

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства для оформления и
согласования служебных командировок**

Студент	_____	_____ <u>Дакина Анастасия Юрьевна</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	_____ <u>Балашова И.Ю.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	_____ <u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	_____ <u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

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

Секретарь ГЭК	_____	_____ <u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Дакина Анастасия Юрьевна _____ гр. 19ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы _____ Программные средства для оформления и согласования
служебных командировок _____

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю. _____

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

1) Назначение: сервис для оформления и согласования служебных командировок _____

2) Среда выполнения: операционная система Ubuntu 20.04, Web-браузер. _____

Языковые средства разработки: C#, JavaScript, UML. _____

Инструментальные средства: Jet Brains Rider, Jet Brains DataGrip. _____

Технологии разработки ООП, сервисно-ориентированный подход. _____

3) Основные функции приложения: _____

а) авторизация; _____

б) добавление сотрудников; _____

в) создание схем согласования; _____

г) создание заявок на согласование фактических расходов; _____

д) создание заявок на согласование на планируемые расходы; _____

е) получение уведомлений об изменении статуса согласования; _____

ж) получение писем на электронную почту об изменении статуса согласования; _____

з) просмотр отчета по согласованиям. _____

5. Перечень подлежащих разработке вопросов

- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование;
- 3) Реализация;
- 4) Контроль качества программного обеспечения;
- 5) План разработки бюджета;
- 6) Оформление пояснительной записки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 78 листов, 26 рисунков, 10 таблиц, 15 использованных источников и 5 приложений.

C#, MSSQL, MICROSOFT SQL, СЕРВЕР БАЗЫ ДАННЫХ, UML, REST, MVC, КОМАНДИРОВКИ, СОГЛАСОВАНИЕ.

Объект исследования – средства автоматизации управления командировками.

Цель выпускной квалификационной работы – разработать программные средства, позволяющие согласовывать и организовывать командировки.

Технологии разработки – разработка сервера приложений проводилась на языке программирования C# с использованием платформы ASP.NET, для создания сервера БД использовался MSSQL. В качестве ORM использовался Entity Framework. В качестве среды разработки использовались интегрированные среды разработки JetBrains DataGrip и Rider.

Среда исполнения – Linux, Web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы была разработана серверная часть приложения по согласованию командировок. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8BP192.07 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства для оформления и согласования служебных командировок. Пояснительная записка	Лит.	Лист	Листов
Разраб.	Дакина А.Ю.						4	78
Пров.	Балашова И.Ю.							
Н. Контр.	Такташкин Д.В					Группа 19ВП2		
Утв.								

Содержание

Введение.....	7
1 Анализ систем управления деловыми поездками.....	9
1.1 Анализ предметной области	9
1.2 Анализ программного обеспечения аналогичного назначения.....	9
1.2.1 Комплексное решение «Битрикс24»	10
1.2.2 Онлайн-система управления деловыми поездками «Starliner»	11
1.2.3 Агентство делового туризма «ОСДИ».....	13
1.2.4 Сервис по организации командировок «Аэротур Business».....	14
1.2.5 Сравнительный анализ продуктов-аналогов	15
1.3 Постановка задачи на разработку программных средств для оформления и согласования служебных командировок	17
1.3.1 Общее описание системы	17
1.3.2 Роли системы	18
1.3.3 Интерфейсы личных кабинетов.....	19
1.3.4 Функциональные требования.....	19
1.3.5 Нефункциональные требования	22
2 Проектирование средств по организации деловых поездок.....	23
2.1 Проектирование базы данных.....	23
2.2 Проектирование программных средств	25
3 Реализация программных средств для оформления и организации командировок.....	28
3.1 Развертывание и заполнение базы данных.....	28
3.2 Разработка серверной части приложения.....	31
3.2.1 Реализация учетных пользовательских бизнес-процессов	31

3.2.2 Реализация аналитических бизнес-процессов.....	34
3.2.3 Реализация автоматически выполняющихся событий.....	36
3.2.4 Поддержка модификации базы данных	37
4 Контроль качества программного обеспечения.....	39
4.1 Функциональное тестирование.....	39
4.2 Модульное тестирование.....	43
4.3 Метрики устойчивости и абстрактности компонент.....	44
4.4 Метрики кода.....	46
5 Планирование разработки и оценка бюджета.....	47
Заключение	50
Список использованных источников	51
Приложение А. Глоссарий	53
Приложение Б. Скрипты развертывания базы данных	55
Приложение В. Листинг приложения	57
Приложение Г. Листинг модульных тестов	74
Приложение Д. Справка о внедрении	77

Введение

Командировки – неотъемлемая часть бизнеса. Многие компании не могут обойтись без деловых поездок. Во-первых, от них зависит эффективность бизнес-процессов, налаживание связей и дальнейшее развитие. Во-вторых, они помогают повышать уровень квалификации сотрудников, получающих во время командировок новые знания и ценный опыт.

Если командировки часть рабочего процесса компании, то важно освоить и внедрить современные технологии, которые помогут упростить оформление всех документов и отчетностей.

При самостоятельном бронировании не всегда есть возможность оперативно согласовать услуги, например, руководитель может быть на совещании. Если затянуть согласование, билет наверняка вырастет в цене. Некоторые авиакомпании предоставляют услуги по резервированию билетов, но за это придется доплачивать.

Система управления командировками – эффективный инструмент оптимизации бизнес-процессов. Она легко интегрируется в любую учетную систему предприятия.

Целью выпускной квалификационной работы является разработка программных средств для оформления и согласования служебных командировок.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести сравнительный анализ программных продуктов-аналогов и выделить основные результаты;
- провести анализ требований к разрабатываемому программному продукту;
- осуществить проектирование программных средств и структур данных разрабатываемого программного продукта, описать их с использованием методологии UML;

- реализовать программные средства в соответствии с поставленными требованиями;
- протестировать полученные при реализации результаты;
- проанализировать качество написанного кода;
- провести планирование и расчет бюджета разрабатываемого программного продукта.

В работе использованы методы сравнительного анализа, методы проектирования и разработки программного обеспечения.

1 Анализ систем управления деловыми поездками

1.1 Анализ предметной области

Служебная командировка – это поездка работника по распоряжению работодателя на определенный срок для выполнения служебного поручения вне места постоянной работы [1].

Обычно процесс согласования командировки в компании выглядит следующим образом: сотрудник формирует заявку на командировку в бумажном виде и ожидает согласования от руководителя на бумаге. В документе прописывают сроки и цель поездки, ФИО и должность командированного.

Можно выделить следующие недостатки традиционного процесса согласования командировки в бумажном виде:

- это довольно длительный процесс, особенно если необходимо согласование нескольких должностных лиц;
- согласование командировки непрозрачно и отнимает время командируемого, его руководителя, бухгалтера;
- бухгалтерия вынуждена тратить время на проверку подтверждающих документов и авансового отчета;
- затянувшееся подписание документов по командировке приводит к проблемам с закрытием отчетного периода.

С помощью онлайн-системы управления командировками сотрудники могут создавать заявки на командировки в пару кликов, согласовывать их по настроенному маршруту, а руководители, сотрудники и бухгалтеры получают возможность оперативно отменять командировку или изменять ее параметры в процессе согласования (например, в случае превышения расходов или переносе сроков).

1.2 Анализ программного обеспечения аналогичного назначения

Сегмент деловых поездок длительный период отставал от индустрии путешествий. При отсутствии автоматизированных сервисов предприятие

сталкивается с проблемой длительного времени согласования бюджета, отведенного на служебные поездки. Сейчас появились удобные и многофункциональные системы автоматизации командировок.

С помощью автоматизированных сервисов компании получили возможность выбирать удобные инструменты для согласования через почту или личный кабинет. Это избавляет сотрудников компании от долгих переписок и ошибок. Другими словами, стало возможным утвердить все детали командировки или туристической поездки в несколько кликов. Преимущества онлайн-платформ также в том, что открыть их получится на компьютере, ноутбуке, планшете или смартфоне в любое нужное время.

Процесс согласования командировок запускается регулярно, поэтому он должен быть максимально удобным.

Рассмотрим некоторые автоматизированные системы по управлению командировками.

1.2.1 Комплексное решение «Битрикс24»

Сервис для управления бизнесом «Битрикс24» (рисунок 1) предлагает множество готовых решений и инструментов по цифровизации часто встречающихся в организации процессов. Одним из таких решений является согласование командировок, представляющее собой оптимизацию следующих бизнес-процессов: согласование командировки с руководителем, оформление командировки и сдача отчетности [2].

В отличие от других систем по управлению командировками, «Битрикс24» представляет собой коробочное решение, т. е. установка происходит на собственный сервер. Систему можно дополнять различными модулями, которые предлагает разработчик системы.

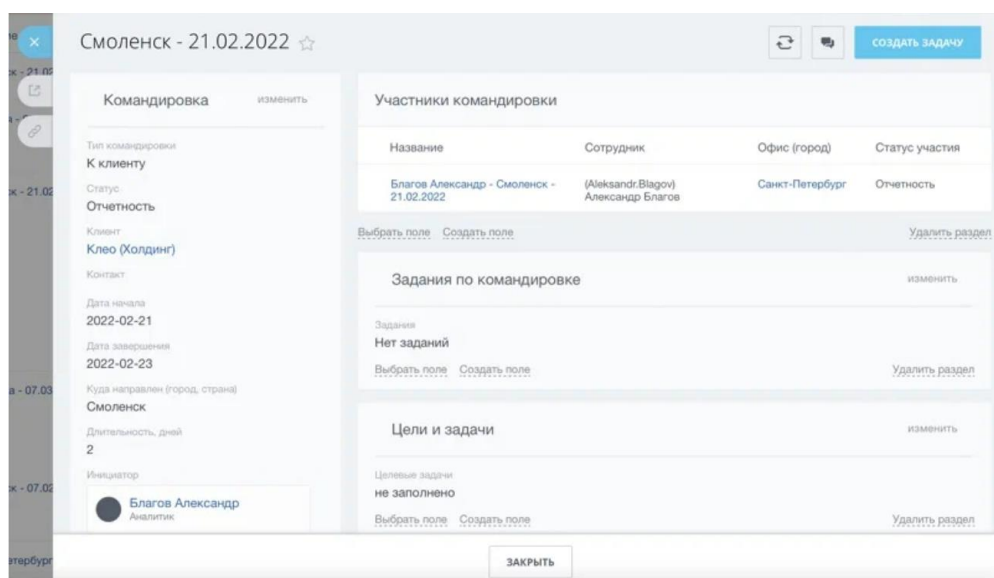


Рисунок 1 – Сервис для управления бизнесом «Битрикс24»

Преимущества платформы:

- хранение данных на собственном сервере (коробочное решение);
- возможность самостоятельной настройки бизнес-процессов;
- наличие системы напоминаний;
- возможность согласования пула командировок в назначенную дату директором.

Недостатки:

- в системе нельзя забронировать услуги для командировок;
- нет функционала согласования фактических расходов;
- в согласованиях участвует предопределенно руководитель отдела, т. е. нет возможности назначить согласующим другое лицо.

Также несомненным преимуществом использования платформы «Битрикс24» является возможность самостоятельного изменения интерфейса системы, а также ее логики под себя.

1.2.2 Онлайн-система управления деловыми поездками «Starliner»

«Starliner» — это онлайн-система управления деловыми поездками, осуществляющая полноценное тревел-сопровождение (рисунок 2). В продукте предложены следующие услуги:

- аэроэкспресс, трансферные компании, услуги такси;
- VIP-сопровождение в аэропортах, аренда бизнес-залов и переговорных комнат;
- визовая, миграционная и страховая поддержка;
- переводческие и экскурсионные услуги;
- автоматическое применение бонусных и мильных карт и программ корпоративной лояльности [3].

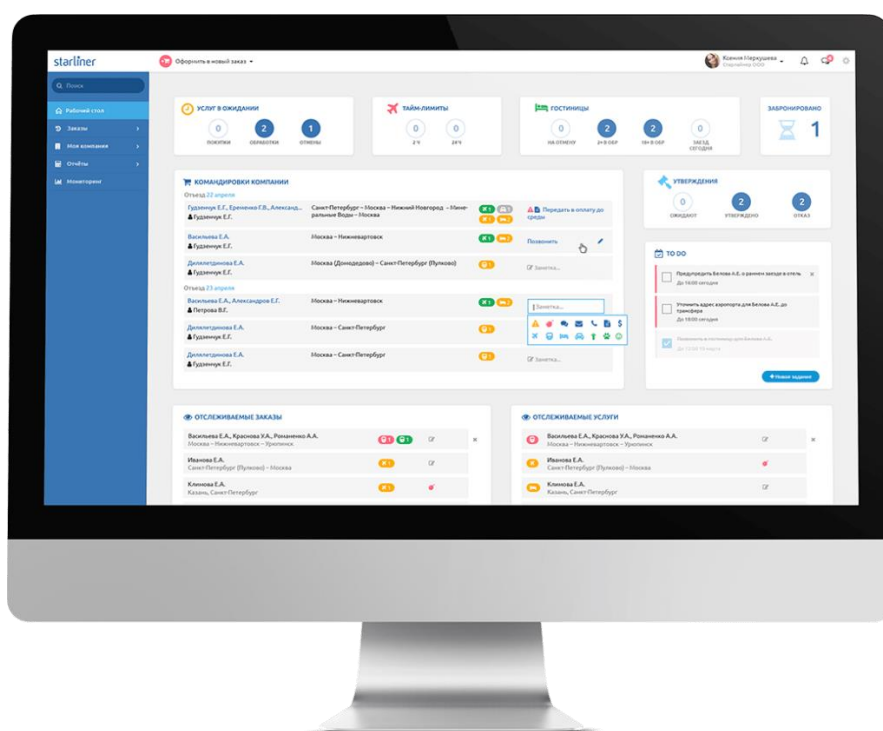


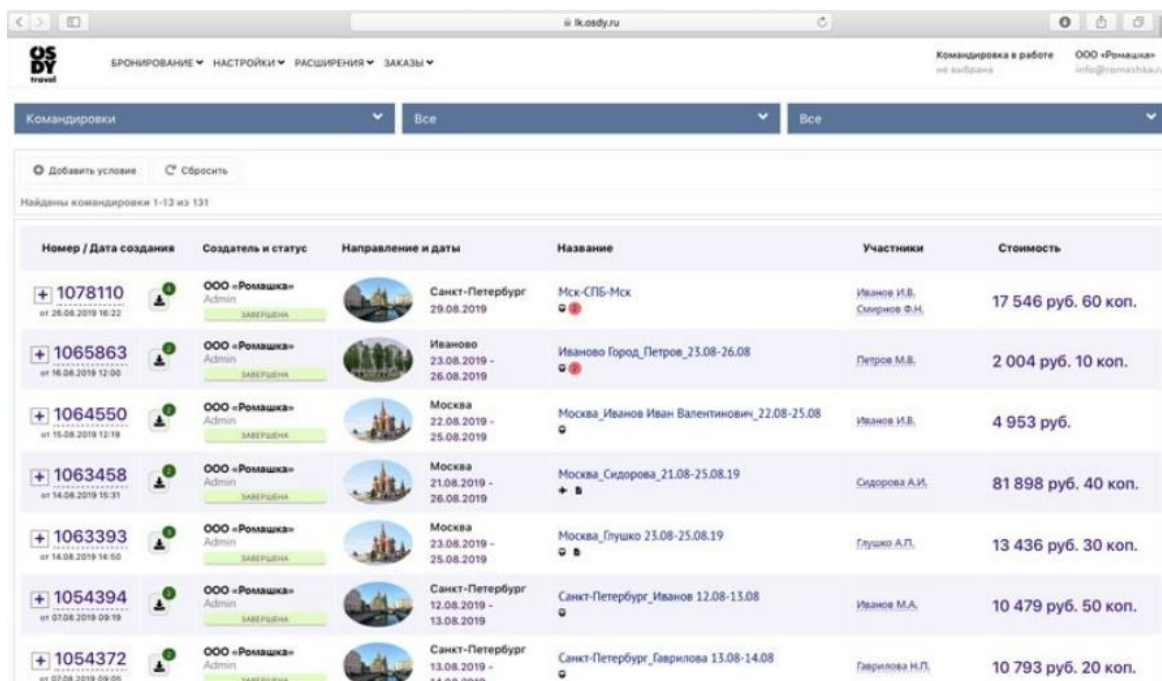
Рисунок 2 – Онлайн-система «Starliner»

Система поддерживает контроль соблюдения бюджета путем задания оптимальной стоимости услуг разных видов, автоформирование командировочных документов, визуализирует график поездок сотрудников, а также предоставляет статистические и аналитические отчеты.

Основной недостаток данной системы заключается в том, что в ней отсутствует функционал согласования командировок. Этот этап очень важен в оформлении деловых поездок и, таким образом, у пользователей данной системы процесс согласования остается в бумажном виде, что не оптимально для компаний.

1.2.3 Агентство делового туризма «ОСДИ»

Агентство делового туризма «ОСДИ» (рисунок 3) занимается организацией командировок для предприятий и предоставляет доступ к специализированной системе бронирования и службу поддержки для консультаций и заказа дополнительных услуг [4].



The screenshot shows a web application for OS DY travel. The header includes navigation links: БРОНИРОВАНИЕ, НАСТРОЙКИ, РАСШИРЕНИЯ, and ЗАКАЗЫ. The main content area displays a table of business trips with the following data:

Номер / Дата создания	Создатель и статус	Направление и даты	Название	Участники	Стоимость
1078110 от 26.08.2019 16:22	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Санкт-Петербург 29.08.2019	Мск-СПб-Мск	Иванов И.В. Смирнов Ф.Н.	17 546 руб. 60 коп.
1065863 от 16.08.2019 12:00	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Иваново 23.08.2019 - 26.08.2019	Иваново Город_Петров_23.08-26.08	Петров М.В.	2 004 руб. 10 коп.
1064550 от 15.08.2019 12:19	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Москва 22.08.2019 - 25.08.2019	Москва_Иванов Иван Валентинович_22.08-25.08	Иванов И.В.	4 953 руб.
1063458 от 14.08.2019 15:31	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Москва 21.08.2019 - 26.08.2019	Москва_Сидорова_21.08-25.08.19	Сидорова А.И.	81 898 руб. 40 коп.
1063393 от 14.08.2019 14:50	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Москва 23.08.2019 - 25.08.2019	Москва_Глушко_23.08-25.08.19	Глушко А.П.	13 436 руб. 30 коп.
1054394 от 07.08.2019 09:19	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Санкт-Петербург 12.08.2019 - 13.08.2019	Санкт-Петербург_Иванов_12.08-13.08	Иванов М.А.	10 479 руб. 50 коп.
1054372 от 07.08.2019 09:00	ООО «Ромашка» Admin ЗАБРОНИРОВАНО	Санкт-Петербург 13.08.2019 - 14.08.2019	Санкт-Петербург_Гаврилова_13.08-14.08	Гаврилова Н.П.	10 793 руб. 20 коп.

Рисунок 3 – Система «ОСДИ»

Особенности системы:

- оформление услуг в одном окне с автозаполнением паспортных данных сотрудников;
- наличие тревел-политик и согласование поездок;
- поездки могут заказывать сами сотрудники, тревел-менеджер, отдел командировок;
- сотрудники могут приобрести только те услуги, которые разрешены правилами компании;
- разные права и ограничения для разных категорий сотрудников;

– все виды услуг: авиа, ЖД, отели, трансфер, групповой перелет, групповое проживание, визы и страховки, VIP-услуги, MICE, культурные мероприятия и другое.

Преимущества системы, связанные с функционалом согласования:

- разные типы авторизации – простая и сложная;
- получение писем на электронную почту по согласованиям, а также уведомлений в личном кабинете;
- система может быть настроена так, чтобы данные отправлялись нескольким уполномоченным лицам;
- можно оформлять услуги как без авторизации, так и с авторизацией при нарушении тревел-политики или всех заявок без исключения.

Недостаток данной системы в том, что в ней отсутствует предкомандировочное согласование, т. е. согласование дат, бюджета поездки и цели командировки.

1.2.4 Сервис по организации командировок «Аэротур Business»

Онлайн сервис организации командировок в России «Аэротур Business» – это возможность бронировать билеты по всей стране и по международным направлениям, отели и гостиницы, трансфер и Аэроэкспресс (рисунок 4). Сервис позволяет оформлять онлайн деловые и туристические визы, а также страховые полисы для международных перелетов [5].



Рисунок 4 – Сервис организации командировок «Аэротур Business»

Преимущества системы:

- возможность настройки модулей системы, тем самым использовать только необходимые;
- возможность дать доступ только ответственным лицам либо всем сотрудникам;
- наличие двух способов согласования командировок: через личный кабинет либо по электронной почте;
- настройка лимитов в личном кабинете.

Онлайн-платформа доступна на компьютере, планшете или смартфоне.

1.2.5 Сравнительный анализ продуктов-аналогов

Для проведения сравнительного анализа программных продуктов на основе их исследования были выбраны критерии и составлены их коэффициенты значимости. Критерии и коэффициенты значимости представлены в таблице 1.

Таблица 1 – Критерии оценки систем для оформления и согласования служебных командировок

№	Наименование	Коэффициент значимости
	Функциональные возможности	
1	Согласование фактических расходов	1
2	Согласование направления, цели и бюджета поездки (предкомандировочное согласование)	1
3	Настройка собственных схем согласования	1
4	Настройка авторизации: покупка только после согласования и без него	0,75
5	Выбор согласующих для конкретных сотрудников	0,75
	Нефункциональные возможности	
6	Бесплатность продукта	0,5
	Технические возможности	
7	Интеграция с ERP-системами	0,5
8	Возможность внесения изменений	0,25

Было решено использовать трехбальную шкалу со следующими оценками:

- 0 баллов: функция не реализована;
- 1 балл: функция реализована частично;
- 2 балла: функция полностью реализована.

Сведем результаты анализа в таблицу 2.

Таблица 2 – Сравнительный анализ продуктов-аналогов

Критериальная оценка	Битрикс24	Starliner	ОСДИ	Аэротур Business
Согласование фактических расходов	2	0	2	2
Согласование направления, цели и бюджета поездки (предкомандировочное согласование)	2	0	0	1
Настройка собственных схем согласования	1	0	2	2
Настройка авторизации: покупка только после согласования и без него	0	1	2	1
Выбор согласующих для конкретных сотрудников	0	0	1	2
Бесплатность продукта	0	0	0	0
Интеграция с ERP-системами	1	2	1	2
Возможность внесения изменений	1	0	2	2

Оценим реализованный функционал описанных выше систем. Для этого произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{пп} = \sum_{i=1}^n (Z_i * K_i),$$

где $O_{пп}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия;

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

В результате подсчета интегральной оценки выяснилось, что сервис организации командировок «Starliner» сильно ограничен по функционалу относительно других сервисов (рисунок 5). Наиболее подходящим программным продуктом оказался «Аэротур Business».

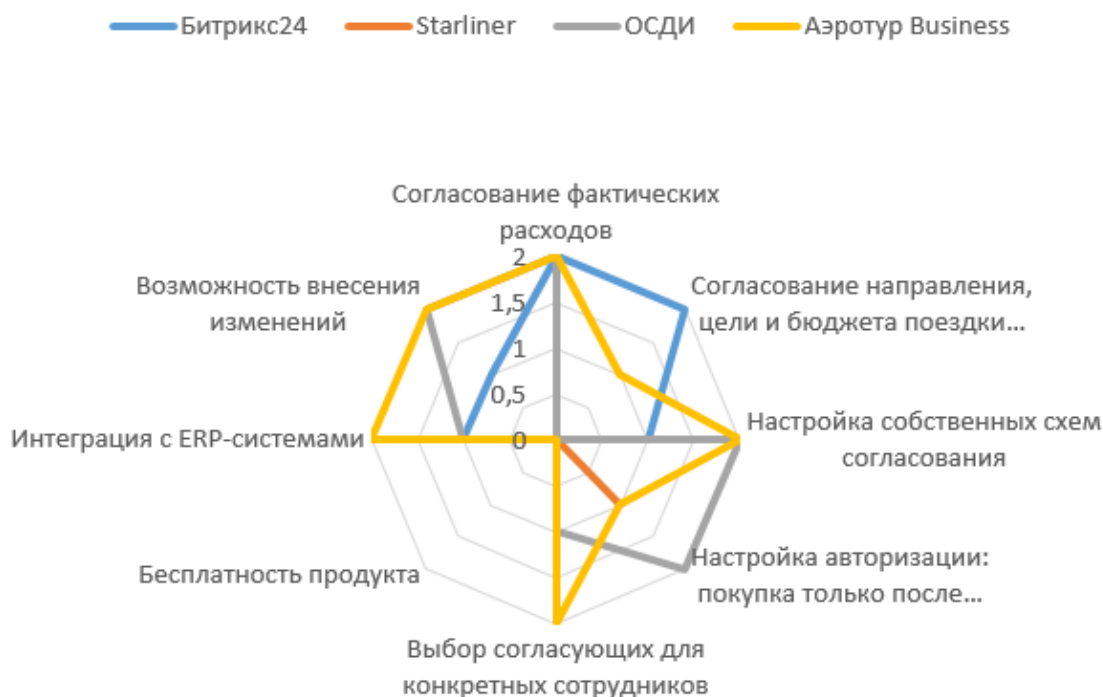


Рисунок 5 – Результаты подсчета интегральной оценки

Подводя итог можно сделать вывод, что все рассмотренные сервисы не способны в полной мере выполнить поставленную задачу.

1.3 Постановка задачи на разработку программных средств для оформления и согласования служебных командировок

1.3.1 Общее описание системы

Разрабатываемое приложение автоматизирует согласование командировки и формирование отчетов. Основной потребитель продукта – юридические компании, имеющие от 10 командировок в месяц.

Система должна обеспечить следующий функционал:

- согласование поездки у всех лиц, принимающих решение;

- фактическое согласование бюджета поездки;
- предварительное согласование направления и цели командировки, а также ее участников;
- уведомление пользователей об изменении статуса согласования как в личном кабинете, так и по почте;
- возможность создавать схемы согласования с выбором количества необходимых шагов и указанием согласующих на каждый шаг;
- выбор типа авторизации: только при нарушении тревел-политики либо всегда;
- предоставление аналитики по командировкам в графическом, текстовом и табличном виде.

Приложение работает в online-режиме и доступно пользователям в сети Интернет.

1.3.2 Роли системы

Администратор:

- создание и редактирование сотрудников и их личных кабинетов;
- создание схем согласования;
- управление бюджетом;
- получение документов для бухгалтерии;
- согласование поездок.

Сотрудник:

- создание заявки на согласование;
- получение решения о согласовании от ответственных лиц в виде уведомлений.

Руководитель отдела:

- создание схем согласования;
- управление бюджетом;
- согласование поездок.

Таким образом были выявлены роли системы и охарактеризованы их функции.

1.3.3 Интерфейсы личных кабинетов

У каждой из ролей есть страницы Главная, Согласование, Профиль (личные данные). Каждая из ролей системы имеет страницы, присущие только данной роли.

Администратор:

- страница управления сотрудниками;
- страница управления согласованиями;
- страница Бухгалтерия;
- аналитика.

Руководитель отдела:

- отделы;
- страница управления сотрудниками;
- страница управления согласованиями.

Сотрудник не имеет страниц, присущих только его роли.

1.3.4 Функциональные требования

Основными функциями приложения являются:

- создание схем согласования;
- создание заявки на согласование;
- согласование заявок на командировки;
- отправка уведомлений на электронную почту сотрудников;
- формирование аналитики по прошедшим и будущим поездкам;
- получение уведомлений об изменении статуса согласования.

Диаграмма вариантов использования приведена на рисунке 6.

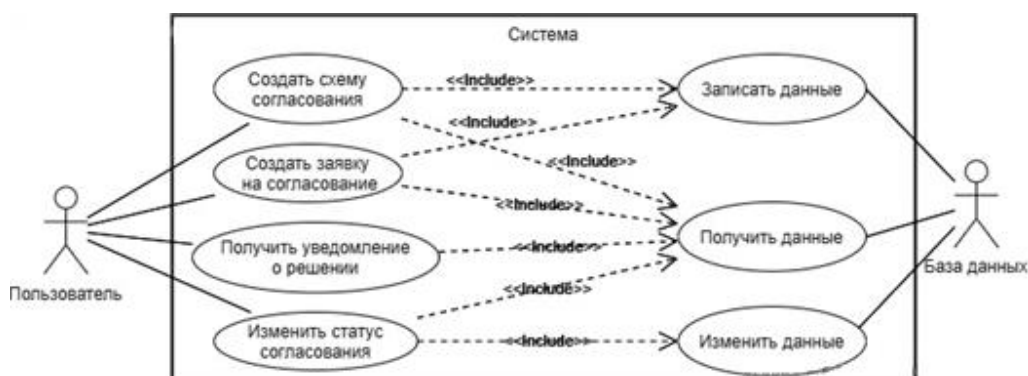


Рисунок 6 – Диаграмма вариантов использования приложения

Ниже приведено описание варианта использования «Создать схему согласования» (рисунок 7) и включенных в него вариантов использования.

Вариант использования: Создать заявку на согласование
ID: 1
Краткое описание: Пользователь согласует направление и бюджет поездки с ответственным лицом
Главные актеры: Пользователь
Второстепенные актеры: База данных
Предусловия: Пользователь авторизован и сформирована корзина с услугами
Основной поток: <ol style="list-style-type: none"> 1. Include (Получить данные) 2. Пользователь открывает корзину с услугами 3. Система загружает корзину и отображает ее пользователю 4. Пользователь инициирует отправку на согласование 5. Система отображает пользователю список согласующих лиц 6. Пользователь выбирает согласующее лицо 7. Пользователь отправляет на согласование 8. Include (Записать данные) 9. Система отправляет уведомление согласующему 10. Если согласующий одобряет поездку <ol style="list-style-type: none"> 10.1 Система открывает пользователю возможность покупки 11. Иначе <ol style="list-style-type: none"> 11.1 Система сообщает пользователю, что ему отказано в согласовании
Постусловие: Пользователю открывается возможность покупки услуг
Альтернативные потоки: Нет.

Рисунок 7 – Сценарий варианта использования «Создать заявку на согласование»

На рисунке 8 приведено описание варианта использования «Записать данные».

Вариант использования: Записать данные
ID: 2
Краткое описание: База данных сохраняет данные в таблицу
Главные актеры: База данных
Второстепенные актеры: Пользователь
Предусловия: Нет.
Основной поток: 1. Пользователь отправляет заявку на согласование 2. Система обрабатывает данные и отправляет запрос на сохранение в БД 3. БД записывает данные
Постусловие: Данные сохранены в таблице
Альтернативные потоки: Нет.

Рисунок 8 – Сценарий варианта использования «Записать данные»

На рисунке 9 приведено описание прецедента «Получить данные».

Вариант использования: Получить данные
ID: 3
Краткое описание: База данных получает данные
Главные актеры: База данных
Второстепенные актеры: Пользователь
Предусловия: Нет.
Основной поток: 1. Пользователь открывает корзину 2. Система отправляет запрос на получение в БД корзины по Id 3. БД получает данные
Постусловие: Система получает данные из БД и отдает их пользователю
Альтернативные потоки: Нет.

Рисунок 9 – Сценарий варианта использования «Получить данные»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы при выполнении определенных функций.

1.3.5 Нефункциональные требования

Локализация: сайт должен быть реализован на русском языке. Предусмотрена возможность перевода интерфейса программы на английский язык. Для английского языка не предусмотрен перевод бизнес-данных.

Дизайн: в цветовой схеме используются только 4 цвета: белый, черный, красный и серый. Стилизация сайта в едином формате.

Поддержка браузеров: разрабатываемое приложение должно корректно открываться и функционировать в текущих версиях (на момент разработки) основных популярных браузеров – Chrome, Firefox, Safari, Яндекс.Браузер, Safari.

2 Проектирование средств по организации деловых поездок

2.1 Проектирование базы данных

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними без учета принятой модели БД и синтаксиса целевой СУБД [6].

Графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для предметной области информационной системы представлено на рисунке 10.

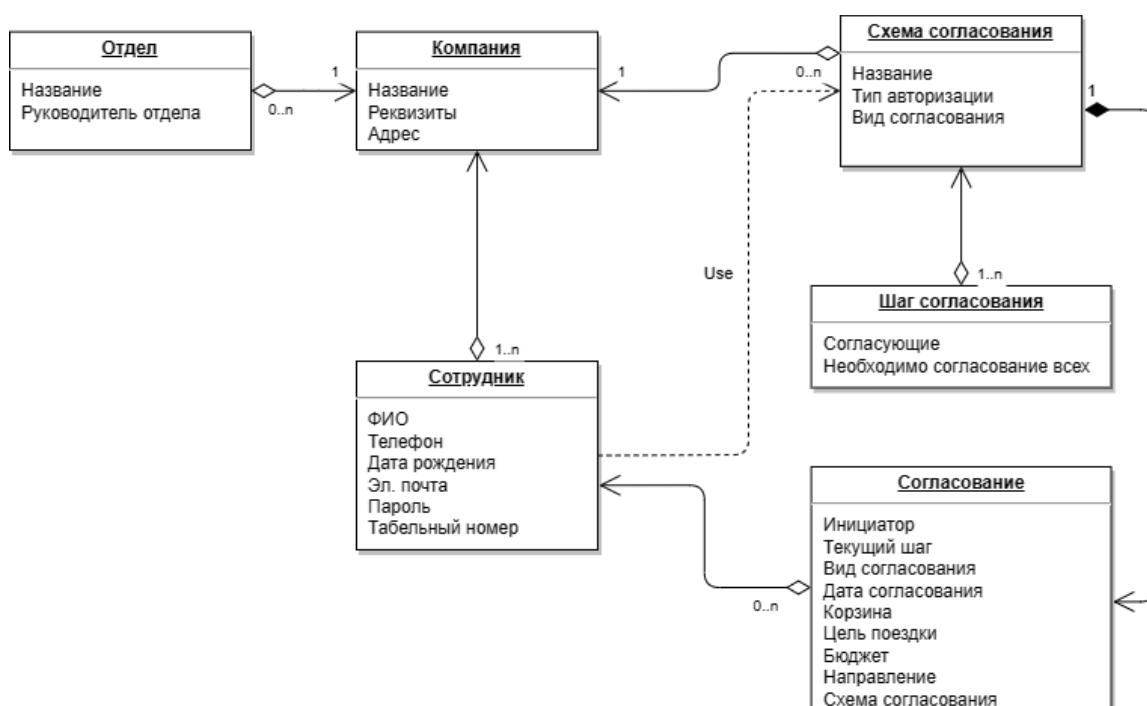


Рисунок 10 – Концептуальная модель базы данных

В таблице 3 представлены сущности, сформированные на основании входных и выходных данных бизнес-процессов.

Таблица 3 – Сущности предметной области

№ п/п	Наименование	Описание	Категория	Количество экземпляров
1	Сотрудник	Пользователь личного кабинета, совершающий покупки либо являющийся согласующим	Стержневая	>900000
2	Согласование	Заявка на одну поездку с указанием сотрудника, согласующего	Ассоциативная	>300000

Продолжение таблицы 3

№ п/п	Наименование	Описание	Категория	Количество экземпляров
3	Компания	Подтверждение покупки с детализацией	Стержневая	>30000
4	Отдел	Ограничения на сотрудников	Обозначающ ая	>10000
5	Схема согласования	Схема согласования, характеризующая процесс, по которому будет проходить согласование	Характеристическая	>5000
6	Шаг согласования	Шаг согласования, который входит в схему. Выполняется по определенной последовательности, имеет указанных согласующих	Характеристическая	>10000

Для сущностей предметной области сформированы связи (таблица 4).

Таблица 4 – Связи между сущностями предметной области

№ п/п	Тип сущности 1	Тип сущности 2	Описание	Мощ ность	Обязательнос ть	Ограниче ния
1	Сотрудник	Согласование	Сотрудник инициирует согласование / Согласование создается Сотрудником	М : М	Обязательная (каждое согласование создается сотрудником)	Нет
2	Компания	Отдел	Компания содержит отделы / Отдел входит в компанию	1 : М	Необязательная (компания может не иметь отделов)	Нет
3	Компания	Схема согласования	Компания создает схемы согласования / Схема согласования относится к компании	1 : М	Необязательная (компания может не иметь схем согласования)	Нет
4	Схема согласования	Шаг согласования	Схема имеет шаги согласования / Шаги согласования входят в схему	1 : М	Обязательное (каждая схема имеет шаги согласования)	Нет

Таким образом была разработана концептуальная модель базы данных.

2.2 Проектирование программных средств

Для разработки приложения было принято решение использовать технологию ASP.NET.

ASP.NET – технология создания веб-приложений и веб-сервисов от компании Майкрософт. Она является составной частью платформы Microsoft .NET и развитием более старой технологии Microsoft ASP [7].

Причины выбора данной технологии следующие:

- ASP.NET легко интегрируется с распространенными клиентскими платформами и библиотеками, в том числе React;
- кроссплатформенность: в Windows, macOS и Linux;
- выполнение развертывания в облаке или локальной среде.

При проектировании было решено использовать паттерн MVC (Model-View-Controller). ASP.NET Core MVC предоставляет функции, которые позволяют создавать веб-интерфейсы API и веб-приложения. Среди них:

- шаблон Model-View-Controller помогает сделать веб-API и веб-приложения тестируемыми;
- веб-API доступны для множества клиентов, включая браузеры и мобильные устройства;
- поддержка нескольких форматов данных;
- согласование содержимого веб-API;
- привязка модели автоматически сопоставляет данные из HTTP-запросов с параметрами методов действия;
- проверка модели автоматически выполняется на стороне сервера и клиента [8].

Фрагмент диаграммы классов программного обеспечения, относящийся к функционалу согласования командировок, представлен на рисунке 11.

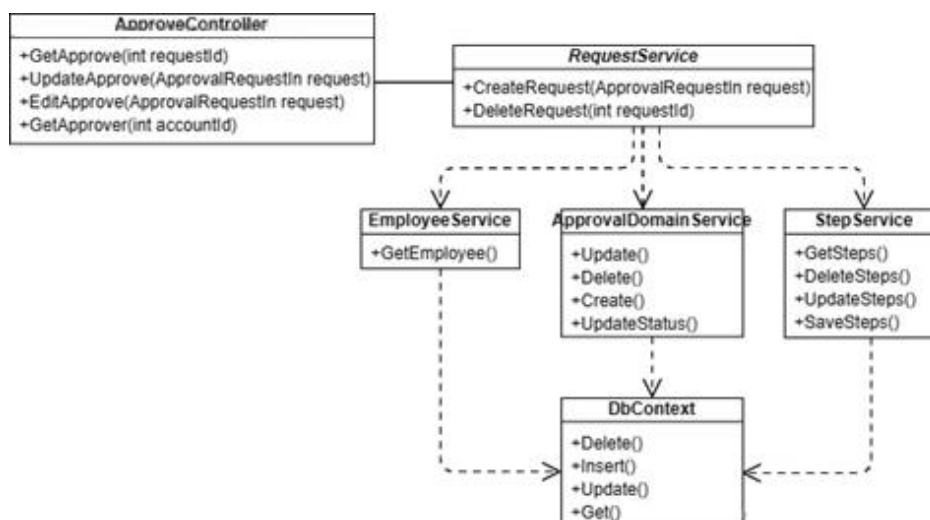


Рисунок 11 – Диаграмма классов микросервиса согласований

Более подробное описание классов приведено в спецификациях ниже. Спецификация класса ApproveController изображено на рисунке 12.

Класс «ApproveController» - класс контроллера, на который приходят запросы от клиента	
GetApprove(int requestId)	Получение данных по согласованию по айди заявки
UpdateApprove(ApprovalRequestIn request)	Обновление статуса согласования
EditApprove(ApprovalRequestIn request)	Изменение данных по согласованию
GetApprover(int accountId)	Получение согласующих в личном кабинете

Рисунок 12 – Спецификация класса ApproveController

Спецификация класса ApprovalDomainScheme приведена на рисунке 13.

Класс «ApprovalDomainScheme» - класс доменного уровня для согласований	
Update(ApproveRequest request)	Обновление данных по согласованию
Create(ApprovalRequest request)	Создание заявки на согласование
Delete(int request)	Удаление заявки на согласование
UpdateStatus(int status, int requestId)	Обновление статуса согласования

Рисунок 13 – Спецификация класса ApprovalDomainScheme

Спецификация класса StepService приведена на рисунке 14.

Класс «StepService» - класс логики шагов в схеме согласования	
UpdateSteps(StepsRequest request)	Обновление шагов в схеме
GetSteps(int schemeId)	Получение шагов схемы
DeleteSteps(int schemeId)	Удаление шагов в схеме
SaveSteps(StepsRequest request)	Создание шагов для схемы согласования

Рисунок 14 – Спецификация класса StepService

Таким образом, для основных модулей системы были спроектированы классы, а также спроектирована концептуальная схема базы данных приложения.

3 Реализация программных средств для оформления и организации командировок

3.1 Развертывание и заполнение базы данных

На основе спроектированной концептуальной модели для программы была разработана база данных. Физическая модель базы данных приведена на рисунке 15.

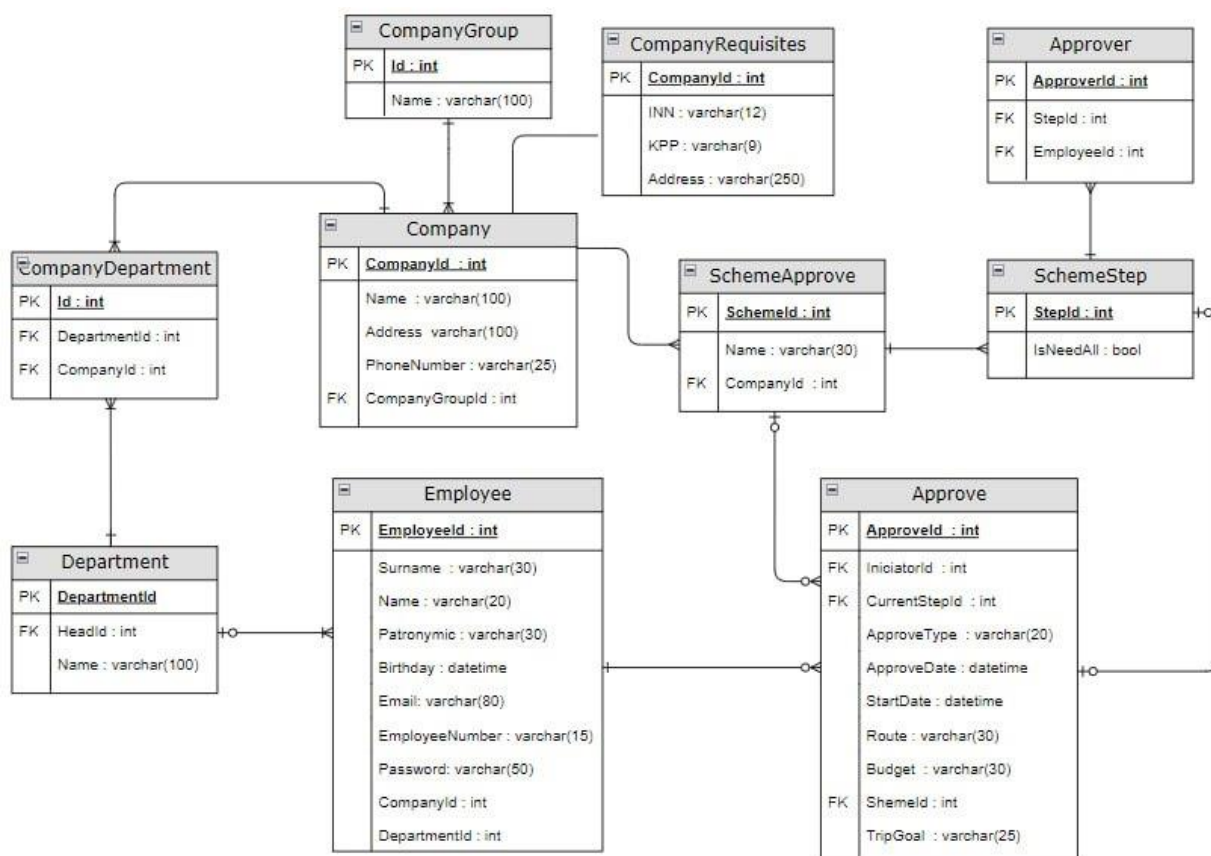


Рисунок 15 – Физическая модель базы данных

Ниже приведено описание некоторых сущностей. Спецификация сущности сотрудника представлена в таблице 5.

Таблица 5 – Спецификация сущности «Employee»

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Обязательное	Роль	Пример
1	EmployeeId	Порядковый номер в базе данных	целый	4	0001-9999	Да	PK	1736

Продолжение таблицы 5

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Обязательное	Роль	Пример
2	Surname	Фамилия	Символьный	100	A-я, A-z, -	Да		Иванов
3	Name	Имя	Символьный	100	A-я, A-z, -	Да		Иван
4	Patronymic	Отчество	Символьный	100	A-я, A-z, -	Нет		Иванович
5	Birthday	Дата рождения	Дата	10	01.01.1950-01.01.2004	Да		01.01.1999
6	Email	Эл. Почтовый адрес	Символьный	80	'A-z', '.', '@', '-'	Да		'mail@mail.ru'
7	Phonenumber	Номер мобильного телефона	Символьный	25	'0-9', '+'	Да		+79542043848
8	Password	Пароль	Символьный	50	A-я, A-z, '0-9', '+', '-'	Да		Password!
9	CompanyId	Идентификатор компании	Целый	4	0001-9999	Да	FK	1
10	DepartmentId	Идентификатор отдела	Целый	4	0001-9999	Да	FK	1

Спецификация сущности согласования представлена в таблице 6.

Таблица 6 – Спецификация сущности «Approve»

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Обязательное	Роль	Пример
1	ApproveId	Порядковый номер в БД	целый	4	0001-9999	Да	PK	1736
2	StartDate	Дата инициации согласования	Дата	10	01.01.2022-...	Да		10.01.2022
3	ApproveType	Согласование происходит по схеме	Булево значение	1	0,1	Да		1
4	InitiatorId	Айди сотрудника, отправившего заявку	целый	4	0001-9999	Да	FK	10
5	Budget	Бюджет планируемой поездки	Символьный	30	'0-9', '+'	Нет		1000

Продолжение таблицы 6

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Обязательное	Роль	Пример
6	Resolution	Решение по согласованию	Целый	1	0,1,2,3	Да		0
7	Route	Направление поездки	Символьный	30	A-я, A-z, '0-9'	Нет		Пенза
8	SchemeId	Идентификатор схемы согласования, по которой происходит согласование	Целый	4	1-9999	Нет	FK	1
9	TripGoal	Цель поездки	Символьный	25	A-я, A-z, '0-9'	Нет		Подписание договора
10	ApproveDate	Дата согласования	Дата	10	01.01.2022-...	Да		10.01.2022

Спецификация сущности схемы согласования представлена в таблице 7.

Таблица 7 – Спецификация сущности «SchemeApprove»

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Обязательное	Роль	Пример
1	ID	Порядковый номер в БД	целый	4	0001-9999	Да	PK	1736
2	CompanyId	Айди компании	Целый	4	1 – 9999	Да	FK	1
3	Name	Название схемы	Символьный	Max	Любые символы	Да		«Схема»

В процессе создания физической модели было принято решение использовать для создания модели БД миграции. При реализации базы данных с помощью Microsoft SQL Server были созданы таблицы и связи между ними согласно физической модели данных.

Далее была внесена информация в таблицы с помощью спроектированных запросов. Запросы для создания базы данных, таблиц и ограничений были зафиксированы в миграциях микросервисов.

Скрипты развертывания базы данных приведены в приложении Б.

3.2 Разработка серверной части приложения

На основе проектирования программных средств произведена их реализация. На рисунке 16 приведена диаграмма компонентов разработанного приложения.

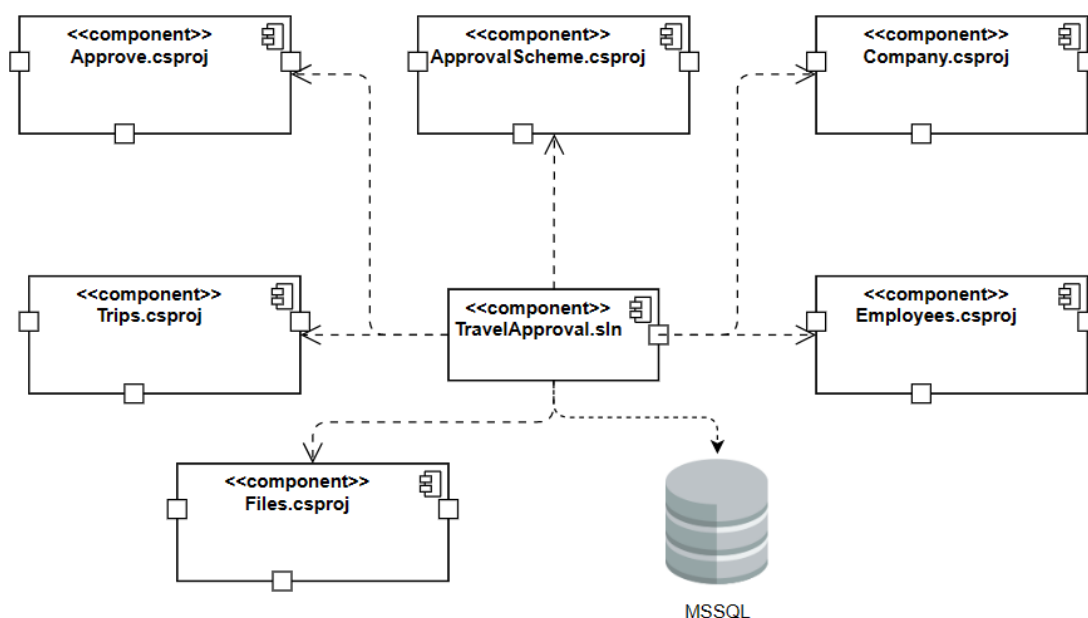


Рисунок 16 – Диаграмма компонентов приложения

Программная реализация классов приложения по согласованию командировок представлена в приложении В.

3.2.1 Реализация учетных пользовательских бизнес-процессов

В приложениях, работающих с базами данными, возникает две задачи при работе с данными: как получить данные из базы данных и как сохранить изменения. В разрабатываемом приложении было принято решение использовать ORM Entity Framework (EF) для создания запросов к базе данных и поддержки модификации базы данных.

Для извлечения данных из базы данных Entity Framework Core использует технологию LINQ to Entities. В основе данной технологии лежит язык интегрированных запросов LINQ (Language Integrated Query). LINQ предлагает

простой и интуитивно понятный подход для получения данных с помощью выражений, которые по форме близки выражениям языка SQL.

Несмотря на то, что при работе с базой данных происходит выполнение запросов LINQ, реляционные базы данных принимают только запросы на языке SQL. Поэтому Entity Framework Core, используя выражения LINQ to Entities, транслирует их в определенные запросы, понятные для используемого источника данных [9].

Рассмотрим бизнес-процесс получения поездок на согласовании. Метод `GetCartsToApprove(int employeeId, DateTime start, DateTime end)` на уровне домена обращается с помощью EF к базе данных и получает список моделей согласования поездок. На вход методу передаются `Id` сотрудника, для которого происходит получение списка согласований, начала периода и конец периода, который являются необязательными.

Запрос к базе данных представлен в листинге 1.

Листинг 1 – Получение поездок на согласовании

```
return _dataContext
    .GetQueryable<DbApprove>()
    .Where(a => a.EmployeeId.Equals(employeeId)
        && a.CartId.HasValue && a.Cart.Items.Any())
    .Select(approve => approve.CartId.Value)
    .ToList();
```

При наличии указанного периода запрос будет выглядеть следующим образом (листинг 2).

Листинг 2 – Получение поездок на согласовании в указанном периоде

```
return _dataContext
    .GetQueryable<DbApprove>()
    .Where(a => a.EmployeeId.Equals(employeeId)
        && a.CartId.HasValue && a.EventDate >= start
        && a.EventDate <= end)
    .Select(approve => approve.CartId.Value)
    .ToList();
```

Ниже приведено описание методов, использованных в запросе.

`Where<TSource> (this IQueryable<TSource> source, Expression<Func<TSource, bool>> predicate)` – метод фильтрации данных. Как видно из запроса, фильтрация происходит по сотруднику, условию существования корзины и, в случае наличия периода, по вхождению даты создания согласования в диапазон.

`Select <TSource, TResult> (this IQueryable<TSource> source, Expression<Func<TSource, TResult>> selector)` – проекция используется для создания выходной последовательности одного типа элементов из входной последовательности элементов другого типа, в нашем случае из модели `Approve` мы хотим получить `int` – `Id` корзины.

`ToList()` – метод преобразует `IQueryable` в `List`.

`GetQueryable<T>(bool trackChanges = true, bool disabled = false)` – предоставляет удаленный доступ к базе данных и позволяет перемещаться по данным как в прямом порядке от начала до конца, так и в обратном порядке. В процессе создания запроса, возвращаемым объектом которого является `IQueryable`, происходит оптимизация запроса. В итоге в процессе его выполнения тратится меньше памяти, меньше пропускной способности сети.

Помимо получения данных, были реализованы запросы на изменение данных. В качестве примера рассмотрим бизнес-процесс изменения статуса согласования (листинг 3).

Листинг 3 – Изменение статуса согласования

```
var dbApprove = _dataContext.GetQueryable<DbApprove>()
    .FirstOrDefault(a => a.UserId.Equals(approve.UserId)
        && a.Id.Equals(approve.Id));
dbApprove.Resolution = (int)approve.Resolution;
dbApprove.Type = (int)approve.Type;
dbApprove.Messages = approve.Messages.ToJson();
dbApprove.EventDate = approve.EventDate;
_dataContext.Update(dbApprove);
_dataContext.SaveChanges();
```

В отличие от метода получения согласования, в данном фрагменте появляются методы контекста `Update<T>()` и `SaveChanges()`. Это связано с тем,

что для обновления данных нужно сначала извлечь нужный объект из базы данных, затем изменить одно или несколько значений его свойств, а затем сохранить изменения в базе данных, используя метод `DbContext.SaveChanges()`. Entity Framework автоматически ищет измененные свойства в прикрепленных сущностных объектах и обновляет только их.

3.2.2 Реализация аналитических бизнес-процессов

Под аналитикой в разрабатываемом приложении понимается статистика согласований в виде количества, реализованного на каждый отдел, либо в виде процентного соотношения. Визуально аналитика представлена в виде круговых диаграмм и гистограмм, а также в текстовом (табличном) формате.

За аналитику отвечает отдельный микросервис «analytics-service», в зону ответственности которого входит сбор аналитики по базе данных MSSQL и запись ее в базу данных PostgreSQL.

Рассмотрим процесс получения аналитики на примере запроса на просмотр аналитики по отделам.

Для реализации аналитических бизнес-процессов был использован архитектурный паттерн Builder, с помощью которого динамически создается SQL-запрос.

Шаблон запроса приведен на листинге 4.

Листинг 4 – Шаблон запроса на аналитику по отделам

```
select
    department_id          as "Id",
    department_name        as "Name",
    round(sum(amount), 2)  as "Amount",
    count(case when version_status = 1 then 1 end) as "Changes",
from analytics
where /**where**/
group by
    department_id,
    department_name
order by /**order by**/
```

Сначала «строитель» собирает в список данные для выборки, условия фильтрации, условия сортировки и группировки, а затем подставляет их в шаблон на соответствующие позиции. Это реализуется с помощью метода `Replace(string oldValue, string? newValue)` класса `String`.

Код метода `Build()` приведен в листинге 5.

Листинг 5 – Метод `Build()`

```
public string Build()
{
    var result = sql;

    if (selectList.Any())
    {
        var selectSql = string.Join(" ", selectList);
        result = result.Replace("/**select**/", selectSql);
    }

    if (whereList.Any())
    {
        var whereSql = string.Join(" ", whereList);
        result = result.Replace("/**where**/", whereSql);
    }

    if (orderList.Any())
    {
        var orderSql = string.Join(" ", orderList);
        result = result.Replace("/**orderby**/", orderSql);
    }

    if (groupList.Any())
    {
        var groupSql = string.Join(" ", groupList);
        result = result.Replace("/**groupby**/", groupSql);
    }

    return result;
}
```

Далее запрос отправляется в базу данных и выполняется там как обычный `sql`-запрос.

При проектировании базы данных аналитики было принято решение хранить данные в одной единственной таблице, несмотря на большое количество столбцов в структуре. Преимущество такого подхода – улучшение

быстродействия за счет отсутствия дорогостоящих операций JOIN, а также ввиду наличия динамических данных – простота масштабирования.

3.2.3 Реализация автоматически выполняющихся событий

При отправке заявки на согласование либо при утверждении решения по согласованию заявки у пользователей в личном кабинете изменяется счетчик активных согласований. Задача заключается в том, что счетчик должен быть всегда актуальным и изменяться в реальном времени. Для этого были реализованы события, обрабатываемые микросервисом на основе библиотеки SignalR.

Каждый раз, когда происходит обновление согласования, создание новой заявки либо удаление заявки, вызывается событие, инициирующее обновление счетчика в пользовательском приложении. Процесс обновления заголовков личного кабинета изображен с помощью диаграммы последовательности на рисунке 17.

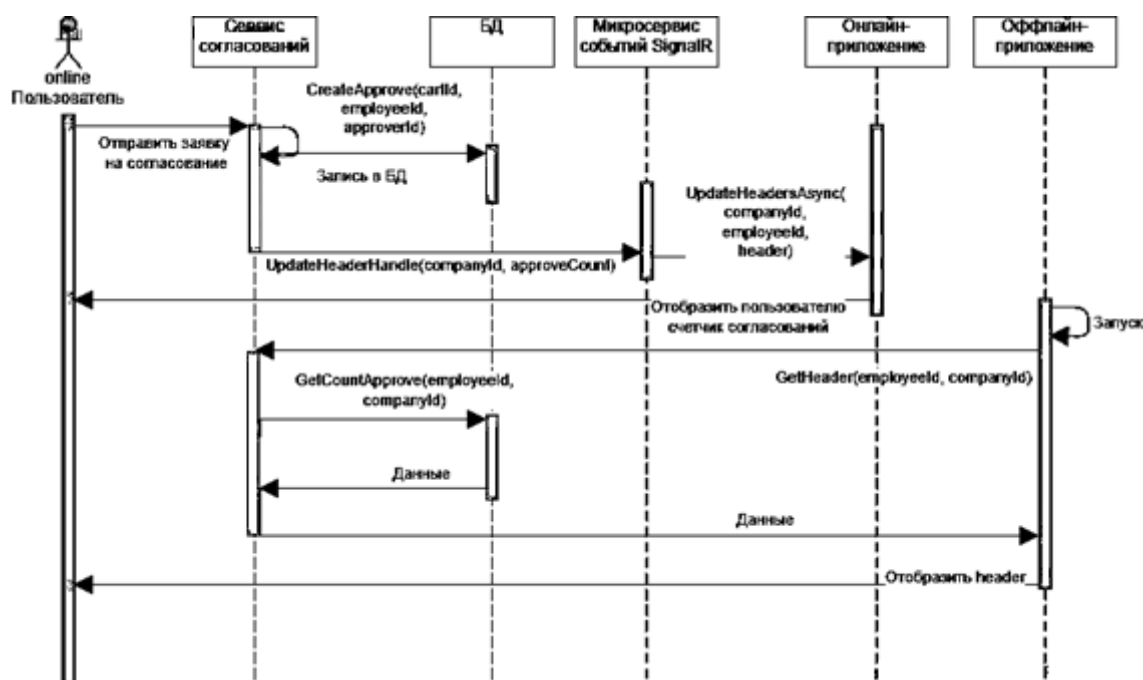


Рисунок 17 – Диаграмма последовательности выполнения события

Ниже приведено описание методов, использованных на диаграмме последовательности.

CreateApprove(cartId, employeeId, approverId) – метод создания и обработки согласования. Данный метод содержит в себе бизнес-логику согласования: проверяет права пользователей, валидность корзины и т. д.

UpdateHeaderEvent(companyId, approveCount) – событие, вызываемое из выше описанного метода. По данному событию осуществляется POST запрос в SignalR, в котором передаются компания, сотрудникам которой нужно выполнить обновление счетчика, и счетчик согласований.

UpdateHeadersAsync(companyId, employeeId, header) – метод отправляет пользовательскому приложению данные, которые необходимо отобразить. Это осуществляется с помощью сокетов. Таким образом, онлайн-пользователи смогут увидеть обновленный счетчик.

GetHeader(employeeId, companyId) – получение счетчика при открытии приложения, т. е. для оффлайн-приложения. Данный метод просто запрашивает данные из базы данных и отдает их пользователю при открытии стартовой страницы.

3.2.4 Поддержка модификации базы данных

При разработке приложения модель может часто изменяться по мере появления новых требований. База данных должна быть синхронизирована с моделью. Функция миграции позволяет вносить изменения в модель, а затем распространять эти изменения на схему базы данных.

Миграция включена по умолчанию в EF Core. Они управляются путем выполнения команд.

Команда, создающая миграцию, приведена на листинге 6.

Листинг 6 – Команды создания миграции в EF

```
[Command Line]
dotnet ef migrations add <name of migration>
[Package Manager console]
add-migration <name of migration>
```

При создании миграции платформа сравнивает текущее состояние модели с предыдущей миграцией, если таковая существует, и создает файл, содержащий класс, наследуемый от метода. Классу присваивается то же имя, которое было указано для миграции. Имя файла – это миграционный префикс с пометкой времени.

Пример миграции на добавление столбца `TravelApproval` в таблицу `Approve` приведен в листинге 7.

Листинг 7 – Миграция в EF

```
public partial class AddTravelApprovalParameter : DbMigration
{
    public override void Up()
    {
        AddColumn("dbo.DbApproves", "TravelApproval",
            c => c.Boolean(nullable: false));
    }

    public override void Down()
    {
        DropColumn("dbo.DbApproves", "TravelApproval");
    }
}
```

После того, как миграция создана, необходимо применить ее, чтобы изменения вступили в силу в базе данных. Существует несколько способов применения миграции: с использованием сценариев SQL, с использованием метода `Database.Migrate` или с использованием методов командной строки [10].

4 Контроль качества программного обеспечения

4.1 Функциональное тестирование

Тестирование черного ящика или поведенческое тестирование – это стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта. Иначе говоря, тестированием черного ящика занимаются инженеры по тестированию, не имеющие доступа к исходному коду приложения [11].

Произведем функциональное тестирование заполнения поля Email при создании пользователя. Администратор должен иметь возможность ввести Email пользователя, которого он регистрирует. Email должен состоять из не более 254 символов, содержать в себе латинские буквы, а также при наличии цифры, иметь знак @, после которого следует почтовый домен. Почтовый домен должен иметь знак «.», но не в начале, и не в конце.

В таблице 8 представлены выделенные классы эквивалентности.

Таблица 8 – Классы эквивалентности

Входные условия	Правильные классы эквивалентности	Неправильные классы эквивалентности
Длина	0-254(1)	>254(2)
Содержание	(a-z, A-Z) (3), 0-9 (необязательно) (4)	Другие(5)
Наличие @	Есть(6)	Нет (7)
Формат домена	Обязательно наличие хотя бы одной «.» между символами(8)	Отсутствие «.» (9) «.» в начале(10) «.» в конце(11)
Длина	0-254(1)	>254(2)

Определим граничные условия. Длина электронной почты по техническом заданию не должна превышать 254 символа. Следовательно, почта с длиной 254 символа находится на границе допустимых значений. Также необходимо проверить ввод почты, которая выходит за границу – почту с длиной 255 символов.

Для того, чтобы построить эффективно-результативные тесты, можно воспользоваться методом функциональных диаграмм. При построении тестов с использованием данного метода в спецификации определяются причины и следствия. Причина – входное условие, а следствие – выход системы. Далее спецификация преобразуется в управляющий граф [12].

Для процесса создания предкомандировочной заявки была построена функциональная диаграмма, представленная на рисунке 18.

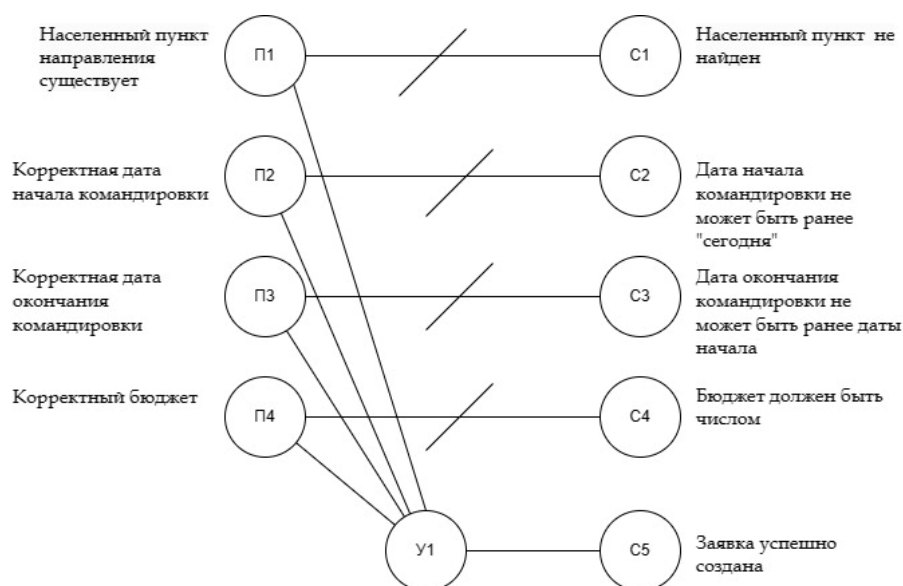


Рисунок 18 – Функциональная диаграмма

Причины и следствия представлены на рисунке 19.

Причины	
П1	Населенный пункт направления существует
П2	Корректная дата начала командировки
П3	Корректная дата окончания командировки
П4	Корректная сумма бюджета
Следствия	
С1	Не найден населенный пункт
С2	Невозможно выбрать дату начала ранее, чем сегодня
С3	Невозможно выбрать дату окончания ранее, чем дату начала
С4	Некорректные символы в поле бюджет
С5	Сообщение: «Заявка успешно создана»

Рисунок 19 – Причины и следствия

На основании функциональной диаграммы была построена таблица решений. Таблица решений представлена в таблице 9.

Таблица 9 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
П1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
П2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
П3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
П4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
С1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
С2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
С3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
С4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
С5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

В таблице 10 представлен набор тестов, составленных на основе таблицы решений.

Таблица 10 – Набор тестов для таблицы решений

№	Входные данные	Ожидаемый результат
1	Направление: Терновка Даты: 01.01.2022-01.12.2021 Бюджет: а	Не найден населенный пункт Невозможно выбрать дату начала ранее, чем сегодня Невозможно выбрать дату окончания ранее, чем дату начала Некорректные символы в поле Бюджет
2	Направление: Пенза Даты: 01.01.2022-01.12.2021 Бюджет: а	Невозможно выбрать дату начала ранее, чем сегодня Невозможно выбрать дату окончания ранее, чем дату начала Некорректные символы в поле Бюджет
3	Направление: Терновка Даты: 30.04.2023-01.12.2021 Бюджет: а	Не найден населенный пункт Невозможно выбрать дату окончания ранее, чем дату начала Некорректные символы в поле Бюджет
4	Направление: Пенза Даты: 30.04.2023-01.12.2021 Бюджет: а	Невозможно выбрать дату окончания ранее, чем дату начала Некорректные символы в поле Бюджет
5	Направление: Терновка Даты: 01.01.2022-01.12.2022 Бюджет: а	Не найден населенный пункт Невозможно выбрать дату начала ранее, чем сегодня Некорректные символы в поле Бюджет

Продолжение таблицы 10

№	Входные данные	Ожидаемый результат
6	Направление: Пенза Даты: 01.05.2023-01.12.2022 Бюджет: а	Невозможно выбрать дату окончания ранее, чем дату начала Некорректные символы в поле Бюджет
7	Направление: Терновка Даты: 01.05.2023-01.12.2023 Бюджет: а	Не найден населенный пункт Некорректные символы в поле Бюджет
8	Направление: Пенза Даты: 01.05.2023-01.12.2023 Бюджет: а	Некорректные символы в поле Бюджет
9	Направление: Терновка Даты: 30.04.2022-01.12.2021 Бюджет: 1000	Не найден населенный пункт Невозможно выбрать дату начала ранее, чем сегодня Невозможно выбрать дату окончания ранее, чем дату начала
10	Направление: Пенза Даты: 30.04.2022-01.12.2021 Бюджет: 1000	Невозможно выбрать дату начала ранее, чем сегодня Невозможно выбрать дату окончания ранее, чем дату начала
11	Направление: Терновка Даты: 01.05.2023-01.12.2022 Бюджет: 1000	Не найден населенный пункт Невозможно выбрать дату окончания ранее, чем дату начала
12	Направление: Пенза Даты: 01.05.2023-01.12.2022 Бюджет: 1000	Невозможно выбрать дату окончания ранее, чем дату начала
13	Направление: Пенза Даты: 01.05.2022-01.12.2022 Бюджет: 1000	Невозможно выбрать дату начала ранее, чем сегодня Невозможно выбрать дату окончания ранее, чем дату начала
14	Направление: Терновка Даты: 01.05.2023-01.12.2023 Бюджет: 1000	Не найден населенный пункт
15	Направление: Пенза Даты: 01.05.2022-01.12.2023 Бюджет: 1000	Невозможно выбрать дату начала ранее, чем сегодня
16	Направление: Пенза Даты: 01.05.2023-01.12.2023 Бюджет: 1000	Заявка успешно создана

Функциональное тестирование показало, что при загрузке набора данных программа работает правильно. Результаты тестирования представлены в приложении Г.

4.2 Модульное тестирование

Модульный тест принимает небольшую единицу приложения, как правило, метод, изолирует его от остальной части кода и проверяет, что он работает должным образом. Его целью является проверка того, что каждая единица функциональности выполняется должным образом, поэтому ошибки не распространяются во всем приложении. Обнаружение ошибки, в которой она возникает, является более эффективной, чем наблюдение за эффектом ошибки косвенно на вторичной точке сбоя.

Модульное тестирование оказывает наибольшее влияние на качество кода, когда это неотъемлемая часть рабочего процесса разработки программного обеспечения. Модульные тесты могут выступать в качестве документации по проектированию и функциональным спецификациям для приложения. Как только метод написан, модульные тесты должны быть написаны, чтобы проверить поведение метода в ответ на стандартные, границы и неправильные входные данные и проверить любые явные или неявные допущения, сделанные кодом [13].

Для приложения были написаны Unit-тесты. Для написания использовались NUnit-пакеты NUnit и Moq.

NUnit – это открытая среда юнит-тестирования приложений для .NET, которая позволяет создавать автоматические тесты. NUnit является производным от популярной платформы тестирования, используемой eXtreme Programming (XP) [14].

Данный вид тестов обладает рядом преимуществ:

- высокое качество программы;
- снижение стоимости;
- безопасность регрессии сети.

Чем выше качество программы, тем меньше средств затрачивается на устранение недостатков проекта. То есть, если найти недостатки в проекте на раннем этапе, решить их будет дешевле.

Примеры модульных тестов приведены в приложении Г.

Таким образом, UNIT-тестирование решает следующие задачи:

- поиск и исправление ошибок на ранних стадиях разработки программного продукта и, следовательно, снижение затрат в дальнейшем;
- лучшее понимание разработчиками базового кода проекта, более простая и быстрая корректировка продукта;
- повторное использование кода, в том числе с переносом (вместе с тестами) в другие продукты;
- использование юнит-тестов как проектной документации, по которой разработчики, не знакомые с кодом, могут понять принцип его работы.

4.3 Метрики устойчивости и абстрактности компонент

Для вычисления позиционной устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

Величина меры неустойчивости изменяется в диапазоне от $[0, 1]$. Если $I = 0$, то компонент максимально устойчив, если $I = 1$, то компонент максимально неустойчив.

Для компонента `RequestService` была рассчитана величина метрики неустойчивости:

$$I = \frac{2}{12 + 2} = 0.143.$$

Мерой абстрактности компонента служит метрика A .

$$A = \frac{Na}{Nc},$$

где Na – количество абстрактных классов в компоненте;

N_c – количество классов в нем же.

Значение метрики A изменяется в диапазоне от 0 до 1. 0 означает полное отсутствие абстрактных классов в компоненте, а 1 означает, что компонент не содержит ничего, кроме абстрактных классов.

Для компонента `RequestService` была рассчитана метрика абстрактности A :

$$A = \frac{12}{25} = 0.48.$$

Вычислив меру устойчивости компоненты и ее абстрактность можно отобразить их на графике. На вертикальной оси необходимо отложить меру абстрактности A , а на горизонтальной оси – меру неустойчивости I .

Компоненты, которые располагаются в верхнем левом углу или в нижнем правом, являются максимально качественными. На практике часто случается так, что полученная точка лежит на диагонали между точкой (1,0) и точкой (0,1).

Для вычисления количественной оценки используется формула вычисления расстояния до главной последовательности:

$$D = \left| \frac{A + I - 1}{\sqrt{2}} \right|.$$

Чем ближе точка располагается к главной последовательности, тем лучше. Расположение точки для оцениваемого компонента изображено на рисунке 20.

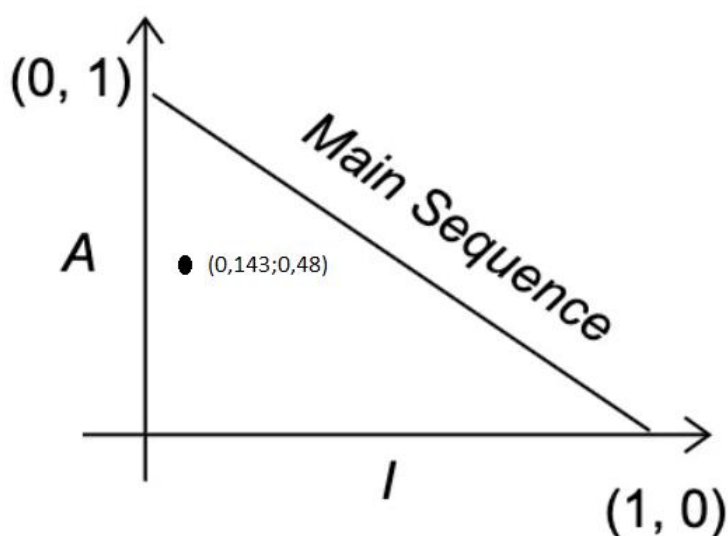


Рисунок 20 – Зоны исключения

Было вычислено расстояние до главной последовательности:

$$D = \left| \frac{0.48 + 0.143 - 1}{\sqrt{2}} \right| = 0.27.$$

Расстояние отношений метрик первой компоненты до главной последовательности достаточно близко к нулю, а значит, ее отношение абстрактности к устойчивости является правильным.

4.4 Метрики кода

Был проведен анализ качества кода реализованной программы с помощью Visual Studio. Метрики кода приведены на рисунке 21.

	Индекс удобства подд...	Сложность организац...	Глубина наследования	Взаимозависимость к...	Строки исходного кода	Строки исполняемого ко...
lication.Analytics (Debug)	95	4	1	8	25	3
lication.Language (Debug)	96	8	1	6	36	5
lication.PrintedForms (Debug)	70	48	1	34	220	104
lication.Report (Debug)	89	5	1	11	51	11
lication.Scheme (Debug)	88	42	1	19	182	49
lication.TravelApproval (Debug)	88	324	3	87	1 416	489
lication.Users (Debug)	93	12	1	9	53	11
on (Debug)	74	18	1	17	191	42
on.Cluster (Debug)	83	41	1	40	163	73
on.Http (Debug)	90	19	1	10	74	18
on.Log (Debug)	94	3	1	8	31	3
on.Notification (Debug)	90	104	1	28	263	100
abase.Sql (Debug)	65	222	2	87	7 974	2 242
.Cart (Debug)	92	181	1	48	411	92
.Companies (Debug)	91	10	1	12	62	10
.Employees (Debug)	91	75	1	20	190	55
.Files (Debug)	87	9	1	19	71	16
.PrintedForms (Debug)	82	164	1	53	531	257
.Scheme (Debug)	89	55	1	43	333	83
.TravelApproval (Debug)	91	142	1	41	604	159
.Trips (Debug)	95	24	1	13	71	11
.Uac (Debug)	98	49	1	36	229	28
(Debug)	97	161	1	19	214	29
al (Debug)	78	81	3	132	587	173

Рисунок 21 – Метрики кода, вычисленные в среде Visual Studio

Индекс удобства поддержки показывает, насколько удобно и легко поддерживать данный код. Рассчитывается по формуле:

$$MI = \max \left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171} \right),$$

где HV – Halstead Volume, вычислительная сложность;

CC – Cyclomatic Complexity, структурная сложность кода;

LoC – количество строк исходного кода [15].

Нормой является значение от 20 до 100. В данном случае значения метрики для каждого модуля близки к 100, что означает, что код удобен в поддержке.

5 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project. Проект был начат 12 апреля 2023 года.

При планировании разработки было выделено 28 задач. Список задач и их описание представлены на рисунке 22.

		Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1		Начало реализации проекта	0 дней	Ср 12.04.23	Ср 12.04.23		
2		▢ Техническое задание	10 дней	Ср 12.04.23	Вт 25.04.23		Интернет[200,00 Р]
3		Разработка технического за	7 дней	Ср 12.04.23	Чт 20.04.23	1	Ведущий программист,Продакт-менеджер
4		Согласование ТЗ с заказчик	3 дней	Пт 21.04.23	Вт 25.04.23	3	Продакт-менеджер
5		Завершение разработки ТЗ	0 дней	Вт 25.04.23	Вт 25.04.23	4	
6		▢ Проектирование	5 дней	Ср 26.04.23	Вт 02.05.23		Интернет[700,00 Р]
7		Создание макетов	1 день	Чт 27.04.23	Чт 27.04.23	5	Продакт-менеджер;Программист2
8		Согласование макетов с зак	1 день	Пт 28.04.23	Пт 28.04.23	7	Продакт-менеджер
9		Проектирование модулей ПК	5 дней	Ср 26.04.23	Вт 02.05.23	5	Ведущий программист,Программист1
10		Завершение проектирования	0 дней	Вт 02.05.23	Вт 02.05.23	8;9	
11		▢ Реализация	18 дней	Ср 03.05.23	Пт 26.05.23		Интернет[3 000,00 Р]
12		Разработка базы данных	3 дней	Ср 03.05.23	Пт 05.05.23	10	Программист1
13		Разработка серверной части	10 дней	Пн 08.05.23	Пт 19.05.23	12	Ведущий программист,Программист1
14		Разработка клиентского при	3 дней	Пн 22.05.23	Ср 24.05.23	13	Ведущий программист[50%];Программист2
15		Интеграция клиентской и се	1 день	Чт 25.05.23	Чт 25.05.23	14	Программист1;Программист2
16		Разработка инфраструктуры	2 дней	Чт 25.05.23	Пт 26.05.23	13;14	Программист1[30%];Ведущий программист
17		Завершение реализации	0 дней	Пт 26.05.23	Пт 26.05.23	16	
18		▢ Тестирование	5 дней	Пн 29.05.23	Пт 02.06.23		Интернет[1 000,00 Р]
19		Написание unit-тестов	1 день	Пн 29.05.23	Пн 29.05.23	17	Программист1;Программист2
20		Функциональное тестирован	3 дней	Пн 29.05.23	Ср 31.05.23	17	Тестировщик
21		Исправление ошибок	1 день	Чт 01.06.23	Чт 01.06.23	19;20	Программист1;Программист2
22		Регресс тестирование	1 день	Пт 02.06.23	Пт 02.06.23	21	Тестировщик
23		Завершение тестирования	0 дней	Пт 02.06.23	Пт 02.06.23	22	
24		▢ Внедрение	6 дней	Пн 05.06.23	Пн 12.06.23		Интернет[200,00 Р]
25		Запуск в эксплуатацию	2 дней	Пн 05.06.23	Вт 06.06.23	23	Ведущий программист,Тестировщик
26		Получение обратной связи	4 дней	Ср 07.06.23	Пн 12.06.23	25	Продакт-менеджер
27		Внедрение завершено	0 дней	Пн 12.06.23	Пн 12.06.23	26	
28		Завершение проекта	0 дней	Пн 12.06.23	Пн 12.06.23	27	

Рисунок 22 – Список задач проекта

Для реализации проекта необходимо выделить основные ресурсы, их тип и стоимость. Описание ресурсов представлено на рисунке 23.

		Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на исполыз.	Начисление	Базовый календарь	Код
1		Ведущий программист	Трудовой		В		100%	00 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный	
2		Программист1	Трудовой		П		100%	60 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональный	Стандартный	
3		Программист2	Трудовой		П		100%	50 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный	
4		Продакт-менеджер	Трудовой		П		100%	40 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный	
5		Тестировщик	Трудовой		Т		100%	30 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный	
6		Интернет	Затраты		И						Пропорциональное		

Рисунок 23 – Список ресурсов проекта

В результате выделения задач и ресурсов была построена диаграмма Ганта (рисунок 24). На ней представлены продолжительность задач и их взаимосвязи друг с другом.

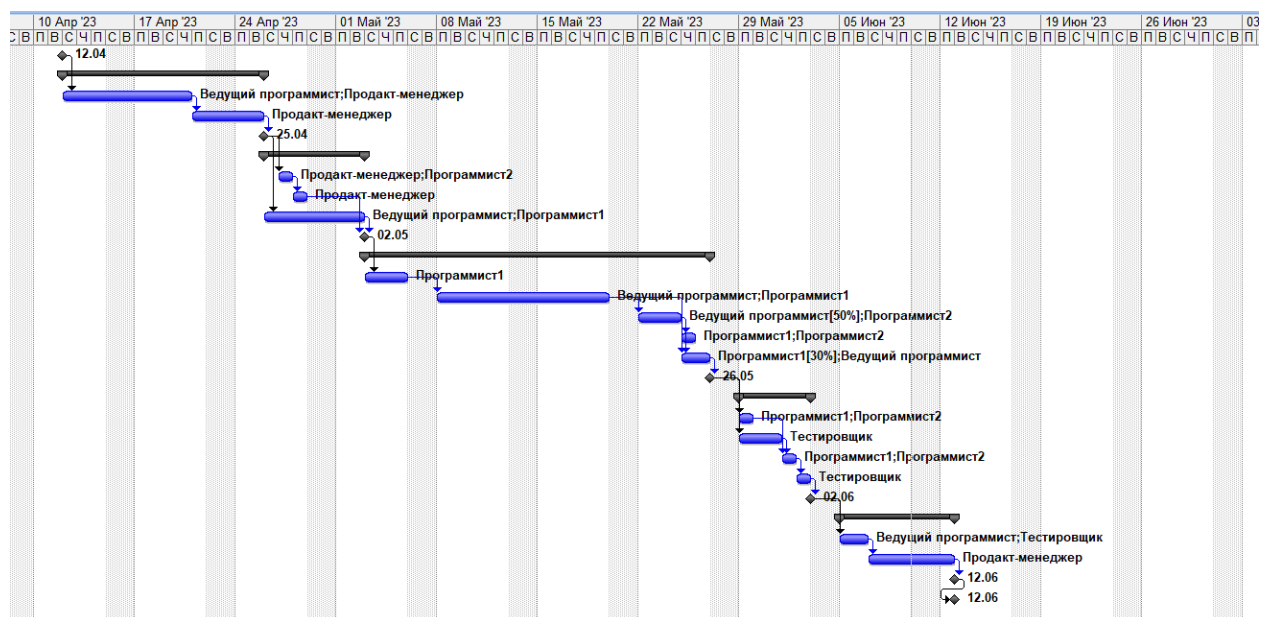


Рисунок 24 – Диаграмма Ганта

На рисунке 25 представлен отчет по бюджету проекта, на котором показаны затраты на каждую задачу.

Отчет о бюджете от Пт 26.05.23
lab6

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты
13	Разработка серверной части	0,00 Р	Пропорциональное	80 000,00 Р
3	Разработка технического задан	0,00 Р	Пропорциональное	49 000,00 Р
9	Проектирование модулей ПО	0,00 Р	Пропорциональное	40 000,00 Р
14	Разработка клиентского прилож	0,00 Р	Пропорциональное	15 000,00 Р
25	Запуск в эксплуатацию	0,00 Р	Пропорциональное	13 000,00 Р
16	Разработка инфраструктуры	0,00 Р	Пропорциональное	11 800,00 Р
12	Разработка базы данных	0,00 Р	Пропорциональное	9 000,00 Р
26	Получение обратной связи	0,00 Р	Пропорциональное	8 000,00 Р
4	Согласование ТЗ с заказчиком	0,00 Р	Пропорциональное	6 000,00 Р
15	Интеграция клиентской и серви	0,00 Р	Пропорциональное	5 500,00 Р
19	Написание unit-тестов	0,00 Р	Пропорциональное	5 500,00 Р
21	Исправление ошибок	0,00 Р	Пропорциональное	5 500,00 Р
7	Создание макетов	0,00 Р	Пропорциональное	4 500,00 Р
20	Функциональное тестирование	0,00 Р	Пропорциональное	4 500,00 Р
8	Согласование макетов с заказч	0,00 Р	Пропорциональное	2 000,00 Р
22	Регресс тестирование	0,00 Р	Пропорциональное	1 500,00 Р
1	Начало реализации проекта	0,00 Р	Пропорциональное	0,00 Р
5	Завершение разработки ТЗ	0,00 Р	Пропорциональное	0,00 Р
10	Завершение проектирования	0,00 Р	Пропорциональное	0,00 Р
17	Завершение реализации	0,00 Р	Пропорциональное	0,00 Р
23	Завершение тестирования	0,00 Р	Пропорциональное	0,00 Р
27	Внедрение завершено	0,00 Р	Пропорциональное	0,00 Р
28	Завершение проекта	0,00 Р	Пропорциональное	0,00 Р
		0,00 Р		260 800,00 Р

Рисунок 25 – Отчет по бюджету

На рисунке 26 показана сводка по проекту, на которой можно увидеть бюджет, длительность, даты проекта и трудозатраты.

<u>Даты</u>			
Начало:	Ср 12.04.23	Окончание:	Пн 12.06.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	44 дней	Оставшаяся:	44 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	44 дней	Процент завершения:	0%
<u>Трудозатраты</u>			
Текущий план:	624,8 ч	Оставшиеся:	624,8 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	624,8 ч	Процент завершения:	0%
<u>Затраты</u>			
Текущий план:	265 900,00 Р	Оставшиеся:	265 900,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	265 900,00 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи	28	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	1
Завершенные задачи:	0	Материальные ресурсы:	0
Итого задач:	28	Итого ресурсов:	6

Рисунок 26 – Обзор по проекту

Таким образом разработка проекта составит 44 полных рабочих дней с 12 апреля 2023 года по 12 июня 2023 года. Трудозатраты составят 625 человеко-часов. Оценка бюджета проекта составляет 266 тысяч рублей.

Заключение

В результате выполнения выпускной квалификационной работы разработаны и протестированы программные средства для согласования и организации командировок.

Рассмотрены программные продукты-аналоги и проведен их сравнительный анализ. В ходе анализа требований к разрабатываемому программному продукту выделены и спроектированы с помощью UML-средств основные бизнес-процессы. Также выявлены функциональные и нефункциональные требования.

Для проекта осуществлено календарное планирование, а также оценена стоимость разработки приложения. На реализацию проекта необходимы 44 дня и 266 000 рублей.

На стадии проектирования приложения разработана концептуальная модель базы данных, спроектирована архитектура программного продукта.

Реализация серверной части происходила в среде разработки JetBrains Rider на языке программирования C# с использованием платформы ASP.NET. В качестве системы управления базами данных выбрана MSSQL, а взаимодействия с ней производились посредством Entity Framework ORM.

Приложение тестировалось двумя видами тестирования: серверная часть протестирована с помощью модульное тестирования, а клиентская часть с помощью функционального. Все тесты были пройдены.

Таким образом, задача, поставленная по реализации программных средств для согласования и организации командировок, выполнена в полном объеме.

Список использованных источников

1. ТК РФ Статья 166. Понятие служебной командировки [Электронный ресурс]. URL: https://www.consultant.ru/document/cons_doc_LAW_34683/9d204a7e952ea1b8293d96849eebc0217b1ab81b/?ysclid=lbxemofwem880286159 (дата обращения: 20.04.2023).
2. Согласование командировок в Битрикс24 [Электронный ресурс]. URL: https://bestrank.ru/solutions/soglasovanie_komandirovok/?ysclid=li4ay5onj9277577078 (дата обращения: 18.04.2023).
3. Модуль бронирования деловых поездок Starliner [Электронный ресурс]. URL: <https://info.starliner.ru/onlajn-platforma/modul-bronirovanija-delovyh-poezdok/?ysclid=li4azrgw1e765266901> (дата обращения: 19.04.2023).
4. OSDY: Организация командировок [Электронный ресурс]. URL: <https://osdy.ru/?ysclid=li4b0pxa1t198672190> (дата обращения: 20.04.2023).
5. Сервис по организации командировок и деловых поездок в России Аэротур Business [Электронный ресурс]. URL: <https://aerob2b.ru/?ysclid=liejvhsw6d717305188> (дата обращения 19.04.2023).
6. Разработка информационной модели [Электронный ресурс]. URL: www.bikmeyev-at.ugatu.su/students/InfSemiot/Lab_01.html (дата обращения: 21.04.2023).
7. Технология ASP.Net: основы, создание сложных веб – страниц [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/602/458/lecture/10259?ysclid=lijyuu5yb8911624745> (дата обращения: 25.04.2023).
8. Сергеева Ю. В. Анализ и сравнительная характеристика современных фреймворков для разработки web-приложений. URL: http://www.rusnauka.com/11_EISN_2011/Informatica/3_85213.doc.htm (дата обращения: 24.04.2023).

9. Запросы LINQ to Entities [Электронный ресурс]. URL: <https://metanit.com/sharp/efcore/5.1.php?ysclid=lbxfc3kq4a82740616> (дата обращения: 11.05.2023).

10. Миграции в Entity Framework Core [Электронный ресурс]. URL: <https://forproger.ru/article/migracii-i-isxodnye-dannye-s-entity-framework-core> (дата обращения: 20.05.2023).

11. Тестирование черного ящика или поведенческое тестирование [Электронный ресурс]. URL: <https://testirovanie24.ru/163868-2/?ysclid=lijz5ipnhb204380055> (дата обращения: 23.05.2023).

12. Системное тестирование: метод функциональных диаграмм [Электронный ресурс]. URL: https://studopedia.net/6_86110_sistemnoe-testirovanie-metod-funktsionalnih-diagramm.html?ysclid=lijzd870cg56091912 (дата обращения: 25.05.2023).

13. Модульное тестирование [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/unit-testing> (дата обращения: 25.05.2023).

14. Unit Testing Using NUnit [Электронный ресурс]. URL: <https://www.codeproject.com/Articles/178635/Unit-Testing-Using-NUnit> (дата обращения: 28.05.2023).

15. Средства оценки качества кода MS Visual Studio [Электронный ресурс]. URL: <https://habr.com/ru/post/111524/> (дата обращения: 19.05.2023).

Приложение А
ГЛОССАРИЙ
(справочное)

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

ORM (Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Приложение Б
СКРИПТЫ РАЗВЕРТЫВАНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг Б.1 – Скрипт создания таблицы Approve

```
create table Approve
(
    Id nvarchar(128) not null constraint [PK_dbo.Approve]
        primary key,
    EmployeeId nvarchar(128) not null
        constraint [FK_dbo.Approves_Employee_EmployeeId]
        references Employee on delete cascade,
    Resolution int not null,
    ApproveType int default 0 not null,
    StartDate datetime,
    Step_Id int
        constraint [FK_dbo.Approves_dbo.ApprovalSteps_Id]
        references ApprovalSteps
)
go
```

Листинг Б.2 – Скрипт вставки данных в таблицу Employee

```
INSERT INTO [Employee]
([MobilePhone], [Code], [Birthday], [Sex], [Status], [Surname],
[Name], [Patronymic], [Email], [ApprovalSchemeId])
VALUES
(<MobilePhone, nvarchar(15),>, <Code, nvarchar(2),>, <Birthday,
datetime,>, <Sex, int,>, <Status, int,>, <Surname, nvarchar(80),>,
<Name, nvarchar(80),>, <Patronymic, nvarchar(80),>, <Email,
nvarchar(80),>, <ApprovalSchemeId, int,>)
GO
```

Листинг Б.3 – Скрипт создания таблицы ApprovalStep

```
create table ApprovalStep
(
    Id int identity constraint [PK_dbo.ApprovalStep] primary key,
    ApprovalSchemeId int not null constraint
        [FK_dbo.ApprovalStep_dbo.ApprovalSchemes_ApprovalSchemeId]
        references ApprovalSchemes,
    TypeTrips int not null,
    Condition int not null
)
create index IX_ApprovalStep_ApprovalSchemeId on ApprovalStep
(ApprovalSchemeId)
```

Приложение В
ЛИСТИНГ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг В.1 – ApproveService.cs

```

public async Task<Request> GetRequest(int accountId, string
userId, int requestId)
{
    var request = await approvalService.GetRequest(requestId);
    if (request == null)
    {
        return null;
    }

    if (request.Approves.Any(approver => approver.UserId ==
userId))
    {
        return request;
    }
    var rights = await uacService.GetRights(userId);
    if (rights == null)
    {
        return null;
    }

    if (rights.Approve == ApproveTripLevel.Available)
    {
        return request;
    }

    var user = await userService.GetUser(accountId, userId);

    if (user == null || !user.EmployeeId.HasValue)
    {
        return null;
    }

    if (rights.BuyTripAccount ==
BuyTripsAccountLevel.ApprovalScheme)
    {
        var employee = await
employeeService.GetEmployeeById(user.EmployeeId.Value);

        if (employee == null)
        {
            return null;
        }

        var companyIds = employee.Companies.Select(company =>
company.CompanyId);

        foreach (var requestEmployee in request.Employees)
        {
            if (requestEmployee.EmployeeId == employee.Id)
            {

```

```

        return request;
    }

    if (!companyIds.Contains(requestEmployee.CompanyId))
    {
        return null;
    }
    return request;
}

if (rights.BuyTripPersonal ==
    buyTripsPersonalLevel.ApprovalScheme)
{
    return request.Employees.Any(employee =>
        employee.EmployeeId == user.EmployeeId.Value) ? request :
        null;
}

return null;
}

public async Task CreateRequest(string token, int accountId,
    string userId, string userName, ApprovalRequestIn requestIn,
    string personalOfficeLink)
{
    var user = await userService.GetUser(accountId, userId);
    if (user?.EmployeeId == null)
    {
        return;
    }

    var employee = await
employeeService.GetEmployeeById(user.EmployeeId.Value);
    if (employee == null)
    {
        return;
    }

    var result = await approvalService.CreateRequest(accountId,
    userId, mapping.MapToDomainRequest(requestIn));
    var requestList = await FillResult(new List<Request>(1) {
result }, accountId);
    var request = requestList.FirstOrDefault();
    if (request == null)
    {
        return;
    }

    var steps = (await
stepService.GetSteps(request.ApprovalSchemeId)).ToList();
    var step = steps.FirstOrDefault();

    if (requestIn.StepsSettings.Any())

```

```

    {
        foreach (var stepSetting in requestIn.StepsSettings)
        {
            var currentStep = steps.FirstOrDefault(s => s.Id ==
stepSetting.Id);
            if (currentStep == null)
            {
                return;
            }

            bool selfApprove = currentStep.Approvers.Count == 1 &&
currentStep.Approvers.FirstOrDefault() == userId
                                || currentStep.Approvers.Count == 0
&&
                                employee.Companies.Any(a =>
a.Departments.Any(b => b.HeadOfDepartmentEmployeeId.HasValue &&
b.HeadOfDepartmentEmployeeId.Value == employee.Id)) &&
                                currentStep.Roles.Any(c => c ==
UserRoles.HeadOfDepartment);

            if (!stepSetting.Approvers.Any() && selfApprove)
            {
                await
approvalService.CreateCustomApprovers(request.Id, new List<string>
{ userId }, stepSetting.Id);
            }
            else
            {
                await
approvalService.CreateCustomApprovers(request.Id,
stepSetting.Approvers, stepSetting.Id);
            }
        }
    }
    else
    {
        await reportClient.PushAsync(request.Id);
        return;
    }
    await reportClient.PushAsync(request.Id);
    await CreateApprove(step, accountId, request, token, userName,
requestIn.Comment, personalOfficeLink, false);
}

private async Task CreateApprove(Step step, int accountId,
ApprovalRequestOut request, string token,
    string userName, string comment, string personalOfficeLink,
bool isApproveFromLetter)
{
    var customApprovers = await
approvalService.GetCustomApprovers(request.Id, step.Id);

    if (!customApprovers.Any())

```

```

    {
        return;
    }

    var requestOwner = await userService.GetUser(accountId,
request.UserId);
    if (requestOwner == null || !requestOwner.EmployeeId.HasValue)
    {
        return;
    }

    var requestOwnerEmployee = (await
employeeService.GetEmployees(new[]
{requestOwner.EmployeeId.Value})).First();

    foreach (var approverId in customApprovers)
    {
        var selfRequest = request.UserId == approverId;
        if (string.IsNullOrEmpty(approverId))
        {
            continue;
        }

        var approverUser = await userService.GetUser(accountId,
approverId);
        if (approverUser == null)
        {
            continue;
        }

        var approve = new Domain.TravelApproval.Models.Approve
        {
            Name = string.Join(", ", request.Destinations.Select(a
=> a.Name)),
            ApproverId = approverId,
            RequestId = request.Id,
            StepId = step.Id,
            Email = approverUser.Email,
            Messages = CreateMessages(comment, userName,
requestOwnerEmployee)
        };

        var result = await approvalService.CreateApprove(approve,
selfRequest);

        await reportClient.PushAsync(request.Id);

        var departmentValidation =
ValidateDepartments(request.Employees, step);

        if (departmentValidation.Any())
        {
            User admin = await userService.GetAdmin(accountId);

```

```

        if (admin == null)
        {
            return;
        }

        await
notificationService.SendEmailToAdmin(FillEmailToAdmin(request,
requestOwnerEmployee, comment, departmentValidation), admin,
result, personalOfficeLink);
        await UpdateHeader(token, accountId, admin.Id,
admin.UserName, isApproveFromLetter);

        return;
    }

    if (!selfRequest)
    {
        string approverLocale = await
languageService.GetLocale(approverUser.MultiCompanyId ?? 0,
approverUser.Id);
        var emailRequest = FillEmailRequest(request,
requestOwnerEmployee, comment);
        await
notificationService.SendEmailToApprover(emailRequest, result,
approverLocale, personalOfficeLink);
        await UpdateHeader(token, accountId, result.UserId,
result.Email, isApproveFromLetter);
        continue;
    }

    var nextStep = await GetNextStep(request, result.StepId);

    if (nextStep == null)
    {
        return;
    }

    var initiator = await userService.GetUser(accountId,
request.UserId);

    await UpdateHeader(token, accountId, result.UserId,
result.Email, isApproveFromLetter);
    await CreateApprove(nextStep, accountId, request, token,
initiator.Email, comment, personalOfficeLink,
isApproveFromLetter);
}
}

```

Листинг В.2 – StepService.cs

```

public async Task DeleteSteps(int accountId, int schemeId)
{
    if (!await schemeService.SchemeRefersAccount(accountId,

```

```

schemeId))
{
    return;
}

await stepService.DeleteSteps(schemeId);
}

public async Task<IEnumerable<ClientStep>> EditSteps(int
accountId, int schemeId, IEnumerable<ClientStep> clientSteps)
{
    if (!await schemeService.SchemeRefersAccount(accountId,
schemeId))
    {
        return Enumerable.Empty<ClientStep>();
    }

    await stepService.DeleteSteps(schemeId);

    foreach (var step in clientSteps)
    {
        step.SchemeId = schemeId;
    }

    return await SaveSteps(clientSteps);
}

public async Task<IEnumerable<ClientStep>> GetAccountSteps(int
accountId)
{
    var schemeIds = await
schemeService.GetAccountSchemeIds(accountId);

    var steps = await stepService.GetSteps(schemeIds);

    return steps.Select(step =>
mappingService.MapToClientStep(step));
}

public async Task<IEnumerable<ClientStep>> GetSteps(int accountId,
int schemeId)
{
    if (!await schemeService.SchemeRefersAccount(accountId,
schemeId))
    {
        return Enumerable.Empty<ClientStep>();
    }

    var steps = await stepService.GetSteps(schemeId);

    return steps.Select(step =>
mappingService.MapToClientStep(step));
}

```

```

public async Task<IEnumerable<ClientStep>> SaveSteps(int
accountId, int schemeId, IEnumerable<ClientStep> steps)
{
    if (!await schemeService.SchemeRefersAccount(accountId,
schemeId))
    {
        return Enumerable.Empty<ClientStep>();
    }

    foreach (var step in steps)
    {
        step.SchemeId = schemeId;
    }

    return await SaveSteps(steps);
}

public async Task<bool> SchemeExists(int accountId, int schemeId)
{
    if (!await schemeService.SchemeRefersAccount(accountId,
schemeId))
    {
        return false;
    }

    return await stepService.SchemeExists(schemeId);
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using OfficeOpenXml;
using Smartway.Common;
using Smartway.Models;
using Smartway.Models.Employee;

namespace Smartway.Application.Report
{
    public class ReportSheetApproval :
IReportSheet<Smartway.Models.Report.Report>
    {
        /// <summary>
        /// Лист для поездов
        /// </summary>
        public void CreateSheet(ExcelPackage package,
List<Smartway.Models.Report.Report> report,
List<Company> companies, List<Department> departments)
        {
            var sheet = package.Workbook.Worksheets[1];

            var projects = report.SelectMany(r =>

```

```

r?.Projects).Where(p => p != null).DistinctBy(p =>
p?.Id)?.OrderBy(p => p.Id).ToList();
    var tags = report.SelectMany(r => r?.Tags).Where(t =>
t != null).DistinctBy(t => t?.Id)?.OrderBy(p => p.Id).ToList();
    var userAnalytics =
        report.SelectMany(r => r.UserAnalytics).Where(a =>
a != null).DistinctBy(a => a?.Id)?.OrderBy(a => a.Id).ToList();

    var allColumns = ExpandHeader(sheet, projects, tags,
userAnalytics);
    SetStyles(sheet, allColumns, report.Count);
    FillDocument(sheet, report, companies, departments,
userAnalytics);
    HideColumns(report, sheet, companies);
    sheet.View.FreezePanels(3, 1);
    sheet.View.ZoomScale = 70; // 70% масштаба
}

private void SetStyles(ExcelWorksheet sheet, int
allColumns, int reportCount)
{
    const int startRow = 4;
    if (reportCount == 1)
    {
        return;
    }

    for (int i = startRow; i < reportCount + startRow - 1;
i++)
    {
        for (int j = 1; j <= allColumns; j++)
        {
            sheet.Cells[i, j].StyleID =
sheet.Cells[startRow - 1, j].StyleID;
            sheet.Cells[i, j].Style.WrapText = true;
        }
    }
}

private void
HideColumns(List<Smartway.Models.Report.Report> report,
ExcelWorksheet sheet, List<Company> companies)
{
    if (!report.Any(i => i.Departments.Any(d => d != 0)))
    {
        // Отдел
        sheet.DeleteColumn(13);
    }
    if (!report.Any(i => i.Companies.Any()))
    {
        // Организация
        sheet.DeleteColumn(12);
    }
}

```

```

        if (report.All(a => !a.PaidDate.HasValue))
        {
            // Дата оплаты
            sheet.DeleteColumn(10);
        }

        if (report.All(a =>
string.IsNullOrEmpty(a.PaidBy)))
        {
            // Кто оплатил
            sheet.DeleteColumn(9);
        }

        if (!report.Any(a => a.DeclinedApprovers.Any()))
        {
            // Кем отказано
            sheet.DeleteColumn(7);
        }

        if (!report.Any(a => a.AcceptedApprovers.Any()))
        {
            // Кем согласовано
            sheet.DeleteColumn(6);
        }
    }

    private int ExpandHeader(ExcelWorksheet sheet,
        ICollection<Project> projects,
        ICollection<Tag> tags,
        ICollection<Smartway.Models.UserAnalytics>
userAnalytics)
    {
        const int HeaderRow = 2;
        var currentColumn = 19;
        var userAnalyticsNames = userAnalytics.Select(a =>
a.Name).ToArray();

        if (projects.Count > 0)
        {
            sheet.Cells[HeaderRow, currentColumn].StyleID =
sheet.Cells[HeaderRow, currentColumn - 1].StyleID;
            sheet.Cells[HeaderRow + 1, currentColumn].StyleID
= sheet.Cells[HeaderRow + 1, currentColumn - 1].StyleID;
            sheet.Cells[HeaderRow, currentColumn].Value =
"Центр затрат";
            currentColumn++;
        }

        if (tags.Count > 0)
        {
            sheet.Cells[HeaderRow, currentColumn].StyleID =
sheet.Cells[HeaderRow, currentColumn - 1].StyleID;
            sheet.Cells[HeaderRow + 1, currentColumn].StyleID

```

```

= sheet.Cells[HeaderRow + 1, currentColumn - 1].StyleID;
    sheet.Cells[HeaderRow, currentColumn].Value =
"Метки";
    currentColumn++;
}

    var userAnalyticsCounter = 0;
    for (int i = currentColumn; i < currentColumn +
userAnalyticsNames.Length; i++)
    {
        sheet.Cells[HeaderRow, i].StyleID =
sheet.Cells[HeaderRow, i - 1].StyleID;
        sheet.Cells[HeaderRow + 1, i].StyleID =
sheet.Cells[HeaderRow + 1, i - 1].StyleID;
        sheet.Cells[HeaderRow, i].Value =
userAnalyticsNames[i - currentColumn];
        userAnalyticsCounter++;
    }

    return currentColumn + userAnalyticsCounter;
}

private void FillDocument(ExcelWorksheet sheet,
List<Smartway.Models.Report.Report> report,
List<Company> companies,
List<Department> departments,
ICollection<Smartway.Models.UserAnalytics>
userAnalytics)
{
    if (!report.Any())
    {
        return;
    }

    if (companies.Count <= 3)
    {
        sheet.Cells["A1"].RichText.Add(string.Join(", ",
companies.Select(i => i.GetName()).ToList()).TrimEnd(', '));
    }

    var currentRow = 3;
    var userAnalyticsIds = userAnalytics.Select(a =>
a.Id).ToList();
    foreach (var r in report)
    {
        var columnCounter = 1;
        // Дата запроса
        sheet.Cells[currentRow, columnCounter].Value =
r.RequestDate.AddHours(3);
        // Статус согласования
        sheet.Cells[currentRow, ++columnCounter].Value =
r.Status;
        // Автор запроса

```

```

        sheet.Cells[currentRow, ++columnCounter].Value =
r.ApproveOwner;
        // Согласующие
        var counter = r.Approvers.Count;
        columnCounter++;
        foreach (var a in r.Approvers)
        {
            sheet.Cells[currentRow,
columnCounter].RichText.Add(a);
            if (counter > 1)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
            }

            counter--;
        }

        columnCounter++;
        if (r.Comments.Any())
        {
            counter = r.Comments.Count;
            foreach (var c in r.Comments)
            {
                // Комментарии
                if (!string.IsNullOrEmpty(c.Text))
                {
                    sheet.Cells[currentRow,
columnCounter].RichText.Add(c.Name).Bold = true;
                    sheet.Cells[currentRow,
columnCounter].RichText.Add($"": {c.Text}").Bold = false;
                    if (counter > 1)
                    {
                        sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
                    }
                }
                counter--;
            }
        }

        columnCounter++;
        if (r.AcceptedApprovers.Any())
        {
            counter = r.AcceptedApprovers.Count;
            // Кем согласовано
            foreach (var a in r.AcceptedApprovers)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add(a);
                if (counter > 1)
                {
                    sheet.Cells[currentRow,

```

```

columnCounter].RichText.Add(Environment.NewLine);
    }

    counter--;
}

columnCounter++;
if (r.DeclinedApprovers.Any())
{
    counter = r.DeclinedApprovers.Count;
    // Кем отказано
    foreach (var a in r.DeclinedApprovers)
    {
        sheet.Cells[currentRow,
columnCounter].RichText.Add(a);
        if (counter > 1)
        {
            sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
        }

        counter--;
    }

    sheet.Cells[currentRow,
2].Style.Font.Color.SetColor(Color.DimGray);
}

// Дата согласования/отказа
sheet.Cells[currentRow, ++columnCounter].Value =
    r.ActionDate.HasValue ?
(object)r.ActionDate.Value.AddHours(3) : string.Empty;
// Кто оплатил
sheet.Cells[currentRow, ++columnCounter].Value =
r.PaidBy;
// Дата оплаты
sheet.Cells[currentRow, ++columnCounter].Value =
r.PaidDate;
// Сотрудники
counter = r.Employees.Count;
columnCounter++;
foreach (var e in r.Employees)
{
    sheet.Cells[currentRow,
columnCounter].RichText.Add(e);
    if (counter > 1)
    {
        sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
    }

    counter--;
}

```

```

    }

    // Организация
    columnCounter++;
    if (r.Companies.Any())
    {
        counter = r.Companies.Count;
        foreach (var c in r.Companies)
        {
            var company = companies.FirstOrDefault(i
=> i.CompanyId == c);
            if (company != null)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add(company.GetName());
                if (counter > 1)
                {
                    sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
                }
            }
            counter--;
        }
    }

    // Отдел
    columnCounter++;
    if (r.Departments.Any(d => d != 0))
    {
        counter = r.Departments.Count;
        foreach (var d in r.Departments)
        {
            var department =
departments.FirstOrDefault(i => i.Id == d);
            if (department != null && department.Id !=
0)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add(department.Name);
                if (counter > 1)
                {
                    sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
                }
            }
            else if (r.Departments.Count > 1 && d ==
0)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add(" ");
                if (counter > 1)
                {
                    sheet.Cells[currentRow,

```

```

columnCounter].RichText.Add(Environment.NewLine);
        }
    }
    counter--;
}

counter = r.Items.Count;
columnCounter++;
foreach (var item in r.Items)
{
    // Услуги
    sheet.Cells[currentRow,
columnCounter].RichText.Add(item);
    if (counter > 1)
    {
        sheet.Cells[currentRow,
columnCounter].RichText.Add(Environment.NewLine);
    }

    counter--;
}

// Дата начала
sheet.Cells[currentRow, ++columnCounter].Value =
r.StartDate;

// Дата завершения
sheet.Cells[currentRow, ++columnCounter].Value =
r.EndDate;

// Итого, стоимость
sheet.Cells[currentRow, ++columnCounter].Value =
r.Amount;

sheet.Cells[currentRow,
columnCounter].Style.Numberformat.Format =
    r.Amount == Math.Truncate(r.Amount) ? "#,##0\\
\\\"₽\\\" : \"#,##0.00\\\" \\\"₽\\\";
// Комментарии о нарушении ТП
sheet.Cells[currentRow, ++columnCounter].Value =
r.TravelPolicyComment;

if (report.SelectMany(x => x.Projects).Any())
{
    columnCounter++;
    counter = r.Projects.Count;
    foreach (var project in r.Projects)
    {
        // Центры затрат
        sheet.Cells[currentRow,
columnCounter].RichText.Add(project.Name);
        if (counter > 1)
        {
            sheet.Cells[currentRow,
columnCounter].RichText.Add("; ");

```

```

        }

        counter--;
    }
}

if (report.SelectMany(x => x.Tags).Any())
{
    columnCounter++;
    counter = r.Tags.Count;
    foreach (var tag in r.Tags)
    {
        // Метки
        sheet.Cells[currentRow,
columnCounter].RichText.Add(tag.Value);
        if (counter > 1)
        {
            sheet.Cells[currentRow,
columnCounter].RichText.Add("; ");
        }

        counter--;
    }
}

if (report.SelectMany(x => x.UserAnalytics).Any())
{
    columnCounter++;
    // ПОЛЬЗОВАТЕЛЬСКИЕ АНАЛИТИКИ
    foreach (var userAnalyticId in
userAnalyticsIds)
    {
        var currentAnalytic =
r.UserAnalytics.FirstOrDefault(a => a.Id == userAnalyticId);

        if (currentAnalytic == null)
        {
            columnCounter++;
            continue;
        }

        var valuesCounter =
currentAnalytic.Values.Count;
        foreach (var value in
currentAnalytic.Values)
        {
            sheet.Cells[currentRow,
columnCounter].RichText.Add(value.Name);
            if (valuesCounter > 1)
            {
                sheet.Cells[currentRow,
columnCounter].RichText.Add("; ");
            }
        }
    }
}

```

```
                valuesCounter--;  
            }  
            columnCounter++;  
        }  
    }  
    currentRow++;  
}  
}  
}
```

Приложение Г
ЛИСТИНГ МОДУЛЬНЫХ ТЕСТОВ
(рекомендуемое)

Листинг Г.1 – Тест проверки корректности ФИО, данные валидны

```
[Test]
public void CheckLatinicOrCyrillicLetters_WhenIsValid()
{
    var value = "Петров";
    var isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Петров-Петровский";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Петров Петровский";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Ivanov";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Ivanov-Petrov";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Ivanov-el-Petrov";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);

    value = "Ivanov Petrov";
    isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.True(isValid);
}
```

Листинг Г.2 – Тест проверки корректности ФИО, данные не валидны

```
[Test]
public void CheckLatinicOrCyrillicLetters_WhenIsNotValid()
{
    var value = "Петров34";
    var isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
    Assert.False(isValid);

    value = "Petrov34";
```

```
        isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
        Assert.False(isValid);

        value = "Petrov34^%^*";
        isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
        Assert.False(isValid);

        value = "Петров^%^*";
        isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
        Assert.False(isValid);

        value = "Петров-Petrov";
        isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
        Assert.False(isValid);

        value = "Petrov-Петров";
        isValid =
validatorService.CheckLatinicOrCyrillicLetters(value);
        Assert.False(isValid);
    }
```

Приложение Д
СПРАВКА О ВНЕДРЕНИИ
(рекомендуемое)

УТВЕРЖДАЮ

руководитель отдела разработки

ООО «Смартвэй»

Мурзаев А. А.

23 мая 2023 года

СПРАВКА

о результатах внедрения решений,
разработанных в выпускной квалификационной работе

Настоящая справка подтверждает то, что результаты выпускной квалификационной работы студентки ФГБОУ ВО «Пензенский государственный университет» факультета ФВТ группы 19ВП2 очной формы обучения Дакиной А. Ю. на тему «Программные средства для оформления и согласования командировок» внедрены в онлайн-сервис организации командировок «Смартвэй».

Руководитель отдела разработки

ООО «Смартвэй»

Мурзаев А. А.

