть предметной области, которая необходима для обеспечения информационных потребителей определенного приложения;
2. Концептуальное представление – обобщение и согласование внешних представлений пользователей, т.е. оно отображает общие информационные потребности всех приложений, согласованных между собой. Концептуальное представление системно независимо, т.е. независимо от конкретной СУБД, операционной системы и аппаратного обеспечения ЭВМ.
3. Представление в терминах конкретной СУБД. Все современные типы ЭВМ поддерживают определенный тип модели данных, в основном сетевой, иерархический или реляционный. Описание предметной области в терминах модели данных, поддерживаемой конкретной СУБД, называется логическим представлением.

Внутреннее представление – это представление о размещении данных в среде хранения и путях доступа к ним. При этом должны быть определены различные типы хранимых записей, существующие индексы, представление хранимых полей, физическая последовательность хранимых записей и т.д.

При построении концептуальных моделей предметной области, используя большинство предлагаемых методик, на этапе построения концептуальной модели выполняются следующие задачи:

1. Выявляются цели, которые преследует администратор предприятия, принимая решения о создании информационной системы.
2. Выделяется на основе этих целей потенциальных пользователей системы.
3. Создаются внешние пользовательские представления.
4. Выявляются и устраняются конфликтные ситуации, возникающие между различными внешними представлениями, и интегрирует их в концептуальное представление.
5. Записывается концептуальное представление на формальном языке, позволяющем легко отображать это представление в одну из логических моделей данных.

Формирование концептуальной модели базы данных включает в себя:

- Идентификацию функциональной деятельности заданной предметной области;
- Идентификацию объектов, которые осуществляют эту функциональную деятельность, и формирование из этих операций последовательность событий, которые могут идентифицировать все сущности и взаимосвязи между ними;
- Идентификацию взаимосвязей между сущностями;

Для того чтобы спроектировать структуру базы данных, необходима исходная информация о предметной области. Желательно, чтобы эта

информация была представлена в формализованном виде. Для этих целей используют искусственные формализованные языковые средства.

Под инфологической моделью понимают описание предметной области, выполненное с использованием специальных языковых средств, не зависящих от используемых в дальнейшем программных средств. Для её описания используются как языки аналитического (описательного) типа, так и графические средства. Последние в настоящее время приобретают все большую популярность. Существует ряд средств автоматизации проектирования, для которых исходные данные представляются в виде совокупности графических схем. Графическое представление является наиболее простым для восприятия и анализа, поэтому воспользуемся графическим способом отображения модели "объект - свойство - отношение". В дипломном проекте используется модель «сущность-связь» (ER - модель) как наиболее распространенная, обеспечивающая вполне достаточное и адекватное описание объектов реального мира, их свойств и взаимосвязей. В приложении Б приведена схема разработанной базы данных.

Визуальная система пользователя предоставляет возможность взаимодействовать с СУБД посредством графического интерфейса. В СУБД MySQL графический пользовательский интерфейс реализуется с помощью форм специальных клиентских плагинов (HeidiSQL).

Формы являются типом объектов базы данных, которые обычно используется для отображения данных в базе данных. Форму можно также использовать как кнопочную форму, открывающую другие формы или отчеты базы данных, а также как пользовательское диалоговое окно для ввода данных и выполнения действий, определяемых введенными данными.

Большинство форм являются присоединенными к одной или нескольким таблицам и из базы данных. Форма не должна включать все поля из каждой таблицы или запроса, на основе которых она создается. Присоединенная форма получает данные из базового источника записей. Другие выводящиеся в форме сведения, такие как заголовок, дата и номера страниц, сохраняются в

макете формы. Графические элементы, такие как линии и прямоугольники, сохраняются в макете формы.

Связь между формой и ее источником записей создается при помощи графических объектов, которые называют. Наиболее часто используемым для вывода и ввода, данных типом элементов управления является поле.

Формы можно также открывать в режиме сводной таблицы или в режиме диаграммы для анализа данных. В этих режимах пользователи могут динамически изменять макет формы для изменения способа представления данных. Существует возможность упорядочивать заголовки строк и столбцов, а также применять фильтры к полям. При каждом изменении макета сводная форма немедленно выполняет вычисления заново в соответствии с новым расположением данных. В режиме сводной таблицы имеется возможность просматривать исходные данные или сводные данные, упорядочивать поля в областях фильтра, строк, столбцов и данных. Кроме этого можно визуально представлять данные, выбрав тип диаграммы и упорядочивая поля в областях фильтра, рядов, категорий и данных.

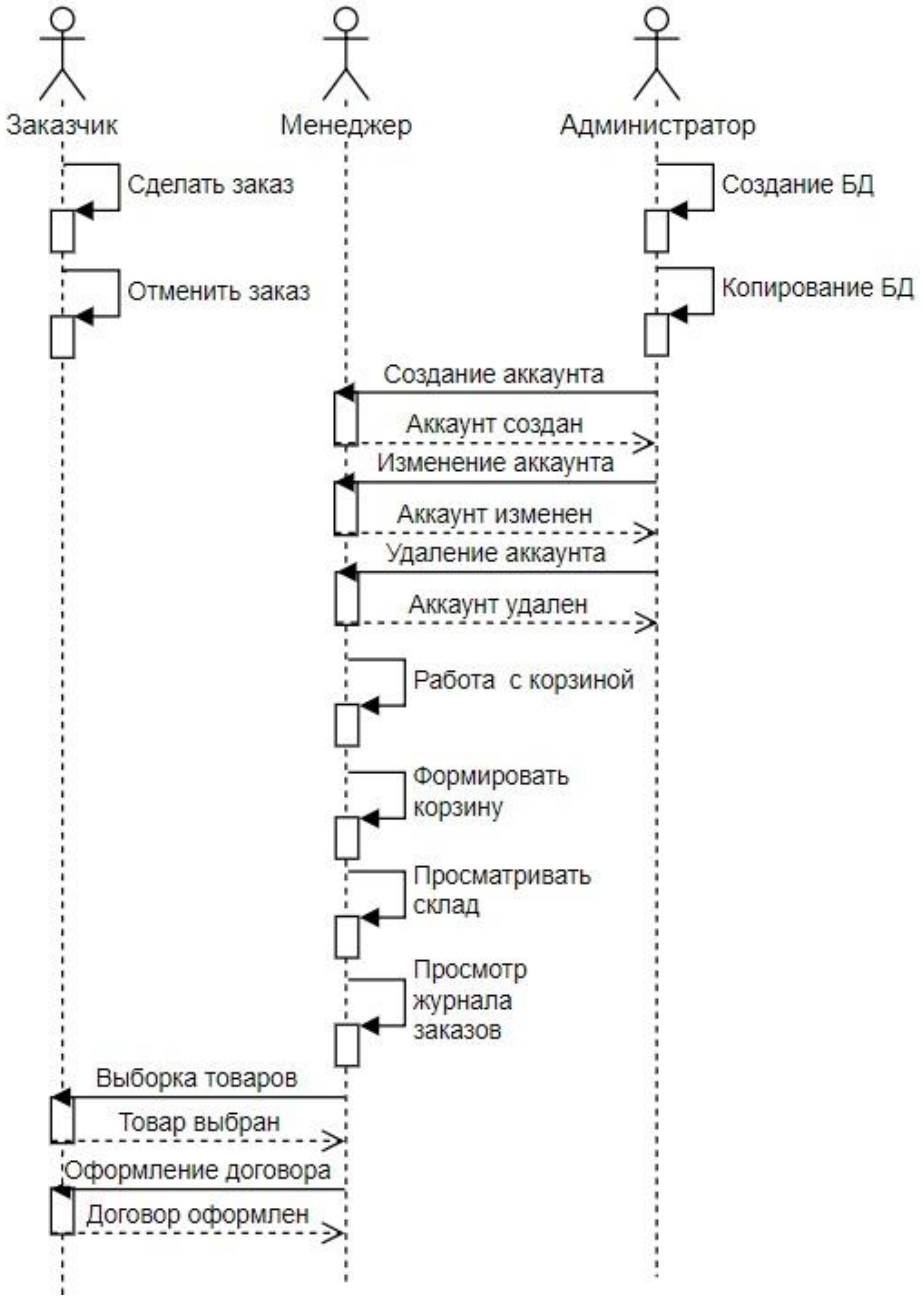


Рисунок 2.2 Схема движения товаров и ролей в организации

2.2 Архитектура программно-аппаратного комплекса

Данный программный комплекс предназначен для развертывания на базе платформы Intel. Рекомендованная конфигурация для сетевой работы AT Pentium IV – 2 ГГц, с объемом оперативной памяти не менее 2 ГБ (рекомендуется 4 ГБ), постоянного запоминающего устройства не менее 500 ГБ, видеокарта с объемом памяти не менее 2 ГБ, дисковод CD-ROM, USB

накопитель. Такие системные требования обусловлены применением требованиями ОС.

Для однопользовательской работы рекомендуется следующая конфигурация – IBM-совместимый компьютере типа IBM PC / AT Pentium IV Celeron – 1,7 ГГц, с объемом оперативной памяти не менее 2 ГБ (рекомендуется 4 ГБ), постоянного запоминающего устройства не менее 10 ГБ, видеокарта с объемом памяти не менее 128 Мб, дисковод CD-ROM, USB накопитель, монитор 17'', средства мультимедиа (звуковая карта, колонки) для распечатки отчетной информации формата А4 рекомендуется любой тип принтера. Для работы пользователя необходима ОС Win32 - Windows 8 и наличие СУБД MySQL.

Сетевая версия программного комплекса может быть установлена на WEB серверах с ОС Windows или ОС UNIX, использоваться на бездисковых рабочих местах, оснащенных средствами мультимедиа.

2.3 Разработка структуры данных

На основе анализа предметной области поставленной задачи по учету движения товарно-материальных ценностей и структуры приходных, расходных и расчетно-платежных документов разработана структура базы данных в среде MySQL HeidiSQL.

База данных по учету движения товарно-материальных ценностей включает следующие таблицы:

Таблица 2.1 - status – предназначена для хранения названий статусов.

Наименование поля	Тип данных	Ключ	Назначение
Id_status	int	PK	Уникальный идентификатор статуса
Name	Varchar(25)	_	Наименование статуса

Таблица 2.2 - order_logs – предназначена для хранения логов всех заказов и изменений статусов соответствующего заказа.

Наименование поля	Тип данных	Ключ	Назначение
Id_order_log	int	PK	Уникальный идентификатор лога заказа
Id_status	int	FK	Идентификатор статуса
Date	Datetime	_	Дата изменения статуса заказа
Id_order	int	FK	Идентификатор заказа

Таблица 2.3 - order – предназначена для обеспечения выполнения заказа.

Наименование поля	Тип данных	Ключ	Назначение
Id_order	int	PK	Уникальный идентификатор заказа
Id_user	int	FK	Идентификатор пользователя

Id_cart	int	FK	Идентификатор корзины заказа

Таблица 2.4 - cart – предназначена для реализации корзины заказа.

Наименование поля	Тип данных	Ключ	Назначение
Id_cart	int	PK	Уникальный идентификатор корзины заказа
Id_product	int	FK	Идентификатор товара
Count	int	_	Количество заказываемого товара
Id_order	int	FK	Идентификатор заказа

Таблица 2.5 - product – предназначена для хранения информации о товарах.

Наименование поля	Тип данных	Ключ	Назначение
Id_product	int	PK	Уникальный идентификатор товара
Name	Varchar(255)	FK	Наименование товара
Id_category	int	FK	Идентификатор категории товара
Id_measure	int	FK	Идентификатор меры товара
Price	Decimal(20,2)	_	Цена товара

Count	int	_	Количество товара на складе
Id_brand	Int	FK	Идентификатор бренда товара
Id_producer	Int	FK	Идентификатор производителя

Таблица 2.6 - measure – предназначена для хранения мер.

Наименование поля	Тип данных	Ключ	Назначение
Id_measure	int	PK	Уникальный идентификатор меры
Name	Varchar(255)	_	Наименование меры

Таблица 2.7 - brand – предназначена для хранения информации о брендах.

Наименование поля	Тип данных	Ключ	Назначение
Id_brand	int	PK	Уникальный идентификатор бренда
Name	Varchar(255)	_	Наименование бренда
Id_producer	int	PK,FK	Идентификатор производителя

Таблица 2.8 - producer – предназначена для хранения информации о производителях.

Наименование поля	Тип данных	Ключ	Назначение
Id_producer	int	PK	Уникальный идентификатор производителя
Name	Varchar(255)	_	Наименование производителя
Phone	Varchar(14)	PK,FK	Телефон производителя
Address	Varchar(255)	_	Адрес производителя

Таблица 2.9 - category – предназначена для хранения информации о категориях товаров.

Наименование поля	Тип данных	Ключ	Назначение
Id_category	int	PK	Уникальный идентификатор категории
Name	Varchar(255)	_	Наименование категории

Таблица 2.10 - priveleges – предназначена для обеспечения доступа менеджеров к категориям.

Наименование поля	Тип данных	Ключ	Назначение
Id_privelege	int	PK	Уникальный идентификатор привилегии
Id_category	Int	PK,FK	Идентификатор категории
Id_user	int	PK,FK	Идентификатор пользователя

Таблица 2.11 - users – предназначена для хранения информации о пользователях.

Наименование поля	Тип данных	Ключ	Назначение
Id_user	Int	PK	Уникальный идентификатор пользователя
login	Varchar(25)	_	Логин производителя
password	Varchar(30)	_	Пароль производителя
F_Name	Varchar(25)	_	Имя пользователя
L_Name	Varchar(20)	_	Фамилия пользователя
S_Name	Varchar(20)	_	Отчество пользователя
Id_group	Int	FK	Идентификатор группы

Таблица 2.12 - group – предназначена для хранения информации о группах.

Наименование поля	Тип данных	Ключ	Назначение
Id_group	int	PK	Уникальный идентификатор группы
Name	Varchar(50)	_	Наименование группы

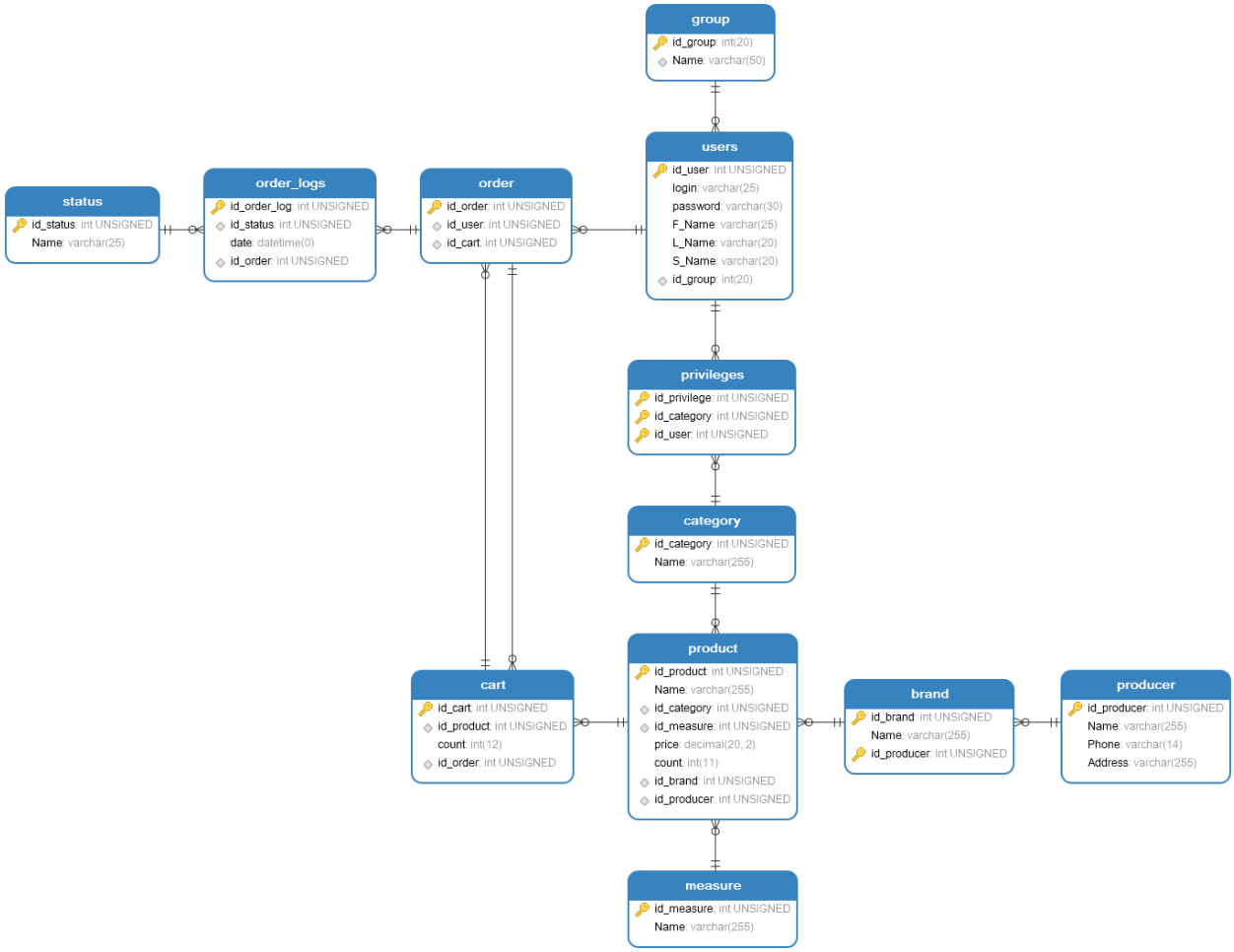


Рисунок 2.3 – Структура базы данных

2.4 Разработка программной части

Программное обеспечение разработано на основе паттерна MVC (Модель – Вид -Контроллер):

– Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние;

– Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели;

– Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

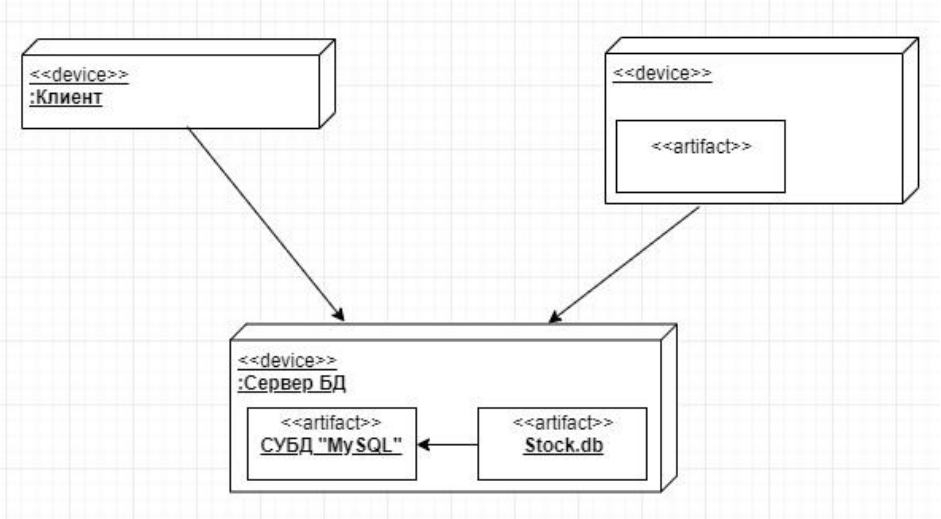


Рисунок 2.4 Сформированное приложение MVC

Фактически разработанное приложение состоит из следующих цепочек MVC – таблицы 2.13-2.18:

Таблица 2.13 - Авторизация

Модель	Представление	Контроллер
Отправка данных на сервер для авторизации. Отправка запроса на сервер для выхода.	Вывод экрана авторизации.	Ввод логина и пароля. Выход из пользователя.

Таблица 2.14 — База данных

Модель	Представление	Контроллер
Запрос списка backup'ов с сервера. Отправка запроса на создание backup'a. Отправка запроса на удаление backup'a.	Вывод списка backup.	Создание, скачивание и удаление backup'a. Поиск по имени и дате.

Таблица 2.15 — Пользователи.

Модель	Представление	Контроллер
Отправка запроса на создание, изменение и удаление пользователя.	Вывод списка пользователей.	Изменение, создание и удаление пользователя. Поиск по имени и дате.

Таблица 2.16 - Склад

Модель	Представление	Контроллер
Отправка списка продуктов корзины на сервер.	Вывод списка товаров.	Добавление товара в корзину и изменение количества товаров в корзине.

Таблица 2.17 — Заказы

Модель	Представление	Контроллер
Отправка новых состояний и запросов на повтор заказа на сервер.	Вывод списка заказов.	Изменение состояния заказов и их повтор.

Таблица 2.18 — Корзина

Модель	Представление	Контроллер
Отправка новых количества товаров, измененных данных заказчика и запросов на создание заказа на сервер.	Вывод списка товаров заказа.	Изменение данных заказчика, количества товаров.

2.5 Описание функций контроллера (бизнес логика)

Ниже описаны функции бизнес логики приложения.

Функция [set]Login реализует вход пользователя в систему и создаёт сессию, входными данными является логин пользователя и пароль, выходные данные отсутствуют.

Функция [get]Logout реализует выход пользователя из системы и завершение сессии, выходные и выходные данные отсутствуют.

Функция [get]Routes реализует вывод всех путей, доступных группе пользователя, группа пользователя берётся из созданной сессии, входные данные отсутствуют.

Функция [set]Backup реализует создание резервной копии базы данных, входные и выходные данные отсутствуют.

Функция [get]Backups реализует вывод всех созданных резервных копий баз данных, входные данные отсутствуют.

Функция [delete]Backup реализует удаление резервной копии, входным параметром является идентификационный номер резервной копии, выходные данные отсутствуют.

Функция [get]Users реализует вывод данных о пользователях, входные данные отсутствуют, выходными данными являются имя, фамилия, отчество, логин, наименование группы.

Функция [get]User реализует вывод данных о текущем пользователе, входные данные отсутствуют, выходными данными являются: идентификационный номер, логин, фамилия, имя, отчество, пути и наименование группы.

Функция [set]User реализует создание пользователя, входными данными являются: логин, пароль, имя, фамилия, отчество, группу, выходные данные отсутствуют.

Функция [change]User реализует изменение данных пользователя, входными данными являются идентификационный номер пользователя, логин, пароль, имя, фамилия, отчество, группа.

Функция [delete]User реализует удаление пользователя, входными данными является идентификационный номер, выходные параметры отсутствуют.

Функция [get]Categories реализует вывод всех существующих категорий товаров, входные данные отсутствуют.

Функция [get]Category реализует вывод категории, входными данными является идентификационный номер.

Функция [set]Category реализует создание категории, входными параметрами является наименование категории, выходные параметры отсутствуют.

Функция [change]Category реализует изменение категории, входными параметрами является идентификационный номер и наименование категории, выходные данные отсутствуют.

Функция [delete]Category реализует удаление категории, входными данными является идентификационный номер категории, выходные данные отсутствуют.

Функция [get]Products реализует получения списка всех товаров, входные данные отсутствуют.

Функция [get]Product реализует получение данных о товаре, выходным параметром является идентификационным номером.

Функция [get]ProductsByCategory реализует получение данных о товарах определённой категории, входными данными является идентификационный номер категории товаров.

Функция [set]Product реализует добавление товара, входными данными являются: наименование, идентификационный номер категории, идентификационный номер меры, цена, количество на складе, идентификационный номер бренда, идентификационный номер производителя, выходные данные отсутствуют.

Функция [change]Product реализует изменение данных пользователя, входными данными являются: идентификационный номер товара, наименование, идентификационный номер категории, идентификационный номер меры, цена, количество на складе, идентификационный номер бренда, идентификационный номер производителя, выходные данные отсутствуют.

Функция [delete]Product реализует удаление товара, входными данными является идентификационный номер товара, выходные данные отсутствуют.

Функция [get]Price реализует получение цены товара, выходными данными является идентификационный номер продукта, выходным параметром является цена товара.

Функция [get]Producers реализует получение списка производителей, входные данные отсутствуют.

Функция [get]Producer реализует получение данных о производителе, входными данными является идентификационный номер производителя.

Функция [set]Producer реализует создание производителя, входными данными является: наименование, телефон, адрес, выходные данные отсутствуют.

Функция [delete]Producer реализует удаление производителя, входными параметрами является идентификационный номер производителя, выходные данные отсутствуют.

Функция [change]Producer реализует изменение данных производителя, входными данными является идентификационный номер производителя, наименование, телефон, адрес, выходные данные отсутствуют.

Функция [get]Brands реализует вывод всех брендов, входные данные отсутствуют.

Функция [set]Brand реализует создание бренда, входными параметрами является идентификационный номер производителя и наименование.

Функция [change]Brand реализует изменение бренда, входными параметрами является идентификационный номер, идентификационный номер производителя, наименование, выходные параметры отсутствуют.

Функция [delete]Brand реализует удаление бренда, входным параметром является идентификационный номер бренда, выходные параметры отсутствуют.

Функция [get]BrandByProducer реализует получение списка брендов производителя, входным параметром является идентификационный номер производителя.

Функция [set]Status реализует установку статуса заказа, входными параметрами являются идентификационный номер заказа и идентификационный номер статуса, выходные параметры отсутствуют.

Функция [set]Privilege реализует установку привилегии, входными параметрами является идентификационный номер пользователя и идентификационный номер категории, выходные данные отсутствуют.

Функция [delete]PrivilegeUser реализует удаление привилегии пользователя, входными параметрами является идентификационный номер пользователя, выходные параметры отсутствуют.

Функция [change]Privilege реализует изменение привилегий пользователя, входными параметрами являются идентификационный номер пользователя, идентификационный номер категории, идентификационный номер привилегии.

Функция [get]PrivilegeByUser реализует вывод привилегий по пользователю входным параметром является идентификационный номер пользователя.

Функция [get]Privileges реализует вывод всех привилегий, входные и выходные параметры отсутствуют.

Функция [get]CategoriesByUser реализует вывод категорий по пользователю, входные и выходные параметры отсутствуют.

Функция [set]ProductInCart реализует добавление товара в корзину, входными данными являются идентификационный номер и количество товара.

Функция [get]UserCart реализует получение корзины пользователя, входные и выходные параметры отсутствуют.

Функция [change]ProductInCart реализует изменение товара в корзине, входными параметрами являются идентификационный номер корзины, идентификационный номер товара, количество, выходные данные отсутствуют.

Функция [get]ClientHistoryOrder реализует выводи информации истории заказов клиента, входным параметром является идентификационный номер клиента, выходными данными является список истории заказов клиента.

Функция [get]UserHistoryOrder реализует вывод истории заказов пользователя, входные параметры отсутствуют, выходными данными является список заказов пользователя.

Функция [change]ClientByCart реализует изменение данных клиента через корзину, входными данными является идентификационный номер корзины, адрес, телефон, имя, фамилия, отчество, электронная почта, выходные данные отсутствуют.

2.4 Разработка модели

Функции контроллера выполняют функции view, insert, delete, read, wrire для моделей.

Модели – это классы, реализованные на PHP, для связи вида, бизнес логики и СУБД (рисунок 2.5).

Фактически данные классы (рисунок 2.5) транслируют данные в таблицы 2.1 -2.18.

В результате реализации приложения на основании паттерна Модель – Вид-Контроллер, реализуется приложение, структура которого показана на рисунке 2.5.

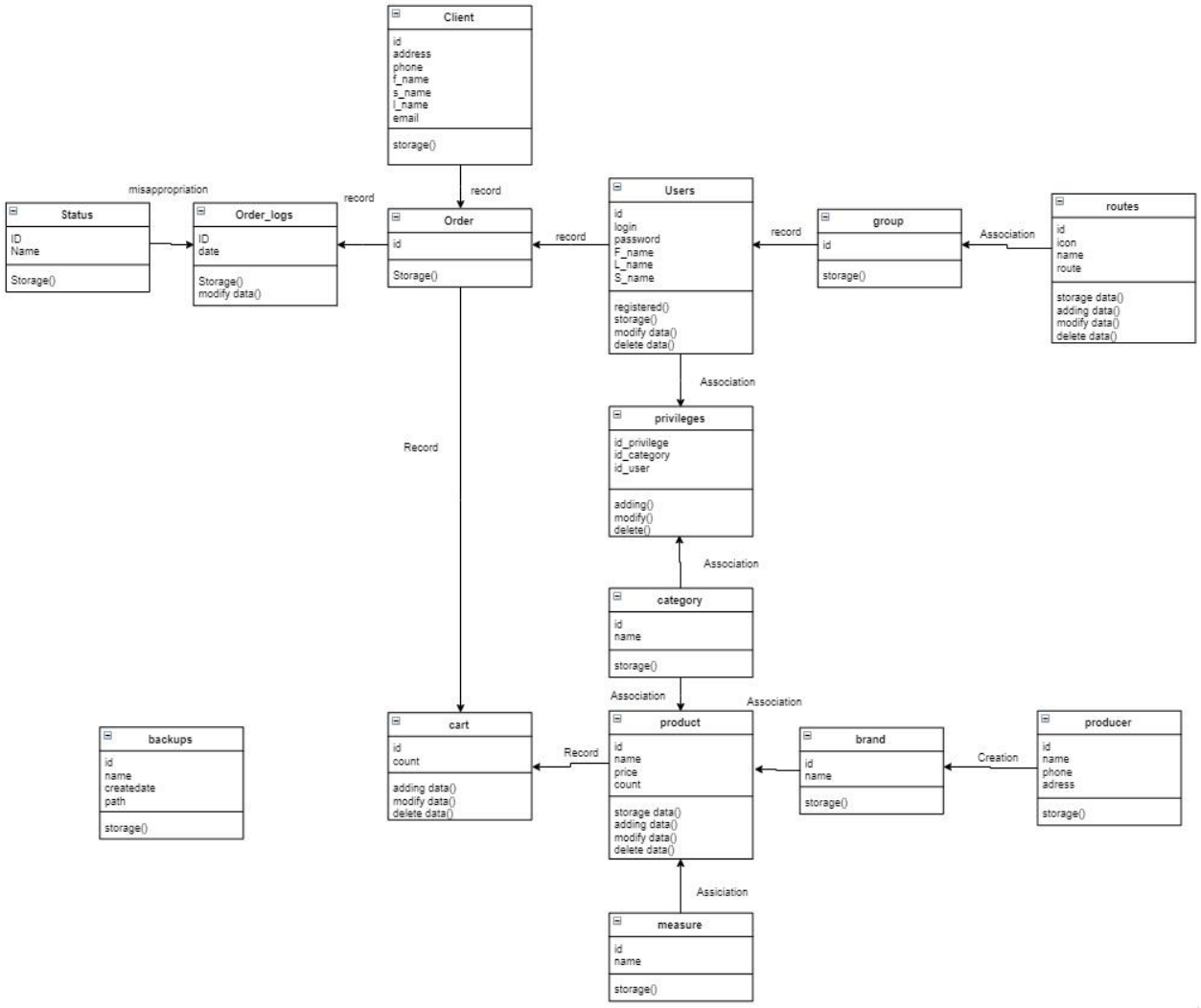


Рисунок 2.5 Классы моделей.

Данные и команды между блоками приложения передаются в формате JSON, с помощью стандартного механизма REST API.

Всё взаимодействие с сервером сводится к 4 операциям (4 - это необходимый и достаточный минимум, в конкретной реализации типов операций может быть больше):

получение данных с сервера (в формате JSON): добавление новых данных на сервер, модификация существующих данных на сервере, удаление данных на сервере.

Операция получения данных не может приводить к изменению состояния сервера.

Для каждого типа операции используется свой метод HTTP-запроса: получение – GET, добавление – POST, модификация – PUT, удаление – DELETE.

Форматы данных передаваемых между формами интерфейса и серверами выглядят следующим образом:

/login
Входные данные:
{"username":"admin","password":"admin"}

Выходные данные:
{"success":true,"message":"Ok","group":"1"}

/user

Входные данные:
-

Выходные данные:
{"success":true,"user":{"id":"2","username":"admin","f_name":"Administrator","s_name":"2","l_name":"1","permissions":{"visit":["1","2"],"role":"Admin"},"avatar":""}}

/routes

Входные данные:
-

Выходные данные:
[{"id":"1","icon":"hdd","name":"\u0411\u0430\u0437\u0430 \u0434\u0430\u043d\u043d\u044b\u0445","route":"\/backups","menuParentId":null,"id_group":"1"},{"id":"2","icon":"user","name":"\u041f\u043e\u043b\u044c\u0437\u043e\u0432\u0430\u0442\u0435\u043b\u0438","route":"\/user","menuParentId":null,"id_group":"1"}]

/users

Входные данные:
-

Выходные данные:
{"data":[{"id":"2","nickName":"admin","f_name":"Administrator","password":"admin","l_name":"1","s_name":"2","group":"Admin"},{"id":"3","nickName":"3","f_name":"3","password":"3","l_name":"3","s_name":"3","group":"Manager"},{"id":"7","nickName":"4","f_name":"1","password":"5","l_name":"3","s_name":"","group":"Admin"}]}

/backups

Входные данные:
-

Выходные данные:

{"data":[{"id":"5","name":"myphp-backup-stock-05052019_203950.sql.gz","createDate":"2019-05-05 20:39:00","path":"myphp-backup-files"},{"id":"6","name":"myphp-backup-stock-05052019_203953.sql.gz","createDate":"2019-05-05 20:39:00","path":"myphp-backup-files"},{"id":"8","name":"myphp-backup-stock-06052019_084830.sql.gz","createDate":"2019-05-06 08:48:00","path":"myphp-backup-files"},{"id":"9","name":"myphp-backup-stock-06052019_090155.sql.gz","createDate":"2019-05-06 09:01:00","path":"myphp-backup-files"}]}

/logout

Входные данные:
-

Выходные данные:
{"res":"true"}

add /user

Входные данные:
{"name":"llo","s_name":"qwe","f_name":"qwe","nickName":"qwe","password":"qwe","group":"1"}

Выходные данные:
{"success":true,"message":"Ok"}

delete /user/8

Входные данные:
{"id":"8"}

Выходные данные:
{"success":true,"message":"User deleted"}

change /user/7

Входные данные:
{"name":"1","s_name":"2","f_name":"1","nickName":"4","password":"5","group":"Admin","id":"7"}

Выходные данные:
{"success":true,"message":"User changed"}

add /backup

Входные данные:
-

Выходные данные:
{"success":true,"message":"myphp-backup-files\/myphp-backup-stock-18052019_082042.sql.gz"}

delete /backup/5

Входные данные:
{"id":"5"}

Выходные данные:
{"success":true,"message":"Backup deleted"}

/userHistoryOrder

Входные данные:
-

Выходные данные:
{"data":[{"id":"1","status":"1","date":"2019-05-05 14:30:00"}]}

/products

Входные данные:
-

Выходные данные:
{"data":[{"id_product":"1","Name":"\u041a\u0430\u0440\u0442\u043e\u0448\u043a\u0430","id_category":"1","id_measure":"1","price":"12.00","count":"123","id_brand":"1","id_producer":"1"}]}

/userCart

Входные данные:
-
Выходные данные:
{"data":[{"title":"\u041a\u0430\u0440\u0442\u043e\u0448\u043a\u0430","id_product":"1","count":"2"}]}

2.5 Формирование шаблонов страниц пользовательского интерфейса на основе модели GOMS

Количественные методы помогают свести спорные вопросы в оценке качества интерфейса к простым вычислениям.

Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель GOMS (the model of goals, objects, methods and selection rules) – модель целей, объектов, методов и выбор правил.

Эта модель основана на оценке скорости ввода. Время, требуемое для выполнения какой-то задачи системой пользователь-компьютер, является суммой всех временных интервалов, которые потребовались системе на выполнение элементарных действий, составляющих данную задачу.

В таблице 2.19 приведены стандартные средние интервалы для некоторых действий, выполняемых различными пользователями.

Для того чтобы определить, в какие моменты будут проходить ментальные операции, необходимо следовать некоторым правилам.

Правило 0. Начальная расстановка операторов М. Оператор М устанавливается перед всеми операторами К (нажатие клавиши) и Р, предназначенными для выбора команд, если Р указывает на аргументы этих команд оператор М не ставится.

Таблица 2.19 – Интервалы некоторых жестов, выполняемых пользователями персональных компьютеров

Название жеста	Условное обозначение	Среднее время, с
Нажатие клавиши	К	0,2
Указание (на какую-либо позицию на экране монитора)	Р	1,1
Перемещение руки с клавиатуры на мышь или с мыши на клавиатуру	Н	0,4
Ментальная задержка	М	1,35
Время задержки ответа компьютера	R	-

Правило 1. Удаление ожидаемых операторов М. Если оператор, следующий за оператором М ожидаемый с точки зрения оператора, предшествующего М, то этот оператор М может быть удален и последовательность РМК превращается в РК.

Правило 2. Удаление операторов М внутри когнитивных единиц.

Если строка типа «МКМКМК…» принадлежит когнитивной единице, то следует удалить все операторы М, кроме первого. Когнитивной единицей

является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент.

Правило 3. Удаление M перед последовательными разделителями. Если оператор K означает лишний разделитель, стоящий в конце когнитивной единицы (например разделитель команды, следующий сразу за разделителем аргумента этой команды) то следует удалить оператор M, стоящий перед ним.

Правило 4. Удаление операторов M, которые являются прерывателями команд. Если оператор K является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор M, стоящий перед ним. Но если оператор K является разделителем для строки аргументов или любой другой изменяемой строки, то M следует сохранить перед ним.

Правило 5. Удаление перекрывающихся операторов M. Любую часть операторов M, которая перекрывает оператор R, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Если имеется модель интерфейса, то с помощью GOMS можно определить время, необходимое пользователю для выполнения любой, четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс. Чтобы ответить на него, необходимо воспользоваться мерой, применяемой в теории информации.

Чтобы сделать правильную оценку времени, необходимого для выполнения задачи с помощью самого быстрого интерфейса, прежде всего, следует определить минимально количество информации, которое пользователь должен ввести, чтобы выполнить задачу. Это минимальное количество не зависит от модели интерфейса. Если методы работы, используемые в предполагаемом интерфейсе, требуют введения такого количества информации, которое превышает минимальное, это означает, что пользователь делает лишнюю работу и поэтому интерфейс следует

усовершенствовать. С другой стороны, если от пользователя требуется ввести именно, то количество информации, которое необходимо для выполнения задачи, то для этой задачи интерфейс нельзя сделать более производительным путем изменения количества информации. В этом случае пути улучшения интерфейса (а также много путей для ухудшения) все же остаются, но, по крайней мере, данная цель повышения производительности будет уже достигнута.

Информационно-теоретическая производительность определяется так же, как понятие производительности определяется в термодинамике – отношением мощности на выходе к мощности на входе процесса. Если в течение какого-то периода времени электрогенератор, работающий от двигателя мощностью в 1000 ватт, производить 820 ватт, то он имеет производительность 820/1000=0,82. Производительность также часто обозначается через проценты.

Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Так же как и в отношении физической производительности, параметр E может изменяться в пределах от 0 до 1. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1 (чтобы избежать деления на 0).

Производительность E может равняться и 0 в случаях, когда пользователь должен ввести информацию, которая бесполезна для выполнения задачи.

В параметре E учитывается только информация, необходимая для задачи и информация, вводимая пользователем.

Так для выбора одного варианта из четырех возможных достаточно двух бит информации, из восьми – трех, из 16 – 4-х и т.д.

Суммарное количество передаваемой информации определяется, как $\log_2 n$.

Количество информации для каждого варианта определяется как $\frac{1}{n}\log_2 n$.

Если вероятности для каждого варианта или элемента не являются равными и i-й вариант имеет вероятность P(i), то информация, передаваемая этим вариантом, определяется как $p(i) * \log_2 \frac{1}{p(i)}$.

Количество информации по всем вариантам является их суммой по всем вариантам выражения, которое при равновероятных вариантах сводится к первому выражению.

В результате использования данного способа были сформированы макеты интерфейса и реализованы шаблоны страниц (выполнено в сервисе формирования мокапов интерфейсных страниц https://ninjamock.com/) – рисунок 2.6 – 2.12.

Рисунок 2.6 Шаблон страницы «Логин»

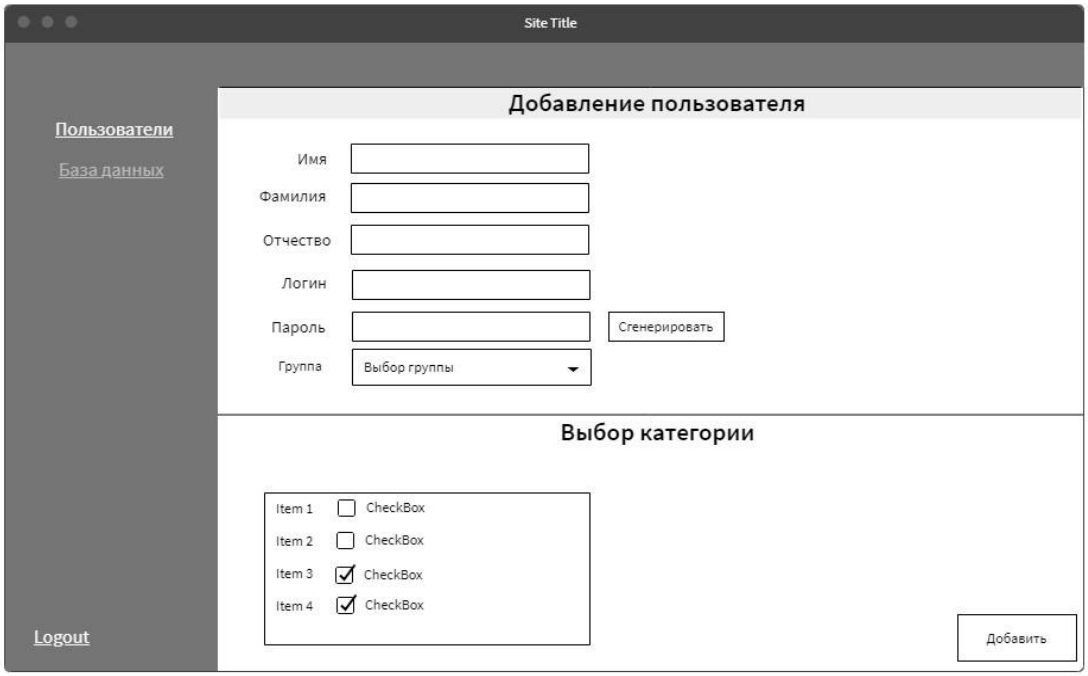


Рисунок 2.7 Шаблон страницы «Добавление пользователя»

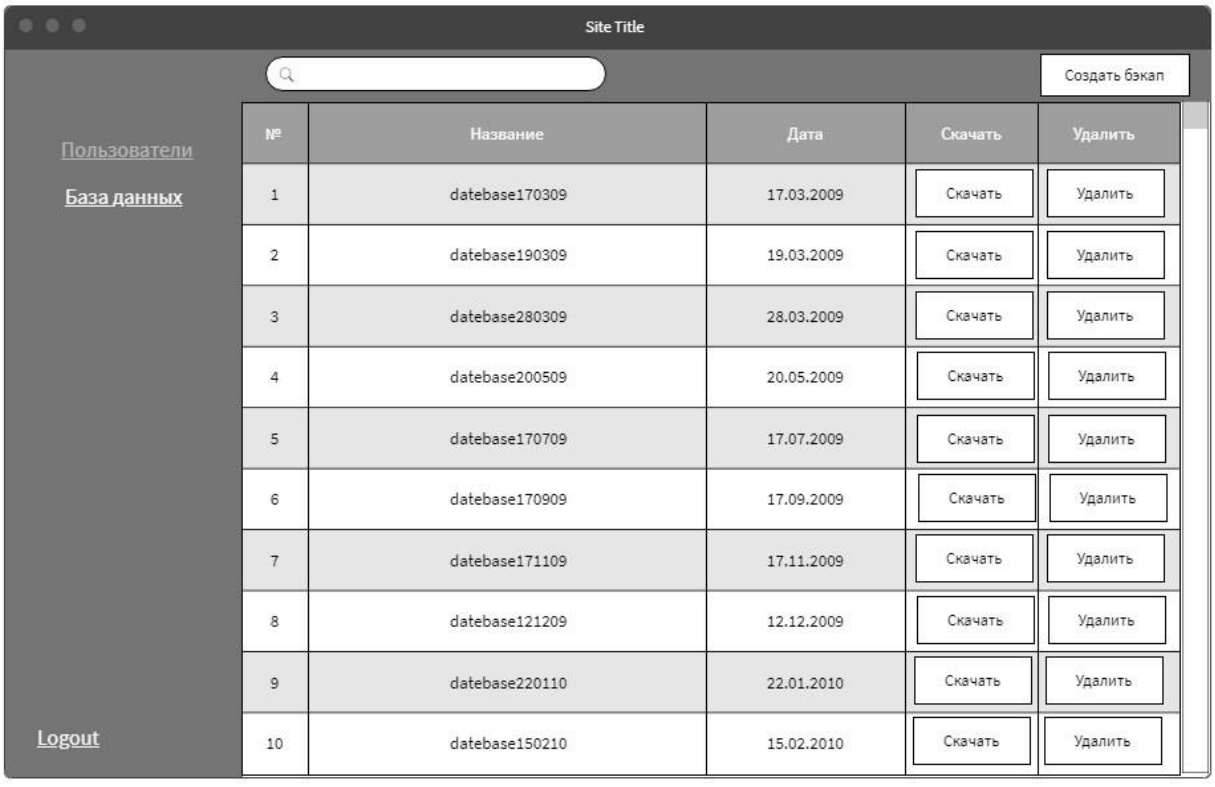


Рисунок 2.8 Шаблон страницы «База данных»

Рисунок 2.9 Шаблон страницы «Пользователи»

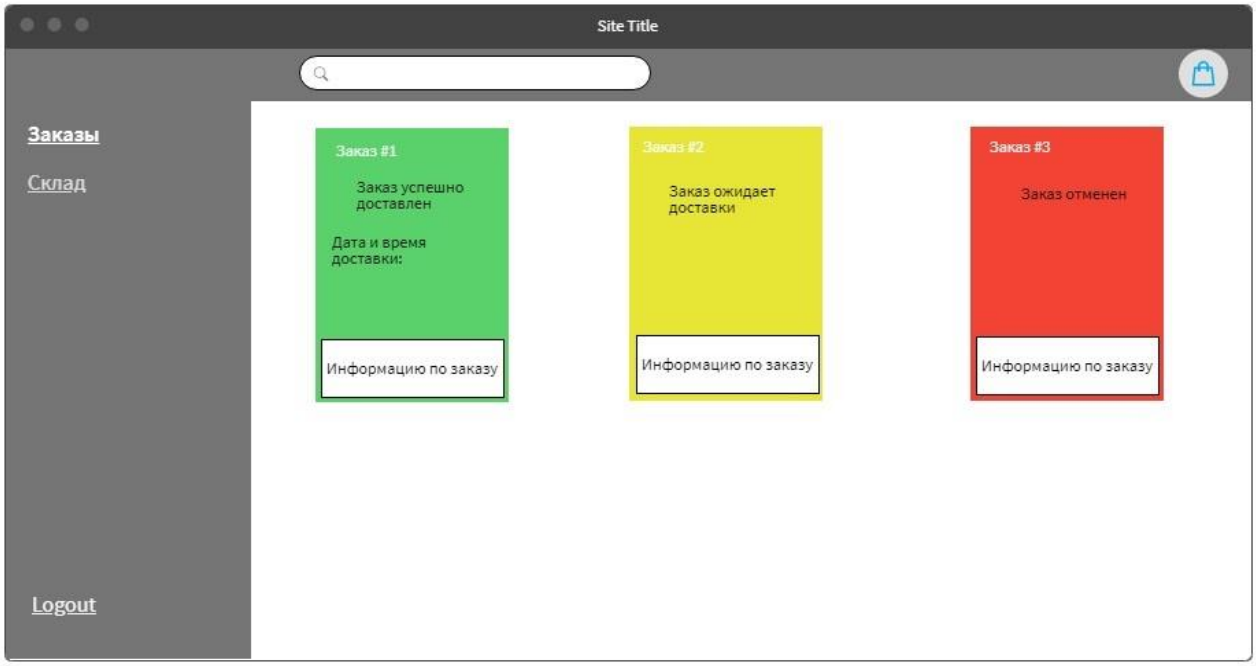


Рисунок 2.10 Шаблон страницы «Заказы»

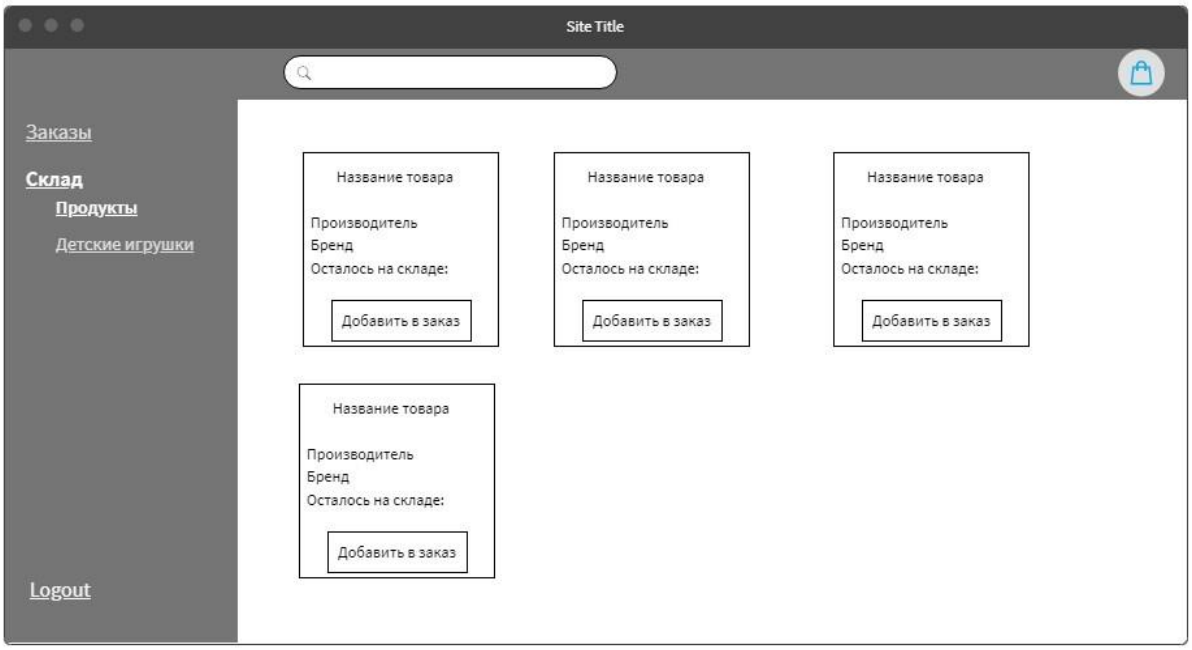


Рисунок 2.11 Шаблон страницы «Склад»

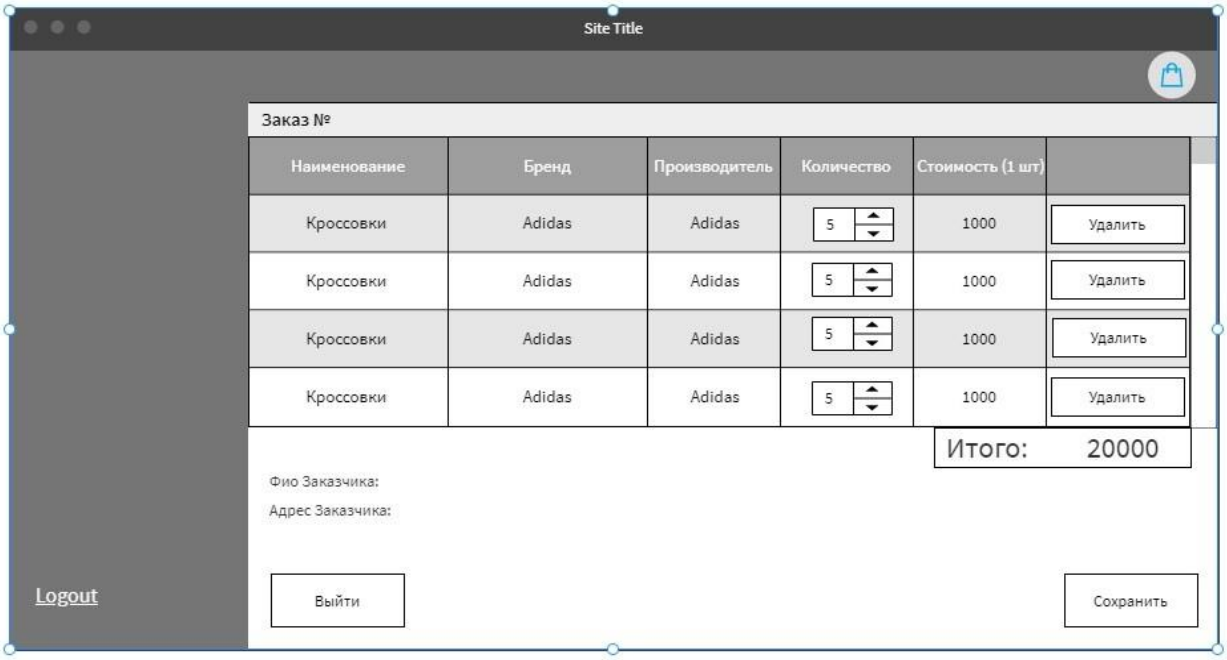


Рисунок 2.12 Шаблон страницы «Формирование заказа»

2.6 Описание работы приложения

После запуска web приложения [localhost] открывается страница аутентификации, в котором пользователь вводит логин и пароль (рисунок 2.13).

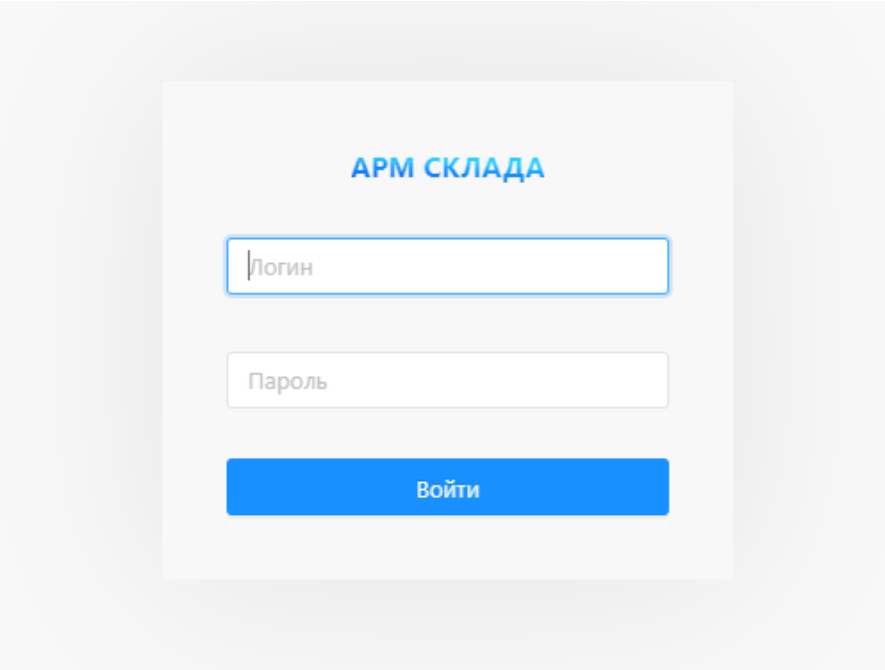


Рисунок 2.13 – Форма аутентификации.

Первоначально требуется ввести логин и пароль «admin» - «admin» для настройки системы. Для чего оператор переходит на страницу администратора (рисунок 2.14). На странице «Операций с пользователем» администратор имеет возможность просмотреть список пользователей, изменить, удалить и создать нового пользователя.

Также администратор имеет возможность поиска пользователя по имени (рисунок 2.16). Для чего с права, на странице имеется кнопка «Добавить» и меню «Операции».

Страница работы с backup (рисунок 2.15) позволяет увидеть уже созданные копии СУБД, а также создать, удалить backup, с помощью меню «Операции».

Вход в систему под пользователем менеджеры, позволяет увидеть три пункта меню: «Склад», «Заказы», «Корзина».

Пункт меню «Склад» содержит описание складских запасов в виде карточек. Каждая карточка содержит информацию о наименовании, стоимости и количестве товара (рисунок 2.18).

Пункт меню «Заказы» содержит карточки трех цветов: зеленые, желтые и красные. Зелены карточки – товар заказан и находится на обработке. Желтые

карточки – товар ждет доставки. Красные карточки – товар ждет доставки, сроки доставки от поставщика просрочены.

Пункт меню «Корзина» - содержит товары, внесенные менеджером для формирования заказа.

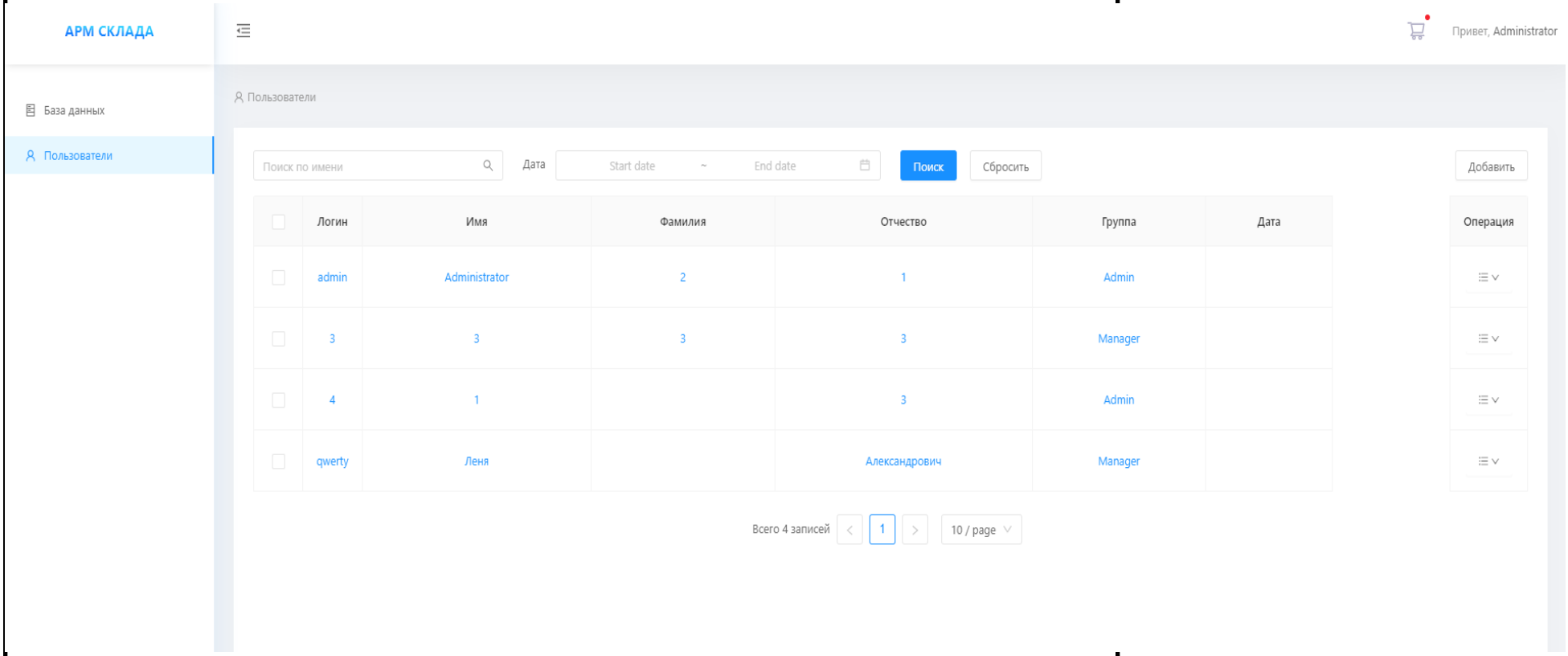


Рисунок 2.14 Страница операций с пользователем (Администратор)

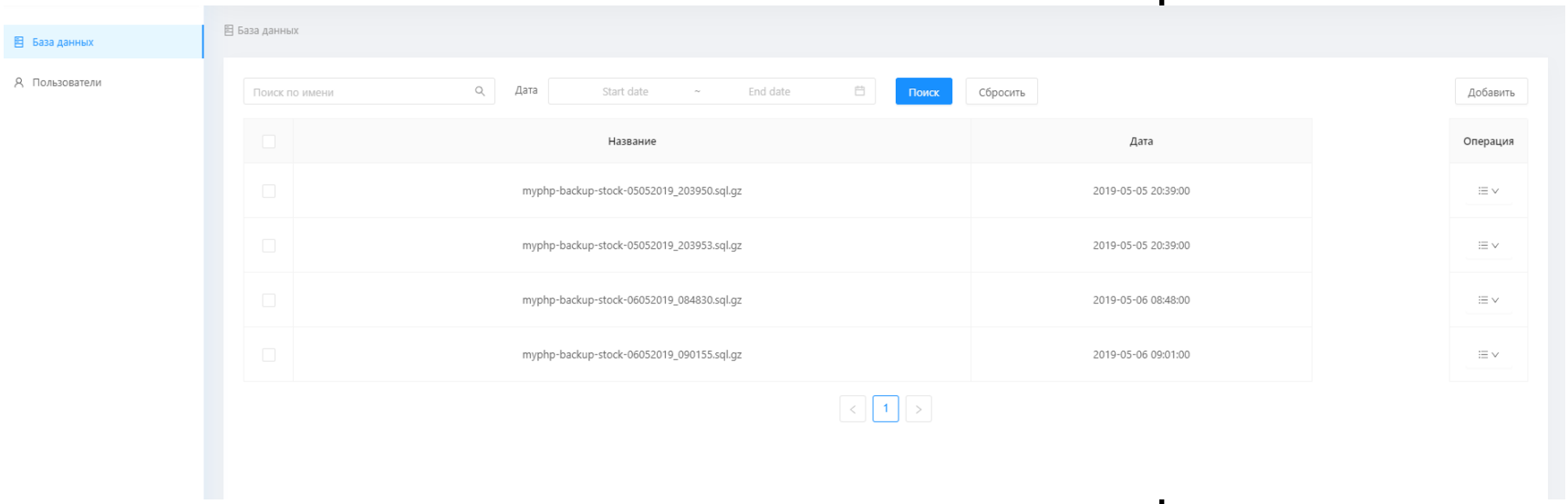


Рисунок 2.15 Страница операций с Backup СУБД (Администратор)

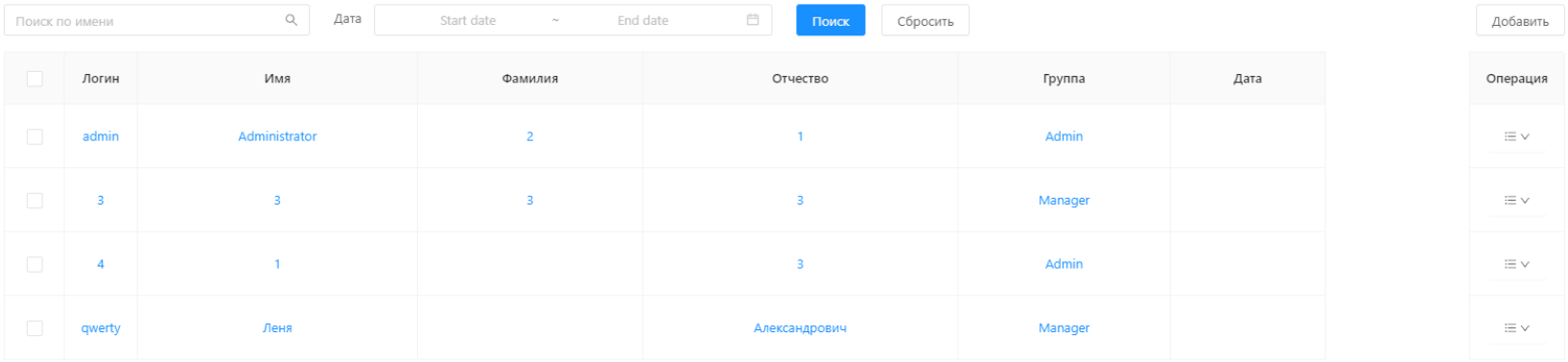


Рисунок 2.16 Страница поиска пользователей (Администратор)

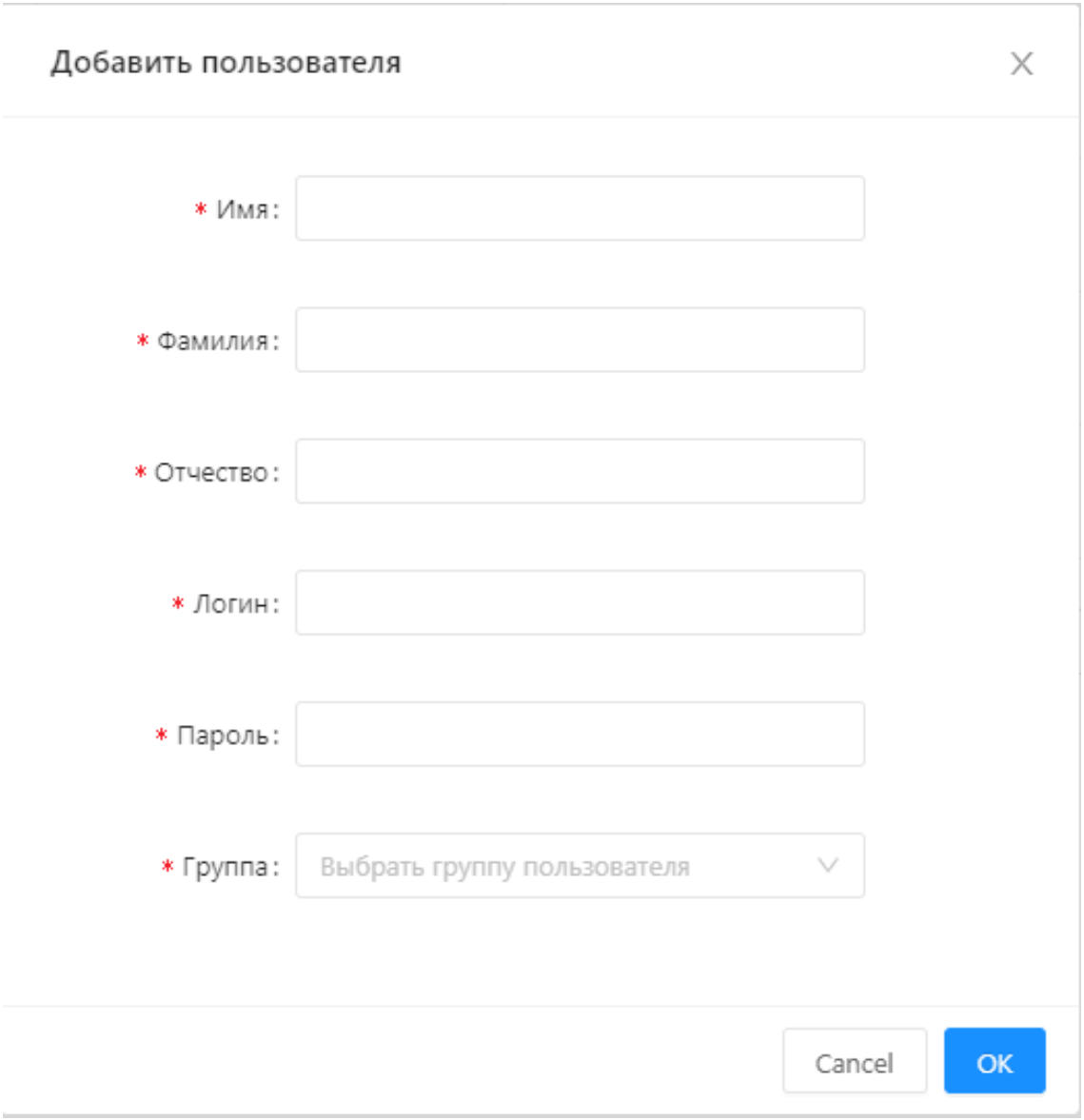


Рисунок 2.17 Страница добавления нового пользователя (Администратор)

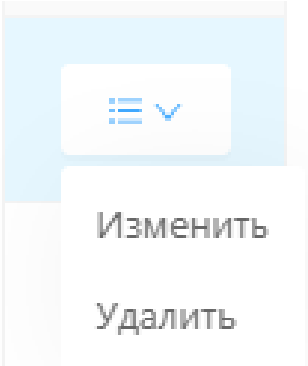


Рисунок 2.18 Меню операции с пользователем (Администратор)

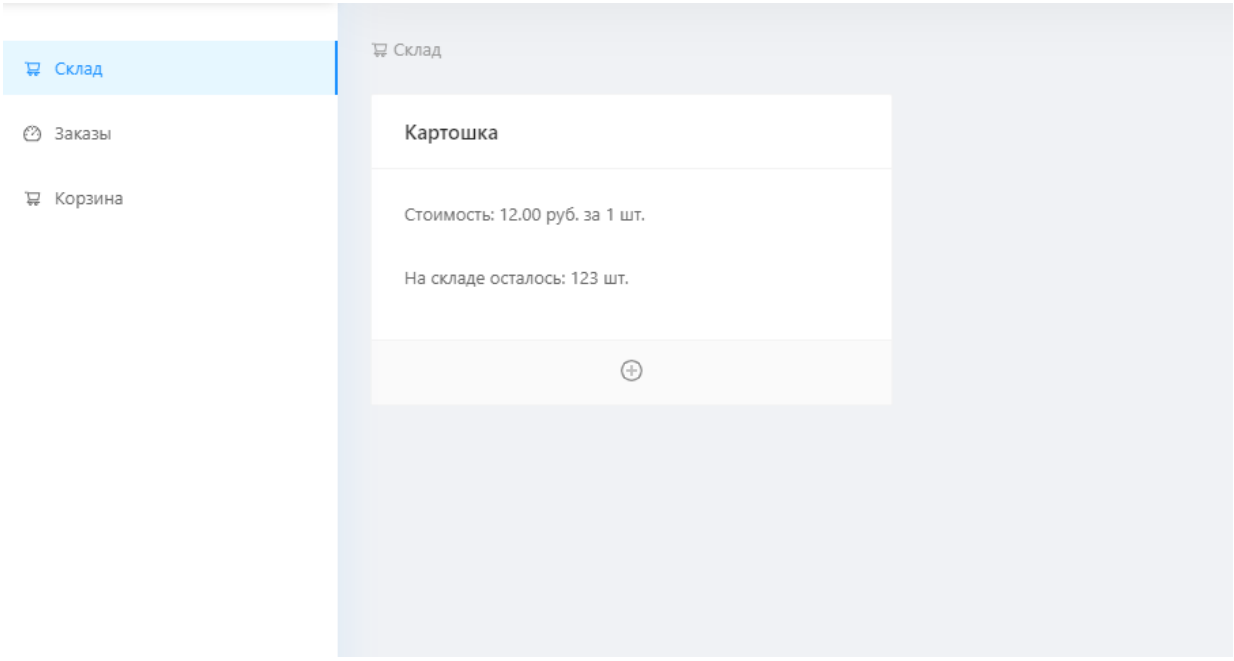


Рисунок 2.18 Страница складского учета (Пользователь)

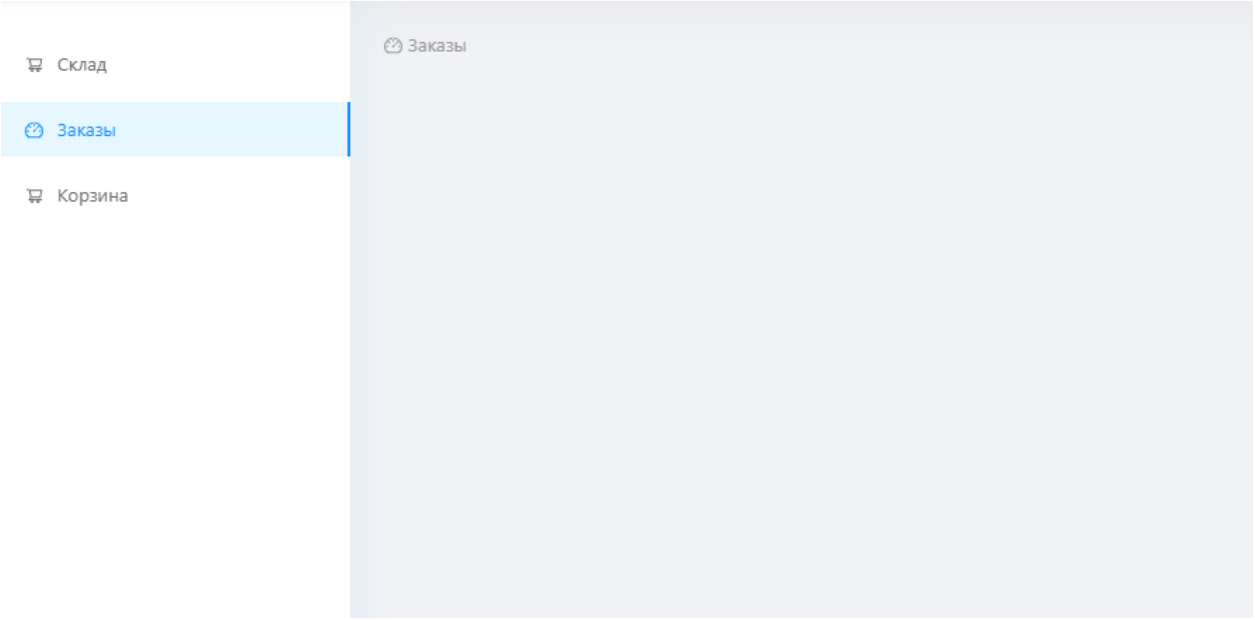


Рисунок 2.19 Страница заказов (Пользователь)

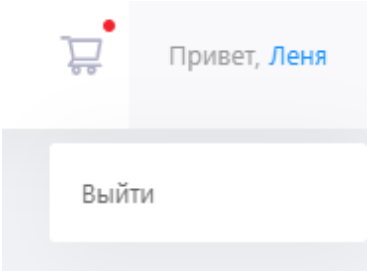


Рисунок 2.19 Меню «Корзина» (Пользователь)

3 Тестирование и развертка программного обеспечения

Для реализации данного дипломного проекта была выбрана среда разработки NetBeans.

Эта среда — бесплатная, быстрая, крос-платформенная, хорошо понимает язык PHP. В ней приятно писать PHP-код, потому что присутствует все необходимые для этого инструменты — подсветка, completion из всех подключенных файлов, браузер классов PHP, всплывающая документация по стандартным функциям и так далее. Вместе с тем, это не просто текстовый редактор. NetBeans умеет скачивать и закачивать файлы на удаленные сервера по FTP и SFTP, имеет развитый SQL-терминал и полностью автоматизирует работу с SVN или CVS.

3.1 Создание проекта в NetBeans

Ниже описана технология создания проекта в среде Eclipse.

Первый шаг, открытие помощника создания проекта - **File > New > Project > PHP > PHP Aplication.**

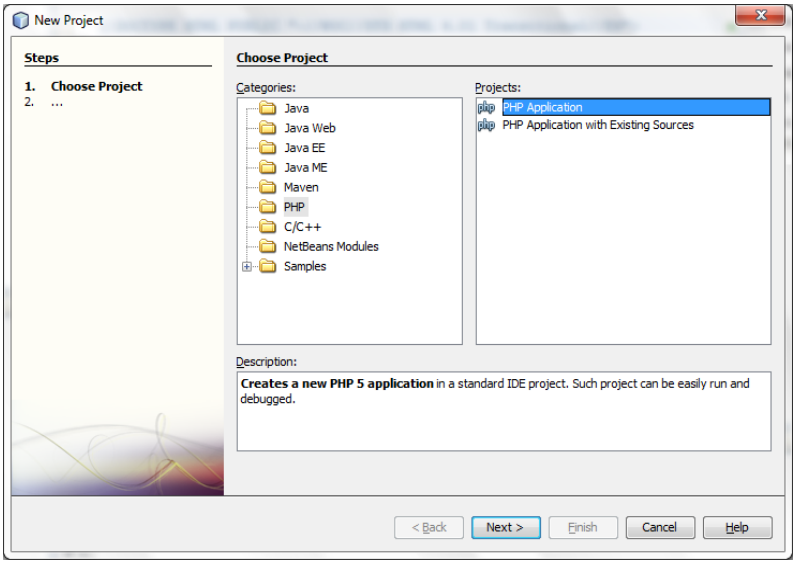


Рисунок 3.1 – Создание нового проекта

Нажмите кнопку **Next** и перейдите в диалог настройки пространства имен проекта (рисунок 3.2).

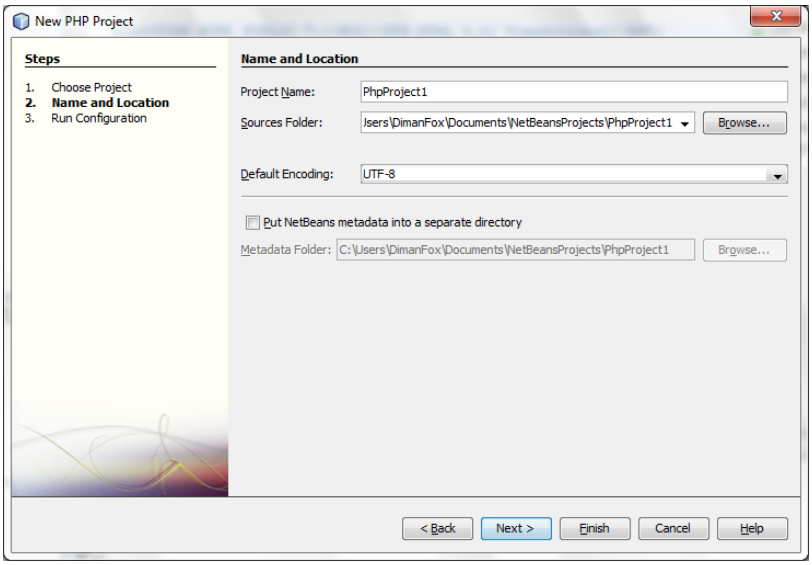


Рисунок 3.2 – Настройка пространства имен проекта

Нажмите кнопку **Next** и перейдите в диалог конфигурации проекта (рисунок 3.3).

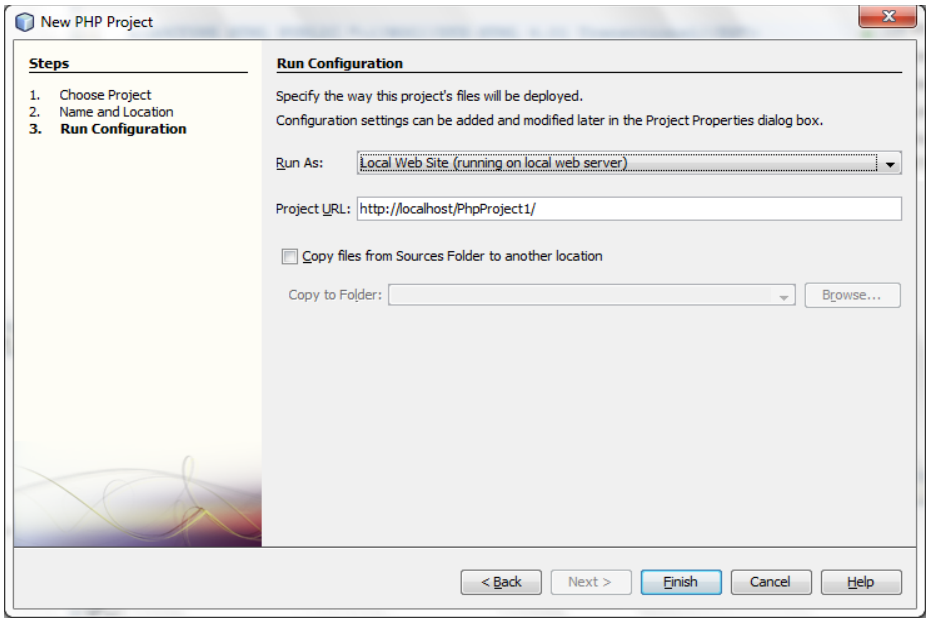


Рисунок 3.3 – Настройка конфигурации проекта

В результате вышеперечисленных операций появляется окно нового проекта, в котором присутствует структура каталога проекта и сгенерированные автоматически файлы настройки (рис. 3.4).

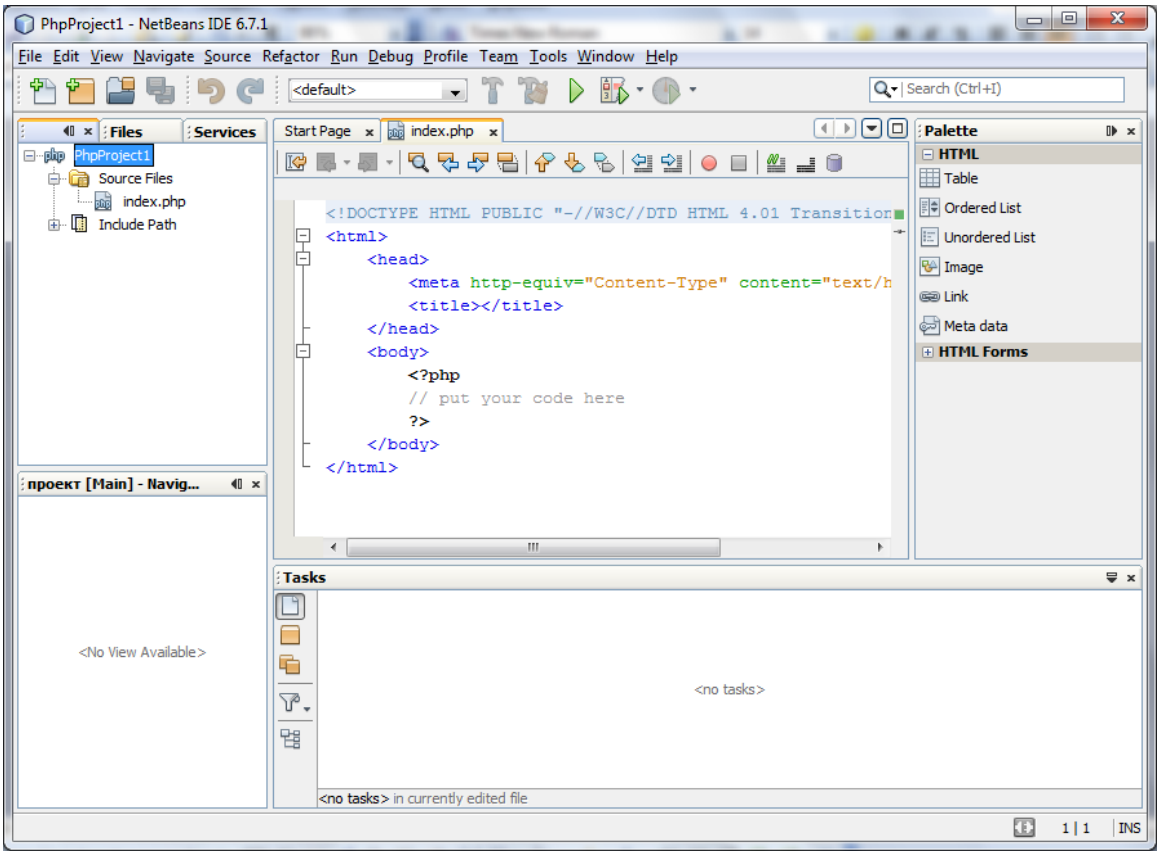


Рисунок 3.4 – Окно нового проекта

По умолчанию в проект добавлен файл index.php, который является главным файлом проекта, и к которому будут подключаться остальные файлы.

Для добавления новых файлов в проект нужно воспользоваться пунктом меню - **File > New > New File.**

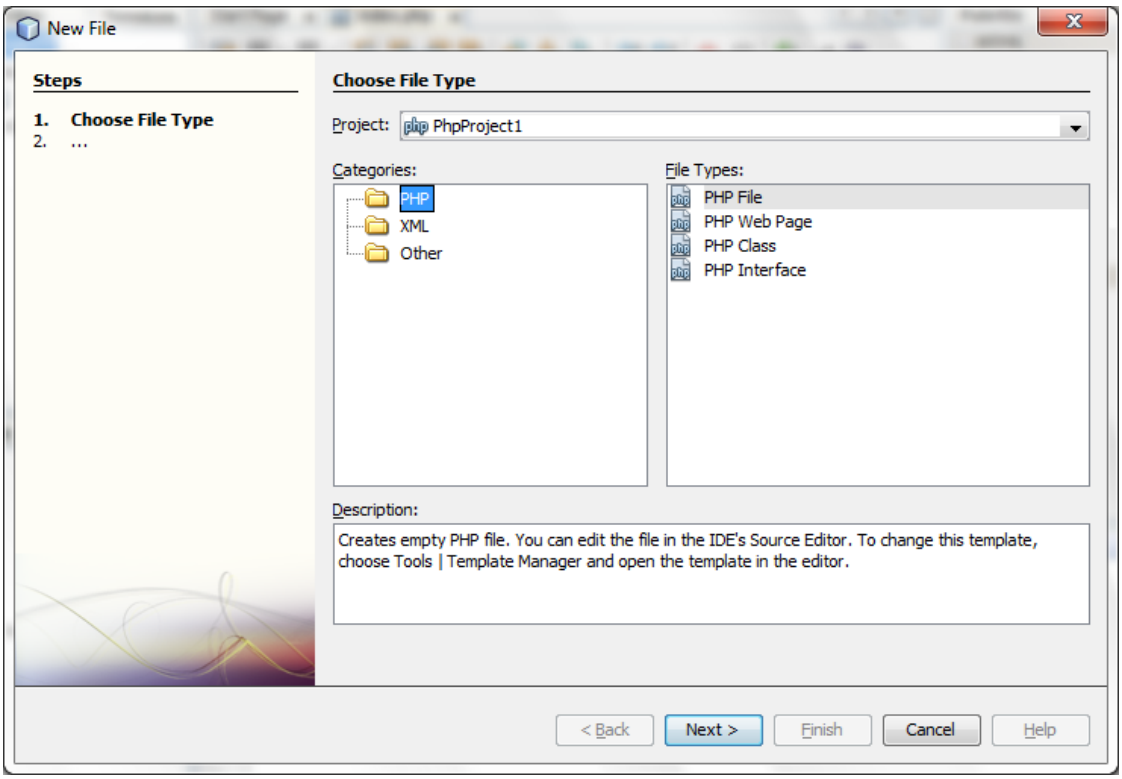


Рисунок 3.5 – Добавление нового файла в проект

После того как выбрали тип файла, который нужно добавить в проект, следует нажать кнопку «Next» для того чтобы задать имя файлу.

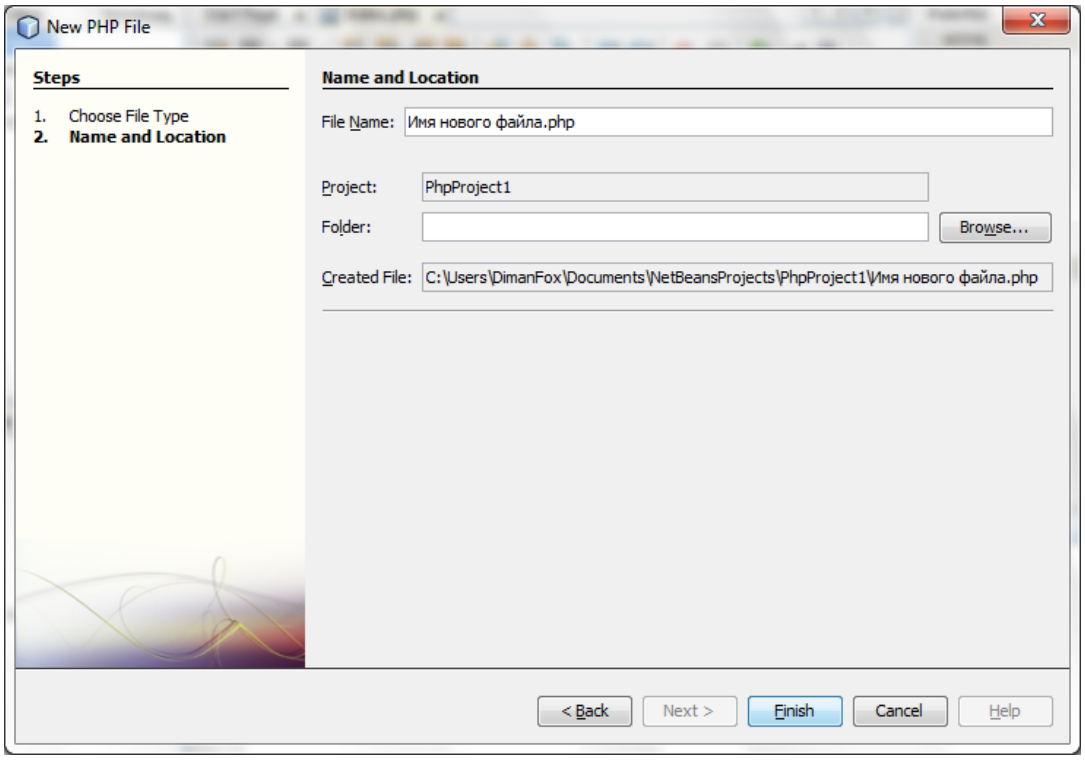


Рисунок 3.6 – Добавление нового файла в проект

После того как задали имя новому файлу, можно нажать кнопку «Finish» и новый файл добавиться в проект с минимум кода-шаблоном.

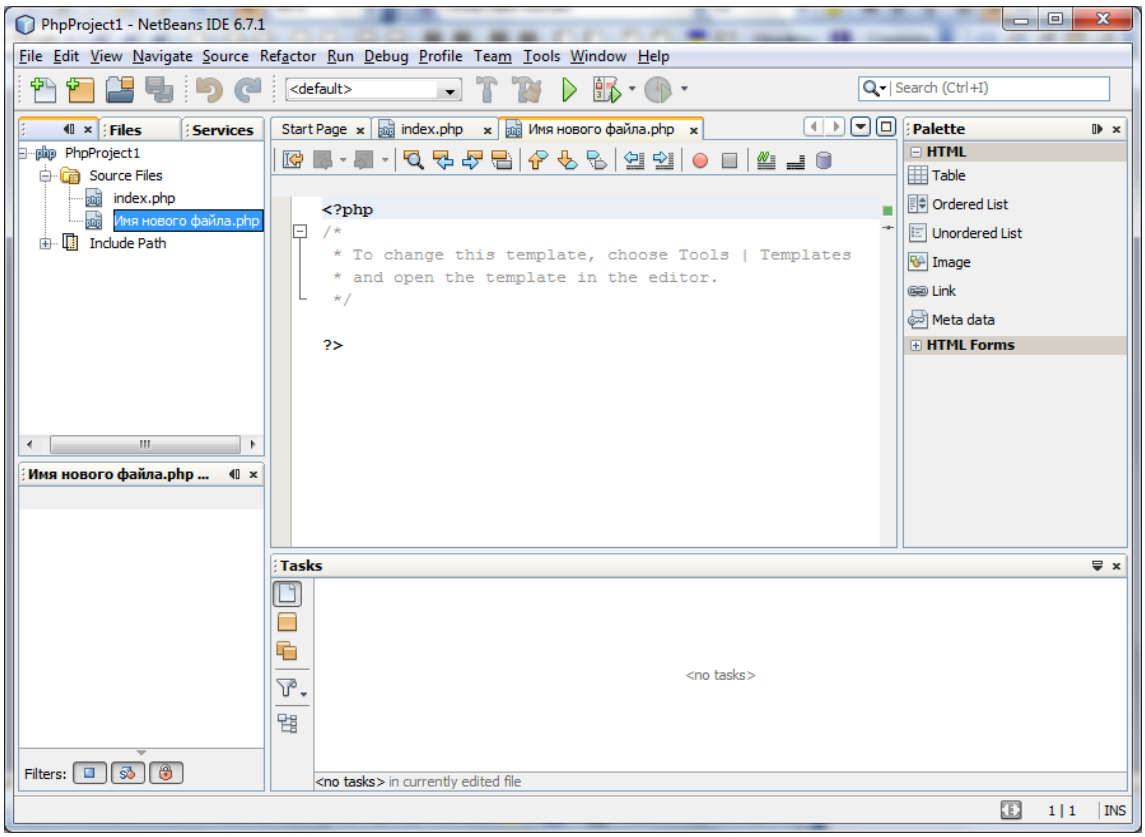


Рисунок 3.7 – Окно проекта

3.2 Отладка кода с помощью XDebug

Отладка программного обеспечения самая ненавистная работа доля разработчика. Большинство используют для отладки связку echo(print_r,var_dump) и exit(die), переходя от одной строке к другой. Однако, если ошибка появляется вновь в этом файле, требуется заново прописывать отладочные команды. Можно конечно отладку обернуть внутри конструкции if, которая будет работать только, когда определена какая-либо константа, например, DEBUG. Но каждый такой if будет слегка замедлять производительность, да и смотреться будет в коде очень ужасно.

Xdebug позволяет создавать логи трассировки, достаточно хороший выход для данной ситуации, и не надо будет изменять программу. Однако лог трассировки, особенно когда он создан для части программы, предоставляет нам много информации, не имеющей никакого отношения к отладке, таким образом, использование отладчика - это оптимальное решение. Отладчик позволит вам остановить запуск программы на некоторое время, проверить

или модифицировать текущее значение переменной, а затем продолжить выполнение программы. Запуская программу шаг за шагом, вы можете близко взглянуть как выполняется код, что поможет найти места ошибок.

В среде разработки NetBeans IDE для отладки PHP кода есть очень удобное решение – Xdebug. Для его настройки необходимо:

В соответствии с версией php, выбрать и скачать библиотеку xdebug .

Скопировать файл php_xdebug.dll в папку /php/ext/.

В файле php.ini добавить секцию XDebug и прописать в ней сладущее:

```
[XDebug]
extension=php_xdebug-2.0.5-5.2.dll
zend_extension_ts="C:\server\php\ext\php_xdebug-2.0.5-5.2.dll"
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
xdebug.remote_host=localhost
xdebug.remote_port=9000
```

Перезапускаем Apache

В NetBeans IDE перейти в Tools-Options-PHP. На вкладке General в секции Command Line указать путь до php.exe. В секции Debugging прописать port 9000 и Session ID netbeans-xdebug.

3.2.1 Как работает отладка

Когда отладка включена в php.ini, xdebug начинает контролировать выполнение программы, это значит, что xdebug может останавливать выполнение программы в любой момент, а потом запускать ее сново.

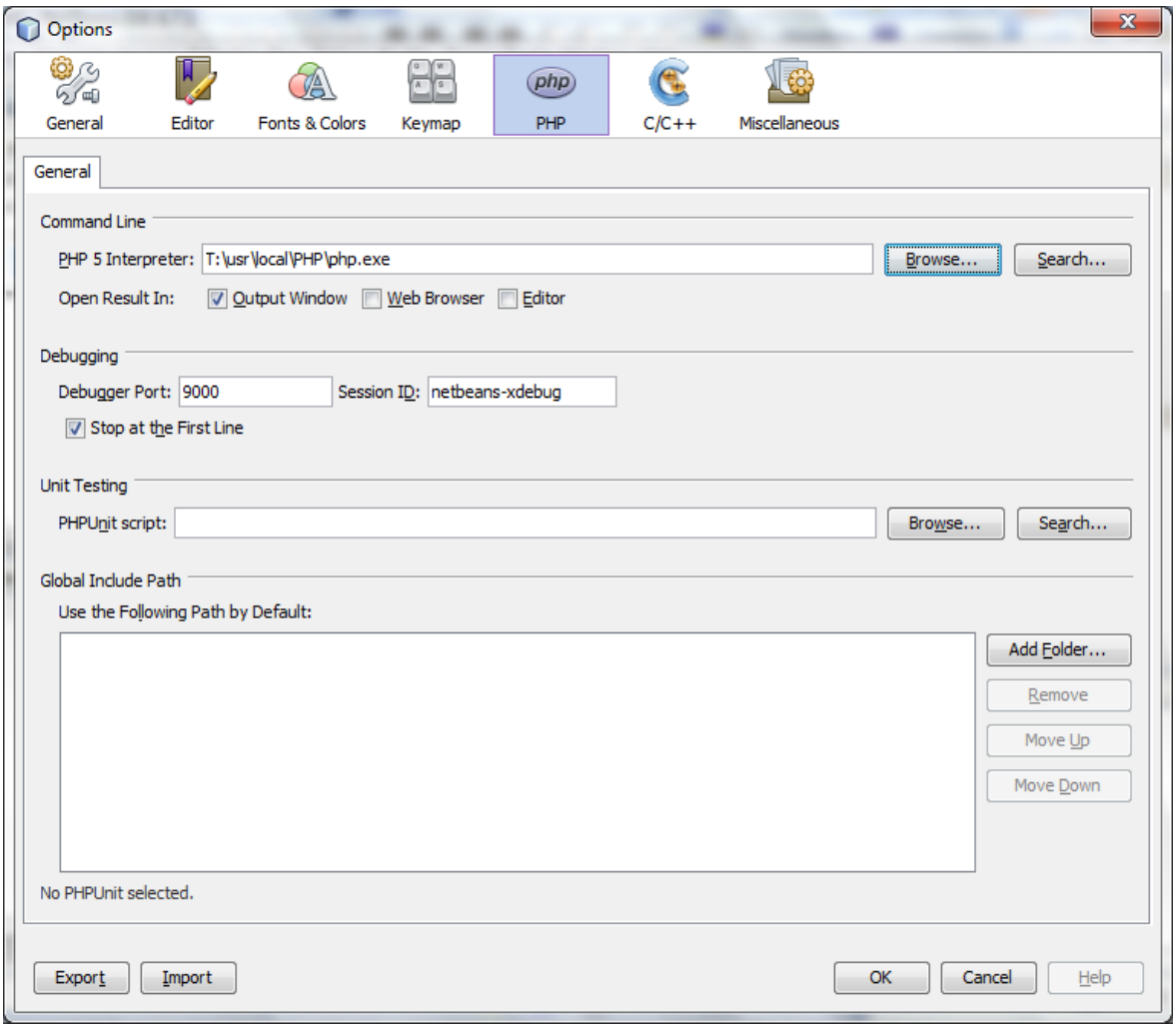


Рисунок 3.8 – Настройка отладчика

Когда выполнение программы остановлено, xdebug может получить информацию о текущем состоянии программы, например, прочитать текущие значения переменных. Также xdebug представляет возможность изменять значения переменных во время выполнения скрипта.

Расширение xdebug – это сервер, ожидающие соединений с клиентами на определенном порту, задаваемом в конфигурации. Существует два протокола, которые могут быть использованы для коммуникации между xdebug-клиентом и xdebug-сервером (GDB и DBGp).

GDB – это старый протокол, который был заменен на DBGp. Посылая команды серверу-xdebug, xdebug-клиент удаленно контролирует выполнение, сообщает PHP о приостановке запуска, выполнении следующей строчки кода или продолжении выполнения программы. Клиент обычно встроен в редактор

или IDE (в нашем случае в NetBeans), поэтому вы не будете иметь дело с протоколом общения напрямую.

Веб-сервер с xdebug может быть запущен на другой операционной системе, нежели xdebug-клиент. Вот почему xdebug называется удаленным отладчиком(remote debugger). Для упрощения, мы настроим сервер и клиент на одном и том же компьютере.

Существует два режима начала сессии отладки. Эти режимы контролируются в php.ini настройкой xdebug.remote_mode. Значение по умолчанию req, xdebug-клиент будет соединятся с сервером всякий раз когда начинается выполнение скрипта. Если вы хотите, чтобы xdebug соединялся с сервером только тогда, когда установлен breakpoint или когда в скрипте возникла ошибка, то можно установить xdebug.remote_mode в jit.

Для того чтобы начать отладку, нужно послать параметр XDEBUG_SESSION_START в скрипт как часть GET, POST или COOKIE. Значение этого параметра это название сессии отладки, которое должно быть уникальным, по имени сессии xdebug может различать различные сессии, запущенные параллельно. Для завершения сессии отладки необходимо послать скрипту команду XDEBUG_SESSION_STOP.

Вместо того чтобы вручную начинать и заканчивать отладочные сессии, следует установить специальный firefox plugin, который поможет легко начинать и заканчивать сессии одним нажатием мышки.

4 Технико-экономическое обоснование разработки программного обеспечения дипломного проекта

4.1 Организация создания программного обеспечения

Целью данного дипломного проекта, является создание программных средств автоматизированной системы контроля успеваемости учащихся.

Программу, как любое техническое решение, необходимо рассматривать с точки зрения экономической целесообразности и пользы.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ЭВМ, машинного времени, пропускной способности каналов передачи данных и др. –

на стадии функционирования; время разработки и денежных ресурсов – на стадии использования программного изделия).

Содержание технико-экономического обоснования разработки программы заключается в следующем:

1) расчет технико-экономических показателей;
2) расчет затрат на разработку;

Приведенные в данном разделе дипломной работы данных позволяют сопоставить результаты разработки и затраты на нее, чтобы сделать вывод об эффективности проекта.

4.2 Расчет цены программного обеспечения

Расчет цены ПО содержит:

1) определение затрат на оплату труда и накладных расходов;
2) установление цены ПО;
3) установление нижнего предела цены и договорной цены с учетом тиражирования.

4.2.1 Определение затрат на разработку программного обеспечения

Затраты на разработку программного обеспечения определяются по формуле:

$$З_{р.по} = З_{р} + З_{д} + З_{с} + З_{н} + Р_{экспл}, \quad (4.1)$$

где $З_р$ – заработная плата разработчиков, руб.;

$З_д$ – дополнительная заработная плата, руб.;

$З_с$ – отчисления на социальные нужды, руб.;

$З_н$ – накладные расходы, руб.;

$Р_{экспл}$ – эксплуатационные расходы, руб.

Оплата труда разработчика 8000 руб. в месяц, с учетом того, что в месяце 24 рабочих дня, получим 333 руб. в день. В общей сложности, разработка ПО велась в течение 40 рабочих дней. Таким образом, общая заработная плата за 40 дней разработки составит

$$З_р = 40 \cdot 333 = 13320 \ (руб.)$$

Дополнительная заработная плата определяется по формуле:

$$З_д = К_д \cdot З_р, \quad (4.2)$$

где $К_д$ – коэффициент, учитывающий дополнительную заработную плату.

Примем $К_д$ равным 0,1, тогда получаем, что дополнительная заработная плата составляет:

$$З_д = 13320 \cdot 0{,}1 = 1332 \ (руб.)$$

Расчет отчислений в страховые фонды с фонда заработной платы всех категорий промышленно-производственного персонала проводится в соответствии с установленным по действующему законодательству значением процента отчислений определяются по формуле:

$$З_с = (З_р + З_д) \cdot 0{,}30 \quad (4.3)$$

Учитывая ранее полученные значения заработной платы разработчика и дополнительной заработной платы, отчисления на социальные нужды составляют:

$$З_с = (13320 + 1332) \cdot 0{,}30 = 4395{,}6\ (руб.)$$

Накладные расходы определяются следующим образом:

$$З_н = \frac{З_р \cdot П}{100}, \qquad (4.4)$$

где П – процент накладных расходов организации, в которой разрабатывается программа.

Примем, что П равен 45 %, получим накладные расходы в размере

$$\boldsymbol{З_н = \frac{13320 \cdot 45}{100} = 5994\ (руб.)}$$

Расчет стоимости одного часа эксплуатации ЭВМ производится по формуле:

$$Р_{экспл} = \frac{С_э + С_о + А + С_р}{Ф_г}, \qquad (4.5)$$

где $С_э$ – стоимость энергии, потребляемой компьютером за год, руб.;

$С_о$ – расходы на обслуживание компьютера в течение года, руб.;

А – амортизационные отчисления, руб.;

$С_р$ – расходы на текущий ремонт, руб.;

$Ф_г$ – годовой полезный фонд времени работы ЭВМ, ч.

Компьютер работает в течение 9 часов в сутки, пять дней в неделю, 249 дней в году. Годовой полезный фонд времени работы компьютера при таких условиях составляет

$$Ф_г = 9 \cdot 249 = 2241(ч.)$$

Стоимость потребляемой за год электроэнергии рассчитывается по формуле:

$$С_э = М_п \cdot Ф_г \cdot Ц_э, \qquad (4.6)$$

где $М_п$ – потребляемая мощность ЭВМ, кВт;

$Ц_э$ – цена 1 кВт*ч электроэнергии, руб.

Согласно последней тарифной ставке 1 кВт*ч электроэнергии стоит 3,44 руб. Потребляемая мощность компьютера 0,2 кВт. Затраты на электроэнергию составят

$$С_э = 0,2 \cdot 2241 \cdot 3,44 = 1541,81 \ (руб.)$$

Расходы на обслуживание компьютера за год определяются по формуле:

$$С_о = \frac{ЗП_о \cdot 12}{Э_о}, \quad (4.7)$$

где $ЗП_о$ – зарплата персонала, обслуживающего компьютеры, руб.;

$Э_о$ – количество обслуживаемых им единиц оборудования;

12 – количество месяцев в году.

В организации используется 5 компьютеров, обслуживанием которых занимается 1 человек. Его заработная плата составляет 5000 руб. в месяц. В соответствии с формулой (4.7), затраты на обслуживание ЭВМ составляют

$$С_о = \frac{5000 \cdot 12}{5} = 12000 \ (руб.)$$

Амортизационные отчисления определяются по формуле:

$$А = \frac{Н_а \cdot С}{100}, \quad (4.8)$$

где $Н_а$ – норма амортизации. Примем ускоренную, равную 50 %;

С – балансовая стоимость компьютера, руб.

Балансовая стоимость складывается из стоимости компьютера, стоимости транспортировки и стоимости установки. Для используемого компьютера балансовая стоимость составляет 15000 руб.

Таким образом, согласно формуле (4.8), амортизационные отчисления составят:

$$А = \frac{50 \cdot 15000}{100} = 7500 \ (руб.)$$

Расходы на текущий ремонт $С_р$ определяются как 5% от балансовой стоимости компьютера. Таким образом, получаем, что расходы составят:

$$С_р = 0,05 \cdot 15000 = 750 \ (руб.)$$

Подставляя вычисленные значения в формулу (4.5) получаем, что стоимость одного часа эксплуатации ЭВМ равна

$$Р_{эксп/ч} = \frac{1541{,}81 + 12000 + 7500 + 750}{2241} = 9{,}72\ (руб.)$$

Из 40 дней, затраченных на разработку ПО, 28 дней приходится на кодирование и 12 дней на отладку ПО. Учитывая, что в день компьютер работает в течение 9 часов, получаем, что эксплуатационные расходы составят:

$$Р_{экспл} = (28 + 12) \cdot 9 \cdot 9{,}72 = 3499{,}2\ (руб.)$$

Согласно формуле (3.1) затраты на разработку программного обеспечения составят:

$$З_{р.по} = 13320 + 1332 + 4395{,}6 + 5994 + 3499{,}2 = 28540{,}8$$ (*руб.)*

4.2.2 Установление цены программного обеспечения

Цена разработанной программы определяется по формуле:

$$\boldsymbol{Ц = З_{р.по} \cdot (1 + Р),} \qquad (4.9)$$

где Р – норматив рентабельности, позволяющий учитывать прибыль организации, разрабатывающей данную программу.

Примем норматив рентабельности равным 20%, тогда цена разрабатываемого программного продукта:

$$Ц = 28540{,}8 \cdot 1{,}2 = 34248{,}96 (руб.)$$

Итак, цена программного продукта, полученного в ходе дипломного проектирования, составляет 34248,96 руб. при норме рентабельности 20%. Себестоимость программы – 28540,8 руб.

4.2.3 Установление нижнего предела цены с учетом тиражирования

Нижний предел цены (Цнп) с учетом желаемой рентабельности (Р), тиражированием (N) и НДС (Н) определяется по формуле:

$$Ц_{НП} = \frac{З_{р.по} \cdot (1+Р)}{N} \cdot (1+Н). \quad (4.10)$$

При установленном НДС равным 20%, и учитывая, что тиражирование программы не предполагается, рассчитываем предел нижней цены.

$$Ц_{НП} = \frac{34248{,}96 \cdot (1+0{,}20)}{1} \cdot (1+0{,}20) = 49318{,}5 \ (руб.)$$

Окончательную цену разрабатываемого ПО примем на уровне 49318,5 рублей.

5 Анализ опасных и вредных факторов, воздействующих на оператора ЭВМ, и способы противодействия им.

5.1 Организация безопасных условий труда на рабочем месте оператора ПЭВМ

Рабочее место - это система функционально и пространственно организованных технических средств и предмета труда, обеспечивающая условия для успешного решения человеком - оператором поставленной перед ним задачи.

На комфортность работы оператора влияет организация рабочего места оператора, средства отображения и органы управления машиной. Они должны быть максимально удобны для человека, чтобы не создавать помех, дискомфорта в процессе работы, а также способствовать наименьшей утомляемости.

Основным способом обеспечения условий комфорта оператора ПЭВМ является рациональная организация его рабочего места.

Под рабочим местом оператора ПЭВМ понимается зона трудовой деятельности в системе «человек-машина» с техническими средствами и вспомогательным оборудованием, необходимым для решения конкретных производственных задач.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7. [1]

Рекомендуемые расположения текста, клавиатуры и экрана компьютера R=50...60 см для типовых компьютеров с размером экрана по диагонали 250..350 мм (предельно минимальное) представлены на рисунке 5.1.

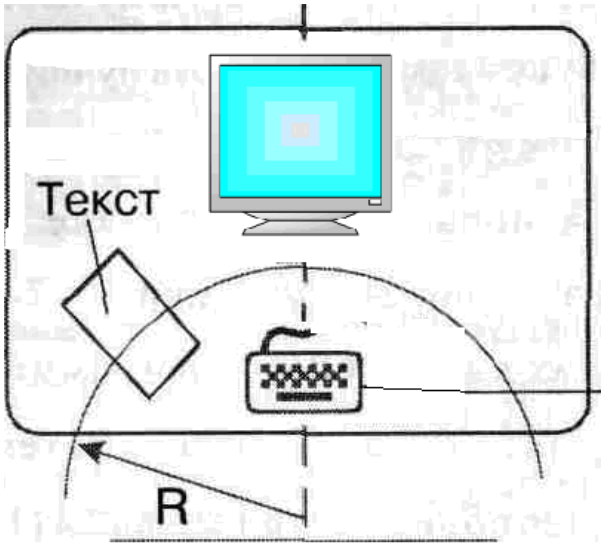


Рисунок 5.1- Рабочее пространство

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм.

Конструкция рабочего стула должна обеспечивать:

- ширину и глубину поверхности сиденья не менее 400 мм;

- поверхность сиденья с закругленным передним краем;

- регулировку высоты поверхности сиденья в пределах 400 - 550 мм и углам наклона вперед до 15 градусов, и назад до 5 градусов;

- высоту опорной поверхности спинки 300 ±20 мм, ширину - не менее 380 мм и радиус кривизны горизонтальной плоскости - 400 мм;

- угол наклона спинки в вертикальной плоскости в пределах ±30 градусов;

- регулировку расстояния спинки от переднего края сиденья в пределах 260 - 400 мм;

- стационарные или съемные подлокотники длиной не менее 250 мм и шириной - 50 - 70 мм;

- регулировку подлокотников по высоте над сиденьем в пределах 230 ±30 мм и внутреннего расстояния между подлокотниками в пределах 350 - 500 мм.

Параметры рабочего стула (кресла) представлены на рисунке 5.2.

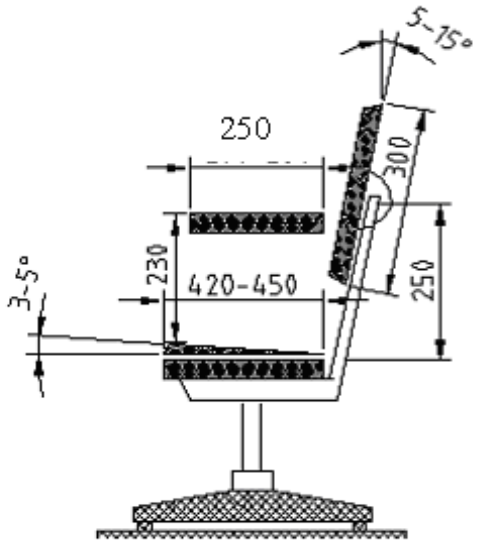


Рисунок 5.2 – Рабочий стул

Рабочее место пользователя ПЭВМ следует оборудовать подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20°. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Экран должен находиться в центре поля обзора, примерно на расстоянии 60 - 70 см от глаз, при этом смотреть на него следует сверху вниз, учебники, тетради должны находиться слева на столе. Положение тела оператора относительно экрана должно соответствовать направлению просмотра экрана компьютера под прямым углом или под углом 75 градусов.[2] Рекомендации по рациональному размещению средств отображения и информации представлены на рисунке 5.3:

1 - возможный наклон плоскости для представления информации,

2 - границы полезной зоны,

3 - граница информации,

4 - горизонталь,

5 - граница оптимальной зоны,

6 - нормальная линия взгляда,

7 - граница оптимальной зоны,

8 - возможный наклон плоскости для представления информации,

9 - граница информации,

10 - граница полезной зоны.

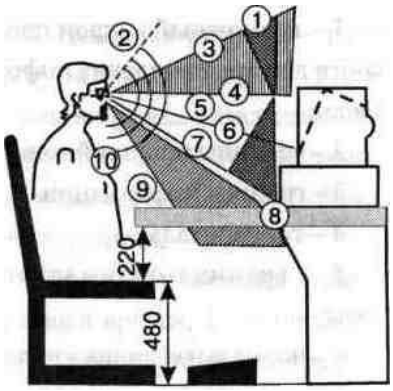


Рисунок 5.3 – Размещение средств отображения и информации

ПЭВМ должны соответствовать требованиям санитарных правил, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке.

Допустимые значения напряженности электрического поля в диапазоне частот 5 Гц – 2 кГц составляет 25 В/м, в диапазоне частот 2 кГц – 400 кГц составляет 2,5 В/м., напряженность электрического поля – 15 кВ/м.

Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций (ПДК), установленных для атмосферного воздуха.

Мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05м от экрана и корпуса монитра при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час).

Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном

положении для обеспечения фронтального наблюдения экрана монитора. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4-0,6 и не иметь блестящих деталей.

Документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям санитарных правил.

Рабочие столы следует размещать таким образом, чтобы мониторы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300лк.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ).

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и

светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Основные и производственные помещения вычислительного центра целесообразно окрашивать в соответствии с цветом технических средств. Выбор цвета определяется рядом факторов, в том числе конструкцией здания, характером выполняемой работы, освещенностью, количеством работающих.

Необходимо учитывать, что цвет является сильным психологическим стимулятором:

красный - увеличивает мускульное напряжение;

оранжевый - стимулирует деятельность;

желтый - стимулирует зрение и нервную систему;

зеленый - успокаивает;

голубой - ослабляет мускульное напряжение и т.д.

Восприятие цвета в большой степени зависит от освещенности. Под влиянием различных источников света цвет поверхности меняет тон.

К системам производственного освещения предъявляются следующие основные требования:

Соответствие уровня освещенности рабочих мест характеру выполняемой зрительной работы.

Достаточно равномерное распределение яркости на рабочих поверхностях в окружающем пространстве.

Отсутствие резких теней, прямой и отражённой блёскости (блёсткость - повышенная яркость светящихся поверхностей, вызывающая ослеплённость).

Постоянство освещённости во времени.

- Оптимальная направленность излучаемого осветительными приборами светового потока.

В вычислительном центре, применяем одностороннее боковое естественное освещение. Причем светопроемы с целью уменьшения солнечной инсоляции устраивают с северной, северо-восточной ориентацией.

В машинных залах рабочие места операторов, работающих с дисплеями, располагают подальше от окон и таким образом, чтобы оконные проемы находились сбоку. Если экран дисплея обращен к оконному проему, необходимы специальные экранирующие устройства. Окна рекомендуется снабжать светорассеивающими шторами, регулируемыми жалюзи или солнцезащитной пленкой с металлизированным покрытием.

Проблемы гигиены труда пользователей ПЭВМ относятся к числу наиболее актуальных, поскольку непрерывно расширяется круг задач, решаемых ПЭВМ, и все больше людей вовлекается в процесс использования ВТ. В последние годы Северо-Западным Научным центром гигиены и общественного здоровья проводились комплексные гигиенические исследования по оценке условий труда и состояния здоровья работающих с ПЭВМ. Анализ полученных результатов, позволяет составлять определенное представление о факторах риска здоровью пользователей ПЭВМ.

Рациональный режим труда и отдыха способствует снижению утомляемости, поддержанию высокой работоспособности оператора ПЭВМ, повышению производительности его труда и сохранению здоровья. При установлении количества перерывов на отдых, определении их длительности и распределении их в течении рабочей смены учитывается специфика производства, напряженность труда, изменение работоспособности человека в процессе работы.[5]

В качестве таких мер рекомендуется применение в рамках режима труда и отдыха различные физкультурно-оздоровительные мероприятия, психологические разгрузки.

Режим труда и отдыха операторов должен организовываться в зависимости от вида и категории трудовой деятельности.

Категория А – работа по считыванию информации с дисплея с предварительным запросом.

Категория Б – работа по вводу информации.

Категория В – творческая работа в режиме диалога с ПЭВМ.

Продолжительность обеденного перерыва определяется Трудовым кодексом.[6]

Система физкультурно-оздоровительных мероприятий должна включать в себя различные формы производственной гимнастики: вводную гимнастику, физкультурную паузу, физкультурные минутки и другие малые формы (микрокомплексы, познотонические упражнения, профилактические приемы, самомассаж и др.), профилактическую и восстановительную гимнастику.

При этом следует знать, что каждое из указанных средств и форм воздействия на организм человека предназначено для определенной цели, а вся система в целом обеспечивает поддержание работоспособности на протяжении рабочего дня.

Помимо физкультурно-оздоровительных мероприятий для снижения нервно-эмоциональной нагрузки рекомендуется посещение комнат психологической разгрузки. Сеансы такой разгрузки в целях снятия усталости могут проводиться как во время регламентированных перерывов, так и по окончании рабочего дня. Они должны проводиться в специально оборудованном для этих целей помещении – комната психологической разгрузки.

Для защиты от статического электричества необходимо использовать нейтрализаторы и увлажнители. Полы должны иметь антистатическое покрытие.

Устройства визуального отображения генерируют несколько типов излучения, в том числе рентгеновское, радиочастотное, видимое и ультрафиолетовое. Однако уровни этих излучений достаточно низки и не превышают действующих норм.

Для снижения значений напряженности электрических и электростатических полей рекомендуется применение защитных фильтров, для снижения воздействия низкочастотных магнитных полей рекомендуется проводить полную внутреннюю защиту мониторов (со стороны, как экрана,

так и корпуса), поскольку существующие приэкранные защитные фильтры в этом отношении являются неэффективными.

В помещениях необходимо контролировать уровень аэроионизации, ибо мягкое рентгеновское излучение, возникающее при напряжении на аноде 20-22 кВ, а также высокое напряжение на токоведущих участках схемы вызывают ионизацию воздуха с образованием положительных ионов, считающихся неблагоприятными для человека.

Оптимальным уровнем аэроионизации в зоне дыхания работающего считается содержание легких аэроионов обоих знаков от 1,5*102 до 5*101 в 1 см3 воздуха.

Для снижения влияния статического электричества рекомендуется устанавливать ионизаторами воздуха.

К непосредственной работе с ПЭВМ допускаются лица, не имеющие медицинских противопоказаний. Женщины со времени установления беременности и в период кормления грудью к выполнению всех видов работ, связанных с использованием ПЭВМ не допускаются.

Соблюдение вышеперечисленных норм и правил, внедрение новых технологий и научных разработок при организации рабочего места оператора ПЭВМ, позволит к минимуму снизить вредное воздействие производственных факторов на операторов и улучит условия рабочего места.

5.2 Расчет освещенности на рабочем месте

Рациональное освещение помещений - один из наиболее важных факторов, от которых зависит эффективность трудовой деятельности человека. Хорошее освещение необходимо для выполнения большинства задач пользователя ЭВМ. Для того чтобы спланировать рациональную систему освещения, учитывается специфика рабочего задания, для которого создается система освещения, скорость и точность, с которой это рабочее

задание должно выполняться, длительность его выполнения и различные изменения в условиях выполнения рабочих операций. [4]

Помещение, в котором находится рабочее место пользователя ЭВМ, имеет следующие характеристики:

- длина помещения 16 м;
- ширина помещения 6 м;
- высота 4 м;
- число окон 3;
- количество рабочих мест 3;
- окраска интерьера - белый потолок, бледно-зеленые стены;
- пол металлический, обтянутый линолеумом зеленого цвета.

В помещении, где находится рабочее место пользователя, используется смешанное освещение, т.е. сочетание естественного и искусственного освещения. В качестве естественного - боковое освещение через окна. Искусственное освещение используется при недостаточном естественном освещении. В данном помещении используется общее искусственное освещение. Расчет его осуществляется по методу светового потока с учетом потока, отраженного от стен и потолка.

Нормами для данных работ установлена необходимая освещенность рабочего места Ен = 300 [лк].

Общий световой поток определяется по формуле:

$$F_{общ} = \frac{Eн \times S \times z_1 \times z_2}{V \cdot N}$$ [лм].

где Ен - нормированная освещенность (Ен =300 [лк]);

S- площадь помещения;

z1 - коэффициент, учитывающий старение ламп и загрязнение светильников (z1=5).

z2 - коэффициент, учитывающий неравномерность освещения помещения (z2=1,1);

N – количество ламп в светильнике;

V- коэффициент использования светового потока, определяется в зависимости от коэффициентов отражения от стен, потолка, рабочих поверхностей, типов светильников и геометрии помещения.

Площадь помещения:

S=A*B=16*6=96 [м2].

Выберем коэффициент использования светового потока по следующим данным:

– коэффициент отражения побеленного потолка Rn = 70%;

– коэффициент отражения от стен, окрашенных в светлую краску Rcm=50%;

– коэффициент отражения от пола, покрытого линолеумом темного цвета Rp=10%;

– индекс помещения:

$$i = \frac{A*B}{h*(A+B)} = \frac{16*6}{4*(16*6)} = 1.1$$

Найденный коэффициент V= 0,34 (для светильника ПВЛ-1). Отсюда определяется общий световой поток:

$$F_{общ} = \frac{300*96*1{,}1*1{,}5}{0{,}34 \cdot 2} = 69882 \text{ [лм].}$$

Для организации общего искусственного освещения выберем лампы типа ЛБ40. Люминесцентные лампы имеют ряд преимуществ перед лампами накаливания: их спектр ближе к естественному; они имеют большую экономичность (больше светоотдача) и срок службы (в 10-12 раз). Наряду с этим имеются и недостатки: их работа сопровождается иногда шумом, хуже работают при низких температурах, их нельзя применять во взрывоопасных помещениях; имеют малую инерционность. Для нашего помещения люминесцентные лампы подходят. Световой поток одной лампы ЛБ40 составляет не менее Fn =3000 [лм]. Число ламп N, необходимых для организации общего освещения определяется по формуле:

$$N = \frac{F_{общ}}{F_{л}} = \frac{69882}{3000} = 24$$

В качестве светильников выбираем ПВЛ-1, 2х40 Вт. Таким образом, чтобы обеспечить световой поток Fобщ = 69882 [лм] надо использовать 12 светильников по 2 лампы ЛБ40 в каждом.

Рабочая высота подвеса определяется выражением:

$$h = H - h_1 = 4 - 0.7 - 0.1 = 3.2 \text{ [м]}.$$

Расстояние между центрами светильников:

$$L = 1{,}4 \cdot h = 1{,}4 \cdot 3.2 = 4{,}48 \text{ [м]}.$$

Число рядов светильников рассчитывается исходя из ширины помещения и расстояния между рядами:

$$n = \frac{B}{L} = \frac{6}{4.48} = 1.34 \approx 2$$

Размещаем светильники в два ряда.

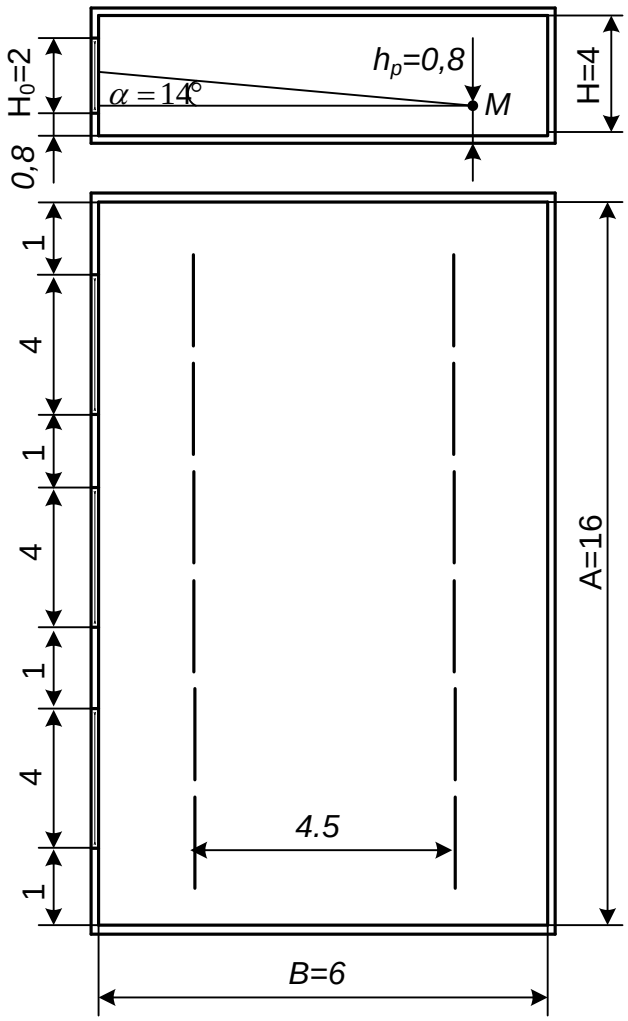


Рисунок 5.4 – Эскиз помещения с расположением светильников

5.3 Параметры микроклимата

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. В санитарных нормах установлены величины параметров микроклимата,

создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (см. табл. 5.1).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5м3/человека с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в табл. 5.2.

Таблица 5.1 – Параметры микроклимата для помещений, где установлены компьютеры.

Период года	Параметр микроклимата	Величина
Холодный	Температура воздуха в помещении	22…24°С
	Относительная влажность	40…60%
	Скорость движения воздуха	до 0,1м/с
Теплый	Температура воздуха в помещении	23…25°С
	Относительная влажность	40…60%
	Скорость движения воздуха	0,1…0,2м/с

Таблица 5.2 – Нормы подачи свежего воздуха в помещения, где расположены компьютеры:

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, м3 /на одного человека в час
Объем до 20м3 на человека	Не менее 30
20…40м3 на человека	Не менее 20

Более 40м3 на человека	Естественная вентиляция

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

5.4 Шум и вибрация

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т.д. Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума (выше 80 дБ) на слух человека приводит к его частичной или полной потере. [3]

В табл. 5.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 5.3 – Предельные уровни звука, дБ, на рабочих местах:

Категория напряженности труда	Категория тяжести труда

Мало напряженный	Легкая	Средняя	Тяжелая
Умеренно напряженный	80	80	75
Напряженный	70	70	65
Очень напряженный	60	60	65
	50	50	55

5.5 Вывод

1. Выполнена организация рабочего места оператора ЭВМ с учетом требований безопасности;

2. Из произведенного в данном разделе расчета следует, что для нормальной работы пользователя рабочего места с видеотерминальным устройством необходимо общее освещение помещения со световым потоком 69882 [лм], для чего необходимо наличие 12 светильников типа ПВЛ-1 с 2-мя лампами типа ЛБ40 каждый. Светильники должны быть расположены в два ряда, что является наиболее оптимальным решением. Кроме того, рекомендуется использовать ряд специальных мер по защите пользователя от вредных факторов в виде бликов на экране монитора, например, использование занавесей на окнах;

3. Соблюдение условий, определяющих оптимальную организацию рабочего места инженера - программиста, позволит сохранить хорошую работоспособность в течение всего рабочего дня. Повысит, как в количественном, так и в качественном отношениях производительность труда программиста, что в свою очередь будет способствовать быстрейшей разработке и отладке программного продукта.

5.6 Список используемой литературы

1. Дзарасов М.Э. Ответственность по нормам трудового права. Учебное пособие. 2018.
2. «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах». СанПиН 2.2.4.3359-16.
3. Бурлак Г.Н. «Безопасность работы на компьютере; организация труда на предприятиях информационного обслуживания». 2012г.
4. «Естественное и искусственное освещение». СНиП 23-05-2010.
5. Н.И.Бобков, Голованова Т.В. «Охрана труда на ВЦ» М:.МАИ.1998
6. Г.Ф.Денисенко «Охрана труда» М.:Высшая школа 2015.

Заключение

В ходе выполнения работы по разработке автоматизированной системы было сделано:

1. Проведен анализ информационного взаимодействия в торговой области;
2. Рассмотрены технологии разработки систем в области торговых предприятий;
3. Рассмотрены существующие технические реализации подобных систем и выявлены из достоинства и недостатки.
4. Проведено формализации разработки системы товарооборота, разработанных алгоритмы работы и определено программное обеспечение, необходимое для разработки.
5. Разработана база данных, позволяющая осуществлять хранение информации.
6. Разработано программное обеспечение для автоматизации процесса управления товарами и системой заказов.

Применение автоматизированной системы управления логистикой торгового предприятия, позволяет значительно повысить эффективность работы сотрудников предприятия. Сократить временные затраты на составление отчетов, журнала товаров, книги покупок и продаж. Кроме этого, в системе существует возможность составления списка сотрудников. При этом, резко уменьшается объем «бумажной» работы в организации.

Использование в программном комплексе СУБД, позволяет обеспечить хранение вводимых данных и резервное копирование.

.

Список использованных источников

1 Интеллектуальные модели анализа информации: курс лекций 2017 BaseGroup Labs.

2 Елманова Н. Введение в технологии PHP. / Н. Елманова, Федоров А. – М.: Диалог-МИФИ, 2017.

3 П. Блаттнер - Special Edition Using MySQL. – М.: Вильямс, 2016.

4 Арлоу Д. Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, И. Нейштадт – Спб: Символ-Плюс, 2017.

5 Мацаяшек А. Анализ требований и проектирование системы. Разработка информационных систем с использованием UML / А. Мацаяшек – М.: Издательский дом «Вильямс», 2015.

6 Фаулер М. UML. Основы / М. Фаулер, К. Скотт – СПб: Символ-Плюс, 2014.

7 Дейтел Х. – PHP in DB / Х. Дейтел, Дейтел П., Листфид Дж., Нието Т., Йегер Ш., Златкина М. - СПб: БХВ-Петербург, 2016.

8 Фримен А. –Язык интегрированных запросов в PHP 2016 для профессионалов / А. Фримен, Раттц-мл Д. – М.: Вильямс, 2018.

9 Фримен А. –MVC Framework с примерами на PHP для профессионалов / Сандерсон С. - М.: Вильямс, 2017.

10 Флэнаган Д. – JavaScript Подробное руководство – СПб: Символ-Плюс, 2018.

11 Мандел Т. Разработка пользовательского интерфейса / Т. Мандел – М.: ДМК Пресс, 2018.

12 Гома Х. UML. Проектирование систем реального времени параллельных и распределенных приложений / Х. Гома – М.: ДМК Пресс, 2015.

13 Котляров В. П. Основы тестирования программного обеспечения: Учебное пособие / В. П. Котляров, Т. В. Коликова – М.: Бином. Лаборатория знаний, 2016.

14 Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения / Дж. Макгрегор, Д. Сайкс. – К.: ООО «ТИД «ДС», 2015.

КОД ПРОГРАММЫ (СЕРВЕРНАЯ ЧАСТЬ)

Приложение А

(обязательное)

Файл .htaccess

```
Options +FollowSymLinks

RewriteEngine on
# if a directory or a file exists, use it directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# otherwise forward it to index.php
RewriteRule . index.html
```

Файл DB.php

```
<?php

class db{
    protected $host = "localhost";
    protected $dbname = "stock";
    protected $user = "root";
    protected $pass = "";
    protected $dbh;

    function __construct() {

        try {

            $this->dbh = new PDO("mysql:host=$this->host;dbname=$this->dbname", $this->user,
$this->pass, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
        }
        catch (PDOException $e) {

            echo $e->getMessage();
        }
    }

    public function closeConnection() {

        $this->dbh = null;
    }
}
```

Файл Model.php

```
<?php
```

```
include_once "db.php";
include_once "myphp-backup.php";
session_start();

class Model extends DB {
    /*
     * Функция авторизации
     */
    public function setLogin($data) {
        $login = $data["username"];
        $pass = $data["password"];
        //$pass1 = md5($pass);
        $STH = $this->dbh->prepare("select * from users
        left join `group` on `group`.id_group = users.id_group
        where (users.nickName = '$login') and (users.password = '$pass')");
        $STH->execute();
        $_group = $STH ->fetchColumn(6);
        $STH->execute();
        if($STH->rowCount() > 0){
            $results = $STH->fetchAll(PDO::FETCH_ASSOC);
            $_SESSION = $results[0];
            $_VIS=$this->dbh->prepare("select id from routes where id_group = '$_group' " );
            $_VIS->execute();
            if($_VIS->rowCount() > 0) {
                $_visit_res = $_VIS ->fetchAll(PDO::FETCH_ASSOC);
                $_SESSION['visit'] = array_map(function($o) { return $o["id"]; }, $_visit_res);
            }
            else {
                $_SESSION['visit'] = [];
            }
            $resault = (object) array("success" => true, "message" => "Ok", "group" => $_group) ;
            return ($resault);
        }
        $res = false;

        return ($res);
    }
    /*
     * Функция выхода
     */
    public function getLogout(){
        session_destroy();
        $res = (object) array("res" => "true");
        return ($res);
    }

        public function getRoutes(){

                $group = $_SESSION['id_group'];

                $_VIS=$this->dbh->prepare("select * from routes where id_group = '$group' " );
                $_VIS->execute();
```

```
                if($_VIS->rowCount() > 0) {
                        $_visit_res = $_VIS ->fetchAll(PDO::FETCH_ASSOC);

                }
                else {
                        $_visit_res = [];
                }

        return ($_visit_res);
           }

    public function setBackup() {

        $backup = new Backup_Database();
        $date = date ('Y-m-d H:i');
        $res = $backup->backupDir . '/' . $backup->backupFile .'.gz';
        $file =$backup->backupFile .'.gz';
        $dir = $backup->backupDir;
        $STH=$this->dbh->prepare("insert into backups(name, createDate, path) VALUES
('$file','$date','$dir')");
        $STH->execute();
        return($res);

    }

           public function getBackups() {
                $STH = $this->dbh->prepare("select * from backups");
                $STH->execute();
                $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
                return ($results);
    }

           public function deleteBackup($id) {

                $STH = $this->dbh->prepare("DELETE FROM backups WHERE id= '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);

        return ("Backup deleted");
           }

    /*
     * Функция получения списка пользователей
     */
    public function getUsers(){
        $STH = $this->dbh->prepare("select users.id_user as id, users.nickName, users.f_name,
users.password, users.l_name, users.s_name, group.Name as 'group' from users
                 join `group` on `group`.id_group = users.id_group");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
```

```
    public function getUser(){
                $temp =
'{"success":true,"user":{"id":null,"username":"","f_name":"","s_name":"","l_name":"","permissi
ons":{"visit":[],"role":""},"avatar":""}}';
                $results = json_decode($temp);

                $results->user->id = $_SESSION["id_user"];
                $results->user->username = $_SESSION["nickName"];
                $results->user->f_name = $_SESSION["f_name"];
                $results->user->s_name = $_SESSION["s_name"];
                $results->user->l_name = $_SESSION["l_name"];
                $results->user->permissions->visit = $_SESSION["visit"];
                $results->user->permissions->role = $_SESSION["Name"];
        return ($results);
    }

        /**/

    public function setUser($data) {
                $login = $data["nickName"];
        $password = $data["password"];
                $fname = $data["name"];
                $lname = $data["f_name"];
                $sname = $data["l_name"];
                $id_group = $data["group"];

        $STH = $this->dbh->prepare("insert into
users(nickName,password,f_name,l_name,s_name,id_group)
VALUES('$login','$password','$fname','$lname','$sname',$id_group)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        $res = (object) array("success" => true, "message" => "Ok");
        return ($res);
    }
    public function changeUser($id, $data){
                $login = $data["nickName"];
        $password = $data["password"];
                $fname = $data["name"];
                $lname = $data["f_name"];
                $sname = $data["l_name"];
                $id_group = $data["group"];
        $STH = $this->dbh->prepare("update users SET nickName ='$login', password
='$password',f_name = '$fname',l_name = '$lname',s_name = '$sname',id_group = '$id_group'
where id_user = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);

        return ("User changed");
    }
    public function deleteUser($id) {
```

```
        $STH = $this->dbh->prepare("DELETE FROM users WHERE id_user= '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ("User deleted");
    }
    public function getCategories(){
        $STH = $this->dbh->prepare("select * FROM category");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getCategory($id){
        $STH = $this->dbh->prepare("select * FROM category where id_category ='$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setCategory($name){
        $STH = $this->dbh->prepare("insert into category(Name) VALUES($name)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeCategory($id,$Name){
        $STH = $this->dbh->prepare("update category SET Name ='$Name' where id_category =
'$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteCategory($id){
        $STH = $this->dbh->prepare("DELETE FROM category WHERE id_category = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getProducts(){
        $STH = $this->dbh->prepare("select * FROM product");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getProduct($id){
        $STH = $this->dbh->prepare("select * FROM product where id_product = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getProductsByCategory($id){
        $STH = $this->dbh->prepare("select Name FROM product where id_category = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
```

```
        return ($results);
    }
    public function
setProduct($name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer){
        $STH = $this->dbh->prepare("insert into
product(Name,id_category,id_measure,price,count,id_brand,id_producer)
VALUES($name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function
changeProduct($id,$name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer){
        $STH = $this->dbh->prepare("update product SET Name ='$name', id_category
='$id_category', id_measure = '$id_measure', price = '$price', count = '$count', id_brand =
'$id_brand', id_producer = '$id_producer'  where id_product = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteProduct($id){
        $STH = $this->dbh->prepare("DELETE FROM product WHERE id_product = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
/*
* Функция получения цены товара
*/
    public function getPrice($id_product) {
        $STH = $this->dbh->prepare("select price from product where id_product ='$id_product'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    /*
     * Функция получения списка издетелей
     */
    public function getProducers() {
        $STH = $this->dbh->prepare("select * from producer");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getProducer($id) {
        $STH = $this->dbh->prepare("select * from producer where id_producer ='$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setProducer($Name,$Phone,$Address) {
        $STH = $this->dbh->prepare("insert into producer(Name,Phone,Address)
```

```
VALUES('$Name','$Phone',$Address)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteProducer($id) { //delete дописать
        $STH = $this->dbh->prepare("DELETE FROM producer WHERE id_producer = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeProducer($id,$name,$phone,$address) { //delete дописать
        $STH = $this->dbh->prepare("update product SET Name ='$name', Phone ='$phone',
Address = '$address' where id_producer = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }

/*
 * Функция получения списка брендов
 */
    public function getBrands() {
    $STH = $this->dbh->prepare("select * from brand ");
    $STH->execute();
    $results = $STH->fetchAll(PDO::FETCH_ASSOC);
    return ($results);
    }
    /*
     * Функция добавления бенда
     */
    public function setBrand($id_producer,$name) {
        $STH = $this->dbh->prepare("insert into brand(Name,id_producer)
VALUES($name,$id_producer)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeBrand($id,$id_producer,$name) {
        $STH = $this->dbh->prepare("update brand SET Name ='$name', id_producer
='$id_producer' where id_brand = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteBrand($id) {
        $STH = $this->dbh->prepare("DELETE FROM brand WHERE id_brand = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getBrandByProducer($producer) {
```

```
        $STH = $this->dbh->prepare("select * from brand where id_producer = '$producer'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    /*
     * Функция установки статуса
    */
    public function setStatus($id_order,$id_status) {
        $date = date('Y-m-d H:i:s');
        $STH = $this->dbh->prepare("insert into order_logs(id_order,id_producer,date)
VALUES('$id_order','$id_status',$date)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
   /*
 * Функция установки привилегии
*/
    public function setPrivilege($id_user,$id_category) {
        $STH = $this->dbh->prepare("insert into privileges(id_category,id_user) VALUES
($id_category,$id_user)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deletePrivilegeUser($id_user) {
        $STH = $this->dbh->prepare("DELETE FROM priveleges WHERE id_user = '$id_user'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }

    public function pathcPrivilege($id_user,$id_category,$id) {
        $STH = $this->dbh->prepare("update priveleges SET Name ='$id_user', id_category
='$id_category' where id_privelege = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getPrivilegeByUser($id_user) {
        $STH = $this->dbh->prepare("select * from priveleges where id_user = '$id_user'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getPrivileges() {
        $STH = $this->dbh->prepare("select * from priveleges");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
```

```
    public function getCategoriesByUser() {
        $id=$_SESSION["id_user"];
        $STH = $this->dbh->prepare("select category.`Name` from `privileges` left join category
on category.id_category = `privileges`.id_category where `privileges`.id_user = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setProductInCart($data){

                $id_product = $data["product_id"];
                $_count = $data["count"];

                $id_user = $_SESSION['id_user'];

        $STH = $this->dbh->prepare("insert into cart(id_product,count,id_user)
VALUES($id_product,$_count,$id_user)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        $res = (object) array("success" => true, "message" => "Ok");
        return ($res);
    }
    public function getUserCart() {///по отдельному заказу?
        $user = $_SESSION['id_user'];
        $STH = $this->dbh->prepare("select product.Name as `title`, cart.id_product,
sum(cart.count) as `count` from `cart` join `product` on `product`.`id_product` =
`cart`.`id_product` where id_user = '$user' GROUP BY cart.id_product");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function patchProductInCart($id_cart,$id_product,$count) {
        $STH = $this->dbh->prepare("update cart SET count = '$count' where id_cart = '$id_cart'
and id_product ='$id_product'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getClientHistoryOrder($id) {
        $STH = $this->dbh->prepare("select id_order from order where id_client = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getUserHistoryOrder() {
        $id_user=$_SESSION['id_user'];
        $STH = $this->dbh->prepare("select `order`.id_order as id, `order_logs`.id_status as status,
`order_logs`.date from `order`
                join `order_logs` on `order_logs`.id_order = `order`.id_order
                where `order`.id_user = '$id_user'");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
```

```
        return ($results);
    }
    public function patchClientByCart($id_cart,$address,$phone,$fname,$sname,$lname,$email)
{
        $STH = $this->dbh->prepare("update client left join `order` on `order`.id_client =
client.id_client  SET address = '$address', phone = '$phone', f_name = '$fname', s_name =
'$sname', l_name ='$lname',email ='$email'   where `order`.id_cart = '$id_cart' ");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }


}
```

Файл Request.php

```
<?php
error_reporting(E_ERROR | E_WARNING | E_PARSE);

include_once("db.php"); //Подключаем класс для работы с бд
include_once("model.php"); //Подключаем класс с функционалом нашего приложения

header('Access-Control-Allow-Origin: *');
if (isset($_SERVER['HTTP_ORIGIN'])) {
        header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
        header('Access-Control-Allow-Credentials: true');
        header('Access-Control-Max-Age: 86400');
}

if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
header("Access-Control-Allow-Headers:
{$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");
}

$con    = new DB();
$model  = new Model();
$postData = file_get_contents('php://input'); //Получаем данные, которые отправили на
сервер
$data = json_decode($postData, true); //Декодируем из json в массив
$url = $_SERVER['REQUEST_URI'];
$parts = explode('/', $url);
$method = $_SERVER['REQUEST_METHOD'];

$mode = $parts[2];
$params = isset($parts[3]) ? $parts[3] : '';
```

```
$api =  ucwords($mode);

function response($d) {
        if($d){
                exit(json_encode($d));
        }
        else {
                header('HTTP/1.0 400 Bad Request', true);
                exit();
        }
}

switch ($method)
{
   case 'GET':
                $grub = 'get'.$api;
                $drar = $model->$grub($params);
                response($drar);
                break;
   case 'POST':
      $grub = 'set'.$api;
                $drar = $model->$grub($data);
                response($drar);
                break;
        case 'DELETE':
      $grub = 'delete'.$api;
                $drar = $model->$grub($params);
                response($drar);
                break;
        case 'PATCH':
      $grub = 'change'.$api;
                $drar = $model->$grub($params, $data);
                response($drar);
                break;
        default:
                break;
}


/*
 * Срабатывает при загрузке файла
 */
if(isset($_FILES['file']))
{
   $uploaddir = '../img/';

   if(!is_dir($uploaddir)) {
      mkdir($uploaddir);
   }
   $uploadfile = $uploaddir . iconv("utf-8","windows-1251",$_FILES['file']['name']);
```

```
    if (move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile)) {
      $res =  (object) array("res" => "true");
       exit(json_encode($res));
    } else {
      $res =  (object) array("res" => "false");
       exit(json_encode($res));
    }
}
```

КОД ПРОГРАММЫ (КЛИЕНТСКАЯ ЧАСТЬ)

Приложение Б

(обязательное)

Глобальное пространство
Файл view.js

```
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Helmet } from 'react-helmet'
import { Loader } from 'components'
import { queryLayout } from 'utils'
import NProgress from 'nprogress'
import config from 'utils/config'
import withRouter from 'umi/withRouter'

import PublicLayout from './PublicLayout'
import PrimaryLayout from './PrimaryLayout'
import './BaseLayout.less'

const LayoutMap = {
primary: PrimaryLayout,
public: PublicLayout,
}

@withRouter
@connect(({ loading }) => ({ loading }))
class BaseLayout extends PureComponent {
previousPath = ''

render() {
const { loading, children, location } = this.props
const Container = LayoutMap[queryLayout(config.layouts, location.pathname)]

const currentPath = location.pathname + location.search
if (currentPath !== this.previousPath) {
NProgress.start()
}

if (!loading.global) {
NProgress.done()
this.previousPath = currentPath
}

return (
<Fragment>
<Helmet>
```

```
<title>{config.siteName}</title>
</Helmet>
<Loader fullScreen spinning={loading.effects['app/query']} />
<Container>{children}</Container>
</Fragment>
)
}
}

BaseLayout.propTypes = {
loading: PropTypes.object,
}

export default BaseLayout

/* global document */
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import withRouter from 'umi/withRouter'
import { connect } from 'dva'
import { MyLayout } from 'components'
import { BackTop, Layout, Drawer } from 'antd'
import { GlobalFooter } from 'ant-design-pro'
import { enquireScreen, unenquireScreen } from 'enquire-js'
import { config, pathMatchRegexp, langFromPath } from 'utils'
import Error from '../pages/404'
import styles from './PrimaryLayout.less'

const { Content } = Layout
const { Header, Bread, Sider } = MyLayout

@withRouter
@connect(({ app, loading }) => ({ app, loading }))
class PrimaryLayout extends PureComponent {
state = {
isMobile: false,
}

componentDidMount() {
this.enquireHandler = enquireScreen(mobile => {
const { isMobile } = this.state
if (isMobile !== mobile) {
this.setState({
isMobile: mobile,
})
}
})
}

componentWillUnmount() {
unenquireScreen(this.enquireHandler)
}
```

```
render() {
const { app, location, dispatch, children } = this.props
const {
user,
theme,
routeList,
collapsed,
permissions,
notifications,
} = app
const { isMobile } = this.state
const { onCollapseChange } = this

const lang = langFromPath(location.pathname)
const newRouteList =
lang !== 'en'
? routeList.map(item => {
const { name, ...other } = item
return {
...other,
name: (item[lang] || {}).name || name,
}
})
: routeList

const currentRoute = newRouteList.find(
_ => _.route && pathMatchRegexp(_.route, location.pathname)
)

const hasPermission = currentRoute
? permissions.visit.includes(currentRoute.id)
: false

const menus = newRouteList.filter(_ => _.menuParentId !== '-1')


}

if(permissions) {
headerProps = {...headerProps, notifications}
}

return (
<Fragment>
<Layout>
{isMobile ? (
<Drawer
maskClosable
closable={false}
onClose={onCollapseChange.bind(this, !collapsed)}
visible={!collapsed}
```

```
placement="left"
width={200}
style={{
padding: 0,
height: '100vh',
}}
>
<Sider {...siderProps} collapsed={false} />
</Drawer>
) : (
<Sider {...siderProps} />
)}
<div
className={styles.container}
style={{ paddingTop: config.fixedHeader ? 72 : 0 }}
id="primaryLayout"
>
<Header {...headerProps} />
<Content className={styles.content}>
<Bread routeList={newRouteList} />
{hasPermission ? children : <Error />}
</Content>
<BackTop
className={styles.backTop}
target={() => document.querySelector('#primaryLayout')}
/>
<GlobalFooter
className={styles.footer}
copyright={config.copyright}
/>
</div>
</Layout>
</Fragment>
)
}
}

PrimaryLayout.propTypes = {
children: PropTypes.element.isRequired,
location: PropTypes.object,
dispatch: PropTypes.func,
app: PropTypes.object,
loading: PropTypes.object,
}

export default PrimaryLayout

import React, { Component } from 'react'
import withRouter from 'umi/withRouter'
import { LocaleProvider } from 'antd'
import { I18nProvider } from '@lingui/react'
import { langFromPath, defaultLanguage } from 'utils'
```

```
import zh_CN from 'antd/lib/locale-provider/zh_CN'
import en_US from 'antd/lib/locale-provider/en_US'

import BaseLayout from './BaseLayout'

const languages = {
zh: zh_CN,
en: en_US,
}

@withRouter
class Layout extends Component {
state = {
catalogs: {},
}

language = defaultLanguage

componentDidMount() {
const language = langFromPath(this.props.location.pathname)
this.language = language
this.loadCatalog(language)
}

shouldComponentUpdate(nextProps, nextState) {
const language = langFromPath(nextProps.location.pathname)
const preLanguage = this.language
const { catalogs } = nextState

if (preLanguage !== language && !catalogs[language]) {
this.loadCatalog(language)
this.language = language
return false
}
this.language = language

return true
}

loadCatalog = async language => {
const catalog = await import(/* webpackMode: "lazy", webpackChunkName: "i18n-[index]" */
`@lingui/loader!../locales/${language}/messages.json`)

this.setState(state => ({
catalogs: {
...state.catalogs,
[language]: catalog,
},
}))
}

render() {
```

```
const { location, children } = this.props
const { catalogs } = this.state

let language = langFromPath(location.pathname)

if (!catalogs[language]) language = defaultLanguage

return (
<LocaleProvider locale={languages[language]}>
<I18nProvider language={language} catalogs={catalogs}>
<BaseLayout>{children}</BaseLayout>
</I18nProvider>
</LocaleProvider>
)
}
}

export default Layout
```

Файл action.js

```
onCollapseChange = collapsed => {
this.props.dispatch({
type: 'app/handleCollapseChange',
payload: collapsed,
})
}

let headerProps = {
menus,
collapsed,
onCollapseChange,
avatar: user.avatar,
name: user.f_name,
fixed: config.fixedHeader,
onAllNotificationsRead() {
dispatch({ type: 'app/allNotificationsRead' })
},
onSignOut() {
dispatch({ type: 'app/signOut' })
},

const siderProps = {
theme,
menus,
isMobile,
collapsed,
onCollapseChange,
onThemeChange(theme) {
dispatch({
```

```
type: 'app/handleThemeChange',
payload: theme,
})
},
}
```

Файл model.js

```
/* global window */

import { router } from 'utils'
import { stringify } from 'qs'
import store from 'store'
import { ROLE_TYPE } from 'utils/constant'
import { queryLayout, pathMatchRegexp } from 'utils'
import { CANCEL_REQUEST_MESSAGE } from 'utils/constant'
import api from 'api'
import config from 'config'

const { queryRouteList, logoutUser, queryUserInfo } = api

export default {
namespace: 'app',
state: {
user: {},
permissions: {
visit: [],
},
routeList: [
{
id: '1',
icon: 'laptop',
name: 'Dashboard',
router: '/dashboard',
},
],
locationPathname: '',
locationQuery: {},
theme: store.get('theme') || 'light',
collapsed: store.get('collapsed') || false,
notifications: [
{
title: 'Зарегистрирован новый пользователь.',
date: new Date(Date.now() - 10000000),
},
{
title: 'Application has been approved.',
date: new Date(Date.now() - 50000000),
},
```

```
],
},
subscriptions: {
setupHistory({ dispatch, history }) {
history.listen(location => {
dispatch({
type: 'updateState',
payload: {
locationPathname: location.pathname,
locationQuery: location.query,
},
})
})
},

setupRequestCancel({ history }) {
history.listen(() => {
const { cancelRequest = new Map() } = window

cancelRequest.forEach((value, key) => {
if (value.pathname !== window.location.pathname) {
value.cancel(CANCEL_REQUEST_MESSAGE)
cancelRequest.delete(key)
}
})
})
},

setup({ dispatch }) {
dispatch({ type: 'query' })
},
},
effects: {
*query({ payload }, { call, put, select }) {
const { success, user } = yield call(queryUserInfo, payload)
const { locationPathname } = yield select(_ => _.app)

if (success && user) {
const { list } = yield call(queryRouteList)
const { permissions } = user
let routeList = list

routeList = list.filter(item => {
const cases = [
permissions.visit.includes(item.id),
item.mpid
? permissions.visit.includes(item.mpid) || item.mpid === '-1'
: true,
item.bpid ? permissions.visit.includes(item.bpid) : true,
]
return cases.every(_ => _)
})
```

```
yield put({
type: 'updateState',
payload: {
user,
permissions,
routeList,
},
})
if (pathMatchRegexp(['/','/login'], window.location.pathname)) {

if(permissions.role === 'Admin') {
router.push({
pathname: '/user',
})
}
else {
router.push({
pathname: '/dashboard',
})
}
}
} else if (queryLayout(config.layouts, locationPathname) !== 'public') {
router.push({
pathname: '/login',
search: stringify({
from: locationPathname,
}),
})
}
},

*signOut({ payload }, { call, put }) {
const data = yield call(logoutUser)
if (data.success) {
yield put({
type: 'updateState',
payload: {
user: {},
permissions: { visit: [] },
menu: [
{
id: '1',
icon: 'laptop',
name: 'Dashboard',
router: '/dashboard',
},
],
},
})
yield put({ type: 'query' })
} else {
```

```
throw data
}
},
},
reducers: {
updateState(state, { payload }) {
return {
...state,
...payload,
}
},

handleThemeChange(state, { payload }) {
store.set('theme', payload)
state.theme = payload
},

handleCollapseChange(state, { payload }) {
store.set('collapsed', payload)
state.collapsed = payload
},

allNotificationsRead(state) {
state.notifications = []
},
},
}
```

Файл service_routes.js

```
export default {
queryRouteList: '/routes',

queryUserInfo: '/user',
logoutUser: '/logout',
loginUser: 'POST /login',

queryUser: '/user/:id',
queryUserList: '/users',
updateUser: 'Patch /user/:id',
createUser: 'POST /user',
removeUser: 'DELETE /user/:id',
removeUserList: 'POST /users/delete',

queryProductList: '/products',
createProduct: 'POST /productInCart',

queryCart: '/userCart',

queryDashboard: '/userHistoryOrder',

queryBackups: '/backups',
```

```
createBackup: 'POST /backup',
removeBackup: 'DELETE /backup/:id',
removeBackupsList: 'POST /backups/delete',

}
```

Файл login.js (View)

```
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Button, Row, Form, Input } from 'antd'

import { Trans, withI18n } from '@lingui/react'

import config from 'utils/config'

import styles from './index.less'
const FormItem = Form.Item

@withI18n()
@connect(({ loading }) => ({ loading }))
@Form.create()
class Login extends PureComponent {

render() {
const { loading, form, i18n } = this.props
const { getFieldDecorator } = form

return (
<Fragment>
<div className={styles.form}>
<div className={styles.logo}>
<span>{config.siteName}</span>
</div>
<form>
<FormItem hasFeedback>
{getFieldDecorator('username', {
rules: [
{
required: true,
},
],
})(
<Input
onPressEnter={this.handleOk}
placeholder={i18n.t`Username`}
/>
```

```
)}
</FormItem>
<FormItem hasFeedback>
{getFieldDecorator('password', {
rules: [
{
required: true,
},
],
})(
<Input
type="password"
onPressEnter={this.handleOk}
placeholder={i18n.t`Password`}
/>
)}
</FormItem>
<Row>
<Button
type="primary"
onClick={this.handleOk}
loading={loading.effects.login}
>
<Trans>Sign in</Trans>
</Button>
</Row>
</form>
</div>
</Fragment>
)
}
}

Login.propTypes = {
form: PropTypes.object,
dispatch: PropTypes.func,
loading: PropTypes.object,
}

export default Login
```

Файл clogin.js (Controller)

```
handleOk = () => {
const { dispatch, form } = this.props
const { validateFieldsAndScroll } = form
validateFieldsAndScroll((errors, values) => {
if (errors) {
return
}
dispatch({ type: 'login/login', payload: values })
})
```

```
}
```

Файл mlogin.js (Model)

```
import { router, pathMatchRegexp } from 'utils'
import api from 'api'

const { loginUser } = api

export default {
namespace: 'login',

state: {},

effects: {
*login({ payload }, { put, call, select }) {
const data = yield call(loginUser, payload)
if (data.success) {
yield put({ type: 'app/query' })
const { locationQuery } = yield select(_ => _.app)
const { from } = locationQuery
if (!pathMatchRegexp('/login', from)) {
if(data.group === "1") {
router.push('/user')
} else {
router.push('/dashboard')
}
} else {
router.push('/dashboard')
}
} else {
throw data
}
},
},
}
```

Файл users.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { router } from 'utils'
import { connect } from 'dva'
import { Row, Col, Button, Popconfirm } from 'antd'
import { withI18n } from '@lingui/react'
import { Page } from 'components'
import { stringify } from 'qs'
import List from './components/List'
import Filter from './components/Filter'
import Modal from './components/Modal'
```

```
@withI18n()
@connect(({ user, loading }) => ({ user, loading }))
class User extends PureComponent {
render() {
const { location, dispatch, user, loading, i18n } = this.props
const { query, pathname } = location
const {
list,
pagination,
currentItem,
modalVisible,
modalType,
selectedRowKeys,
} = user


return (
<Page inner>
<Filter {...filterProps} />
{selectedRowKeys.length > 0 && (
<Row style={{ marginBottom: 24, textAlign: 'right', fontSize: 13 }}>
<Col>
{`Selected ${selectedRowKeys.length} items `}
<Popconfirm
title="Are you sure delete these items?"
placement="left"
onConfirm={handleDeleteItems}
>
<Button type="primary" style={{ marginLeft: 8 }}>
Remove
</Button>
</Popconfirm>
</Col>
</Row>
)}
<List {...listProps} />
{modalVisible && <Modal {...modalProps} />}
</Page>
)
}
}

User.propTypes = {
user: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
loading: PropTypes.object,
}

export default User

import React, { PureComponent } from 'react'
```

```
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Page } from 'components'
import styles from './index.less'

@connect(({ userDetail }) => ({ userDetail }))
class UserDetail extends PureComponent {
render() {
const { userDetail } = this.props
const { data } = userDetail
const content = []
for (let key in data) {
if ({}.hasOwnProperty.call(data, key)) {
content.push(
<div key={key} className={styles.item}>
<div>{key}</div>
<div>{String(data[key])}</div>
</div>
)
}
}
return (
<Page inner>
<div className={styles.content}>{content}</div>
</Page>
)
}
}

UserDetail.propTypes = {
userDetail: PropTypes.object,
}

export default UserDetail

/* global document */
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import moment from 'moment'
import { FilterItem } from 'components'
import { Trans, withI18n } from '@lingui/react'
import { Form, Button, Row, Col, DatePicker, Input, Cascader } from 'antd'
import city from 'utils/city'

const { Search } = Input
const { RangePicker } = DatePicker

const ColProps = {
xs: 24,
sm: 12,
style: {
marginBottom: 16,
```

```
},
}

const TwoColProps = {
...ColProps,
xl: 96,
}

@withI18n()
@Form.create()
class Filter extends PureComponent {
handleFields = fields => {
const { createTime } = fields
if (createTime.length) {
fields.createTime = [
moment(createTime[0]).format('YYYY-MM-DD'),
moment(createTime[1]).format('YYYY-MM-DD'),
]
}
return fields
}

handleSubmit = () => {
const { onFilterChange, form } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields = this.handleFields(fields)
onFilterChange(fields)
}

handleReset = () => {
const { form } = this.props
const { getFieldsValue, setFieldsValue } = form

const fields = getFieldsValue()
for (let item in fields) {
if ({}.hasOwnProperty.call(fields, item)) {
if (fields[item] instanceof Array) {
fields[item] = []
} else {
fields[item] = undefined
}
}
}
setFieldsValue(fields)
this.handleSubmit()
}
handleChange = (key, values) => {
const { form, onFilterChange } = this.props
const { getFieldsValue } = form
```

```
let fields = getFieldsValue()
fields[key] = values
fields = this.handleFields(fields)
onFilterChange(fields)
}

render() {
const { onAdd, filter, form, i18n } = this.props
const { getFieldDecorator } = form
const { name } = filter

let initialCreateTime = []
if (filter.createTime && filter.createTime[0]) {
initialCreateTime[0] = moment(filter.createTime[0])
}
if (filter.createTime && filter.createTime[1]) {
initialCreateTime[1] = moment(filter.createTime[1])
}

return (
<Row gutter={24}>
<Col {...ColProps} xl={{ span: 5 }} md={{ span: 8 }}>
{getFieldDecorator('name', { initialValue: name })(
<Search
placeholder={i18n.t`Search Name`}
onSearch={this.handleSubmit}
/>
)}
</Col>
<Col
{...ColProps}
xl={{ span: 7 }}
md={{ span: 8 }}
sm={{ span: 12 }}
id="createTimeRangePicker"
>
<FilterItem label={i18n.t`CreateTime`}>
{getFieldDecorator('createTime', {
initialValue: initialCreateTime,
})(
<RangePicker
style={{ width: '100%' }}
onChange={this.handleChange.bind(this, 'createTime')}
getCalendarContainer={() => {
return document.getElementById('createTimeRangePicker')
}}
/>
)}
</FilterItem>
</Col>
<Col
{...TwoColProps}
```

```
xl={{ span: 12 }}
md={{ span: 24 }}
sm={{ span: 24 }}
>
<Row type="flex" align="right" justify="space-between">
<div>
<Button
type="primary"
className="margin-right"
onClick={this.handleSubmit}
>
<Trans>Search</Trans>
</Button>
<Button onClick={this.handleReset}>
<Trans>Reset</Trans>
</Button>
</div>
<Button type="ghost" onClick={onAdd}>
<Trans>Create</Trans>
</Button>
</Row>
</Col>
</Row>
)
}
}

Filter.propTypes = {
onAdd: PropTypes.func,
form: PropTypes.object,
filter: PropTypes.object,
onFilterChange: PropTypes.func,
}

export default Filter


import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Table, Modal } from 'antd'
import { DropOption } from 'components'
import { Trans, withI18n } from '@lingui/react'
import Link from 'umi/link'
import styles from './List.less'

const { confirm } = Modal

@withI18n()
class List extends PureComponent {
handleMenuClick = (record, e) => {
const { onDeleteItem, onEditItem, i18n } = this.props
```

```
if (e.key === '1') {
onEditItem(record)
} else if (e.key === '2') {
confirm({
title: i18n.t`Are you sure delete this record?`,
onOk() {
onDeleteItem(record.id)
},
})
}
}

render() {
const { onDeleteItem, onEditItem, i18n, ...tableProps } = this.props

const columns = [
{
title: <Trans>NickName</Trans>,
dataIndex: 'nickName',
key: 'nickName',
width: 72,
fixed: 'left',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>Name</Trans>,
dataIndex: 'f_name',
key: 'f_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>F_name</Trans>,
dataIndex: 's_name',
key: 's_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>L_name</Trans>,
dataIndex: 'l_name',
key: 'l_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>Group</Trans>,
dataIndex: 'group',
key: 'group',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>CreateTime</Trans>,
dataIndex: 'createTime',
key: 'createTime',
```

```
},
{
title: <Trans>Operation</Trans>,
key: 'operation',
fixed: 'right',
render: (text, record) => {
return (
<DropOption
onMenuClick={e => this.handleMenuClick(record, e)}
menuOptions={[
{ key: '1', name: i18n.t`Update` },
{ key: '2', name: i18n.t`Delete` },
]}
/>
)
},
},
]

return (
<Table
{...tableProps}
pagination={{
...tableProps.pagination,
showTotal: total => i18n.t`Total ${total} Items`,
}}
className={styles.table}
bordered
scroll={{ x: 1200 }}
columns={columns}
simple
rowKey={record => record.id}
/>
)
}
}

List.propTypes = {
onDeleteItem: PropTypes.func,
onEditItem: PropTypes.func,
location: PropTypes.object,
}

export default List

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Form, Input, Checkbox , Modal, Select, Layout } from 'antd'
import { withI18n } from '@lingui/react'

const FormItem = Form.Item
```

```
const {
Header, Content,
} = Layout

const CheckboxGroup = Checkbox.Group;

const plainOptions = ['Apple', 'Pear', 'Orange'];

const formItemLayout = {
labelCol: {
span: 6,
},
wrapperCol: {
span: 14,
}
}
@withI18n()
@Form.create()
class UserModal extends PureComponent {
state = {
group: null,
checkedList: null,
indeterminate: true,
checkAll: false,
}

handleOk = () => {
const { item = {}, onOk, form } = this.props
const { validateFields, getFieldsValue } = form

validateFields(errors => {
if (errors) {
return
}
const data = {
...getFieldsValue(),
key: item.key,
}

onOk(data)
})
}

handleSelect = (val) => {
this.setState({
group: val
})
}

onChange = (checkedList) => {
this.setState({
checkedList,
```

```
indeterminate: !!checkedList.length && (checkedList.length < plainOptions.length),
checkAll: checkedList.length === plainOptions.length,
});
}

onCheckAllChange = (e) => {
this.setState({
checkedList: e.target.checked ? plainOptions : [],
indeterminate: false,
checkAll: e.target.checked,
});
}

render() {
const { item = {}, onOk, form, i18n, ...modalProps } = this.props
const { getFieldDecorator } = form

return (
<Modal {...modalProps} onOk={this.handleOk}>
<Form layout="horizontal">
<FormItem label={i18n.t`Name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('name', {
initialValue: item.f_name,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`F_name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('s_name', {
initialValue: item.s_name,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`L_name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('f_name', {
initialValue: item.f_name,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>

<FormItem label={i18n.t`NickName`} hasFeedback {...formItemLayout}>
```

```
{getFieldDecorator('nickName', {
initialValue: item.nickName,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`Password`} hasFeedback {...formItemLayout}>
{getFieldDecorator('password', {
initialValue: item.password,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`Group`} hasFeedback {...formItemLayout}>
{getFieldDecorator('group', {
rules: [
{
required: true,
},
],
initialValue: item.group,
})(
<Select
placeholder="Выбрать группу пользователя"
onChange={this.handleSelect}
>
<Select.Option value="2">Менеджер</Select.Option>
<Select.Option value="1">Администратор</Select.Option>
</Select>
)}
</FormItem>
{
this.state.group && this.state.group === "manager" ? (
<FormItem >
<Layout style={{
backgroundColor: "transparent"
}}>
<Header style={{
backgroundColor: "transparent",
textAlign: 'center',
fontSize: 20
}}>
{i18n.t`Select category`}
</Header>
<Content>
<Checkbox
```

```
indeterminate={this.state.indeterminate}
onChange={this.onCheckAllChange}
checked={this.state.checkAll}
>
Выбрать всё
</Checkbox>
<br />
{getFieldDecorator('category', {
initialValue: item.category,
rules: [
{
required: true,
},
],
})(
<CheckboxGroup options={plainOptions} onChange={this.onChange} />
)}
</Content>
</Layout>
</FormItem>
) : null

}
</Form>
</Modal>
)
}
}

UserModal.propTypes = {
type: PropTypes.string,
item: PropTypes.object,
onOk: PropTypes.func,
}

export default UserModal
```

Файл cusers.js (Controllers)

```
const handleRefresh = newQuery => {
router.push({
pathname,
search: stringify(
{
...query,
...newQuery,
},
{ arrayFormat: 'repeat' }
),
```

```
})
}

const modalProps = {
item: modalType === 'create' ? {} : currentItem,
visible: modalVisible,
maskClosable: false,
confirmLoading: loading.effects[`user/${modalType}`],
title: `${
modalType === 'create' ? i18n.t`Create User` : i18n.t`Update User`
}`,
centered: true,
onOk(data) {
dispatch({
type: `user/${modalType}`,
payload: data,
}).then(() => {
handleRefresh()
})
},
onCancel() {
dispatch({
type: 'user/hideModal',
})
},
}

const listProps = {
dataSource: list,
loading: loading.effects['user/query'],
pagination,
onChange(page) {
handleRefresh({
page: page.current,
pageSize: page.pageSize,
})
},
onDeleteItem(id) {
dispatch({
type: 'user/delete',
payload: id,
}).then(() => {
handleRefresh({
page:
list.length === 1 && pagination.current > 1
? pagination.current - 1
: pagination.current,
})
})
},
onEditItem(item) {
dispatch({
```

```
type: 'user/showModal',
payload: {
modalType: 'update',
currentItem: item,
},
})
},
rowSelection: {
selectedRowKeys,
onChange: keys => {
dispatch({
type: 'user/updateState',
payload: {
selectedRowKeys: keys,
},
})
},
},
}

const filterProps = {
filter: {
...query,
},
onFilterChange(value) {
handleRefresh({
...value,
page: 1,
})
},
onAdd() {
dispatch({
type: 'user/showModal',
payload: {
modalType: 'create',
},
})
},
}

const handleDeleteItems = () => {
dispatch({
type: 'user/multiDelete',
payload: {
ids: selectedRowKeys,
},
}).then(() => {
handleRefresh({
page:
list.length === selectedRowKeys.length && pagination.current > 1
? pagination.current - 1
: pagination.current,
```

```
})
})
}
```

Файл cusers.js (Models)

```
/* global window */
import modelExtend from 'dva-model-extend'
import { pathMatchRegexp } from 'utils'
import api from 'api'
import { pageModel } from 'utils/model'

const {
queryUserList,
createUser,
removeUser,
updateUser,
removeUserList,
} = api

export default modelExtend(pageModel, {
namespace: 'user',

state: {
currentItem: {},
modalVisible: false,
modalType: 'create',
selectedRowKeys: [],
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/user', location.pathname)) {
const payload = location.query || { page: 1, pageSize: 10 }
dispatch({
type: 'query',
payload,
})
}
})
},
},

effects: {
*query({ payload = {} }, { call, put }) {
const data = yield call(queryUserList, payload)
if (data) {
yield put({
type: 'querySuccess',
```

```
payload: {
list: data.data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
}
},

*delete({ payload }, { call, put, select }) {
const data = yield call(removeUser, { id: payload })
const { selectedRowKeys } = yield select(_ => _.user)
if (data.success) {
yield put({
type: 'updateState',
payload: {
selectedRowKeys: selectedRowKeys.filter(_ => _ !== payload),
},
})
} else {
throw data
}
},

*multiDelete({ payload }, { call, put }) {
const data = yield call(removeUserList, payload)
if (data.success) {
yield put({ type: 'updateState', payload: { selectedRowKeys: [] } })
} else {
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createUser, payload)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
throw data
}
},

*update({ payload }, { select, call, put }) {
const id = yield select(({ user }) => user.currentItem.id)
const newUser = { ...payload, id }
const data = yield call(updateUser, newUser)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
```

```
throw data
}
},
},

reducers: {
showModal(state, { payload }) {
return { ...state, ...payload, modalVisible: true }
},

hideModal(state) {
return { ...state, modalVisible: false }
},
},
})


import { pathMatchRegexp } from 'utils'
import api from 'api'

const { queryUser } = api

export default {
namespace: 'userDetail',

state: {
data: {},
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
const match = pathMatchRegexp('/user/:id', pathname)
if (match) {
dispatch({ type: 'query', payload: { id: match[1] } })
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryUser, payload)
const { success, message, status, ...other } = data
if (success) {
yield put({
type: 'querySuccess',
payload: {
data: other,
},
})
} else {
```

```
throw data
}
},
},

reducers: {
querySuccess(state, { payload }) {
const { data } = payload
return {
...state,
data,
}
},
},
}
```

Файл Dashboard.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Row, Col, Card, Icon } from 'antd'

import { Page } from 'components'

import styles from './index.less'

const Color = {
green: '#64ea91',
blue: '#8fc9fb',
purple: '#d897eb',
red: '#f69899',
yellow: '#f8c82e',
peach: '#f797d6',
borderBase: '#e5e5e5',
borderSplit: '#f4f4f4',
grass: '#d6fbb5',
sky: '#c1e0fc',
}

const statuses = [
{
title: "Заказ успешно доставлен",
color: Color.green
},
{
title: "Заказ ожидает доставки",
color: Color.yellow
},
```

```
{
title: "Заказ отменен",
color: Color.red
}
]

@connect(({ app, dashboard, loading }) => ({
username: app.user.username,
dashboard,
loading,
}))
class Dashboard extends PureComponent {
render() {
const { dashboard } = this.props
const {
data,
} = dashboard

const numberCards = () => data.map((item, key) => (
<Col key={key} lg={6} md={12}>
<Card
title={`Заказ #${item.id}`}
style={{minHeight: 180, backgroundColor: statuses[item.status -1].color}}
actions={[<Icon type="eye" />, <Icon type="edit" />]}
>
<p style={{ textAlign: 'center' }}>{statuses[item.status -1].title}</p>
<p style={{ paddingTop: 15 }}>{item.date}</p>

</Card>
</Col>
))

return (
<Page
className={styles.dashboard}
>
<Row gutter={24}>
{data && numberCards()}
</Row>
</Page>
)
}
}

Dashboard.propTypes = {
username: PropTypes.string,
dashboard: PropTypes.object,
loading: PropTypes.object,
}

export default Dashboard
```

```
import React from 'react'
import PropTypes from 'prop-types'
import { Icon, Card } from 'antd'
import styles from './numberCard.less'

function NumberCard({ icon, color, title, status, date, countUp }) {
return (
<Card
className={styles.numberCard}
bordered={false}
bodyStyle={{ padding: 10 }}
>
<Icon className={styles.iconWarp} style={{ color }} type={icon} />
<div className={styles.content}>
<p className={styles.title}>{title || 'No Title'}</p>
<p className={styles.title}>
{status}
</p>
<p className={styles.title}>
{date}
</p>
</div>
</Card>
)
}

NumberCard.propTypes = {
icon: PropTypes.string,
color: PropTypes.string,
title: PropTypes.string,
status: PropTypes.string,
date: PropTypes.string,
countUp: PropTypes.object,
}

export default NumberCard
```

Файл Dashboard.js (Controllers)

```
import { request, config } from 'utils'

const { api } = config
const { dashboard } = api

export function query(params) {
return request({
url: dashboard,
withCredentials: true,
```

```
method: 'get',
data: params,
})
}
```

Файл Dashboard.js (Model)

```
import { parse } from 'qs'
import modelExtend from 'dva-model-extend'
import api from 'api'
import { pathMatchRegexp } from 'utils'
import { model } from 'utils/model'

const { queryDashboard } = api

export default modelExtend(model, {
namespace: 'dashboard',
state: {
sales: [],
numbers: [],
recentSales: [],
user: {},
subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
if (
pathMatchRegexp('/dashboard', pathname) ||
pathMatchRegexp('/', pathname)
) {
dispatch({ type: 'query' })
}
})
},
},
effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryDashboard, parse(payload))
yield put({
type: 'updateState',
payload: data,
})
},
},
})
```

Файл Products.js (Model)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Row, Col, Card, Icon, Pagination } from 'antd'

import { Page } from 'components'
import { router } from 'utils'
import { stringify } from 'qs'
import { withI18n } from '@lingui/react'

@withI18n()
@connect(({ products, loading }) => ({ products, loading }))
class Products extends PureComponent {

render() {
const { products, loading, location, dispatch, i18n } = this.props
const { list } = products

const productCards = list.map((item, key) => (
<Col key={key} lg={6} md={12}>
<Card
title={item.Name}
style={{minHeight: 180, marginBottom: 30 }}
actions={[<Icon type="plus-circle" onClick={ () => handleClick({product_id: item.id_product,
count: 1}) }/>]}
>
<p style={{ }}>{`Стоимость: ${item.price} руб. за 1 шт.`}</p>
<p style={{ paddingTop: 15 }}>{`На складе осталось: ${item.count} шт.`}</p>

</Card>
</Col>
))

return (
<Page
>
<Row gutter={24}>
{productCards}
</Row>
</Page>
)
}
}

Products.propTypes = {
products: PropTypes.object,
loading: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
}

export default Products
```

```
import React, { PureComponent } from 'react'
import { Table, Avatar } from 'antd'
import { withI18n } from '@lingui/react'
import { Ellipsis } from 'ant-design-pro'
import styles from './List.less'

@withI18n()
class List extends PureComponent {
render() {
const { i18n, ...tableProps } = this.props
const columns = [
{
title: i18n.t`Image`,
dataIndex: 'image',
render: text => <Avatar shape="square" src={text} />,
},
{
title: i18n.t`Title`,
dataIndex: 'title',
render: text => (
<Ellipsis tooltip length={30}>
{text}
</Ellipsis>
),
},
{
title: i18n.t`Author`,
dataIndex: 'author',
},
{
title: i18n.t`Categories`,
dataIndex: 'categories',
},
{
title: i18n.t`Tags`,
dataIndex: 'tags',
},
{
title: i18n.t`Visibility`,
dataIndex: 'visibility',
},
{
title: i18n.t`Comments`,
dataIndex: 'comments',
},
{
title: i18n.t`Views`,
dataIndex: 'views',
},
{
title: i18n.t`Publish Date`,
```

```
dataIndex: 'date',
},
]

return (
<Table
{...tableProps}
pagination={{
...tableProps.pagination,
showTotal: total => i18n.t`Total ${total} Items`,
}}
bordered
scroll={{ x: 1200 }}
className={styles.table}
columns={columns}
simple
rowKey={record => record.id}
/>
)
}
}

export default List
```

Файл cProducts.js (Controllers)

```
const handleClick = data => {
dispatch({
type: `products/create`,
payload: data,
}).then(() => {

})
}

handleChange = (current,pageSize) => {
const {
location
} = this.props

const {
query,
pathname
} = location

router.push({
pathname,
search: stringify({
...query,
page: current,
```

```
pageSize: pageSize,
}),
})
}
```

Файл mProducts.js (Model)

```
import modelExtend from 'dva-model-extend'
import api from 'api'
import { pathMatchRegexp } from 'utils'
import { pageModel } from 'utils/model'

const {
queryProductList,
createProduct,
} = api

export default modelExtend(pageModel, {
namespace: 'products',

subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/products', location.pathname)) {
dispatch({
type: 'query',
payload: {
...location.query,
},
})
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryProductList, payload)
if (data.success) {
yield put({
type: 'querySuccess',
payload: {
list: data.data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
} else {
```

```
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createProduct, payload)
if (data.success) {

} else {
throw data
}
},
},
})
```

Файл Backups.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { router } from 'utils'
import { connect } from 'dva'
import { Row, Col, Button, Popconfirm } from 'antd'
import { withI18n } from '@lingui/react'
import { Page } from 'components'
import { stringify } from 'qs'
import List from './components/List'
import Filter from './components/Filter'

@withI18n()
@connect(({ backups, loading }) => ({ backups, loading }))
class Backups extends PureComponent {
render() {
const { location, dispatch, backups, loading, i18n } = this.props
const { query, pathname } = location
const {
list,
pagination,
selectedRowKeys,
} = backups


return (
<Page inner>
<Filter {...filterProps} />
{selectedRowKeys.length > 0 && (
<Row style={{ marginBottom: 24, textAlign: 'right', fontSize: 13 }}>
<Col>
{`Выбрано ${selectedRowKeys.length} записей `}
<Popconfirm
title="Вы уверены, что хотите удалить?"
```

```
placement="left"
onConfirm={handleDeleteItems}
>
<Button type="primary" style={{ marginLeft: 8 }}>
Удалить
</Button>
</Popconfirm>
</Col>
</Row>
)}
<List {...listProps} />
</Page>
)
}
}

Backups.propTypes = {
backups: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
loading: PropTypes.object,
}

export default Backups

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import moment from 'moment'
import { FilterItem } from 'components'
import { Trans, withI18n } from '@lingui/react'
import { Form, Button, Row, Col, DatePicker, Input, Cascader } from 'antd'
import city from 'utils/city'

const { Search } = Input
const { RangePicker } = DatePicker

const ColProps = {
xs: 24,
sm: 12,
style: {
marginBottom: 16,
},
}

const TwoColProps = {
...ColProps,
xl: 96,
}

@withI18n()
@Form.create()
class Filter extends PureComponent {
```

```
handleFields = fields => {
const { createTime } = fields
if (createTime.length) {
fields.createTime = [
moment(createTime[0]).format('YYYY-MM-DD'),
moment(createTime[1]).format('YYYY-MM-DD'),
]
}
return fields
}

handleSubmit = () => {
const { onFilterChange, form } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields = this.handleFields(fields)
onFilterChange(fields)
}

handleReset = () => {
const { form } = this.props
const { getFieldsValue, setFieldsValue } = form

const fields = getFieldsValue()
for (let item in fields) {
if ({}.hasOwnProperty.call(fields, item)) {
if (fields[item] instanceof Array) {
fields[item] = []
} else {
fields[item] = undefined
}
}
}
setFieldsValue(fields)
this.handleSubmit()
}
handleChange = (key, values) => {
const { form, onFilterChange } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields[key] = values
fields = this.handleFields(fields)
onFilterChange(fields)
}

render() {
const { onAdd, filter, form, i18n } = this.props
const { getFieldDecorator } = form
const { name } = filter
```

```
let initialCreateTime = []
if (filter.createTime && filter.createTime[0]) {
initialCreateTime[0] = moment(filter.createTime[0])
}
if (filter.createTime && filter.createTime[1]) {
initialCreateTime[1] = moment(filter.createTime[1])
}

return (
<Row gutter={24}>
<Col {...ColProps} xl={{ span: 5 }} md={{ span: 8 }}>
{getFieldDecorator('name', { initialValue: name })(
<Search
placeholder={i18n.t`Search Name`}
onSearch={this.handleSubmit}
/>
)}
</Col>
<Col
{...ColProps}
xl={{ span: 7 }}
md={{ span: 8 }}
sm={{ span: 12 }}
id="createTimeRangePicker"
>
<FilterItem label={i18n.t`CreateTime`}>
{getFieldDecorator('createTime', {
initialValue: initialCreateTime,
})(
<RangePicker
style={{ width: '100%' }}
onChange={this.handleChange.bind(this, 'createTime')}
getCalendarContainer={() => {
return document.getElementById('createTimeRangePicker')
}}
/>
)}
</FilterItem>
</Col>
<Col
{...TwoColProps}
xl={{ span: 12 }}
md={{ span: 24 }}
sm={{ span: 24 }}
>
<Row type="flex" align="right" justify="space-between">
<div>
<Button
type="primary"
className="margin-right"
onClick={this.handleSubmit}
>
```

```
<Trans>Search</Trans>
</Button>
<Button onClick={this.handleReset}>
<Trans>Reset</Trans>
</Button>
</div>
<Button type="ghost" onClick={onAdd}>
<Trans>Create</Trans>
</Button>
</Row>
</Col>
</Row>
)
}
}

Filter.propTypes = {
onAdd: PropTypes.func,
form: PropTypes.object,
filter: PropTypes.object,
onFilterChange: PropTypes.func,
}

export default Filter

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Table, Modal } from 'antd'
import { DropOption } from 'components'
import { Trans, withI18n } from '@lingui/react'
import Link from 'umi/link'
import styles from './List.less'

const { confirm } = Modal

@withI18n()
class List extends PureComponent {
handleMenuClick = (record, e) => {
const { onDeleteItem, onDownloadItem, i18n } = this.props

if (e.key === '1') {
onDownloadItem(record)
} else if (e.key === '2') {
confirm({
title: i18n.t`Are you sure delete this record?`,
onOk() {
onDeleteItem(record.id)
},
})
}
}
```

```
render() {
const { onDeleteItem, onDownloadItem, i18n, ...tableProps } = this.props

const columns = [
{
title: <Trans>Title</Trans>,
dataIndex: 'name',
key: 'name',
},
{
title: <Trans>CreateTime</Trans>,
dataIndex: 'createDate',
key: 'createDate',
},
{
title: <Trans>Operation</Trans>,
key: 'operation',
fixed: 'right',
render: (text, record) => {
return (
<DropOption
onMenuClick={e => this.handleMenuClick(record, e)}
menuOptions={[
{ key: '1', name: i18n.t`Download` },
{ key: '2', name: i18n.t`Delete` },
]}
/>
)
},
},
]

return (
<Table
{...tableProps}

className={styles.table}
bordered
columns={columns}
simple
rowKey={record => record.id}
/>
)
}
}

List.propTypes = {
onDeleteItem: PropTypes.func,
onEditItem: PropTypes.func,
location: PropTypes.object,
}
```

```
export default List

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Page } from 'components'
import styles from './index.less'

@connect(({ backupDetail }) => ({ backupDetail }))
class BackupDetail extends PureComponent {
render() {
const { backupDetail } = this.props
const { data } = backupDetail
const content = []
for (let key in data) {
if ({}.hasOwnProperty.call(data, key)) {
content.push(
<div key={key} className={styles.item}>
<div>{key}</div>
<div>{String(data[key])}</div>
</div>
)
}
}
return (
<Page inner>
<div className={styles.content}>{content}</div>
</Page>
)
}
}

BackupDetail.propTypes = {
backupDetail: PropTypes.object,
}

export default BackupDetail
```

Файл cBackups.js (Controllers)

```
const handleRefresh = newQuery => {
router.push({
pathname,
search: stringify(
{
...query,
...newQuery,
},
{ arrayFormat: 'repeat' }
```

```
),
})
}

const listProps = {
dataSource: list.data,
loading: loading.effects['backups/query'],
onDownloadItem(item) {
window.open(`${document.location.origin}/${item.path}/${item.name}`, '_blank')
},
onDeleteItem(id) {
dispatch({
type: 'backups/delete',
payload: id,
}).then(() => {
handleRefresh()
})
},
rowSelection: {
selectedRowKeys,
onChange: keys => {
dispatch({
type: 'backups/updateState',
payload: {
selectedRowKeys: keys,
},
})
},
},
}

const filterProps = {
filter: {
...query,
},
onFilterChange(value) {
handleRefresh({
...value,
page: 1,
})
},
onAdd(data) {
dispatch({
type: `backups/create`,
payload: data,
}).then(() => {
handleRefresh()
})
},
}

const handleDeleteItems = () => {
```

```
dispatch({
type: 'backups/multiDelete',
payload: {
ids: selectedRowKeys,
},
}).then(() => {
handleRefresh({
page:
list.length === selectedRowKeys.length && pagination.current > 1
? pagination.current - 1
: pagination.current,
})
})
}
```

Файл cBackups.js (Models)

```
/* global window */
import modelExtend from 'dva-model-extend'
import { pathMatchRegexp } from 'utils'
import api from 'api'
import { pageModel } from 'utils/model'

const {
queryBackups,
createBackup,
removeBackup,
removeBackupsList,
} = api

export default modelExtend(pageModel, {
namespace: 'backups',

state: {
currentItem: {},
modalVisible: false,
modalType: 'create',
selectedRowKeys: [],
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/backups', location.pathname)) {
const payload = location.query || { page: 1, pageSize: 10 }
dispatch({
type: 'query',
payload,
})
}
})
```

```
},
},

effects: {
*query({ payload = {} }, { call, put }) {
const data = yield call(queryBackups, payload)
if (data) {
yield put({
type: 'querySuccess',
payload: {
list: data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
}
},

*delete({ payload }, { call, put, select }) {
const data = yield call(removeBackup, { id: payload })
const { selectedRowKeys } = yield select(_ => _.user)
if (data.success) {
yield put({
type: 'updateState',
payload: {
selectedRowKeys: selectedRowKeys.filter(_ => _ !== payload),
},
})
} else {
throw data
}
},

*multiDelete({ payload }, { call, put }) {
const data = yield call(removeBackupsList, payload)
if (data.success) {
yield put({ type: 'updateState', payload: { selectedRowKeys: [] } })
} else {
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createBackup, payload)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
throw data
}
```

```
},
},

reducers: {
showModal(state, { payload }) {
return { ...state, ...payload, modalVisible: true }
},

hideModal(state) {
return { ...state, modalVisible: false }
},
},
})

import { pathMatchRegexp } from 'utils'
import api from 'api'

const { queryBackups } = api

export default {
namespace: 'backupDetail',

state: {
data: {},
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
const match = pathMatchRegexp('/backup/:id', pathname)
if (match) {
dispatch({ type: 'query', payload: { id: match[1] } })
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryBackups, payload)
const { success, message, status, ...other } = data
if (success) {
yield put({
type: 'querySuccess',
payload: {
data: other,
},
})
} else {
throw data
}
},
```

```
},

reducers: {
querySuccess(state, { payload }) {
const { data } = payload
return {
...state,
data,
}
},
},
}
```