Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределенные системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры ЭИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПАССАЖИРСКИМИ ПЕРЕВОЗКАМИ НА ЖЕЛЕЗНОДОРОЖНОМ ТРАНСПОРТЕ»**

БГУИР КП 1-28 01 02 016 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  КРУЧОК Вероника Олеговна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| [Введение …………………………………..…………………………………...](#_heading=h.4d34og8) 3  [1 Анализ и моделирование предметной области программного средства ..](#_heading=h.2s8eyo1) 4  [1.1 Описание предметной области …………………….…………………..](#_heading=h.17dp8vu) 4  [1.2 Разработка функциональной модели предметной области ……...…..](#_heading=h.2et92p0) 5  [1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований …………………………8](#_heading=h.2et92p0)  [1.4 Разработка информационной модели предметной области ………....10](#_heading=h.2et92p0)  [1.5 Модели представления программного средства и их описание ……. 1](#_heading=h.2et92p0)0  [2 Проектирование и конструирование программного средства …….…....](#_heading=h.3rdcrjn) 14  [2.1 Постановка задачи ……………………………………………………..14](#_heading=h.17dp8vu)  [2.2 Архитектурные решения ……………………………………..……… 14](#_heading=h.2et92p0)  [2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства ………………………….](#_heading=h.2et92p0) 15  [2.4 Проектирование пользовательского интерфейса …………………….](#_heading=h.2et92p0) 17  [2.5 Обоснование выбора компонентов и технологий для реализации программного средства ……………………………………………….](#_heading=h.2et92p0) 17  [Приложение А Блок-схема……………………………………….…….…… 19](#_heading=h.44sinio) |  |

# **ВВЕДЕНИЕ**

В настоящее время автоматизация транспортных систем стала неотъемлемой частью развития экономики и повседневной жизни людей. Железнодорожный транспорт не является исключением, и в последнее время наблюдается увеличение спроса на автоматизированные системы управления пассажирскими перевозками.

Для достижения цели курсового проекта было решено создать веб приложение.

В настоящий момент отрасль веб разработки является одной из самых востребованных в IT. Доступность интернета ежегодно возрастает, а вместе с этим и востребованность отрасли веб разработки растёт изо дня в день.

Современная веб разработка охватывает широкую область применений. Социальных сети, мессенджеры, электронная коммерция, онлайн магазины, облачные сервисы, игровые сервисы – всё это реализуемо в рамках области веб разработки. Зачастую в данной отрасли задействованы отдельные студии или самозанятые разработчик специализирующиеся на создании и проектировании веб-сервисов под заказ.

Целью данного курсового проекта является разработка автоматизированной системы управления пассажирскими перевозками на железнодорожном транспорте, которая будет улучшать эффективность и надежность перевозок, сокращать время в пути и обеспечивать безопасность пассажиров.

Для разработки данного клиент-серверное приложение необходимо решить следующие задачи: выполнить анализ предметной области, выбрать инструменты разработки, спроектировать и реализовать клиент-серверное приложение, провести тестирование.

В результате выполнения данного курсового проекта будет разработана прототип автоматизированной системы управления пассажирскими перевозками на железнодорожном транспорте, который может быть использован в реальных условиях эксплуатации.

# **АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

* 1. **Описание предметной области**

Железнодорожный транспорт является одним из наиболее распространенных и важных видов транспорта в мире, который обеспечивает перевозку грузов и пассажиров на большие расстояния. Однако, как и любая другая система, железнодорожный транспорт нуждается в постоянной оптимизации и улучшении, чтобы соответствовать современным требованиям быстроты, эффективности, безопасности и комфорта для пассажиров.

Автоматизированная система управления пассажирскими перевозками на железнодорожном транспорте включает в себя комплекс технических средств и программных решений, позволяющих управлять движением поездов, обеспечивать безопасность пассажиров и персонала, контролировать процесс продажи и бронирования билетов, а также осуществлять мониторинг и управление всеми техническими и технологическими процессами, связанными с перевозками.

Одной из основных целей автоматизации железнодорожных перевозок является повышение эффективности и экономической эффективности системы путем сокращения времени на подготовку рейсов, улучшения качества обслуживания пассажиров и уменьшения затрат на эксплуатацию и обслуживание оборудования.

Цифровая трансформация во всем мире не могла пройти мимо железнодорожных перевозок. Наличие автоматизированных систем для работников становиться обязательными, ведь это позволяет вести удобную документацию и позволяет эффективно работать с огромными объемами информации.

Сейчас на рынке программного обеспечения для бизнеса представлено немало различных *CRM* систем (сокр. «*customer relationship management*») для автоматизированного управления деятельностью. Подобные системы представляют собой компьютерные программы, устанавливаемые на персональные компьютеры пользователей или предоставляемые в аренду в виде облачных веб-приложений, выполняющие все основные операции, с которыми имеют дело работники железной дороги.

Внедрить необходимую автоматизированную систему в компании можно, используя различные способы приобретения.

Различают несколько способов приобретения информационных систем:

* разработка, то есть способ, подразумевающий проектирование и программирование собственными силами средствами компании;
* покупка готового решения, то есть один из способов приобретения ИС, который подразумевает покупку готовой ИС у компании, разрабатывающей и/или реализующей разного рода ИС, и ее использование;
* покупка готового решения и его доработка, то есть один из способов приобретения ИС, который подразумевает покупку готового ядра требуемой ИС у компании, разрабатывающей и/или реализующей разного рода ИС, для того, чтобы внести в продукт свои значительные изменения и только, после этого, использовать.

Самостоятельное изготовление АРМ – самый дешевый способ. Имеет как преимущества, так и недостатки. Преимущества в том, что исключены лишние затраты на оплату готовых решений, проект делается исключительно для своих нужд и своими силами – что увеличивает приспособленность системы под бизнес-процессы предприятия. Недостаток же только в том, что надежность и время создания системы своими силами зависит от компетенции и профессионализма программистов.

Создание системы силами собственных сотрудников гарантирует: уникальность кода, собственные методы защиты от взлома, создание оптимизированных участков кода, масштабируемость системы в будущем, так как собственную программу человеку менять намного легче, чем разбираться в чужой программе, сопровождение и улучшение программы возможно все время, пока на фирме есть хоть один ИТ-сотрудник, обладающий соответствующими знаниями.

Обычно в число основных функций, которые должны быть представлены в CRM системах, входят следующие функции:

* управление клиентами;
* управление продажами;
* управление бухгалтерией;
* управление документооборотом;
* управление аналитикой и отчетами.
  1. **Разработка функциональной модели предметной области**

Для моделирования процесса работы системы управления пассажирскими перевозками воспользуемся методологией *SADT* (*IDEF0*). Данная методология при описании функционального аспекта информационной системы конкурирует с методами, ориентированными на потоки данных (*DFD*). В отличие от них *IDEF0* позволяет описывать любые системы, а не только информационные (*DFD* предназначена для описания программного обеспечения) и создавать описание системы и ее внешнего окружения до определения окончательных требований к ней. Иными словами, с помощью данной методологии можно постепенно выстраивать и анализировать систему даже тогда, когда трудно еще представить ее воплощение.

Таким образом, *IDEF0* может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению.

Основу методологии *IDEF0* составляет графический язык описания процессов. Модель в нотации *IDEF0* представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать 4 типа диаграмм:

* контекстную диаграмму;
* диаграммы декомпозиции;
* диаграммы дерева узлов;
* диаграммы только для экспозиции (*FEO*).

Контекстная диаграмма (или же диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т.е. определяются функции, из которых состоит основная.

Далее функции делятся на подфункции и так до достижения требуемого уровня детализации исследуемой системы. Диаграммы, которые описывают каждый такой фрагмент системы, называются диаграммами декомпозиции. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных процессов созданным диаграммам. Найденные несоответствия устраняются, после чего приступают к дальнейшей детализации процессов.

Диаграмма дерева узлов показывает иерархическую зависимость функций (работ), но не связи между ними. Их может быть сколько угодно, поскольку дерево можно построить на произвольную глубину и с произвольного узла.

Диаграммы для экспозиции строятся для иллюстрации отдельных фрагментов модели с целью отображения альтернативной точки зрения на происходящие в системе процессы (например, с точки зрения руководства организации).

Контекстная диаграмма работы системы управления пассажирскими перевозками приведена на рисунке 1.1.

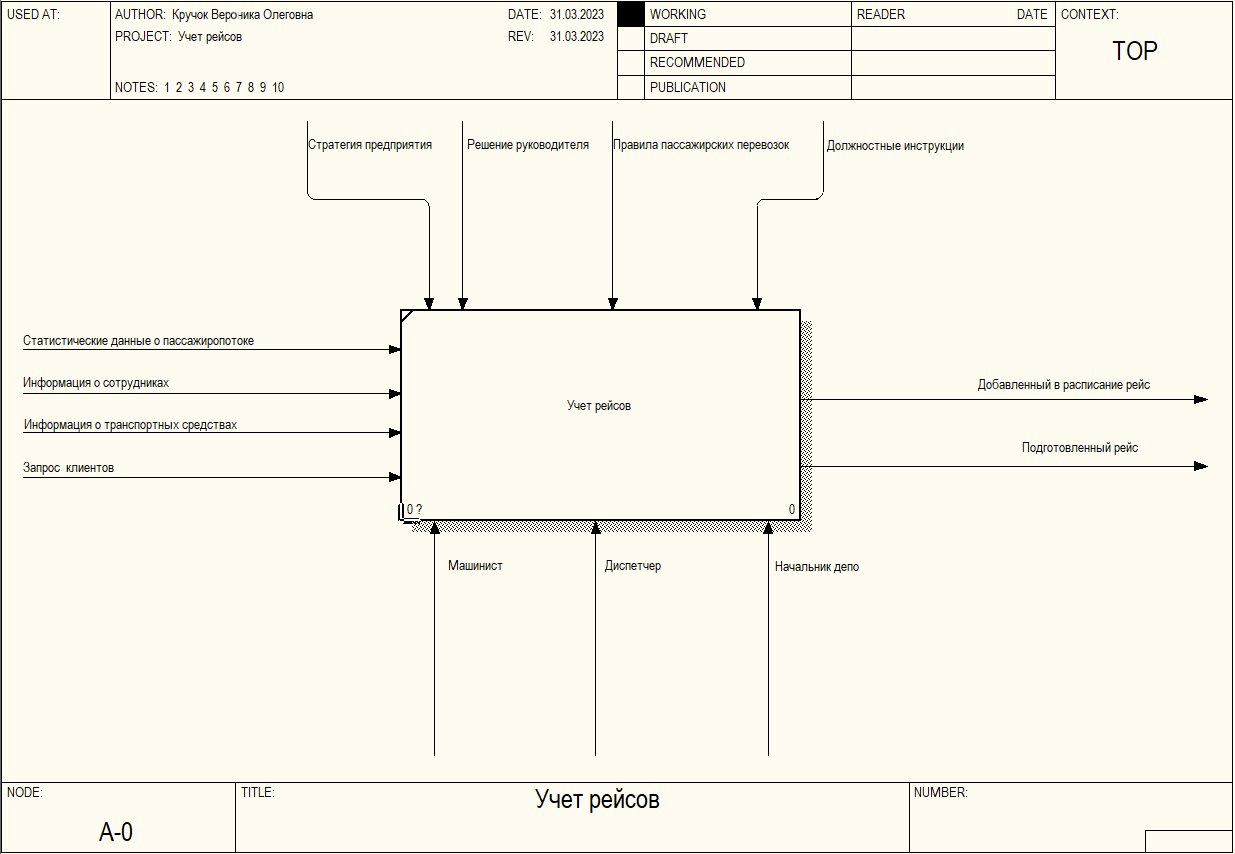


Рисунок 1.1 – Контекстная диаграмма верхнего уровня

Входными данными является информация о клиентах, сотрудниках и транспортных средствах. Результатом работы программы будет являться добавление рейса в расписание, подготовка рейса. В качестве механизма выступают машинист, диспетчер, начальник депо. Управлением будут являться стратегия предприятия, решение руководителя, правила пассажирских перевозок, должностные инструкции.

После декомпозиции верхнего уровня диаграммы процесс раскладывается на следующие этапы:

* формирование запроса на рейс;
* назначение транспортного средства;
* назначение машинист.

Эти этапы представлены на рисунке 1.2.



Назначение машиниста

Рисунок 1.2 - Декомпозиция контекстной диаграммы «Учет рейсов»

На рисунке представлена работа функционального блока, выполненная при помощи декомпозиции и построена диаграмма второго уровня. На ней представлено три функциональных блока декомпозиции, распределение входных данных и управляющих механизмов.

В итоге *IDEF0* предоставляет преобразование одного сложного процесса на систему подпроцессов.

* 1. **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Функциональные требования определяют, какие функции должно выполнять программное обеспечение, чтобы пользователи могли выполнить свои задачи в соответствии с бизнес-требованиями. Их еще называют требованиями поведения, так как они описывают, что система должна делать. Примером такого требования может быть отправка пользователю подтверждения о заказе по электронной почте. Для анализа требований часто используют диаграммы *Use Case* языка *UML*, которые являются важным инструментом в современной разработке программного обеспечения.

Диаграммы вариантов использования (*Use Case*) описывают, как различные варианты использования ПО взаимодействуют между собой и зависят от участников, которые участвуют в процессе. Они представляют собой графическое описание функций и возможностей, которые ПО предоставляет для решения задач пользователей. Использование диаграмм вариантов использования помогает лучше понять требования пользователей и способствует более эффективной разработке программного обеспечения.

В данном курсовом проекте актерами будут являться сотрудники железнодорожной компании всех должностей, работа которых будет автоматизироваться с помощью данного курсового проекта, а именно: администратор, диспетчер, начальник депо, машинист. У всех сотрудников есть общие возможности, поэтому они состоят в отношении обобщения с актёром «Пользователь». Этому актёру доступны следующие базовые возможности: авторизация, смена пароля, выход из аккаунта, просмотр данных о своём аккаунте, просмотр ближайших рейсов, просмотр текущей даты и времени.

Актёр «Администратор» введён в первую очередь для управления сотрудниками железнодорожной компании. Он может зарегистрировать пользователя (было принято решение не делать саморегистрацию пользователя, так как приложение создаётся для внутреннего пользования), просмотреть список пользователей, редактировать данные пользователя, в том числе и его заблокировать (данный узел находится в отношении расширения («*extend*») с узлом «Редактирование cписка пользователей»). Для узла «Просмотр списка пользователей» предусмотрен узел «Формирование графика распределения сотрудников по должностям», находящийся с ним в отношении расширения («*extend*»).

Актёру «Диспетчер» доступен функционал для работы с рейсами. Ему доступны функции просмотра, добавления и редактирования рейсов. Также этот актёр занимается ведением списка остановочных пунктов и составлением маршрутов.

Актёр «Машинист» с помощью приложения будут иметь возможность посмотреть свои рейсы. В качестве фильтра для этого списка выступает поле «Номер рейса». Для облегчения работы водителя предусмотрена возможность просмотра маршрута каждого рейса на карте.

Актёр «Начальник депо» управляет парком транспортных средств, находящихся в распоряжении компании. Он имеет возможность просмотреть список всех автобусов, зачислять их на баланс организации и редактировать данные о каком-то из них.

Для каждого узла, содержащего в названии слово «Просмотр», предусмотрены узлы «Сохранение в файл», «Сортировка» и «Поиск», находящиеся с ним в отношении расширения.

Всего на диаграмме представлен двадцать восемь вариантов использований от четырёх видов актёров.

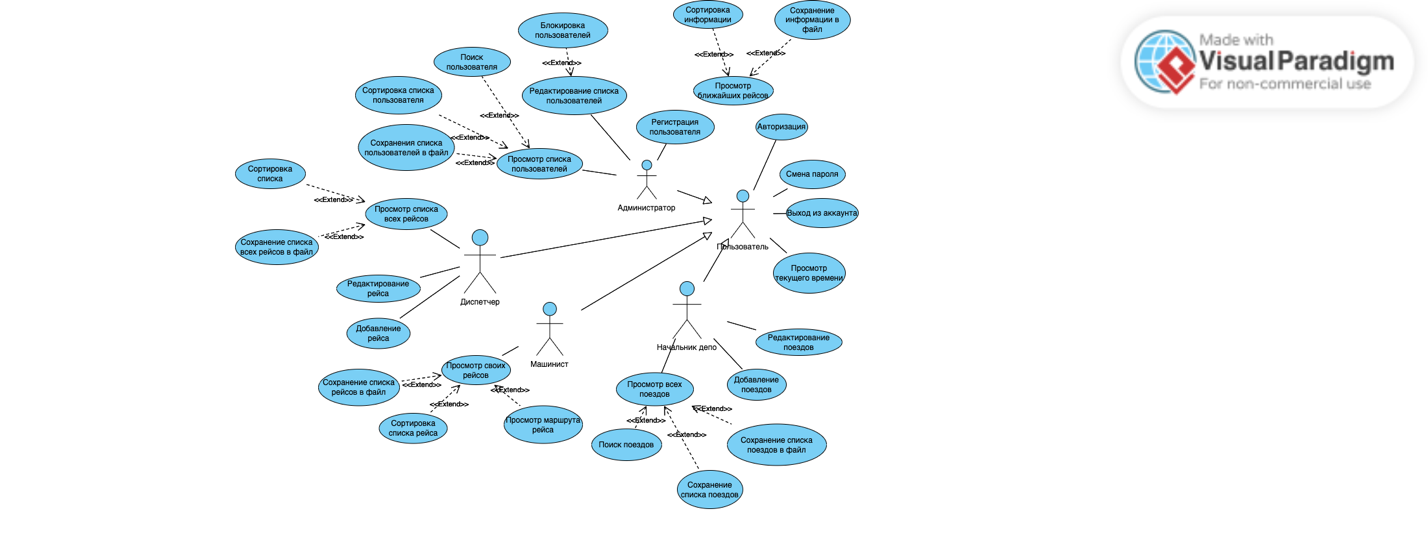


Рисунок 1.3 – Диаграмма вариантов использования

* 1. **Разработка информационной модели предметной области**

Информационная система – это компьютерная система, которая собирает, хранит, обрабатывает и передает информацию для управления. Она предоставляет работникам необходимую информацию о предметной области, которая должна храниться в базе данных. Для обеспечения эффективности и минимизации объема данных, информационная модель должна быть приведена к нормальной форме. Это помогает обеспечить надежное хранение и обновление данных, чтобы снизить риск потери или искажения информации при внесении ее в базу данных.

Проектирование информационной системы включает три этапа: концептуальное, логическое и физическое проектирование. Концептуальное проектирование заключается в анализе предметной области и ее описании. Логическое проектирование описывает логическую структуру данных, используя систему управления базами данных (СУБД), для которой проектируется база данных. Этот этап основывается на концептуальной модели и включает описание таблиц, связей между таблицами и атрибутов.

* 1. **Модели представления программного средства и их описание**

Модель программного обеспечения представляет собой структурированное описание системы на абстрактном уровне, которое включает в себя набор диаграмм и документов определенного формата. Каждая модель фокусируется на конкретном аспекте системы и используется разными людьми с различными интересами, ролями и задачами. Одна из таких диаграмм – диаграмма последовательности – наглядно показывает временной аспект взаимодействия между сущностями системы. Она использует два измерения: слева направо – для указания порядка вовлечения экземпляров сущностей во взаимодействие, и сверху вниз – для указания порядка обмена сообщениями. Она не указывает масштаб времени и отображает только упорядоченность взаимодействия по времени.

Диаграмма последовательности данного курсового проекта представлена на рисунке 1.4.

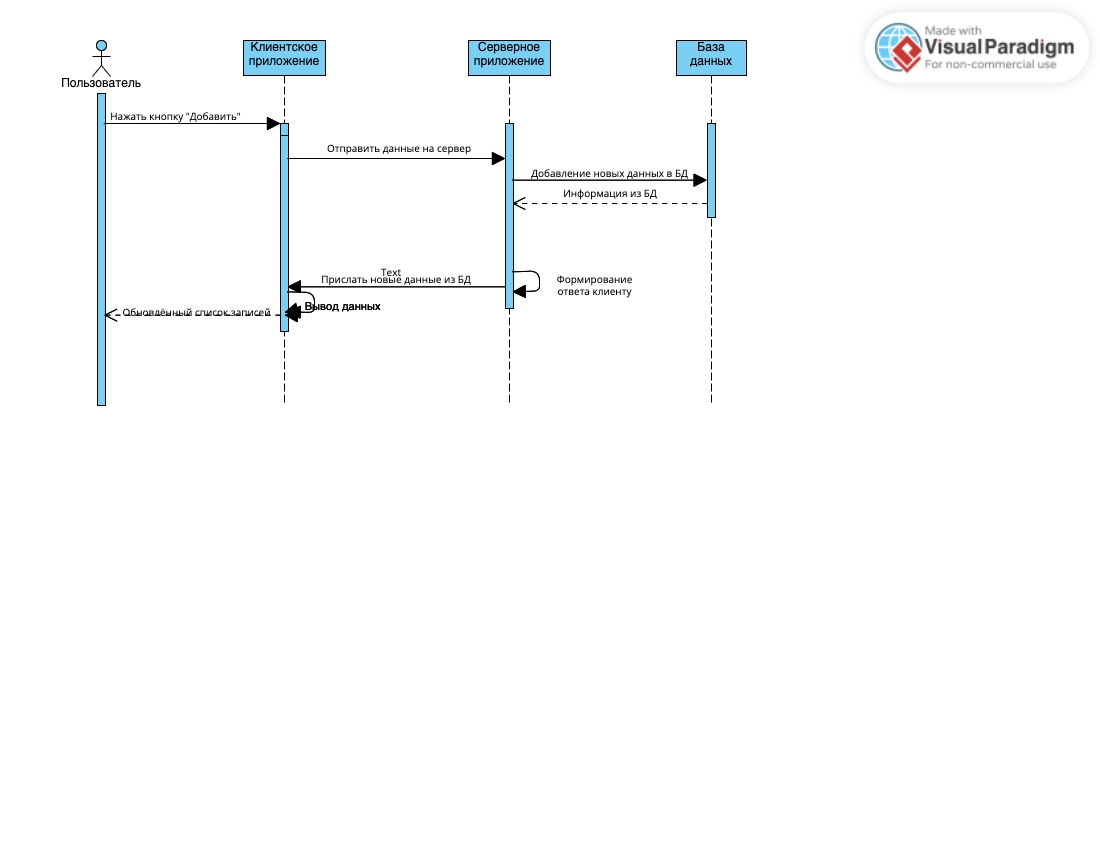


Рисунок 1.4 – Диаграмма последовательности

На диаграмме состояний, изображённой размещённой в приложении, отражены состояния, в которых может находиться пользователь при прохождении аутентификации. При запуске клиентского веб-приложения открывается страница авторизации. В этот момент приложение переходит в состояние ожидания ввода учётных данных пользователя. При любых изменениях в полях ввода информации происходит валидация введенных данных. При условии нажатия на кнопку «*Login*» происходит переход в состояние отправки данных для аутентификации. На входе в это состояние данные преобразуются в формат *JSON*, после формируется объект запроса. Если отправленные данные были корректны, то клиентское приложение получит данные пользователя и отобразит главное окно приложения. Если данные некорректны, то на выходе из этого состояния сформируется объект ответа, произойдёт переход к начальному состоянию и отобразится сообщение об ошибке. При нажатии на кнопку «закрыть» работа с веб-приложением завершится.

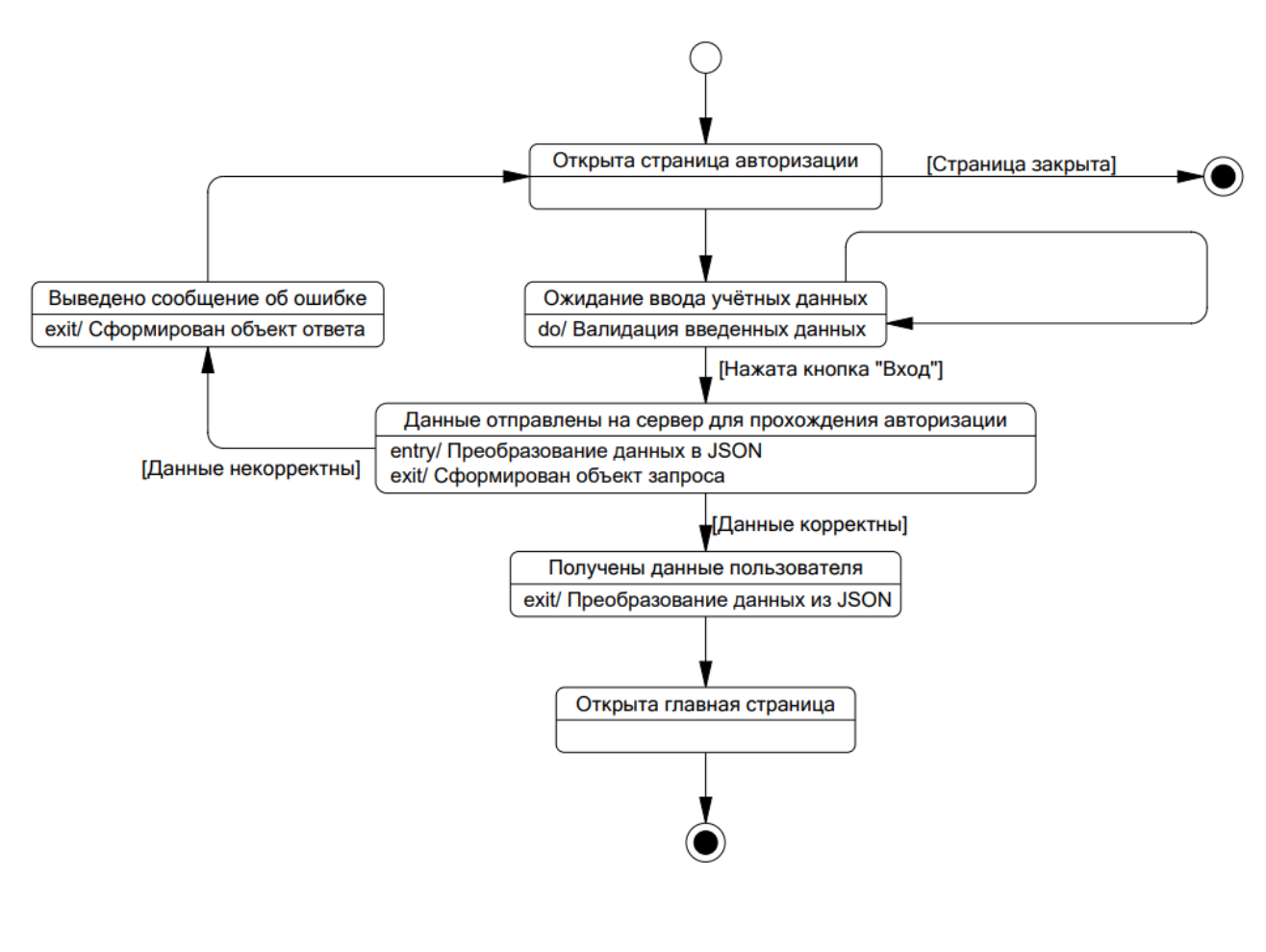


Рисунок 1.5 – Диаграмма состояний

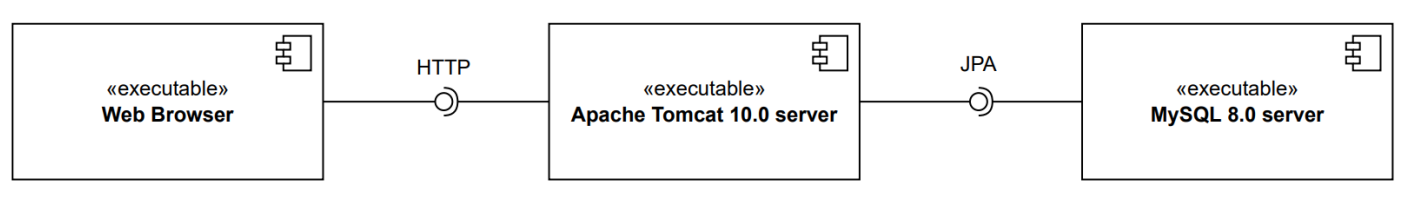
На диаграмме компонентов, представленной в приложениях, представлено три компонента: модуль «*Web Browser*», модуль «*Apache Tomcat* 9.0.71» и модуль «*PostgreSQL* 13».

Приложение разрабатывалось с учётом архитектурного шаблона *MVC*, поэтому в клиентском и серверном модуле классы сгруппированы на модели, которые описывают логику работы с объектами на основе объектно-ориентированного подхода.

Модуль «*Web Browser*» отвечает за функционирование клиентской части приложения. Этот модуль содержит *HTML*-файлы, которые выступают в качестве контроллеров. В классах-контроллерах реализована вся логика по представлению данных, полученных с сервера, в удобном для понимания виде для пользователя. Для общения клиентской части с серверной был создан сервисный слой «\_*services*», в котором содержатся классы, необходимые для корректной отправки, получения и обработки запросов.

Модуль «*Apache Tomcat* 9.0.71» выступает в качестве контейнера для серверной части приложения. Компоненты этого модуля отвечают за обработку запросов клиента и работу с модулем базы данных. Этот модуль содержит приложение, разработанное с использованием *Spring Framework*. Клиентская и серверная части приложения связываются между собой по протоколу *HTTP*.

Для взаимодействия с базой данных необходим сервер *PostgreSQL*. Серверная часть приложения устанавливает связь с базой данных посредством протоколов, работающих по спецификации *JPA*.



«executable»

PostgreSQL 13

«executable»

Apache Tomcat 9.0.71

«executable»

Web Browser

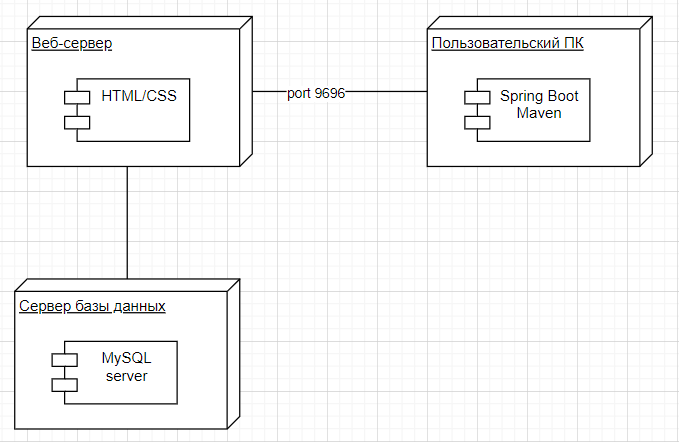
Рисунок 1.6 – Диаграмма компонентов

На рисунке 1.7 представлена диаграмма развёртывания разрабатываемой системы. Как можно заметить, основными элементами являются клиентский компьютер, сервер приложения и сервер системы управления базами данных.

В рамках данной системы к клиентским терминалам можно отнести компьютеры, установленные на рабочих местах сотрудников, часть рабочих процессов которых автоматизируется. Этот узел в себя включает себя компонент «веб-браузер», с помощью которого отображается клиентская часть приложения.

Узел сервера включает в себя исполняемый файл приложения, который выполняется с помощью *Java Virtual Machine*, которая в свою очередь развернута на сервере «*Apache Tomcat* 9.0.71». Серверная и клиентская части приложения обмениваются данными по протоколу *HTTP*.

На сервере СУБД в результате выполнения заранее созданного скрипта создается база данных. Серверная часть приложения получает данные с этого сервера посредством протоколов, работающих по спецификации JPA.



port 8080

PostgreSQL 13

server

Рисунок 1.7 – Диаграмма развертывания компонентов

1. **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

# 

# **Постановка задачи**

Для достижения цели курсового проекта необходимо разработать такое приложение, которое предоставит возможность работникам железнодорожной компании вносить и просматривать необходимую информацию.

Целевой аудиторией разрабатываемого на данный момент приложения будут являться некоторые сотрудники железнодорожной компании, осуществляющей пассажирские перевозки в регулярном сообщении. Например, диспетчер сможет вносить в базу данных рейсы, редактировать их статус. Машинисты должны иметь возможность просмотра своих рейсов. У машиниста также появится возможность просмотреть информацию о дорожных условиях. Начальник депо будет обладать функционалом для мониторинга депо транспортных средств.

Созданное приложение должно обеспечить выполнение следующих операций:

* регистрация сотрудников;
* авторизация сотрудников;
* получение списка всех пользователей;
* блокировка пользователя;
* выгрузка списка пользователей по различным критериям;
* регистрация поступлений новых транспортных средств в парк;
* регистрация выбытия транспортных средств из парка;
* выгрузка списка транспортных средств, находящихся в депо, по различным критериям;
* внесение в базу новых рейсов;
* смена статуса рейса.

Для максимального удобства клиентское приложение должно быть реализовано в виде веб-приложения.

# **Архитектурные решения**

Разработанное программное средство базируется на архитектуре типа «клиент-сервер». Эта архитектура предусматривает разделение процессов предоставления услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других. Несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервера, которая также может называться хост-системой.

В данном программном средстве реализована трехуровневая архитектура.

Первый уровень – клиентский. Клиент – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

Сервер приложения является вторым уровнем этой архитектуры. Это очень мощный компьютер или специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам.

В качестве третьего уровня выступает сервер базы данных. Результаты работы этого уровня и использует сервер приложения. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение, которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность.

Функциональные возможности, реализуемые на серверной стороне, представлены ниже:

* работа с поступающими клиентскими запросами;
* защита данных и доступ к ним;
* работа по отправке ответа клиенту.

Функциональные возможности, реализуемые на клиентской стороне, следующие:

* отрисовка пользовательского интерфейса;
* формирование запроса к серверу и его отправка;
* получение ответа сервера и предоставление его пользователю.

Особенности такой модели заключаются в том, что пользователь отправляет определенный запрос на сервер, где тот системно обрабатывается и конечный результат отсылается клиенту. В возможности сервера входит одновременное обслуживание сразу нескольких клиентов.

Если одновременно поступает более одного запроса, то такие запросы помещаются в определенную очередь и выполняются сервером по очереди. Нередко у запросов может быть приоритет – запросы с более высокими приоритетами будет постоянно выполняться в первую очередь.

Преимуществами трехуровневой клиент-серверной архитектуры являются целостность данных, высокая безопасность и защищенность базы данных от несанкционированных проникновений.

# **Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства**

Бизнес-логика описывает правила и процессы, которые определяют поведение объектов и процессов в бизнесе, и используется для представления этих процессов в информационных системах. Это включает в себя методы и алгоритмы анализа данных и способы передачи результатов пользователям. Бизнес-правила определяют, каким образом должны выполняться процессы. Программные модули и уровни системы, которые реализуют бизнес-логику, также называются бизнес-логикой.

Бизнес-логика может быть описана в текстовой форме, концептуальных моделях предметной области, бизнес-правилах, алгоритмах, диаграммах и моделях бизнес-процессов. В фазе анализа и проектирования системы она может быть представлена в виде диаграмм языка *UML* или других подобных языков, а в фазе программирования – в виде классов и методов или процедур и функций.

Бизнес-логика является ключевым элементом в разработке информационных систем, поскольку она определяет, как должны работать процессы и объекты предметной области в системе. Это позволяет разработчикам создавать программное обеспечение, которое соответствует требованиям бизнеса и эффективно решает задачи.

Кроме того, бизнес-логика может быть использована для автоматизации бизнес-процессов, что позволяет повысить эффективность и качество работы компании. Она также может использоваться для оптимизации производственных процессов, управления запасами, маркетинга и других аспектов бизнеса.

Бизнес-логика также может быть изменена в ходе развития компании или изменения требований рынка. Поэтому она должна быть гибкой и адаптивной, чтобы обеспечивать долгосрочный успех бизнеса.

Одним из основных процессов в программе является авторизация. Блок-схема этого процесса представлена на рисунке 2.1.

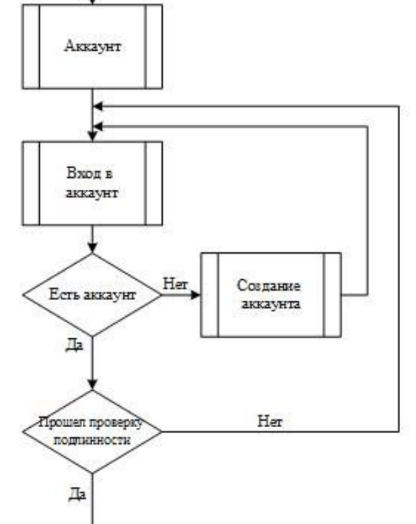


Рисунок 2.1 – Блок-схема алгоритма авторизации

Полная блок-схема представлена в Приложении А.

# **Проектирование пользовательского интерфейса**

Проектирование пользовательского интерфейса имеет цель создание четкого понимания того, как должен выглядеть и функционировать интерфейс системы, чтобы помочь в решении бизнес-задач и обеспечить конкурентное преимущество. Для достижения этой цели необходимо ответить на важные вопросы о целевой аудитории, целях продукта и ключевых задачах, а также выполнить предпроектный анализ и подготовить детальные схемы страниц и сценарии взаимодействия. Однако наиболее эффективный результат достигается с помощью интерактивного прототипа системы, который включает основные страницы и процессы работы.

Разработка интерактивного прототипа системы позволяет проверить и оптимизировать пользовательский интерфейс, выявить проблемы и улучшить его функциональность и удобство использования. Этот этап является важным для тестирования гипотез и получения обратной связи от потенциальных пользователей системы. После того, как интерактивный прототип был протестирован и улучшен, можно начинать работу над финальным дизайном и разработкой полноценной системы.

# **Обоснование выбора компонентов и технологий для реализации программного средства**

Язык программирования был выбран *Java.* Это современный, строго типизированный объектно-ориентированный язык программирования, выпущенный и поддерживаемый компанией S*un Microsystems*. На сегодняшний момент язык программирования *Java* один из самых производительных, быстро развивающихся и востребованных языков в *IT*-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, ежедневно обслуживающих миллионы пользователей.

*Java* поддерживает управление версиями, чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки *Java* как раздельные модификаторы *virtual* и *override*, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Объектно-ориентированный подход позволяет решить задачи по построению крупных и одновременно гибких, масштабируемых и расширяемых приложений. И *Java* продолжает активно развиваться, и с каждой новой версией происходит расширение функциональности языка.

При разработке будет использоваться фреймворк *Spring Framework* – универсальный фреймворк с открытым исходным кодом для *Java*-платформы. Совместима с такими операционными системами как *Windows, Linux* и *macOS*. Была выпущена компанией *Pivotal Softwaret*.

*Spring Framework* представляет собой просто контейнер внедрения зависимостей, с несколькими удобными слоями например: доступ к базе данных, прокси, *RPC*, веб-инфраструктура *MVC*.

*Spring boot* это дополнение к *Spring*, которое облегчает и ускоряет работу с ним. Сам *Spring Boot* представляет собой набор утилит, автоматизирующих настройки фреймворка. *Spring boot* упаковывает зависимости в *starter*-пакеты, автоматически конфигурирует приложения с помощью *jar*-зависимостей, создаёт веб-сервер, что позволяет запускать веб приложения. А также позволяет без использования *CSS* придавать различным объектам веб страницы свойства и стили.

*Spring Security* это фреймворк, предоставляющий различные механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для веб-сервисов, созданных с помощью *Spring Framework.*

*Spring MVC* фреймворк обеспечивающий архитектуру паттерна *Model – View – Controller* (Модель – Отображение– Контроллер) при помощи слабо связанных готовых компонентов. Паттерн *MVC* разделяет аспекты приложения (логику ввода, бизнес-логику и логику *UI*), обеспечивая при этом свободную связь между ними.

В качестве системы управления базами данных будет использована *PostgreSQL*. *PostgreSQL –* свободная реляционная система управления базами данных. Благодаря своему функционалу и возможностям *PostgreSQL* является отличным решением для малых и средних приложений. Главными достоинствами являются гибкость, лёгкость и удобство использования

*PostgreSQL* базы данных способны хранить большие объемы структурированной информации. В *PostgreSQL* все самые распространённые типы хранимых данных.

В качестве среды разработки была выбрана *IntelliJ IDEA Ultimate* от компании *JetBrains*. Данная среда разработки обладает удобным интерфейсом, возможностью подключения различных плагинов, что в значительной мере расширяет функционал данной среды разработки, кроме этого, встроенный функционал облегчает ускоряет и облегчает работу.

# **ПРИЛОЖЕНИЕ А**