

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ ДОКУМЕНТООБОРОТА ОТДЕЛА  
ПО ОРГАНИЗАЦИИ КОМАНДИРОВОК

БГУИР КП 1-40 01 01 021 ПЗ

Студент: гр. 081073 Самойло А.А.

Руководитель: Ступень З.Ю.

Минск 2022

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. Постановка задачи	5
1.1 Анализ предметной области	5
1.2 Обзор существующих аналогов	6
1.3 Постановка задачи	9
1.4 Входные и выходные данные	9
1.5 Выбор и обоснование средств разработки	10
2 Функциональное проектирование	12
2.1 Инфологический этап проектирования БД	12
2.2 Диаграмма деятельности	17
2.3 Функциональная модель	18
2.4 Диаграмма компонентов	22
2.5 Диаграмма развертывания	24
3 Программная реализация	26
3.1 Структурная схема	26
3.2 Физическая модель базы данных	27
3.3 Интерфейс разрабатываемого приложения	31
3.4 Описание алгоритмов работы программы	34
4 Тестирование	36
5 Руководство пользователя	39
ЗАКЛЮЧЕНИЕ	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	45
Приложение А	47
Приложение Б	48
Приложение В	49
Приложение Г	50
Приложение Д	51

## ВВЕДЕНИЕ

В практической деятельности предприятий, учреждений и организаций нередки случаи, когда необходимо произвести закупки за наличный расчет на других предприятиях, в розничной торговле, у физических лиц или произвести расчеты с командированными работниками. В таких случаях обычно работнику выдаются наличные денежные средства для выполнения определенных действий по поручению организации. Так же выдаются деньги для предстоящих командировочных расходов, для оплаты хозяйственных расходов, для покупки продукции, для оплаты выполненных работ, оказанных услуг, а также на иные цели.

Целью данного курсового проекта является создание веб-приложения позволяющего вести документооборот в сфере организации командировок и командировочных расходах.

Проблема автоматизации производственных процессов и процессов управления как средства повышения производительности труда всегда являлась и остается актуальной. Необходимость автоматизации управления объясняется задачами облегчения труда управленческого персонала, и в частности, менеджера, сдерживанием, вызываемым развитием производства; усложнением производственных связей; увеличением объемов управленческой функции.

Важную роль играет задача соответствия технической базы управления аналогичной базе производства, в отношении которого производится автоматизация.

На современном этапе автоматизации управления производством наиболее перспективным является автоматизация планово-управленческих функций на базе персональных ЭВМ, установленных непосредственно на рабочих местах специалистов. Эти системы получили широкое распространение в организационном управлении под названием автоматизированных рабочих мест (АРМ). Это позволит использовать

систему людям, не имеющим специальных знаний в области программирования, и одновременно позволит дополнять систему по мере надобности.

Для достижения цели курсовой работы необходимо выполнить следующие задачи:

- изучить сферы деятельности организации и определить обязанности менеджера;
- проанализировать рабочее место специалиста по организации командировок, определить уровень автоматизации рабочего места, выявить недостатки;
- подготовить набор требований к программному обеспечению;
- проанализировать аналоги;
- спроектировать базу данных;
- определить необходимые средства разработки;
- разработать программное обеспечение;
- протестировать разработанное программное обеспечение;
- подготовить руководство пользователя.

## 1. Постановка задачи

### 1.1 Анализ предметной области

Предприятие нуждалось в программном продукте, который мог хранить информацию о командировках и расходах. Раньше эти данные хранились в excel таблице и постоянно изменялись. Менеджер часто редактировал данные, следил за тем, где сейчас находятся сотрудники и вручную создавал необходимые отчеты. Однако весь этот процесс занимает достаточно большой промежуток времени. Для автоматизации и упрощения работы менеджера в первую очередь необходимо определить основные функции менеджера:

- планирование командировок;
- редактирование данных о сотрудниках предприятия;
- редактирование данных о командировках;
- расчет стоимости командировок;
- обеспечение руководителя оперативной информацией о текущих командировках;
- создание и печать отчета о командировке.

Обладая необходимыми программными и техническими средствами, менеджер улучшает свою производительность труда, сокращает время обработки различных запросов. При достаточном уровне автоматизации сокращается время, затраченное на обслуживание одного клиента, что очень важно при современном темпе жизни человека.

Основной задачей разрабатываемого программного средства является ускорение и упрощение процесса обработки данных. Менеджер должен без

особого труда найти необходимую ему информацию. Для этого будет разработана система страниц, между которыми можно будет переключаться для выполнения определенных действий.

Для хранения всей необходимой информации необходимо спроектировать БД, а также предусмотреть возможность расширения. В базе данных необходимо сохранять информацию как минимум о сотрудниках, билетах и командировках.

## 1.2 Обзор существующих аналогов

Аналогов в сети интернет немного и их основной недостаток в том, что они созданы для средних и крупных бизнесов, а также платные и как правило весьма дорогостоящие. Такое решение больше подходит для крупных компаний с большим количеством сотрудников. В связи с этим актуальность программного продукта для небольших компаний становится весьма высокой.

Рассмотрим несколько таких решений:

- Ozon Командировки<sup>[1]</sup>;
- Aviasales<sup>[2]</sup>.

Ozon в рамках своего сервиса для бронирования отелей и билетов Ozon.Travel запустил систему оформления командировок Ozon Командировки. Она позволит автоматизировать согласование рабочих поездок с учётом специфики среднего и крупного бизнеса. Сотрудники смогут сами собрать «корзину» услуг: авиа- и железнодорожные билеты, отель, трансферы, визы и другие. При этом система позволяет определить «роли» для разных работников, установив для них лимиты на расходы.

Ключевые особенности Ozon Командировки:

- web-Приложение, все данные хранятся на серверах Ozon;

- у сотрудников есть роли и лимиты;
- доступно только для юридических лиц;
- у командировок есть статусы;
- документы формируются автоматически.

Главная особенность программы состоит в том, что командировки создаются сотрудниками и отправляются на согласование менеджеру. Данное решение позволяет снизить загруженность менеджера, но это также добавляет немало неудобств сотрудникам.

К достоинствам можно отнести интерфейс. Главная страница Ozon Командировки изображена на рисунке 1.1. Данный интерфейс удобен и содержит всю необходимую информацию.

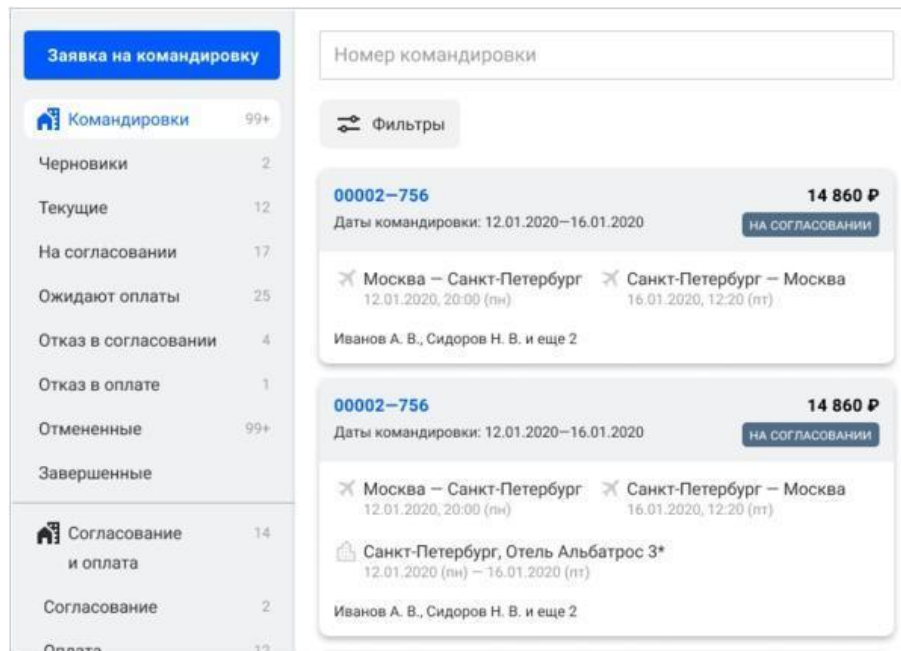


Рисунок 1.1 – Интерфейс Ozon Командировки

Недостаток Ozon Командировки состоит в том, что за использование приложения взимается плата, которая зависит как от количества сотрудников,

так и от количества командировок. Так же к недостаткам можно отнести плохую совместимость с другими программными продуктами, например невозможно использовать единую базу данных сотрудников.

Сервис бронирования авиабилетов Aviasales решил выйти на корпоративный рынок и создал функционал для организации командировок. Он позволит вести учет командировок, создавать отчетные документы (документация будет приходить на почту бухгалтера и других ответственных сотрудников сразу после бронирования), осуществлять контроль и анализ расходов в поездках, настраивать политику бизнес-путешествий, производить оплату заказов с депозита и по постоплате.

Ключевые особенности Aviasales:

- web-Приложение, все данные хранятся на серверах Aviasales;
- быстрый поиск необходимых билетов;
- у командировок есть статусы;
- бесплатный сервис.

Главная особенность, а также и главный недостаток программы состоит в том, что командировки оформляются только с билетами купленными на Aviasales. Сервис предоставляет удобный поиск и покупку билетов, но отсутствует возможность указать дополнительную информацию о командировке. К примеру, невозможно указать дополнительные расходы на командировку.



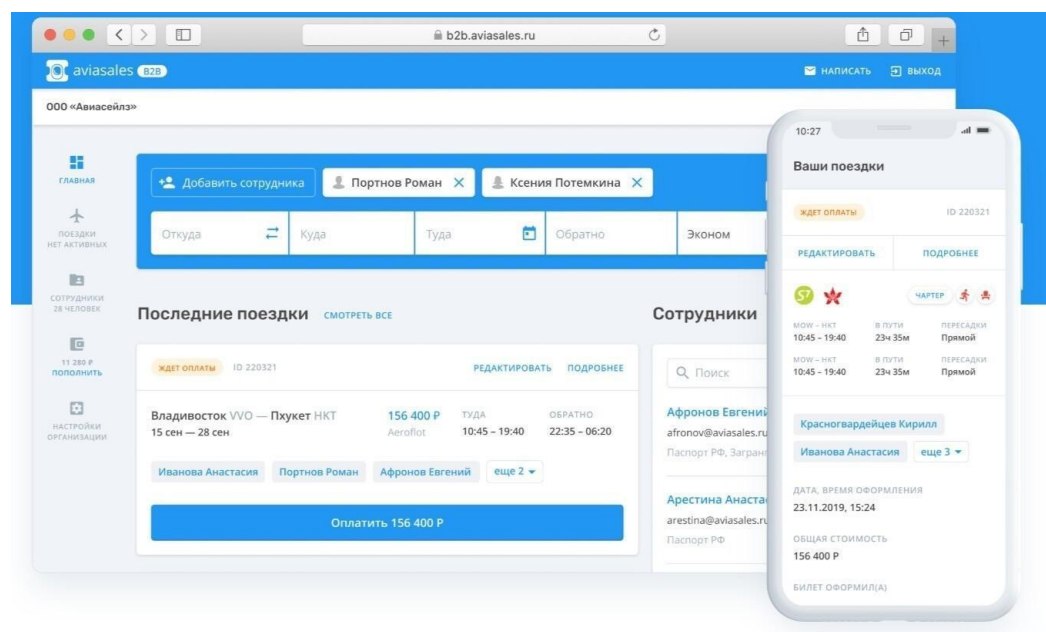


Рисунок 1.2 – Интерфейс Aviasales

На основании анализа существующих аналогов следует сделать вывод о том, что данные сервисы пользуются спросом как у малых, так и крупных бизнесов.

### 1.3 Постановка задачи

Необходимо разработать веб-приложение. Должны быть реализованы следующие функции:

- добавление сотрудника, билета, расходов, командировки;
- редактирование информации о сотруднике, билете, расходах, командировке;
- удаление сотрудника, билета, расходов, командировки;
- поиск билетов по месту отправления/прибытия;

- поиск сотрудника по имени и фамилии;
- поиск командировки по названию;
- генерация отчета о командировке.

Помимо вышеперечисленных функций необходимо предусмотреть возможность расширения. В будущем менеджеру может потребоваться хранение дополнительной информации, а также выполнение других различных функций. Добавление нового функционала не должно сильно влиять на корректную работоспособность старого.

Таким образом разработано задание на проектирование приложения.

#### 1.4 Входные и выходные данные

Информационная база задачи состоит из входной, выходной и постоянной информации.

Входная информация – информация, поступающая в систему в виде данных, внесенных пользователем.

Выходная информация – информация, получаемая в результате выполнения функций или запросов.

Постоянная информация – информация, которая остается неизменной в течение длительного периода времени и многократно используется при обработке переменной информации.

Входные данные разрабатываемого программного средства:

- данные о компании;
- данные о сотрудниках;
- данные о командировках;
- данные о билетах;
- данные об отделах;

- различные запросы пользователя, выполняемые во время использования сайта.

Выходные данные разрабатываемого программного средства:

- результаты выполнения запросов или функций;
- информация, отображаемая на экране.

Постоянные данные разрабатываемого программного средства представлены в виде информации о сайте, а также различные неизменяемые данные, например, роль пользователя, тип транспорта в билете.

### 1.5 Выбор и обоснование средств разработки

Разрабатываемый продукт будет разрабатываться с помощью операционной системы Microsoft Windows 11<sup>[3]</sup>.

Microsoft Windows 11 обладает множеством преимуществ над своими младшими версиями такими, как визуальное оформление, быстроедействие, поддержка новых технологий и тому подобное, что позволяет не только разрабатывать программный продукт в полном соответствии со всеми заданными условиями, но и создать удобный и интуитивно понятный интерфейс программы, что играет немаловажную роль в проектировании информационно-справочных систем.

Программное обеспечение будет написано на java<sup>[4]</sup>.

В java 13<sup>[5]</sup> включены все необходимые компоненты и библиотеки для создания полноценного приложения. Основой для приложения послужит фреймворк Spring boot<sup>[6]</sup>, для интерфейса будет использоваться фреймворк Thymeleaf.

Запросы для манипулирования данными будут написаны на JPQL с использованием фреймворка Spring Data. JPQL — это объектно-ориентированный язык запросов, его особенностью является абстрагирование от структуры БД, запросы создаются на основе java объектов, и с помощью фреймворка автоматически преобразуются в SQL.

Для сборки проекта будет использоваться Gradle<sup>[7]</sup>. Gradle — это инструмент автоматизации сборки для многоязычной разработки программного обеспечения. Он контролирует процесс разработки от задач компиляции и упаковки до тестирования, развертывания и публикации.

В качестве среды разработки приложения будет использоваться IntelliJ IDEA<sup>[8]</sup>. IntelliJ IDEA – это умная и удобная среда разработки для Java, включающая поддержку всех последних технологий и фреймворков. Она предоставляет инструменты для продуктивной работы и идеально подходит для создания коммерческих, мобильных и веб-приложений.

В качестве базы данных будет использоваться PostgreSQL<sup>[9]</sup>. PostgreSQL представляет собой одну из самых распространенных сегодня систем управления базами данных в сети Интернет. Данная система используется для работы с достаточно большими объемами информации, поддерживая при этом высокий уровень производительности. Однако PostgreSQL идеально подходит как для небольших, так и для крупных интернет-проектов.

## 2 Функциональное проектирование

### 2.1 Инфологический этап проектирования БД

Концептуальное (Инфологическое) проектирование технических систем — начальная стадия проектирования, на которой принимаются решения, определяющие последующий облик, и проводится исследование и согласование параметров созданных технических решений с возможной их организацией. Термин «концепция» применяется для описания принципа

действия не только в технических системах, но и в научных, художественных и прочих видах деятельности. Концепт — содержание понятия, смысл. Таким образом, проектирование на концептуальном уровне — на уровне смысла или содержания понятия систем.


Основной объем задач концептуального проектирования относится к ранним стадиям разработки технических систем: при постановке задачи на проектирование, выработке массива вариантов технических и оформительских решений.

Одним из наиболее важных этапов концептуального проектирования является построение диаграммы «сущность-связь»<sup>[10]</sup>.

Модель «сущность — связь» используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Важным свойством модели «сущность — связь» является то, что она может быть представлена в виде графической схемы. Это значительно облегчает анализ предметной области. Существует несколько вариантов обозначения элементов (или сущностей) диаграммы «сущность — связь», каждый из которых имеет свои положительные черты. Элементы представлены в таблице 2.1.

Таблица 2.1 - Элементы диаграммы «сущность — связь».

Обозначение	Значение
	Независимая (сильная) сущность
	Зависимая (слабая) сущность

	Атрибут
	Ключевой атрибут
	Связь

Помимо различных элементов, в диаграмме также определены связи между элементами. Виды связей представлены в таблице 2.2.

Таблица 2.2 - Виды связей.

Обозначение	Значение
	один-к-одному
	один-ко-многим
	многие-ко-многим

Применимо к разрабатываемой задаче можно выделить следующие сущности, представленные в таблице 2.3.

Таблица 2.3 — Описание сущностей

Сущность	Описание
Пользователи	В этой сущности идентифицируются все зарегистрированные пользователи и администраторы.

Учетные данные	В этой сущности идентифицируются данные пользователей необходимые для авторизации.
Роли	В этой сущности идентифицируются роли пользователей.
Пол	В этой сущности идентифицируется пол пользователей.
Контакты	Эта сущность идентифицирует контакты пользователей (номер телефона, email).
Отделы	Эта сущность описывает отделы в компании.
Паспорта	Эта сущность описывает сохраненные паспорта сотрудников.
Участники	Данная сущность описывает пользователей, которые являются участниками командировок.
Командировки	Данная сущность идентифицирует командировки.
Билеты	В этой сущности идентифицируются приобретенные билеты для командировки.

Продолжение таблицы 2.3

Транспорт	Данная сущность описывает типы транспорта, на который был куплен билет.
Страны	Данная сущность описывает страны.
Суточные	В этой сущности идентифицируются расходы на сотрудника в сутки.

Следующим шагом является идентификация связей. Связь – это ассоциация между двумя сущностями, значимая для рассматриваемой предметной области.

Логические зависимости (связи), которые прослеживаются между объектами нашей предметной области, будут как обязательными, так и необязательными. Причем, по характеру объединения данных установленные связи относятся к видам «Один – ко – многим», «Один – к – одному» и «Многие – ко – многим». Описание связи между сущностями представлено в таблице 2.4.

Таблица 2.4 — Описание связи между сущностями

Сущность 1	Связь	Вид связи	Сущность 2
Пользователи	Пользователь может быть участником в нескольких командировках.	Многие ко многим	Командировки
	У пользователя могут быть только одни учетные данные.	Один к одному	Учетные данные

Продолжение таблицы 2.4

Сущность 1	Связь	Вид связи	Сущность 2
	Пользователь может быть только в одном отделе.	Один ко многим	Отделы



Пользователи	У пользователя может быть только один паспорт.	Один к одному	Паспорта
	У пользователя может быть только одна роль.	Один ко многим	Роли
	У пользователя может быть только один пол.	Один ко многим	Пол
	У пользователя может быть несколько контактов.	Многие ко многим	Контакты
Участники	Одному участнику может полагаться несколько суточных	Многие ко многим	Суточные
	У одного участника может быть несколько билетов.	Многие ко многим	Билеты
Билеты	Несколько билетов могут иметь один тип транспорта.	Один ко многим	Транспорт
Суточные	Несколько расходов могут быть прикреплены к одной стране.	Один ко многим	Страны

Последним шагом моделирования является идентификация атрибутов. Атрибуты сущностей будут изображены на конечной диаграмме. Полученная диаграмма «сущность — связь» представлена на рисунке 2.1.



Прямоугольники с закруглениями — действия (операция). Узел управления (control node) — это абстрактный узел действия, которое координирует потоки действий.

Ромбы — решения. Узел решения предназначен для определения правила ветвления и различных вариантов дальнейшего развития сценария. В точку ветвления входит ровно один переход, а выходит — два или более.

Широкие полосы — начало (разветвление) и окончание (схождение) ветвления действий. Узел объединения имеет два и более входящих узла и один исходящий.

Чёрный круг — начало процесса (начальный узел). Начальный узел деятельности (или начальное состояние деятельности) (activity initial node) является узлом управления, в котором начинается поток (или потоки) при вызове данной деятельности извне.

Черный круг с обводкой — окончание процесса (финальный узел). Конечный узел деятельности (или конечное состояние деятельности) (activity final node) является узлом управления, который останавливает все потоки данной диаграммы деятельности. На диаграмме может быть более одного конечного узла.

Диаграмма представлена в приложении А.

### 2.3 Функциональная модель

Диаграмма вариантов использования<sup>[12]</sup> (use case diagrams), предназначенная для моделирования функциональных требований к системе (в виде сценариев взаимодействия пользователей с системой) и отражает функциональные возможности и требования системы с использованием действующих лиц и вариантов использования.

Диаграммы вариантов использования показывают взаимодействия между вариантами использования и действующими лицами, отражая функциональные требования к системе с точки зрения пользователя. Цель

построения диаграмм вариантов использования — это документирование функциональных требований в самом общем виде, поэтому они должны быть предельно простыми.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). В нашем случае действующим лицом (актером), будет выступать пользователь, а именно пользователь, у которого будет открыт доступ к базе данных.

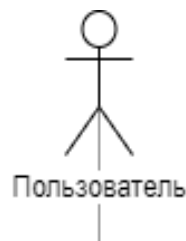


Рисунок 2.2 - Актер

Вариант использования изображается на диаграмме эллипсом внутри, которого содержится его краткое название или имя.



Рисунок 2.3 — Вариант использования

Вариант использования описывает типичное взаимодействие между Пользователем и системой и отражает представление о поведении системы с точки зрения пользователя. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые

он хотел бы реализовать, или целей, которые он преследует по отношению к разрабатываемой системе.

Также есть еще один базовый элемент — операция отношения. Отношение — это семантическая связь между отдельными элементами модели. В нашем случае будем пользоваться только тремя из них:

1. Отношение ассоциации. Оно указывает на то, что актер инициирует соответствующий вариант использования. Пользователь, перед тем перейти к базе данных, запускает программу

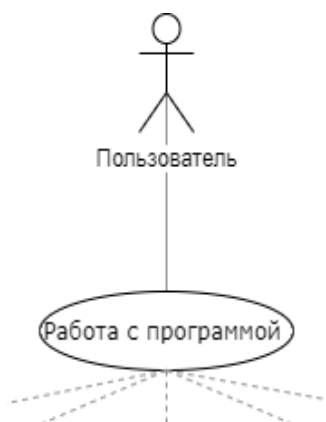


Рисунок 2.4 — Отношение ассоциации

2. Отношение включения означает, что поведение или выполнение одного варианта использования включения в качестве составного объекта в поведение или выполнение другого варианта использования



Рисунок 2.5 — Отношение включения

3. Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров.



Рисунок 2.6 — Отношение расширения

Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Диаграмма вариантов использования представлена в приложении Б.

## 2.4 Диаграмма компонентов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Для представления физических сущностей в языке UML применяется специальный термин – компонент (component). Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели. Для графического представления компонента может использоваться специальный символ – прямоугольник со вставленными слева двумя более мелкими прямоугольниками. Внутри огибающего прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация. Пример компонента изображен на рисунке 2.7.

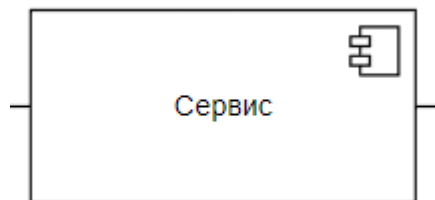


Рисунок 2.7 - Компонент

В языке UML выделяют три вида компонентов:

- компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций: динамически подключаемые библиотеки с расширением dll, Web-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp.
- компоненты-рабочие продукты: файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++.
- компоненты исполнения, представляющие исполнимые модули – файлы с расширением exe.

Следующим элементом диаграммы компонентов являются интерфейс. Этот элемент уже рассматривался ранее, поэтому отметим только его особенности, которые характерны для представления на диаграммах компонентов. В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок. Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

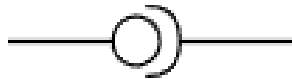


Рисунок 2.8 - Интерфейс

Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой. Пример связи изображен на рисунке 2.9.

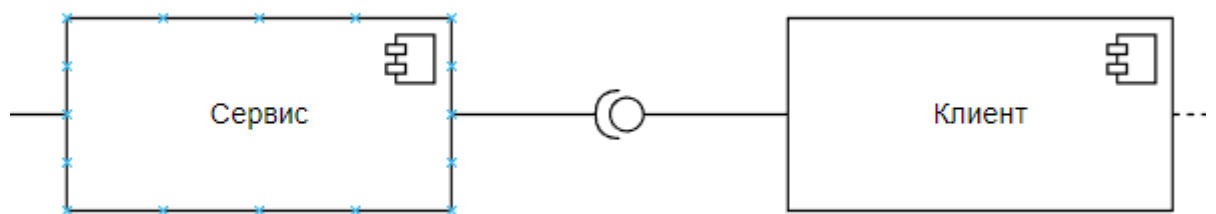


Рисунок 2.9 - Пример связи

Диаграмма компонентов представлена в приложении В.



## 2.5 Диаграмма развертывания

В контексте UML, диаграмма развертывания входит в семейство структурных диаграмм, так как описывает один из аспектов самой системы, а именно — физическое развертывание информации, выработанной программой на аппаратных компонентах. В отношении информации, которую генерирует программа, применяется термин «артефакт».

В состав диаграммы развертывания входит несколько типов фигур UML. Трехмерные блоки, или узлы, символизируют базовые программные или аппаратные элементы системы. Линии, идущие от одного узла к другому, применяются для обозначения связей, а фигуры поменьше, расположенные внутри блоков, — для программных артефактов, которые развертываются на узлах.

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны.

Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который представляет собой сервис, выполняющий другие исполняемые программные элементы. Узлы среды выполнения и устройств представлены на рисунке 2.10.

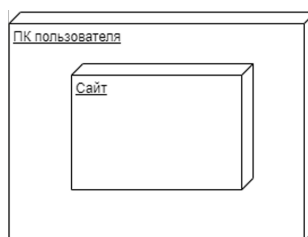


Рисунок 2.10 - Узлы среды выполнения и устройств

Артефакты – это конкретные элементы, которые вызваны процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и т.д. Пример изображения артефакта представлен на рисунке 2.11.

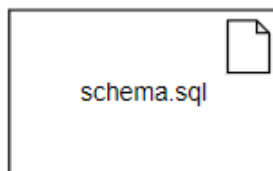


Рисунок 2.11 - Пример артефакта

Путь связи - обозначения схемы развертывания. Это представлено сплошной линией между двумя узлами. Он показывает путь связи между узлами. Пример связи изображен на рисунке 2.12.



Рисунок 2.12 - Пример связи

Диаграмма развертывания представлена в приложении Г.

### 3 Программная реализация

#### 3.1 Структурная схема

Структурная схема сайта<sup>[15]</sup> – это логическое построение всех страниц и разделов ресурса. Благодаря ей пользователь понимает, где располагаются основные разделы сайта, как перемещаться между страницами или вернуться на главную. Визуально структуру можно представить в виде схемы.

Типы структуры сайта в виде схем:

Линейная. Здесь все очень логично устроено – страницы ссылаются друг на друга и одновременно на главную. Этот тип структуры данных сайта отлично подходит для портфолио, презентационных ресурсов и других специфических интернет-площадок, целью которых является последовательное ознакомление пользователей со всем имеющимся контентом.

Линейная с ответвлениями. Этот тип похож на линейный вариант. Только здесь можно применять одновременно несколько продуктов. Самый распространенный пример такой схемы – онлайн-библиотека одного автора с его произведениями. Читатель здесь, опять же, будет двигаться от главной страницы. Для SEO-продвижения эта структура сайтов интернета не подходит.

Блочная. В данном случае каждая страница ссылается на несколько других, равнозначных между собой. Эту конструкцию можно использовать для какого-то определенного продукта, размещая отдельные записи с описанием его преимущества/свойства или совокупностей характеристик.

Древовидная. Универсальная конструкция, которую используют 99 % всех ресурсов. Для каждого направления здесь имеется своя ветка, для каждого товара или услуги – отдельное ответвление.

Записи объединяются в привычные всем разделы и подразделы. В данном случае внимание посетителей концентрируется не только на главной, но и на разделах.

В данном проекте используется древовидная структура сайта. Структура представлена на рисунке 3.1.

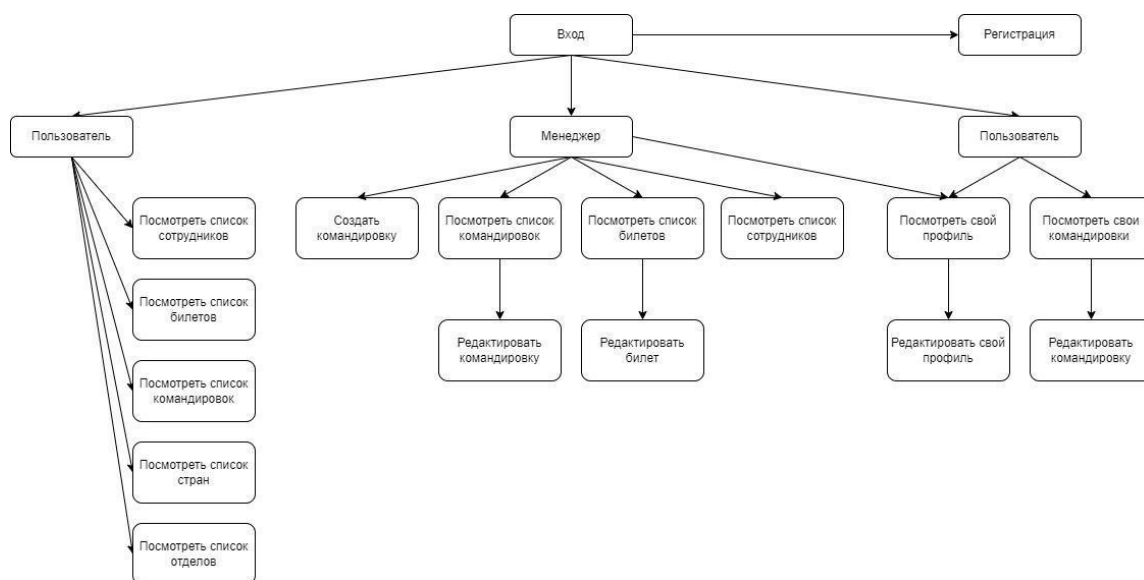


Рисунок 3.1 - Структура сайта

### 3.2 Физическая модель базы данных

На этапе физического проектирования базы данных приводится состав таблиц базы данных. Для каждого поля таблицы необходимо указать используемый тип данных и, размер поля (количество символов), т.е. составить таблицу спецификаций. Для первичных ключей необходимо ввести запрет неопределенных значений. Для остальных полей возможность запрета неопределенных значений определяется семантикой предметной области.

Физическая модель базы данных<sup>[16]</sup> — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных.

На первом этапе происходит выделение сущностей. Сущность (entity) представляет тип объектов, которые должны храниться в базе данных. Каждая таблица в базе данных должна представлять одну сущность. Как правило, сущности соответствуют объектам из реального мира.

У каждой сущности определяют набор атрибутов. Атрибут (элемент данных) - наименьшая единица структуры данных. Обычно каждому элементу при описании базы данных присваивается уникальное имя. По этому имени к нему обращаются при обработке. Элемент данных также часто называют полем.

Каждый столбец должен хранить один атрибут сущности. А каждая строка представляет отдельный объект или экземпляр сущности.

Физическая модель базы данных будет построена на основе диаграммы “сущность-связь”, а также таблицы с описанием атрибутов сущностей. Описание базы данных представлено в таблице 3.1.

Таблица 3.1 — Описание базы данных

Сущность	Атрибут	Тип данных
credentials	user_id (FK)(PK)	int8
	login	varchar
	password	varchar
user	id (PK)	bigserial
	name	varchar
	surname	varchar
	patronymic	varchar
	birthday	timestamp
	gender_id (FK)	int8

	role_id (FK)	int8
	department_id (FK)	int8
role	id (PK)	bigserial
	value	varchar
gender	id (PK)	bigserial
	value	varchar
	fullValue	varchar
contact	id (PK)	bigserial
	user_id (FK)	int8
	value	varchar
	type	varchar

Продолжение таблицы 3.1

department	id (PK)	bigserial
	name	varchar
	description	varchar
passport	user_id (FK)(PK)	int8
	serial	varchar
	number	int8
	personal_number	int8
member	id (PK)	bigserial
	user_id (FK)	int8
	trip_id (FK)	int8
	role	varchar

trip	id (PK)	bigserial
	title	varchar
	description	varchar
	place	varchar
	date_start	timestamp
	date_end	timestamp
	expenses	float8
ticket	id (PK)	bigserial
	member_id (FK)	int8
	from	varchar
	to	varchar
	date	timestamp
	price	float8
	type_id (FK)	int8
transport	id (PK)	bigserial
	value	varchar
country	id (PK)	bigserial
	value	varchar
allowance	id (PK)	bigserial
	value	float8
	country_id (FK)	int8

Продолжение таблицы 3.1

member_allowance	member_id (FK)(PK)	int8
	allowance_id (FK)(PK)	int8
	days	int4
user_login	id (PK)	bigserial
	user_id (FK)	int8
	date	timestamp
	success	bool

На основе концептуальной и логической моделей, была создана физическая модель базы данных, адаптированная под СУБД MySQL. Физическая модель базы данных представлена на рисунке 3.2.

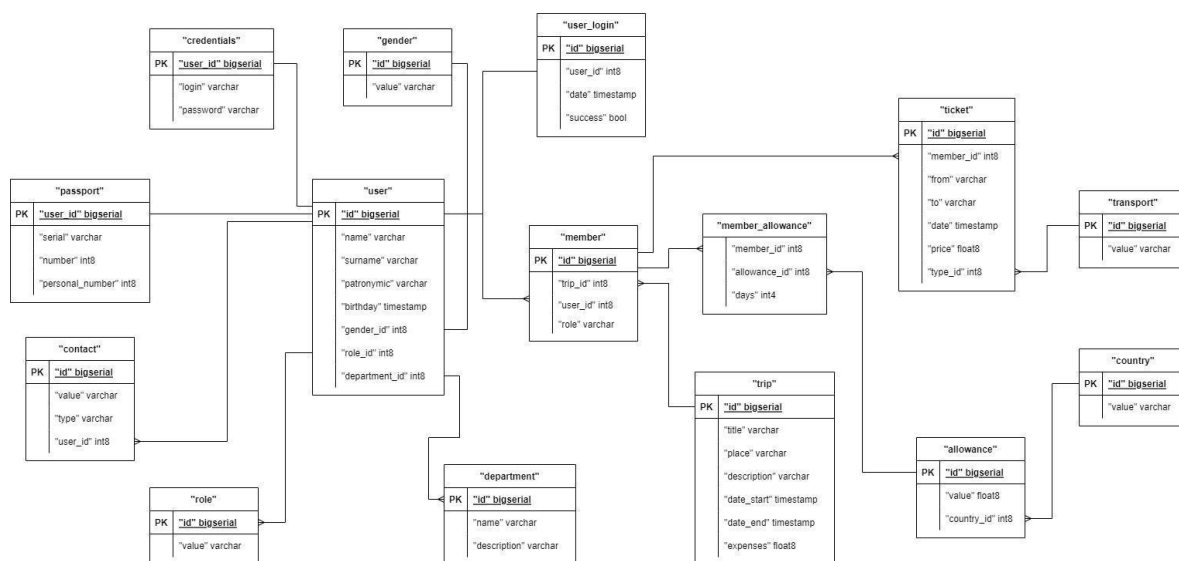


Рисунок 3.2 — Физическая модель базы данных



### 3.3 Интерфейс разрабатываемого приложения

При разработке интерфейса программного средства были использованы фреймворк Thymeleaf<sup>[17]</sup>, язык разметки пользовательского интерфейса HTML, а также для визуального оформления сайта был использован CSS.

Пользовательский интерфейс оснащён всеми необходимыми функциями. В качестве которых: поиск, добавление командировки, сотрудников, билетов и расходов, редактирование и удаление любой информации, создание отчета, а также другие функции.

На рисунке 3.3 представлена страница входа на сайт. Для входа пользователю потребуется ввести логин и пароль от своей учетной записи. В зависимости от роли пользователя будут доступны определенные функции.

Для добавления новых пользователей. Для добавления новых пользователей используется форма регистрации, изображенная на рисунке 3.4.

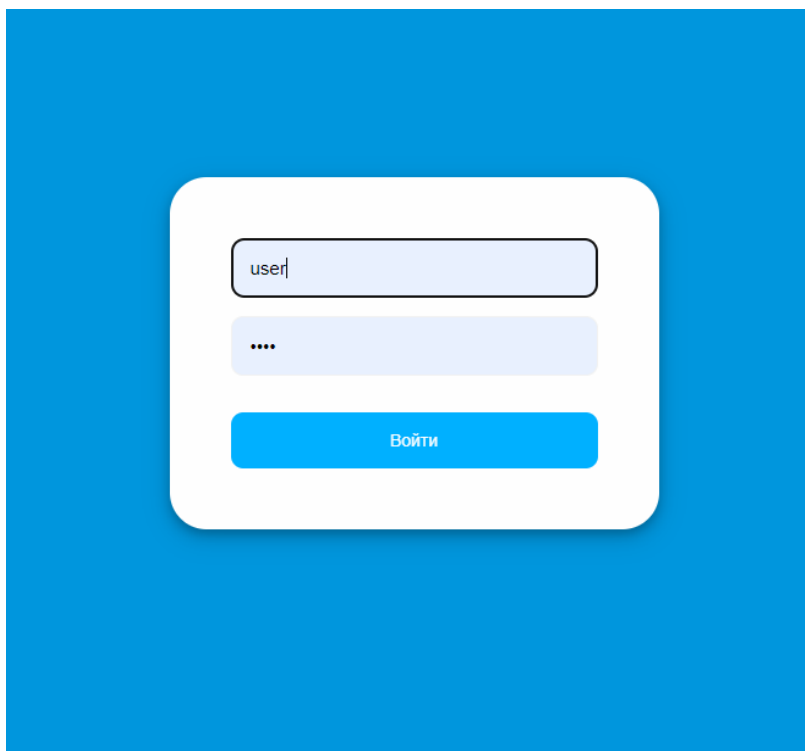
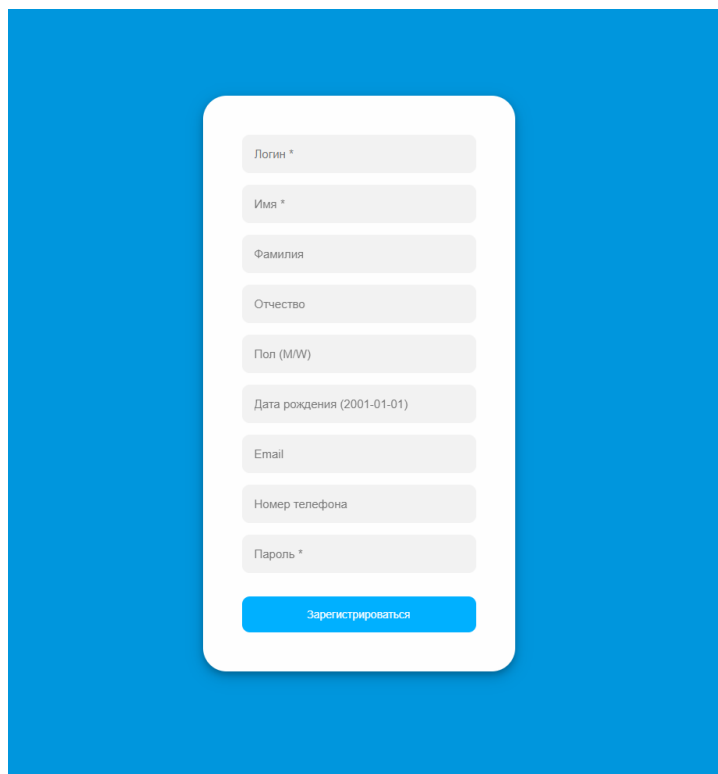


Рисунок 3.3 — Страница входа



Логин \*

Имя \*

Фамилия

Отчество

Пол (M/W)

Дата рождения (2001-01-01)

Email

Номер телефона

Пароль \*

Зарегистрироваться

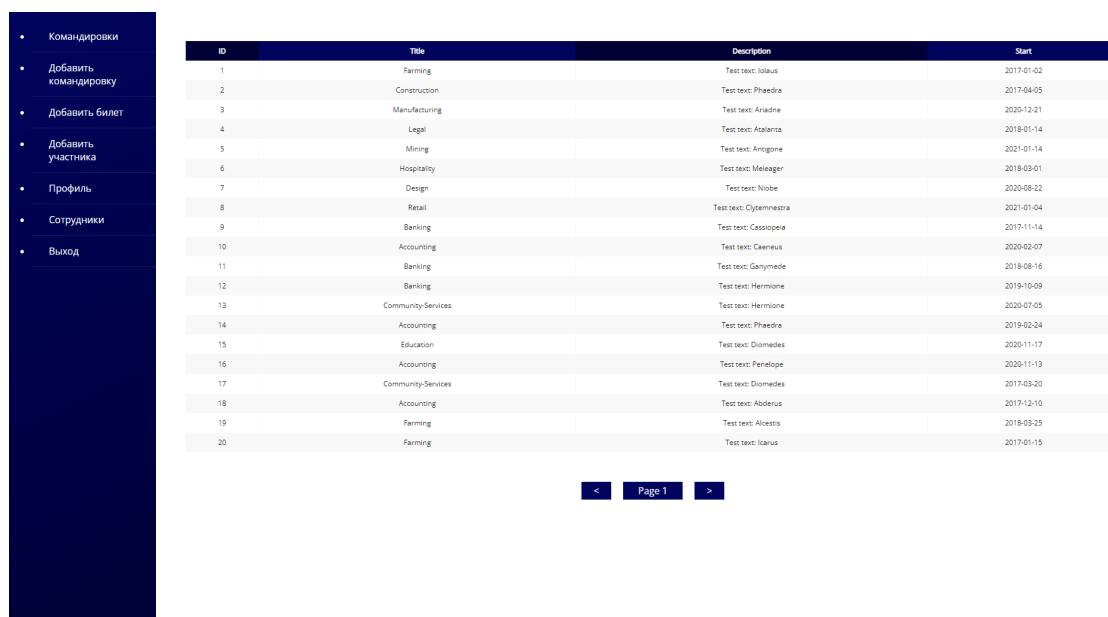
Рисунок 3.4 — Страница регистрации пользователей

На рисунке 3.5 представлена главная страница сайта с уровнем доступа “Администратор”. На ней присутствуют элементы управления с помощью которых можно просматривать, редактировать сотрудников, командировки и т.д. Боковое меню изменяется в зависимости от уровня доступа.

- Командировки
- Сотрудники
- Отделы
- Страны
- Билеты
- Создать отчет
- Выход

Рисунок 3.5 — Главная страница

На рисунке 3.6 представлена страница администратора со списком командировок, на ней присутствуют элементы управления с помощью которых можно выбирать определенные командировки и редактировать информацию о них.

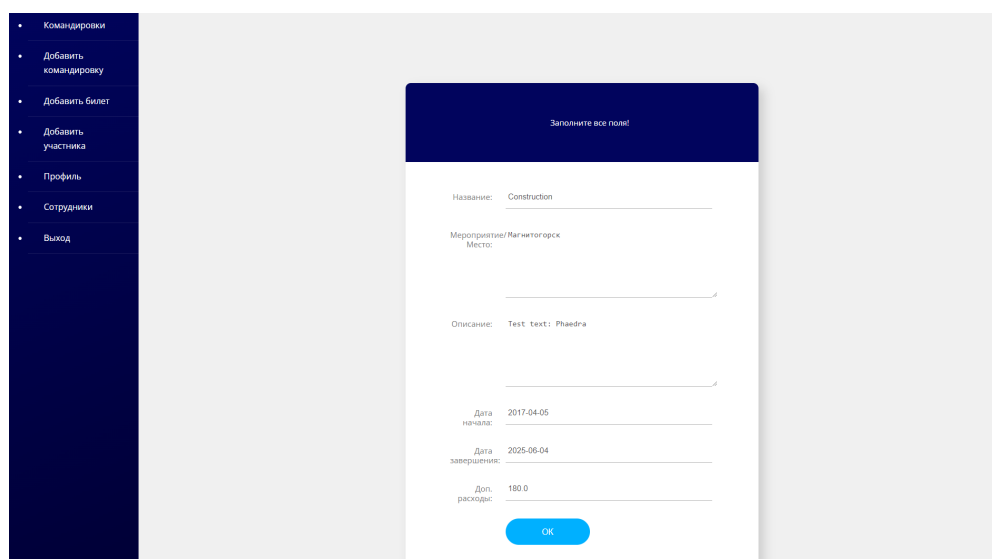


The screenshot shows a web application interface. On the left is a dark blue sidebar with a menu containing the following items: "Командировки", "Добавить командировку", "Добавить билет", "Добавить участника", "Профиль", "Сотрудники", and "Выход". The main area displays a table with 20 rows of business trip data. The table has four columns: ID, Title, Description, and Start. Below the table is a pagination control showing "< Page 1 >".

ID	Title	Description	Start
1	Farming	Test text: Iolaua	2017-01-02
2	Construction	Test text: Phaedra	2017-04-05
3	Manufacturing	Test text: Ariadne	2020-12-21
4	Legal	Test text: Atalanta	2018-01-14
5	Mining	Test text: Antigone	2021-01-14
6	Hospitality	Test text: Meleager	2018-03-01
7	Design	Test text: Niobe	2020-08-22
8	Retail	Test text: Clytemnestra	2021-01-04
9	Banking	Test text: Cassiopeia	2017-11-14
10	Accounting	Test text: Caeneus	2020-02-07
11	Banking	Test text: Ganymede	2018-08-16
12	Banking	Test text: Hermione	2019-10-09
13	Community-Services	Test text: Hermione	2020-07-05
14	Accounting	Test text: Phaedra	2019-02-24
15	Education	Test text: Diomedes	2020-11-17
16	Accounting	Test text: Penelope	2020-11-13
17	Community-Services	Test text: Diomedes	2017-03-20
18	Accounting	Test text: Abderus	2017-12-10
19	Farming	Test text: Alceste	2018-03-25
20	Farming	Test text: Icarus	2017-01-15

Рисунок 3.6 — Страница со списком командировок

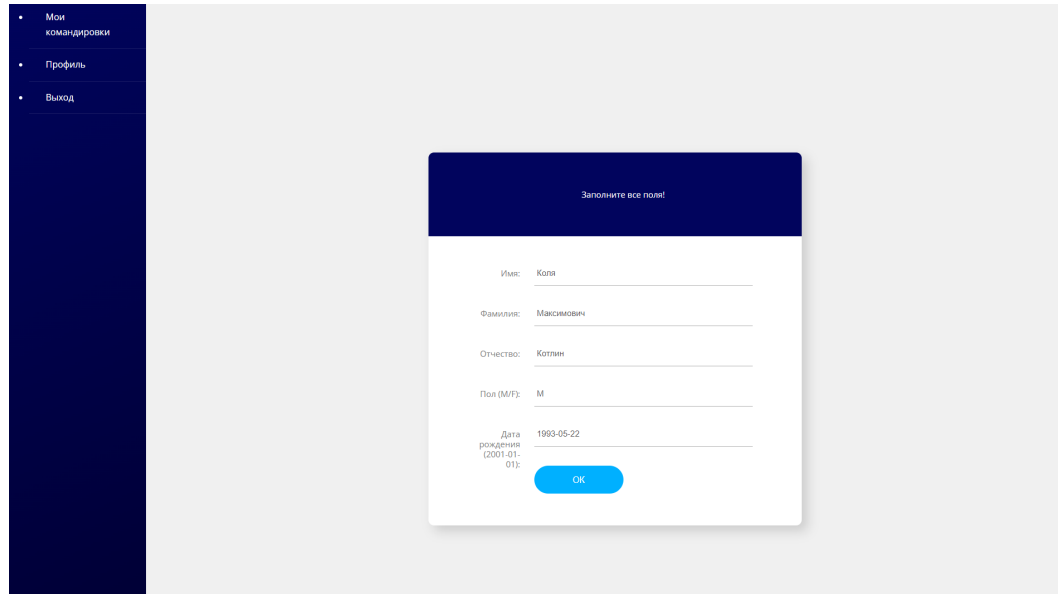
Для добавления, а также редактирования командировки используется форма, изображенная на рисунке 3.7.



The screenshot shows a form for adding or editing a business trip. The form is titled "Заполните все поля!" (Fill in all fields!). It contains the following fields: "Название:" (Name) with the value "Construction", "Мероприятие/Материалное Место:" (Event/Material Place) with a text input field, "Описание:" (Description) with the value "Test text: Phaedra", "Дата начала:" (Start Date) with the value "2017-04-05", "Дата завершения:" (End Date) with the value "2025-06-04", and "Доп. расходы:" (Additional expenses) with the value "180.0". At the bottom of the form is a blue "OK" button.

### Рисунок 3.7 — Страница редактирования командировок

Для редактирования профиля используется форма изображенная на рисунке 3.8.



• Мои командировки

• Профиль

• Выход

Заполните все поля!

Имя: Коя

Фамилия: Максимович

Отчество: Катлин

Пол (М/Ж): М

Дата рождения (YYYY-MM-DD): 1993-05-22

OK

### Рисунок 3.8 — Страница редактирования профиля

## 3.4 Описание алгоритмов работы программы

В ходе разработки проекта было использовано огромное количество различных функций. Многие из них отвечают за извлечение данных из бд, а также отображение этих данных на пользовательском интерфейсе. Одной из наиболее объёмных функций является функция расчета расходов командировки, код функции представлен ниже:

```
public static double calcTicketExpenses(Collection<Member> members) {  
    return members.stream()  
        .mapToDouble(ExpensesCalculator::calcTicketExpensesForMember)  
        .sum();  
}
```

```

public static double calcAllowanceExpenses(Collection<Member> members) {
    return members.stream()
        .mapToDouble(ExpensesCalculator::calcAllowanceExpensesForMember)
        .sum();
}

private static double calcTicketExpensesForMember(Member member) {
    return member.getTickets().stream().mapToDouble(Ticket::getPrice).sum();
}

private static double calcAllowanceExpensesForMember(Member member) {
    int allDays = member.getMemberAllowances()
        .stream().mapToInt(MemberAllowance::getDays).sum();
    double maxDayPrice = member.getMemberAllowances()
        .stream().mapToDouble(ma -> ma.getAllowance().getValue())
        .max().orElse(0);
    return allDays * maxDayPrice;
}

```

Для более подробного описания работы функции будет создана блок-схема, описывающая каждый этап выполнения функции. Блок-схема будет представлена в приложении Д.

## 4 Тестирование

Тестирование ПО — это процесс его исследования с целью получения информации о качестве продукта. Целью тестирования является выявление дефектов в ПО. С помощью тестирования нельзя доказать отсутствие дефектов и корректность функционирования анализируемой программы. Тестирование сложных программных продуктов является творческим процессом, не сводящимся к следованию строгим и четким процедурам.

Дефект — изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию.

Покрытие — это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

Существует большое количество различных видов тестирования. Рассмотрим два основных:

Модульное тестирование. Каждая сложная программная система состоит из отдельных частей - модулей, выполняющих ту или иную функцию в составе системы. Для того, чтобы удостовериться в корректной работе всей системы, необходимо сначала протестировать каждый модуль системы по отдельности. В случае возникновения проблем при тестировании системы в целом это позволяет проще выявить модули, вызвавшие проблему, и устранить соответствующие дефекты в них.

В Java за модульное тестирование отвечают специальные Test классы, которые работают с помощью Junit 5<sup>[18]</sup>. JUnit — это библиотека для модульного тестирования программного обеспечения на языке Java. Принцип работы данной библиотеки заключается в сравнении ожидаемого результата с полученным. Тестирование запускается при компиляции кода чтобы обеспечить работоспособность готового приложения.

С помощью данной библиотеки будет осуществлено тестирование некоторых функций приложения таких как расчет расходов на командировку.

На рисунке 4.1 приведен пример тестирования расчета общей стоимости билетов, а также командировочных расходов. Сущность тестирования заключается в сравнении ожидаемого результата с результатом, полученным с помощью выполнения функции. Тест считается успешным если ожидаемые и актуальные данные сошлись. На рисунке 4.2 изображен результат тестирования метода.

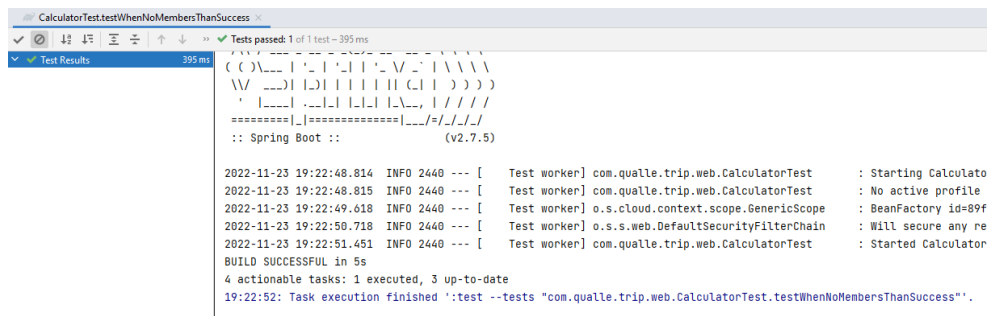
```
@Test
public void testWhenMemberHasTwoTicketsAndTwoAllowancesThanSuccess() {
    List<Member> members = new ArrayList<>();

    Member member1 = Member.builder()
        .id(1L)
        .tickets(List.of(
            Ticket.builder().id(1L).price(100).build(),
            Ticket.builder().id(2L).price(200).build()
        ))
        .memberAllowances(List.of(
            MemberAllowance.builder().days(10).allowance(Allowance.builder().id(1L).value(10).build()).build(),
            MemberAllowance.builder().days(20).allowance(Allowance.builder().id(2L).value(20).build()).build()
        ))
        .build();

    members.add(member1);

    assertEquals( expected: 300d, ExpensesCalculator.calcTicketExpenses(members));
    assertEquals( expected: 600d, ExpensesCalculator.calcAllowanceExpenses(members));
}
```

Рисунок 4.1 – Тестирование расчета расходов



```
CalculatorTest.testWhenNoMembersThanSuccess
Tests passed: 1 of 1 test - 395 ms
Test Results
395 ms
:: Spring Boot :: (v2.7.5)
2022-11-23 19:22:48.814 INFO 2440 --- [ Test worker] com.qualle.trip.web.CalculatorTest : Starting Calculato
2022-11-23 19:22:48.815 INFO 2440 --- [ Test worker] com.qualle.trip.web.CalculatorTest : No active profile
2022-11-23 19:22:49.618 INFO 2440 --- [ Test worker] o.s.cloud.context.scope.GenericScope : BeanFactory id=89f
2022-11-23 19:22:50.718 INFO 2440 --- [ Test worker] o.s.s.web.DefaultSecurityFilterChain : Will secure any re
2022-11-23 19:22:51.451 INFO 2440 --- [ Test worker] com.qualle.trip.web.CalculatorTest : Started Calculator
BUILD SUCCESSFUL in 5s
4 actionable tasks: 1 executed, 3 up-to-date
19:22:52: Task execution finished ':test --tests "com.qualle.trip.web.CalculatorTest.testWhenNoMembersThanSuccess"'.
```

Рисунок 4.2 – Результат тестирования

Путем модульного тестирования приложения были выявлены некоторые непредвиденные ошибки, такие как невозможность рассчитать расходы на командировку при отсутствии каких-либо расходов. В данном случае функция ожидала наличие как минимум одного билета для расчета. При отсутствии билетов функция завершалась с ошибкой

NullPointerException. Путем проверки что количество билетов больше нуля был устранен данный дефект.

Функциональное тестирование - тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции обрабатывали на различных значениях.

Тест-кейс – это описанный алгоритм тестирования программы, специально созданный для определения возникновения в программе определённой ситуации, определённых выходных данных.

В Таблице 4.1 представлено тестирование основных действий пользователя на главной странице, ожидаемый результат и полученный.

Таблица 4.1 – Тестирование функций главной страницы

Действие	Ожидаемый результат	Полученный результат	Вывод
Нажатие на пункт меню “Сотрудники”	Вывод таблицы со всеми сотрудниками компании	Вывод таблицы со всеми сотрудниками компании	+
Нажатие на пункт меню “Командировки”	Вывод таблицы со всеми командировками	Вывод таблицы со всеми командировками	+
Нажатие на пункт меню “Билеты”	Вывод таблицы со всеми билетами	Вывод таблицы со всеми билетами	+
Нажатие на пункт меню “Отделы”	Вывод таблицы со всеми отделами	Вывод таблицы со всеми отделами	+
Нажатие на пункт меню “Страны”	Вывод таблицы со всеми странами	Вывод таблицы со всеми странами	+



Нажатие на пункт меню “Создать отчет”	Вывод формы с выбором командировки	Вывод формы с выбором командировки	+
---------------------------------------	------------------------------------	------------------------------------	---

## 5 Руководство пользователя

При открытии сайта пользователь увидит страницу входа на сайт. На данной странице размещена форма входа. Для входа пользователю необходимо ввести свой логин и пароль, который он указывал при регистрации. Пример формы входа изображен на рисунке 5.1. После нажатия кнопки “Войти” пользователь автоматически перейдет на главную страницу.

Рисунок 5.1 – Форма входа

Если пользователь не зарегистрирован, то ему необходимо перейти по ссылке, полученной от администратора. Будет открыта страница регистрации, где пользователю необходимо будет ввести свои данные.

После входа на сайт будет открыта главная страница, на которой обычный пользователь сможет увидеть свои командировки, а также свой профиль. Пример главной страницы с командировками изображен на рисунке 5.2.

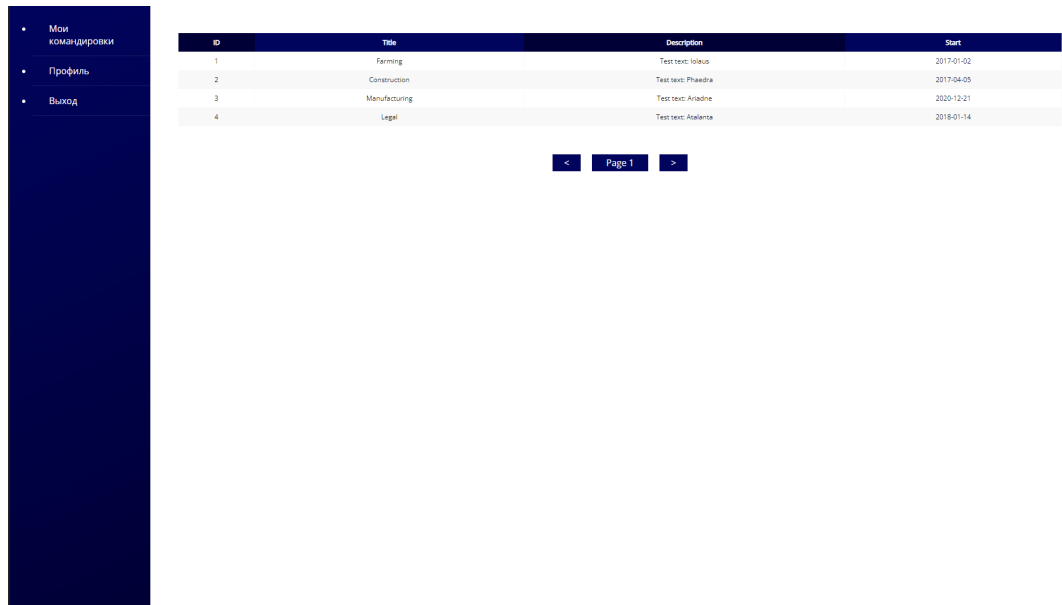


Рисунок 5.2 – Главная страница пользователя

В случае если на сайт входит пользователь с ролью менеджера, то данный пользователь получает доступ к редактированию командировок, а также к просмотру дополнительных данных. Пример главной страницы менеджера изображен на рисунке 5.3.

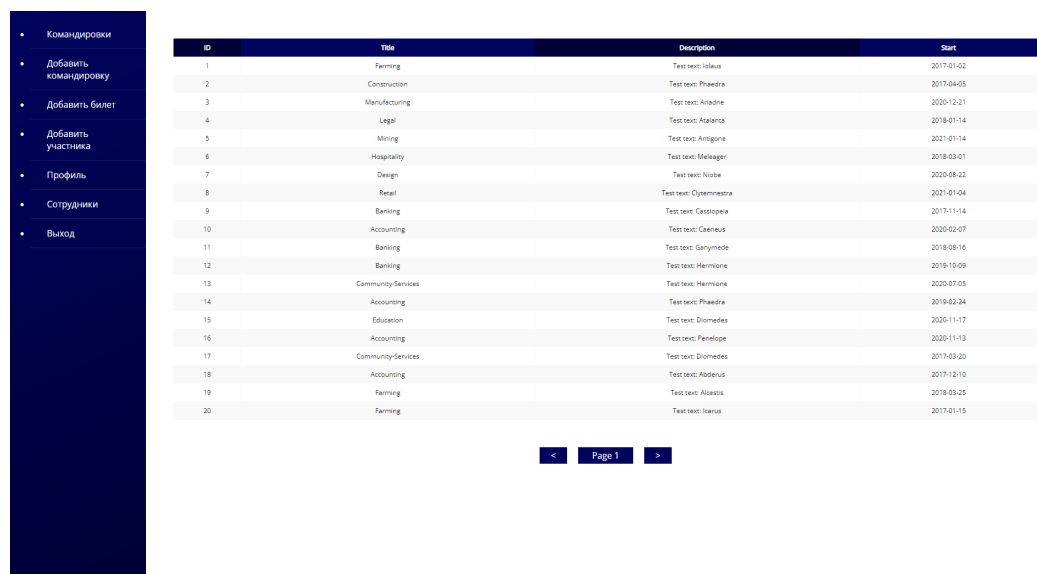
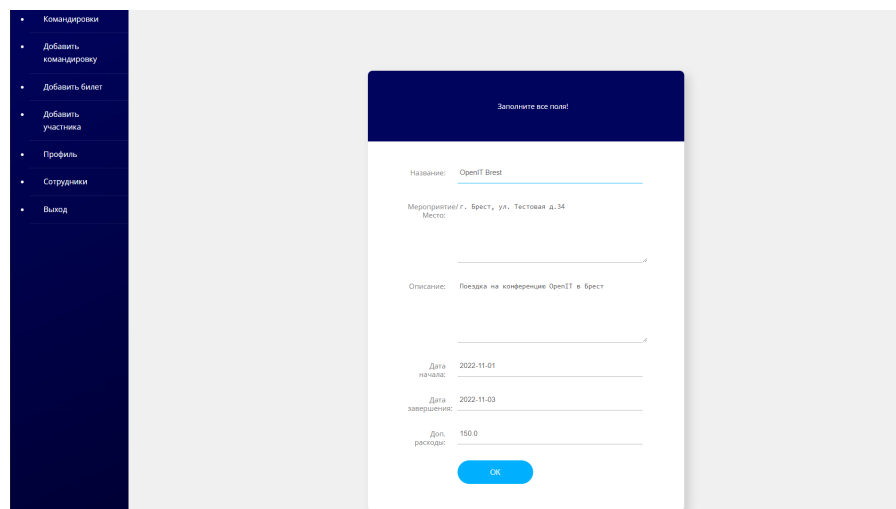


Рисунок 5.3 – Главная страница менеджера

Для создания командировки необходимо нажать на пункт “Добавить командировку” в боковом меню. Будет открыто окно создания командировки, на котором необходимо указать информацию о командировке. Пример создания командировки изображен на рисунке 5.4.



The screenshot shows a web application interface. On the left is a dark blue sidebar menu with white text links: "Командировки", "Добавить командировку", "Добавить билет", "Добавить участника", "Профиль", "Сотрудники", and "Выход". The main area is light gray and contains a white modal form titled "Заполните все поля!". The form has the following fields: "Название:" with the value "ОрелIT West", "Мероприятие/г.: Брест, ул. Тестовая д.34" and "Место:" (empty), "Описание:" with the value "Поездка на конференцию OpenIT в Брест", "Дата начала:" with the value "2022-11-01", "Дата завершения:" with the value "2022-11-03", and "Доп. расходы:" with the value "150.0". A blue "OK" button is at the bottom of the form.

Рисунок 5.4 – Пример создания командировки

После создания командировки необходимо добавить участников командировки. Для этого можно воспользоваться страницей добавления участников. На данной странице необходимо выбрать пользователя, командировку, а также выбрать страну, в которой будет проходить командировка, это необходимо для расчета командировочных расходов, которые также можно указать на данной форме. Пример добавления участника изображен на рисунке 5.5.

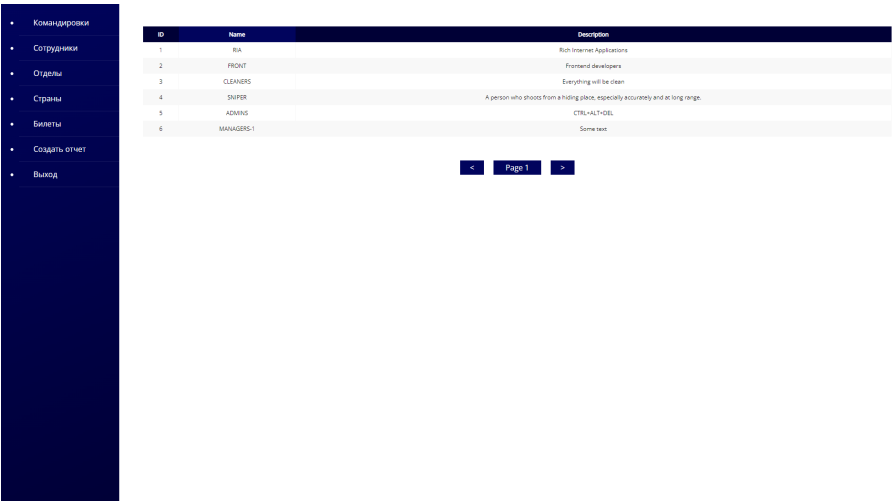
Рисунок 5.5 – Пример создания участника командировки

Также если командировка проходит в другом городе или стране и участнику необходим билет на транспорт, его можно добавить с помощью формы добавления билета. В данной форме необходимо указать участника, командировку тип транспорта, а также цену. После создания билета стоимость будет включена в отчет и соответственно компенсирована сотруднику.

Рисунок 5.6 – Пример создания участника командировки

Пользователю с ролью администратора доступны такие функции как создание отчета, просмотр различных данных таких как доступные страны,

билеты и т.д. Пример главной страницы администратора изображен на рисунке 5.7.



The screenshot displays the administrator interface. On the left is a dark blue sidebar with a menu containing the following items: Командировки, Сотрудники, Отделы, Страны, Билеты, Создать отчет, and Выход. The main content area features a table with the following data:

ID	Name	Description
1	RIA	Rich Internet Applications
2	FRONT	Frontend developers
3	CLEANERS	Everything will be clean
4	SNIPER	A person who shoots from a hiding place, especially accurately and at long range.
5	ADMINS	CTRL+ALT+DEL
6	MANAGERS-1	Some text

Below the table, there are navigation controls: a left arrow, a button labeled "Page 1", and a right arrow.

Рисунок 5.7 – Главная страница администратора

Для создания отчета пользователь может перейти на страницу создания отчета, с помощью соответствующего пункта меню. После выбора необходимой командировки, администратор сможет скачать отчет в формате .docx. Пример созданного отчета изображен на рисунке 5.8.

Начальнику отдела  
кадров ООО  
«ТестМаркет»  
Тестову Т. С.

ОТЧЕТ О КОМАНДИРОВКЕ  
от 23.11.2022 г.

**Период командировки:** 01.11.2022 г. – 03.11.2022 г.  
**Командированные сотрудники:** Коля Максимович, Жанна Власова, Таисия Третьяков, София Петухова.  
**Отдел:** Engineers  
**Цель командировки:** Brest [OpenIT](#)  
**Место проведения командировки:** г. Брест  
**Описание командировки:** Поездка на конференцию [OpenIT](#) в Брест.  
**Расходы на суточные:** 200.0 р.  
**Расходы на билеты:** 761.0 р.  
**Дополнительные расходы:** 236.0 р.  
**Общие расходы (без дополнительных):** 961.0 р.

Старший менеджер ООО «ТестМаркет» Иванов И.И.

Начальнику отдела кадров ООО «ТестМаркет» Тестов И. И.

23.11.2022 г.

Рисунок 5.8 – Пример созданного отчета

## ЗАКЛЮЧЕНИЕ

Целью курсового проекта являлось создание программы позволяющей автоматизировать процесс создания и редактирования информации о командировках. Особенностью данной программы является то, что любую информацию можно поменять в любой момент, а также все запросы на изменение данных выполняются как транзакции. Был использован read committed уровень изолированности транзакций, данный уровень имеет

высокую скорость выполнения и среднюю согласованность данных, предоставляя защиту от чтения незафиксированных изменений другой транзакцией.

Все задачи были выполнены в полном объёме. В процессе выполнения курсового проекта была подробно изучена деятельность предприятия, а также обязанности менеджера, что позволило составить подробные требования к программному продукту. Перед началом разработки были проанализированы схожие по функционалу программы и были выявлены их достоинства и недостатки.

В процессе разработки программного продукта была спроектирована база данных, которая позволила хранить всю необходимую информацию.

Цель курсового проекта достигнута, было разработано приложение позволяющее автоматизировать процесс создания и редактирования информации о командировках. После завершения разработки, программа была полностью протестирована. Также для конечных пользователей разработано подробное руководство по установке и использованию приложения.

Разработанное программное обеспечение соответствует требованиям, и может быть использовано конечными пользователями.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Ozon Командировки [Электронный ресурс]. – Режим доступа: <https://www.ozon.travel/promo/corporate/> – Дата доступа: 10.10.2022.
- [2] Aviasales Командировки [Электронный ресурс]. – Режим доступа: <https://b2b.aviasales.ru/> – Дата доступа: 12.10.2022.
- [3] Microsoft [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/ru-ru/windows> – Дата доступа: 17.10.2022.
- [4] Java документация [Электронный ресурс]. – Режим доступа: <https://docs.oracle.com/en/java/> – Дата доступа: 19.10.2022.
- [5] Руководство Java [Электронный ресурс]. – Режим доступа: <https://www.baeldung.com/> – Дата доступа: 29.10.2022.
- [6] Руководство по Spring Boot [Электронный ресурс]. – Режим доступа: <https://spring.io/guides/> – Дата доступа: 10.11.2022.
- [7] Gradle [Электронный ресурс]. – Режим доступа: <https://docs.gradle.org/current/userguide/userguide.html> – Дата доступа: 12.11.2022.
- [8] JetBrains IntelliJ IDEA [Электронный ресурс]. – Режим доступа: <https://www.jetbrains.com/idea/> – Дата доступа: 13.11.2022.
- [9] Документация по PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/> – Дата доступа: 15.11.2022.
- [10] Модель «сущность — связь» [Электронный ресурс]. – Режим доступа: <https://www.lucidchart.com/pages/ru/erd-диаграмма> – Дата доступа: 17.11.2022.
- [11] Диаграмма деятельности [Электронный ресурс]. – Режим доступа: <https://studizba.com/lectures/diagrammy-deyatelnosti.html> – Дата доступа: 18.11.2022.
- [12] Диаграммы UML [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/UML> – Дата доступа: 20.11.2022.



[13] Диаграммы компонентов [Электронный ресурс]. – Режим доступа: <https://planerka.info/item/diagrammy-komponentov-uml/> – Дата доступа: 20.11.2022.

[14] Диаграммы развертывания [Электронный ресурс]. – Режим доступа: <https://creately.com/blog/ru/uncategorized-ru/> – Дата доступа: 21.11.2022.

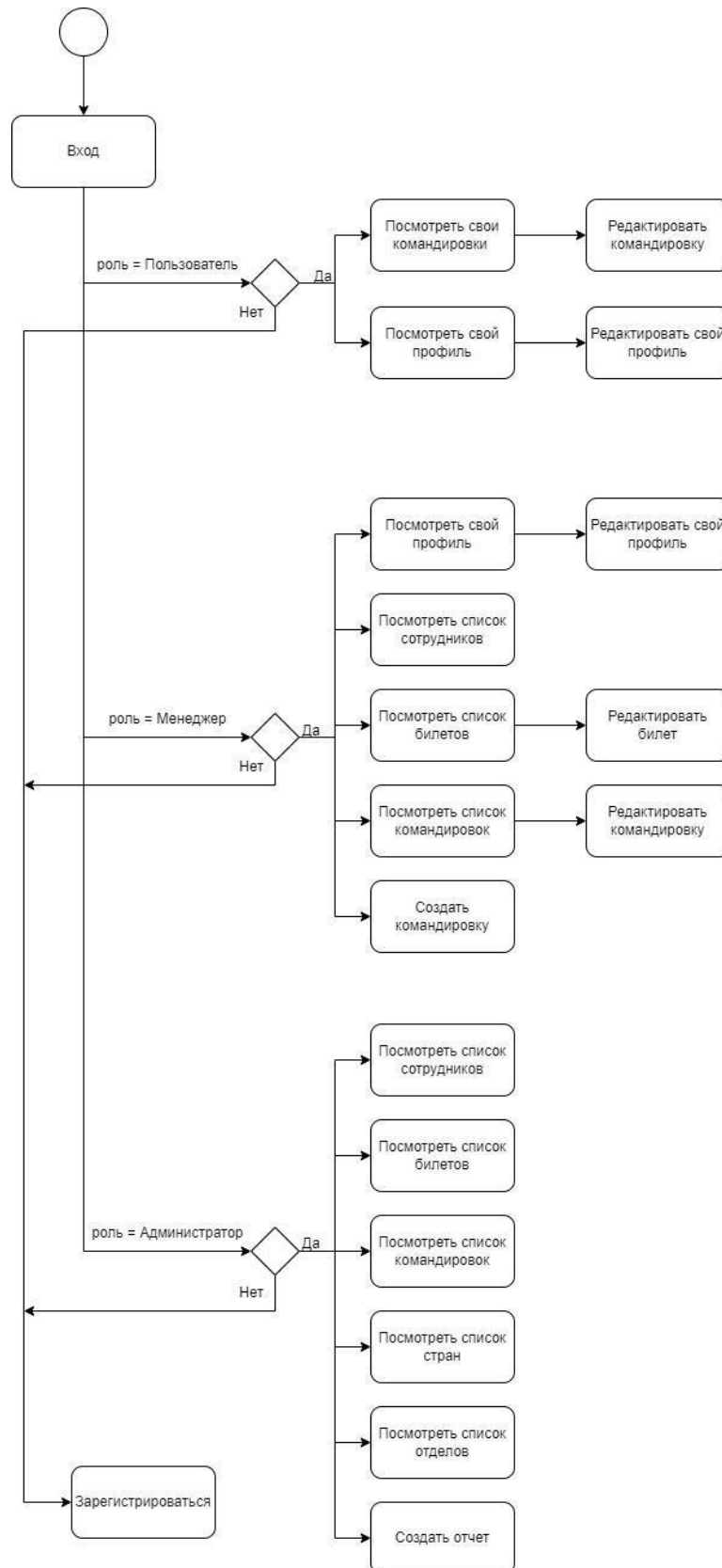
[15] Структурная схема сайта [Электронный ресурс]. – Режим доступа: <https://sales-generator.ru/blog/struktura-sayta/> – Дата доступа: 22.11.2022.

[16] Модель данных [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Model\\_dannich](https://ru.wikipedia.org/wiki/Model_dannich) – Дата доступа: 23.11.2022.

[17] Thymeleaf [Электронный ресурс]. – Режим доступа: <https://www.thymeleaf.org/> – Дата доступа: 24.11.2022.

[18] Junit 5 [Электронный ресурс]. – Режим доступа: <https://junit.org/junit5/> – Дата доступа: 24.11.2022.

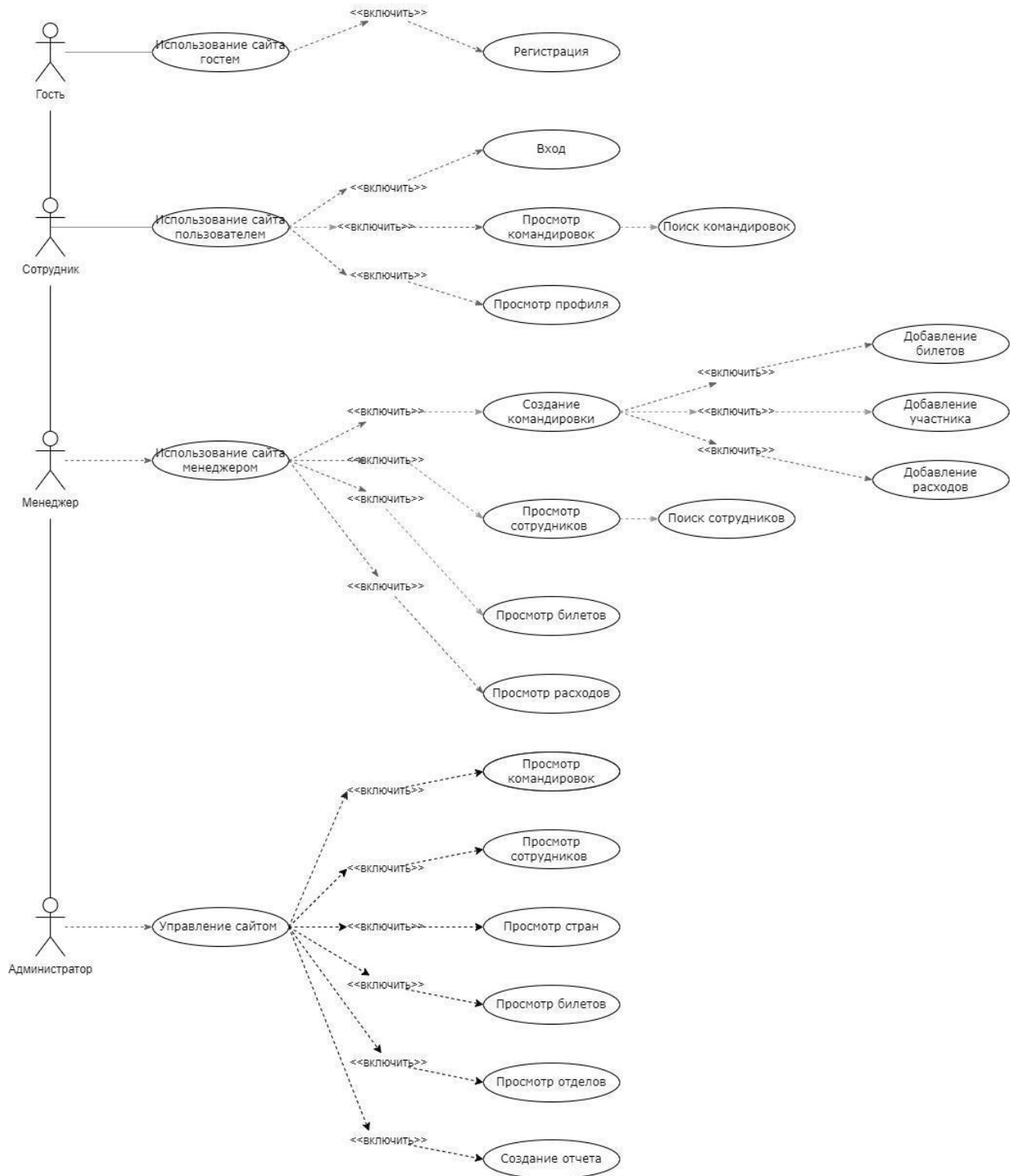
Приложение А  
(обязательное)  
Диаграмма деятельности



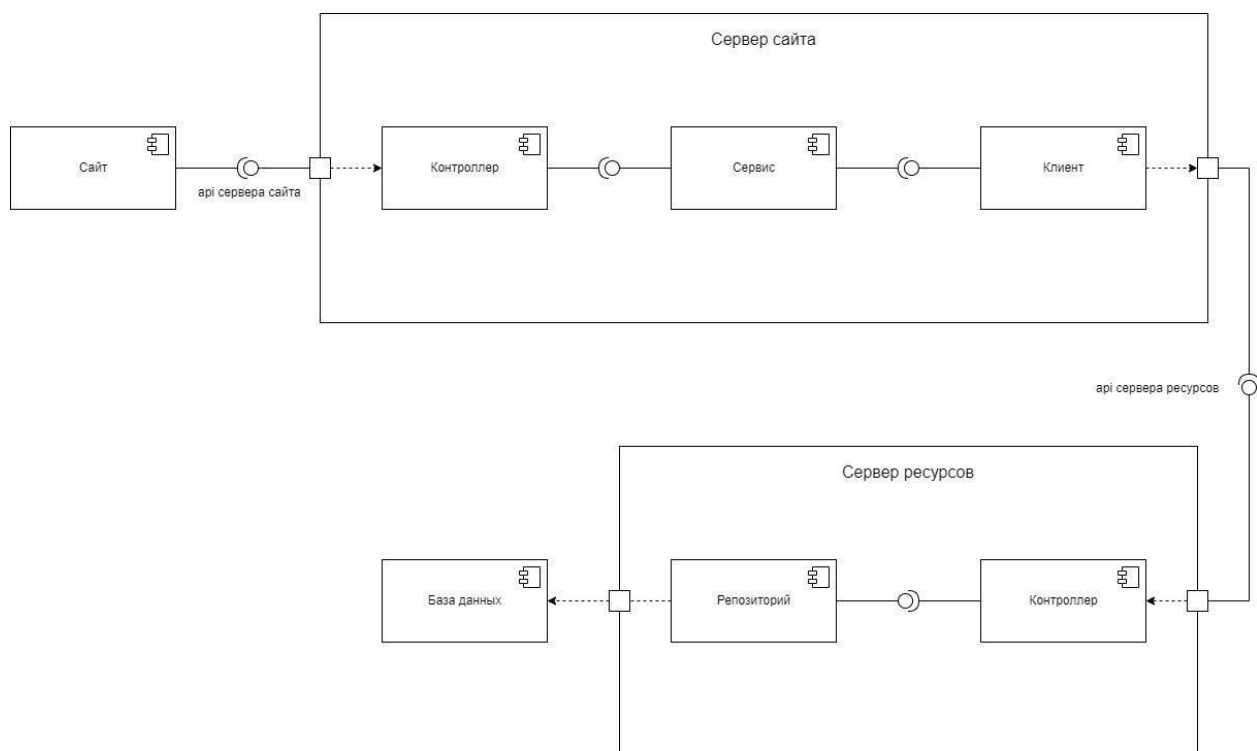
## Приложение Б

(обязательное)

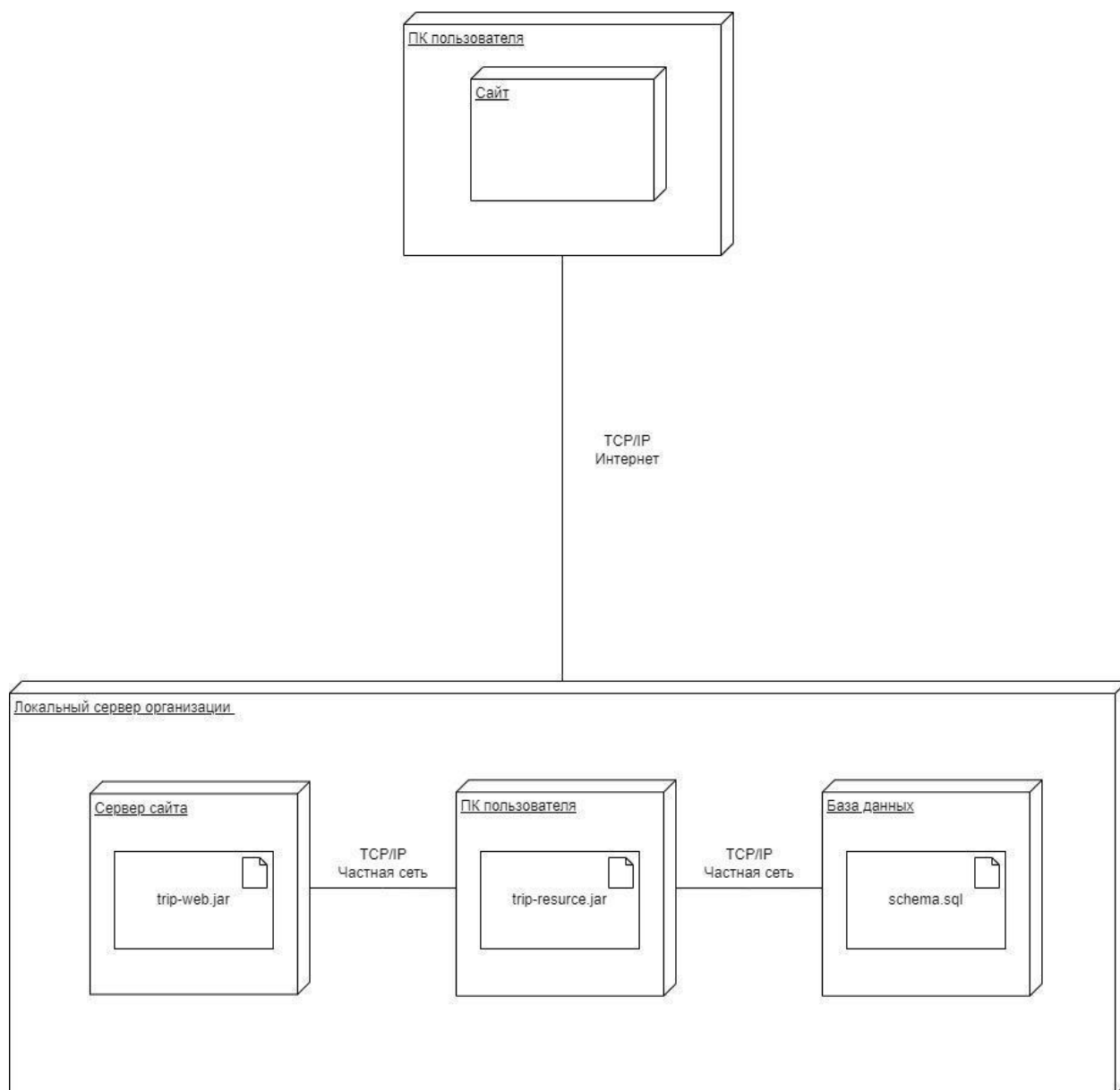
### Диаграмма вариантов использования



Приложение В  
(обязательное)  
Диаграмма компонентов



Приложение Г  
(обязательное)  
Диаграмма развертывания



## Приложение Д

(обязательное)

### Блок схема работы программы

