

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»  
Факультет инженерно-экономический  
Кафедра экономической информатики  
Дисциплина «Средства и технологии анализа и разработки  
информационных систем»

«К ЗАЩИТЕ ДОПУСТИТЬ»  
Руководитель курсового проекта  
ассистент кафедры ЭИ  
\_\_\_\_\_.\_\_\_\_\_.2023 К.П. Воробей

## **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ УЧЁТА ПО ПРОКАТУ  
АВТОМОБИЛЕЙ»**

БГУИР КП 1-40 01 02-08 011 ПЗ

Выполнил студент группы 072303  
Колбик Артём Александрович

\_\_\_\_\_  
(подпись студента)  
Курсовой проект представлен на  
проверку \_\_\_\_\_.\_\_\_\_\_.2023

\_\_\_\_\_  
(подпись студента)

Минск 2023

## РЕФЕРАТ

БГУИР КП 1-40 05 01-02 011 ПЗ

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ УЧЁТА ПО ПРОКАТУ АВТОМОБИЛЕЙ: курсовой проект / А. А. Колбик – Минск: БГУИР, 2023, – п. з. 55 с.

Пояснительная записка 55 с., 37 рис., 10 источников, 3 приложения.

**Ключевые слова:** салон, прокат, автомобиль, база данных, учёт, приложение, администратор, пользователь, саппорт, разработка программного средства.

Целью курсового проекта является совершенствование и оптимизация процесса работы с учетом проката автомобилей, а также автоматизации бизнес-процессов, непосредственно связанных с управлением салоном проката автомобилей (отслеживание выручки, подача заявок на заказ и др.), путем разработки программного средства для автоматизации работы салона проката автомобилей.

Объект исследования – салон проката автомобилей и системы автоматизации работы салона проката автомобилей.

Предмет исследования – программное средство для автоматизации работы салона проката автомобилей.

Актуальность работы: программное средство для автоматизации работы салона требует углублений в рамках мобильности его использования и наличия возможности формирования документации и статистики. Для таких задач целесообразна автоматизация работы салона проката автомобилей.

Методология проведения работы: в процессе разработки системы использованы методы анализа данных, современные подходы к обработке данных, функциональный анализ процессов, принципы построения баз данных, моделирование системы с помощью UML-диаграмм.

Результаты работы: создано web-приложение, являющееся программным средством осуществления работы салона проката автомобилей. Проанализированы основные бизнес-процессы предметной области. Разработан интуитивно понятный интерфейс программного продукта, не вводящий пользователя в заблуждение. Техническое обоснование разработки программного модуля показало целесообразность его внедрения.

## СОДЕРЖАНИЕ

Введение.....	3
1 Анализ и моделирование предметной области программного средства.....	5
1.1 Описание предметной области .....	5
1.2 Разработка функциональной модели предметной области .....	6
1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.....	11
1.4 Разработка информационной модели предметной области .....	13
1.5 UML-модели представления программного средства и их описание ...	17
2 Проектирование и конструирование программного средства .....	21
2.1 Постановка задачи .....	21
2.2 Обоснование выбора компонентов и технологий для реализации программного средства .....	21
2.3 Архитектурные решения .....	22
2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства .....	23
2.5 Проектирование пользовательского интерфейса .....	26
2.6 Методы и средства, используемые для обеспечения безопасности данных.....	27
3 Тестирование и проверка работоспособности программного средства.....	29
4 Руководство по развертыванию и использованию программного средства .....	32
4.1 Руководство по установке (развертыванию) программного средства ..	32
4.2 Руководство пользователя.....	33
Заключение .....	45
Список использованных источников .....	46
Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» .....	47
Приложение Б (обязательное) Листинг скрипта генерации базы данных .....	48
Приложение В (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику .....	51

## **ВВЕДЕНИЕ**

Современный мир стремительно меняется, в результате чего современные технологии стали играть важную роль в различных сферах человеческой деятельности. И салоны проката автомобилей не стали исключением из этого правила.

Прокат автомобилей – это услуга, которая становится все более популярной в наше время. Она предоставляет возможность арендовать автомобиль на определенный период времени и использовать его для различных целей – от поездок по городу до дальних путешествий.

В связи с этим, программное средство для автоматизации основных бизнес-процессов салона проката автомобилей становится важной составляющей в управлении салоном и его успешности. Это сочетание аппаратно-программного обеспечения, которое помогает владельцам салонов автоматизировать бизнес-процессы, сократить временные затраты на рутинные операции, повысить качество услуг и удовлетворение клиентов.

Программное средство для автоматизации салона проката автомобилей может быть разработано под различные задачи и нужды. Кроме того, такое программное обеспечение позволяет сэкономить время на поиске и учете информации, ведение книги учета, управление базой данных клиентов и многое другое. В результате салон проката автомобилей может сосредоточиться на своем основном бизнесе – предоставлении качественных услуг клиентам, не тратя силы и время на мелкие, но важные рутинные операции.

Таким образом, применение программного средства для автоматизации основных бизнес-процессов салона проката автомобилей – это эффективный способ управления, повышения качества услуг, удовлетворения клиентов и привлечения новых клиентов. Автоматизация бизнес-процессов с помощью программного обеспечения, теперь является необходимостью для современных салонов проката автомобилей. Это инвестиция, которая выигрывает со временем, приносит выгоду и помогает оставаться конкурентоспособным на рынке.

Объект исследования – салон проката автомобилей и системы автоматизации работы салона проката автомобилей.

Предмет исследования – программное средство для автоматизации работы салона проката автомобилей.

Целью данного курсового проекта является совершенствование и оптимизация процесса работы с учетом проката автомобилей, а также

автоматизации бизнес-процессов, непосредственно связанных с управлением салоном проката автомобилей (отслеживание выручки, создание заказов и др.), путем разработки программного средства для автоматизации работы салона проката автомобилей.

Поставленная цель потребовала решения следующих задач:

- сделать анализ и модель предметной области салона проката автомобилей;
- выволнить проектирование и конструирование программного средства для автоматизации работы салона проката автомобилей;
- провести тестирование и проверку работоспособности программного средства для автоматизации работы салона проката автомобилей;
- предоставить руководство по развертыванию и использованию программного средства для автоматизации работы салона проката автомобилей пользователем.

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

Прокат автомобилей – это услуга, которая становится все более популярной в наше время. Она предоставляет возможность арендовать автомобиль на определенный период времени и использовать его для различных целей – от поездок по городу до дальних путешествий.

## **1.1 Описание предметной области**

Основная цель салона проката автомобилей – это предоставление высококачественных услуг и удовлетворение потребностей клиентов при поездке. Эта цель может быть достигнута только благодаря комплексному подходу к управлению салоном.

Салон проката автомобилей – это компания, которая предоставляет услуги по аренде автомобилей на определенный период времени. В салонах проката автомобилей представлены различные модели автомобилей, начиная от маленьких городских машин и заканчивая большими внедорожниками. Каждая компания имеет свой флот автомобилей, который может отличаться по маркам, моделям, возрасту и техническому состоянию.

Главным преимуществом салонов проката автомобилей является удобство. Вы можете взять автомобиль в любое время, в любом месте и на любой срок. Салоны проката автомобилей располагаются в различных точках города, а также в аэропортах и на железнодорожных станциях. Это позволяет вам быстро и удобно получить автомобиль и начать свое путешествие.

Кроме того, салоны проката автомобилей предлагают широкий выбор дополнительных услуг. Например, вы можете заказать навигатор или детское кресло для безопасности вашего ребенка. Также вы можете заказать дополнительную страховку, чтобы быть защищенным в случае несчастного случая.

Одним из главных критериев при выборе салона проката автомобилей является репутация компании. Вы можете почитать отзывы о компании в интернете, а также узнать мнение знакомых, которые уже пользовались услугами данной компании. Также важно обратить внимание на условия аренды – количество свободных километров, возможность использования автомобиля за пределами города или страны, наличие дополнительных услуг (например, навигатора, детского кресла и т.д.).

Салоны проката автомобилей предлагают различные тарифы на аренду автомобилей. Цена может зависеть от многих факторов – от модели автомобиля до срока аренды. Некоторые компании предлагают специальные тарифы на длительную аренду автомобиля или на аренду автомобиля на выходные дни. Также важно обратить внимание на стоимость дополнительных услуг, например, наличие страховки или возможность вернуть автомобиль в другом городе.

В салонах проката автомобилей вы можете получить профессиональную консультацию по выбору автомобиля. Сотрудники компании помогут вам определиться с моделью автомобиля, которая подходит именно вам, а также дадут советы по маршруту и условиям использования автомобиля.

Одним из важных факторов при выборе салона проката автомобилей является техническое состояние автомобиля. Хороший салон проката автомобилей должен предоставлять автомобили в отличном техническом состоянии. Однако, перед тем как арендовать автомобиль, необходимо проверить его наличие повреждений и работоспособность основных узлов и агрегатов. Также важно ознакомиться с правилами использования автомобиля, чтобы избежать неприятных ситуаций на дороге.

Салоны проката автомобилей предоставляют услуги как для частных лиц, так и для бизнеса. Например, компании могут арендовать автомобиль для своих сотрудников во время командировок. Также салоны проката автомобилей могут предоставлять услуги по организации свадебных кортежей или других торжественных мероприятий.

В целом, салоны проката автомобилей – это удобное и экономичное решение для тех, кто хочет совершать поездки без лишних затрат времени и денег. Однако, чтобы выбрать подходящую компанию и избежать неприятных сюрпризов, необходимо учитывать все факторы и тщательно изучать условия аренды.

## **1.2 Разработка функциональной модели предметной области**

IDEF0 – нотация описания бизнес-процессов. Основана на методологии SADT.

SADT - графические обозначения и подход к описанию систем. Разработка SADT началась в 1969 году и была опробована на практике в компаниях различных отраслей (аэрокосмическая отрасль, телефония и т.д.).

Публично появилась на рынке в 1975 г и получила очень широкое распространение в мире.

IDEF0 является результатом программы компьютеризации промышленности, которая была предложена ВВС США. Автоматизация деятельности предприятий потребовала соответствующих методик и инструментов. Перед тем, как разрабатывать программное обеспечение, необходимо было четко и понятно описать бизнес-процессы. Инструменты, разработанные для задач программирования, так же могут быть полезны и для задач менеджмента.

Нотация может быть использована для моделирования широкого круга автоматизированных и неавтоматизированных систем.

Идея IDEF0 лежит в том, что бизнес-процесс отображается в виде прямоугольника, в которой входят и выходят стрелки.

Для IDEF0 имеет значение сторона процесса и связанная с ней стрелка:

- слева входящая стрелка – вход бизнес-процесса – информация (документ) или ТМЦ, который будет преобразован в ходе выполнения процесса;

- справа исходящая стрелка – выход бизнес-процесса – преобразованная информация (документ) или ТМЦ;

- сверху входящая стрелка – управление бизнес-процесса – информация или документ, который определяет, как должен выполняться бизнес-процесс, как должно происходить преобразование входа в выход;

- снизу входящая стрелка – механизм бизнес-процесса – то, что преобразовывает вход в выход: сотрудники или техника. Считается, что за один цикл процесса не происходит изменения механизма.

Выход 1-го бизнес-процесса является входом/управлением/механизмом другого бизнес-процесса. На диаграмме процессы принято располагать по диагонали с верхнего левого угла в нижний правый. Количество процессов не более 6-8.

Основными потребителями нотации IDEF0 являются руководители, которым необходимо видеть и понимать взаимосвязь процессов, не вникая в мелочи [1].

Верхним уровнем в данном случае является процесс автоматизации процесса расчёта студенческой стипендий. Данное окно представлено на рисунке 1.1.



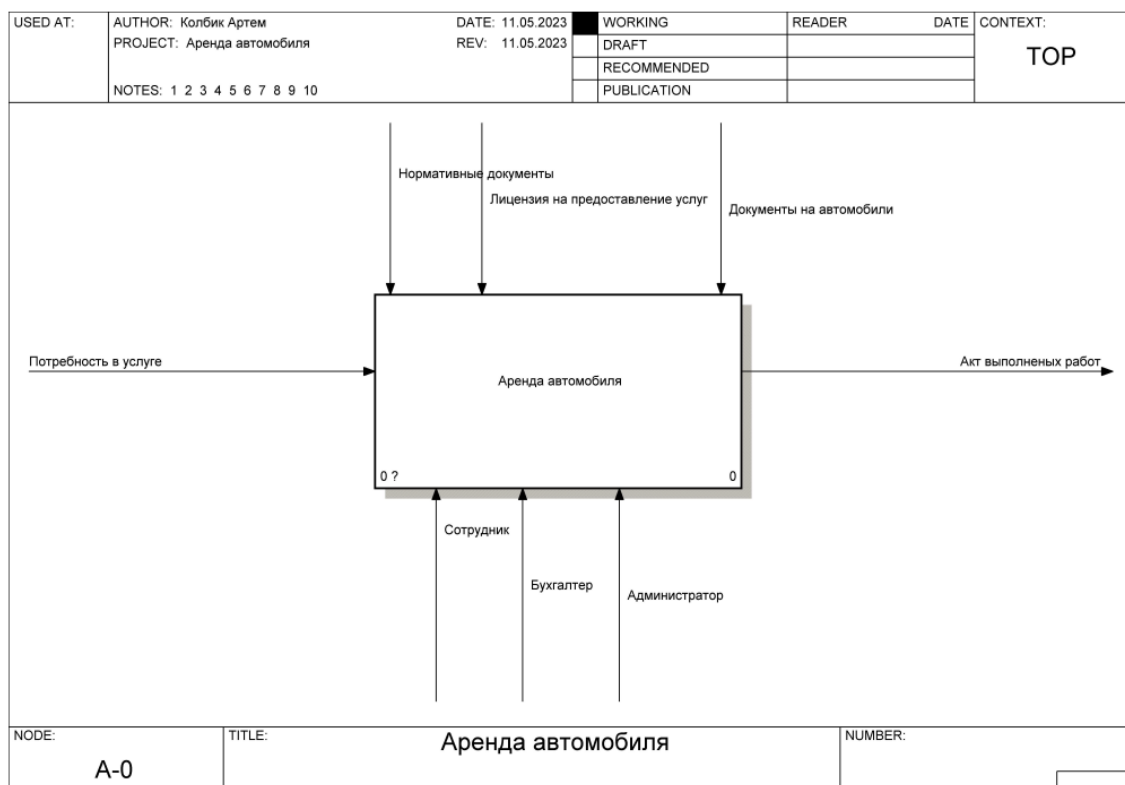


Рисунок 1.1 – Контекстная диаграмма верхнего уровня «Аренда автомобиля»

Далее будет представлено разбиение процесса автоматизации системы аренды автомобиля. Данное окно представлено на рисунке 1.2.

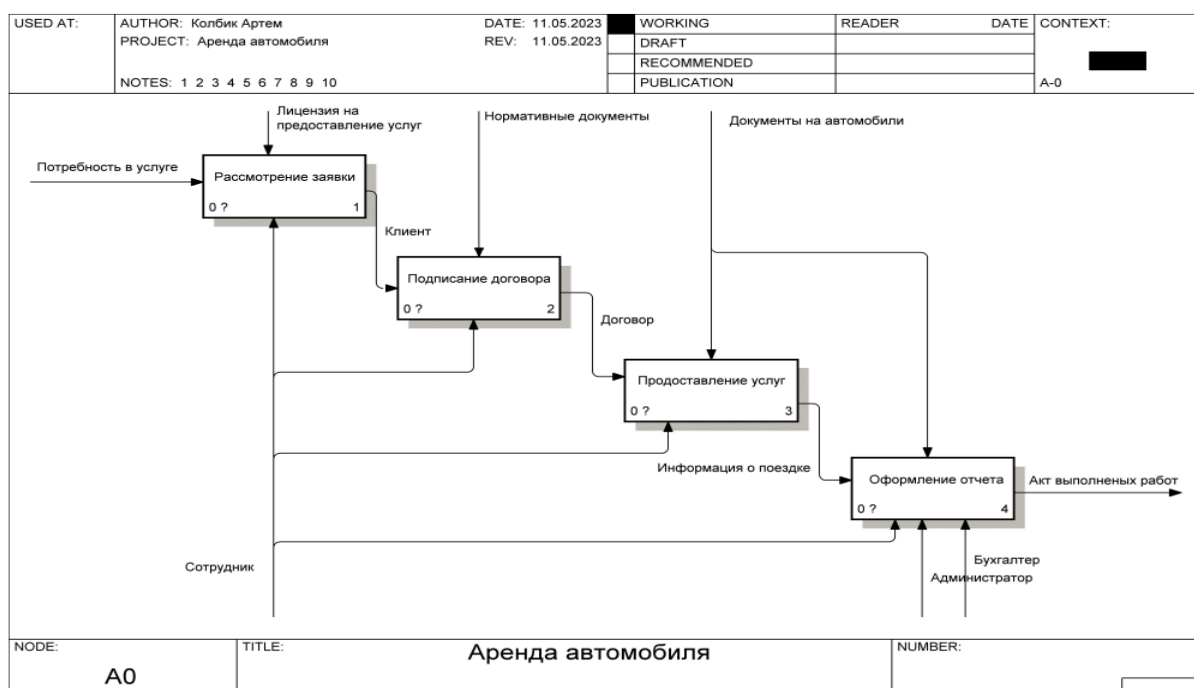


Рисунок 1.2 – Декомпозиция контекстной диаграммы верхнего уровня

Далее будет произведён процесс разбиения рассмотрения заявки. Данное окно представлено на рисунке 1.3.

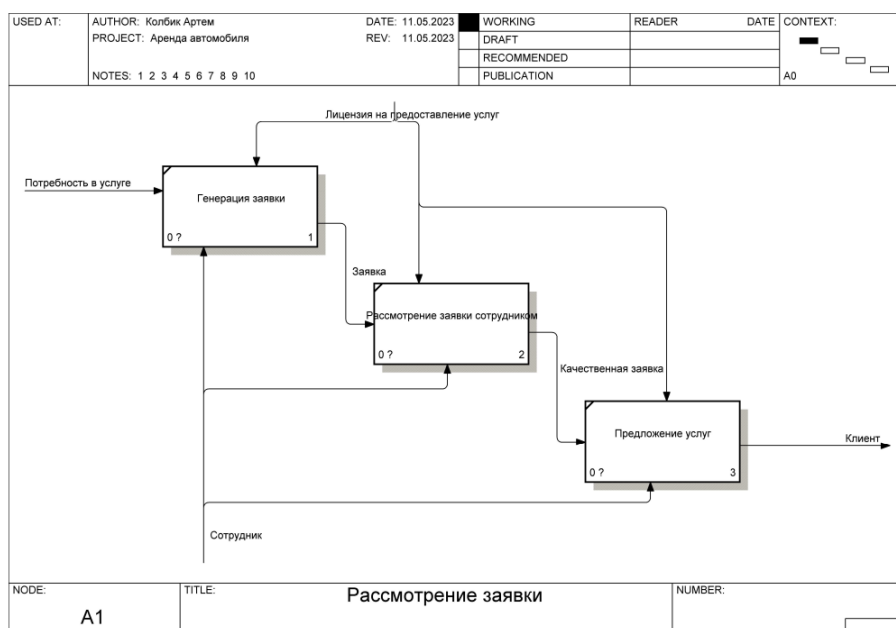


Рисунок 1.3 – Детализация блока «Рассмотрение заявки»

Далее будет произведён процесс разбиения подписания договора. Данное окно представлено на рисунке 1.4.

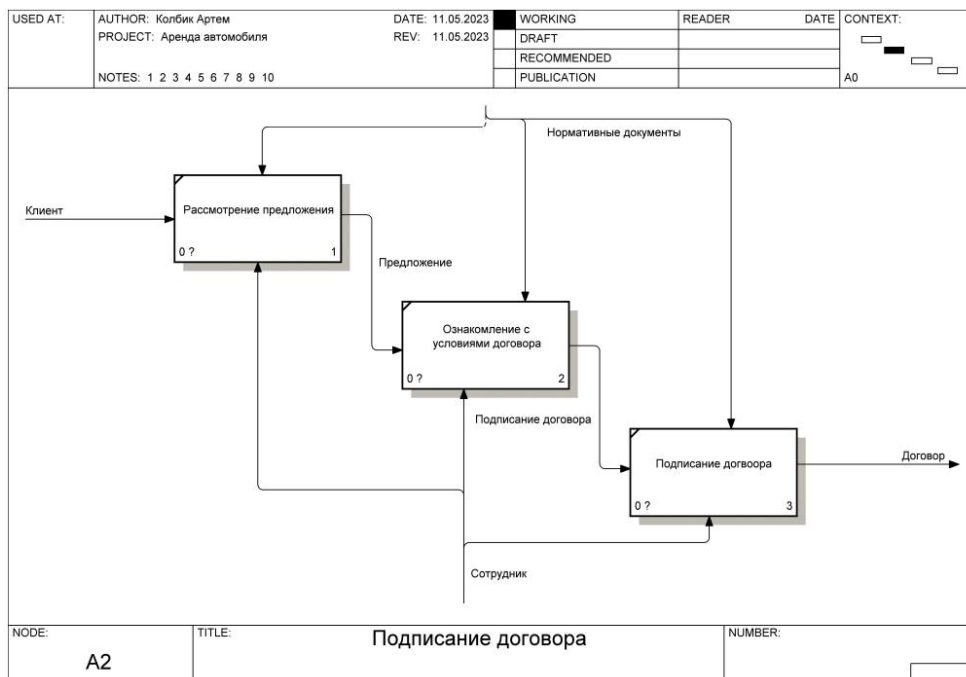


Рисунок 1.4 – Детализация блока «Подписание договора»

Далее будет представлено разбиение процесса предоставления услуг. Данное окно представлено на рисунке 1.5.

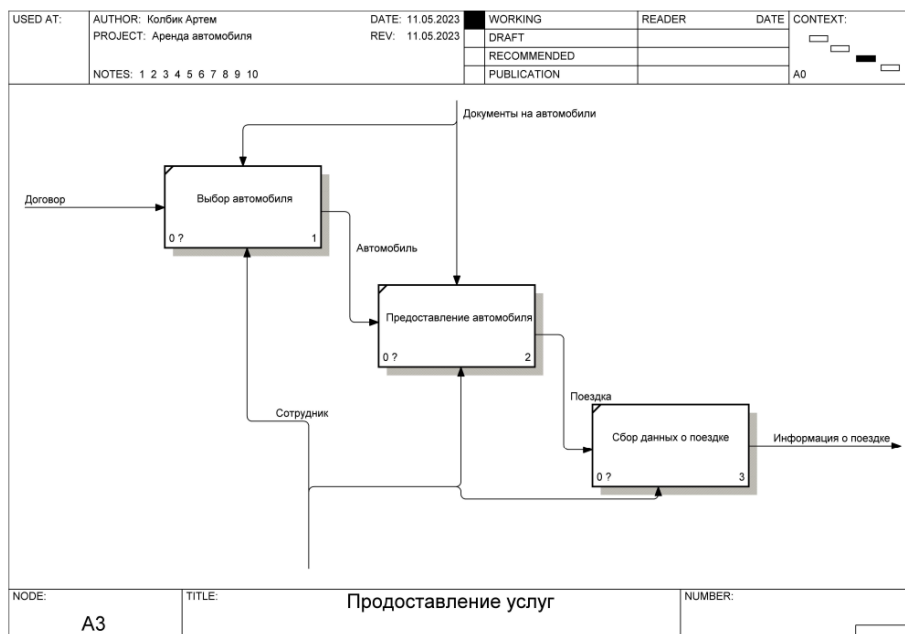


Рисунок 1.5 – Детализация блока «Предоставление услуг»

Далее будет представлено разбиение процесса реализации применения расчётов в бухгалтерии. Данное окно представлено на рисунке 1.6.

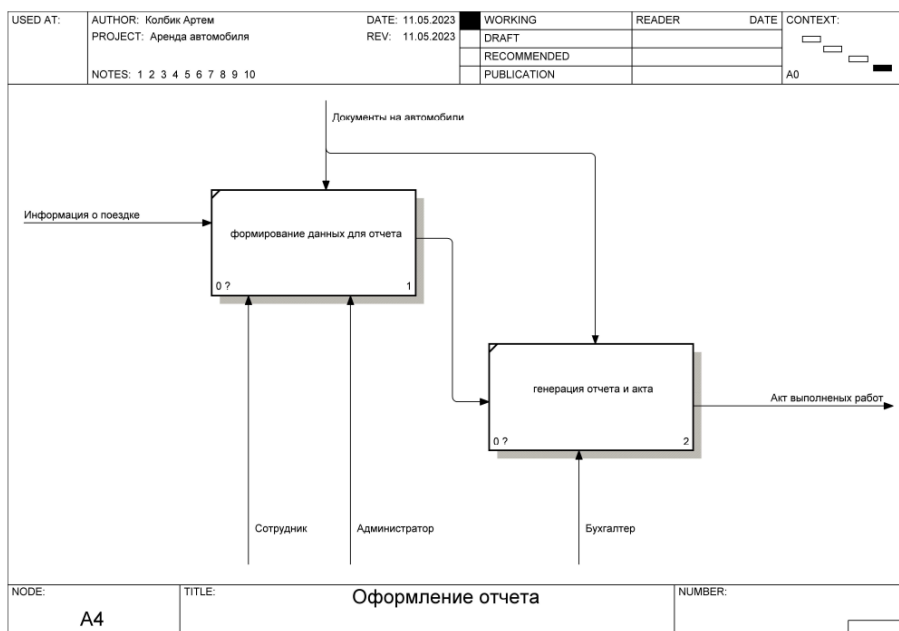


Рисунок 1.6 – Детализация блока «Оформление отчёта»

Перед проектированием программного средства следует определить, какие требования должны быть предъявлены к нему, так как невыполнение некоторых требований, которые были сформированы еще на ранней стадии, говорит о том, что разработанный сервис не сможет эффективно и с максимальной пользой использоваться, как было задумано изначально.

### 1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Так как у каждого пользователя имеются персональные данные, необходимо спроектировать базу данных, которая осуществляла бы их хранение в электронном виде. Это позволит облегчить поиск информации для

работников бухгалтерии. Для корректной работы необходимо предусмотреть возможность добавления, удаления и редактирования информации в базе данных. Для хранения информации будет использован MySQL Server.

Осуществлять регистрацию пользователя в систему будет или сам пользователь или Администратор.

Для ведения отчетности, Администратор сможет составлять отчёты по общему количеству заказов и сумме выручки на текущий день. Это позволит скорректировать работу бухгалтерии, что в дальнейшем может быть полезно для оптимизации её работы.

Диаграмма вариантов использования (англ. use case diagram) в UML – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмма вариантов использования представляет собой ценный инструмент для понимания функциональных требований к системе. Первый вариант такой диаграммы должен составляться на ранней стадии выполнения проекта. Более подробные версии должны появляться непосредственно перед реализацией конкретного прецедента.

Важно понимать, что данная диаграмма представляет взгляд на систему со стороны. А раз так, то не стоит ждать полного соответствия между прецедентами на диаграмме и их программной реализацией.

Отчасти именно поэтому зачастую получается так, что чем больше прецедентов находится на диаграмме, тем менее ценной может оказаться диаграмма вариантов использования.

Самым важным при построении данного вида диаграмм является определение того, какие актеры выполняют тот или иной прецедент, какие прецеденты включают или расширяют другие прецеденты.

В языке UML при построении диаграммы вариантов использования основными отношениями являются «include» (включает) и «extend» (расширяет).

На рисунке 1.7 представлена диаграмма вариантов использования приложения, которая демонстрирует, какие функции доступны для пользователя, какие – для администратора, какие – для саппорта, а какие – являются общими.

На диаграмме показаны лишь самые важные отношения между основными вариантами использования.

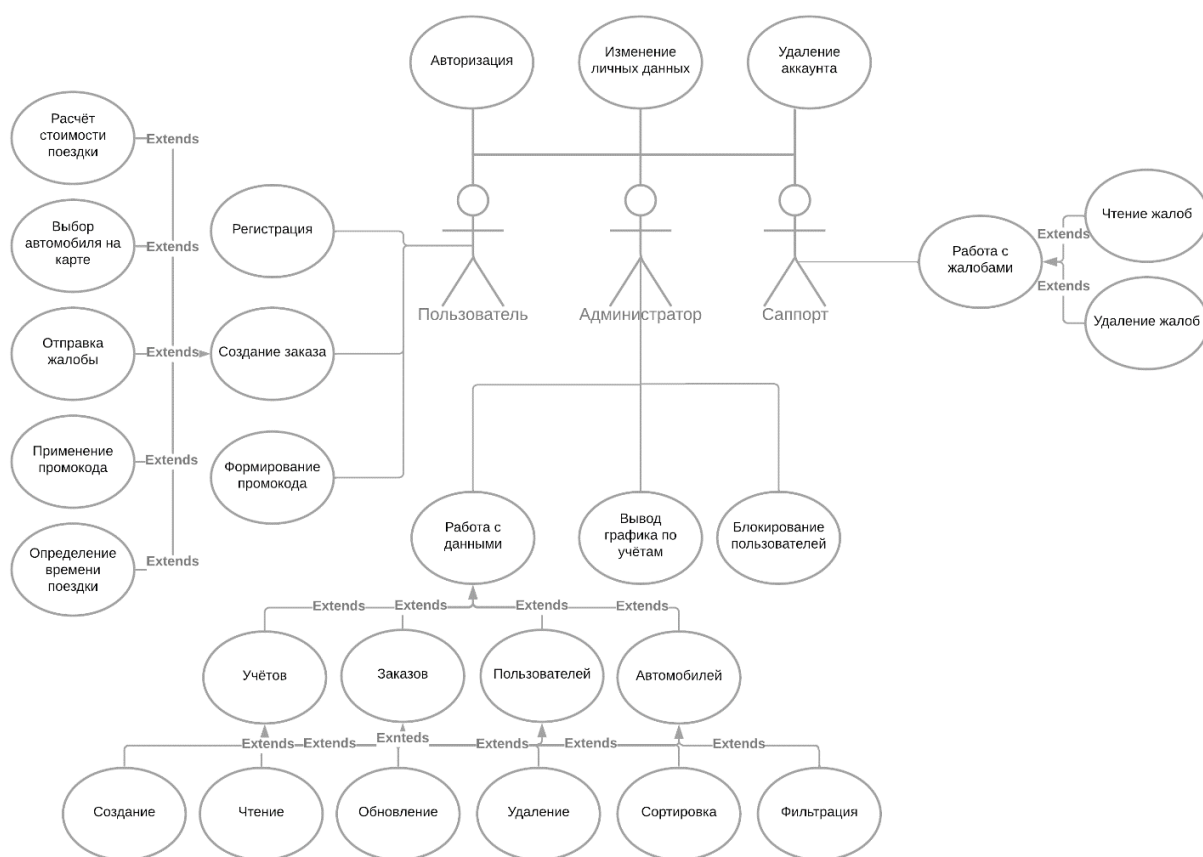


Рисунок 1.7 – Диаграмма вариантов использования

На данной диаграмме показано, что войти в систему возможно под тремя ролями: Пользователь, Администратор и Саппорт. Прежде чем войти в систему, пользователю необходимо авторизоваться в системе. Следует заметить, что наибольший функционал сосредоточен в ролях Администратора и Пользователя.

#### 1.4 Разработка информационной модели предметной области

Сначала необходимо создать информационную модель, чтобы максимально подробно отобразить суть в базе данных SQL. Набор различных таблиц в базе данных, их связи и атрибуты являются представлением информационной модели.

У всех объектов имеется определённый набор атрибутов, среди них выделяют существенные и малозначительные. Для того, чтобы упростить процедуру формализации предметной области часто прибегают к процессу дробления всего множества объектов ПО на мелкие подгруппы объектов, однородных по своей структуре и поведению. Они называются типами объектов.

Экземпляры такого рода объектов имеют одинаковый набор атрибутов, но отличаются значением хотя бы одного атрибута.

Каждому объекту присваивается определённый идентификатор – ключевой атрибут или набор атрибутов. Такие идентификаторы называются первичным ключом. Его значение всегда будет являться уникальным и необходимым.

Информационные потребности всей совокупности пользователей будущей системы определяются этапе анализа предметной области информационной системы. Которые, в свою очередь, определяют информацию, которая будет храниться в базах данных.

Информационный объект – это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных реквизитов (информационных элементов). Такими сущностями для информационных объектов могут служить: цех, склад, материал, вуз, студент, сдача экзаменов и т.д.

В процессе построения информационной модели были выделены следующие основные сущности:

- Таблица пользователей;
- Таблица автомобилей;
- Таблица заказов;
- Таблица жалоб;
- Таблица отчётов.

Сущность «Таблица пользователей» используется для регистрации и авторизации пользователей. Она имеет следующие атрибуты:

- id используется для хранения индивидуального номера пользователя в системе;
- first\_name используется для хранения в базе данных имени пользователя;
- last\_name используется для хранения фамилии пользователя в базе данных;

- email используется для хранения в базе данных электронной почты пользователя;
- phone используется для хранения номера телефона пользователя в базе данных;
- password используется для хранения пароля пользователя в базе данных;
- date\_of\_birth используется для хранения даты рождения пользователя в базе данных;
- status используется для хранения в базе данных статуса пользователя;
- role используется для хранения в базе данных роли администратора, саппорта или пользователя для последующей работы в системе.

Сущность «Таблица автомобилей» используется для внесения автомобилей в систему:

- id используется для хранения уникального номера автомобиля в базе данных;
- brand используется для хранения марки автомобиля в базе данных;
- engine\_capacity используется для хранения объёма двигателя в базе данных;
- is\_maintained используется для хранения готовности автомобиля к работе в базе данных.
- location используется для хранения координат автомобиля в базе данных.
- name используется для хранения названия автомобиля в базе данных.
- plate\_number используется для хранения номера автомобиля в базе данных.
- seats\_quantity используется для хранения количества мест в автомобиле в базе данных.
- stereo используется для хранения наличия магнитолы в автомобиле в базе данных.
- tariff используется для хранения тарифа автомобиля в базе данных.
- year используется для хранения года выпуска автомобиля в базе данных.

Сущность «Таблица заказов» используется для записи данных о заказах и имеет следующие атрибуты:



- id используется для хранения уникального номера заказа в базе данных;
- drive\_length используется для хранения в базе данных длительности поездки;
- drive\_price используется для хранения в базе данных стоимости поездки;
- order\_date используется для хранения даты поездки в системе;
- order\_time используется для хранения времени поездки в системе;
- promocode используется для хранения в системе примененного к оплате промокода;
- car\_id используется для хранения в системе индивидуального номера автомобиля, участвовавшего в заказе;
- user\_id используется для хранения в системе индивидуального номера пользователя, участвовавшего в заказе;

Сущность «Таблица жалоб» используется для хранения данных о жалобах на автомобили:

- id используется для хранения в базе данных индивидуального номера жалобы;
- message используется для хранения сообщения жалобы в системе;
- report\_date используется для хранения даты жалобы в системе;
- report\_time используется для хранения времени жалобы в системе;
- car\_id используется для хранения в системе индивидуального номера автомобиля, на который поступила жалоба;

Сущность «Таблица отчётов» используется для хранения всех данных об отчётах и имеет следующие атрибуты:

- id используется для хранения индивидуального номера отчёта в системе;
- last\_order\_time используется для хранения в базе данных времени последнего заказа;
- order\_quantity используется для хранения количества заказов в базе данных;
- order\_sum используется для хранения в базе данных суммы выручки от заказов;

- rental\_date используется для хранения даты создания отчёта в базе данных;
- rental\_time используется для хранения времени создания отчёта в базе данных;

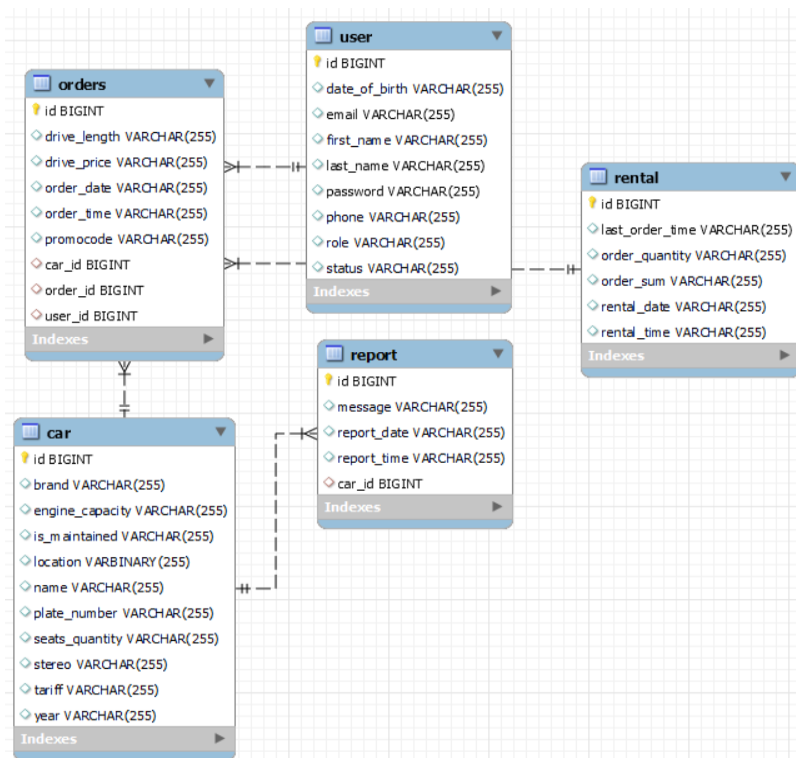


Рисунок 1.8 – Информационная модель базы данных

Таким образом была построена информационная модель базы данных для реализации системы учёта проката автомобилей, все сущности задействованы в логике приложения и без взаимной работы всех таблиц ожидаемый результат будет казаться неполным.

## 1.5 UML-Модели представления программного средства и их описание

Диаграмма состояний. Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются

соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграмма состояний отражает поведение, которое определяет последовательность состояний в ходе существования объекта, выделяя поток управления, следующий от состояния к состоянию.

На данной диаграмме (рисунок 1.9) показаны все возможные состояния авторизации.



Рисунок 1.9 – Диаграмма состояний авторизации

Диаграмма последовательности. Диаграмма последовательности представлена на рисунке 1.10. Когда пользователь хочет создать заказ, отправляет запрос на сервер на вывод только готовых к работе автомобилей, после чего сервер достаёт необходимые автомобили из базы данных и отправляет их клиенту. После того, как пользователь сформирует заказ, то бишь выберет автомобиль и проедет на нем какое-то количество времени, то эти данные отправляются на сервер, где ведется подсчёт стоимости поездки и её сохранение в базу данных. В конце, результаты подсчёта поездки и данные о заказе отправляются клиенту.

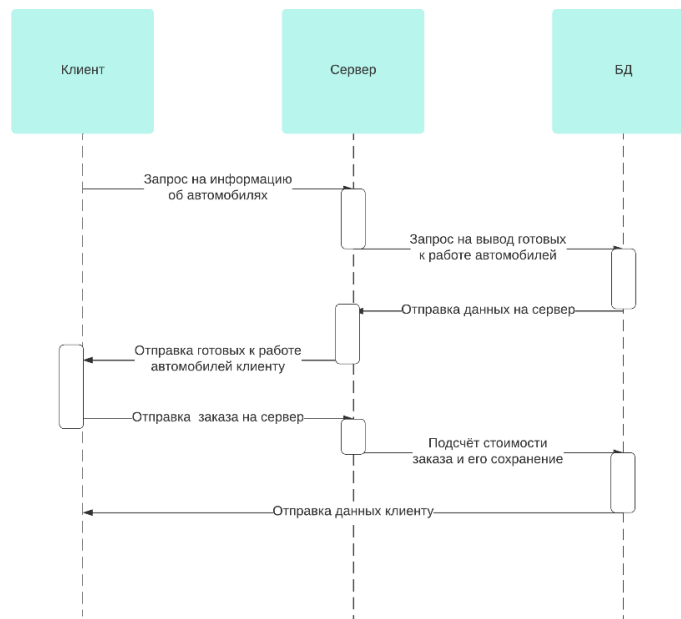


Рисунок 1.10 – Диаграмма последовательности формирования заказа

Диаграмма развёртывания. С помощью диаграммы развёртывания отобразим элементы и компоненты системы, существующие на этапе ее исполнения. Диаграмму развёртывания системы приведена на рисунке 1.11.

Как видно из диаграммы, для развёртывания приложения необходимо предоставить базу данных, запустить серверное приложение, затем запустить клиентское приложение на устройстве.

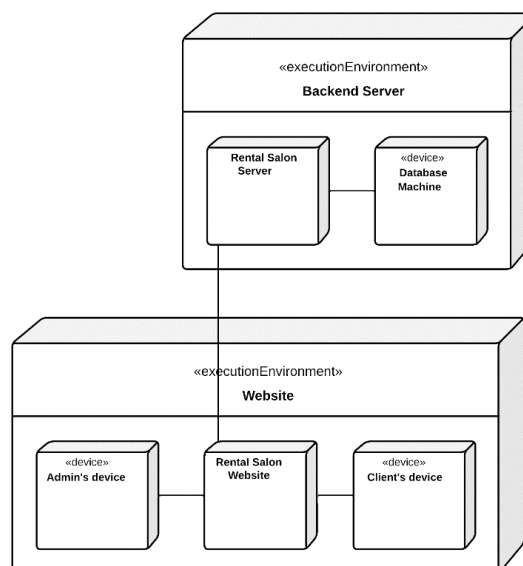


Рисунок 1.11 – Диаграмма развёртывания

Таким образом, диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения, к которым относятся исполнимые файлы, динамические библиотеки, таблицы базы данных. Те компоненты, которые не используются на этапе исполнения (например, исходные тексты программ), на диаграмме не показываются.

Подводя итоги, в данной главе представлены моделирование предметной области и разработка требований к программному средству, анализ и формализация бизнес-процессов службы управления персоналом, разработаны спецификация функциональных требований, информационная модель программного средства, а также модели представления программного средства на основе языка UML.

## **2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

### **2.1 Постановка задачи**

Так как у каждого пользователя имеются персональные данные, необходимо спроектировать базу данных, которая осуществляла бы их хранение в электронном виде. Это позволит облегчить поиск информации для работников бухгалтерии. Для корректной работы необходимо предусмотреть возможность добавления, удаления и редактирования информации в базе данных. Для хранения информации будет использован MySQL Server.

Осуществлять регистрацию пользователя в систему будет или сам пользователь или Администратор.

Для ведения отчетности, Администратор сможет составлять отчёты по общему количеству заказов и сумме выручки на текущий день. Это позволит скорректировать работу бухгалтерии, что в дальнейшем может быть полезно для оптимизации её работы.

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

В качестве основного средства разработки серверной части был выбран объектно-ориентированный язык программирования Java, так как он отлично подходит для разработки серверных приложений. В качестве его преимуществ можно выделить безопасность, независимость от платформы, стабильность и высокую производительность.

Для клиентской части был выбран фреймворк React, который подразумевает работу с JSX, где JSX – JavaScript XML, то бишь HTML и JavaScript. С помощью данного фреймворка можно создавать выразительные и понятные в использовании интерфейсы.

Основной средой разработки программного приложения является IntelliJ IDEA и VSCode. Сегодня IntelliJ IDEA – это наиболее интеллектуальная среда. Она по ходу написания и выполнения может проанализировать код, выявить ошибки и предложить достойное решение. Не менее важным инструментом является и графический редактор для создания интерфейса. Это значительно удобнее, чем постоянно писать шаблонный код. VSCode – один из самых

популярных редакторов кода, потому что он бесплатный и открытый, его можно сделать каким угодно под свои задачи.

Серверное приложение было запущено на облачном сервере. При этом база данных, используемая в приложении, отделена от машины, на которой оно запускается, таким образом, можно запустить несколько различных поставщиков данных (приложений) на многих устройствах для повышения надежности. В реализации используется реляционная база данных MySQL.

В целом, выбранные архитектурные решения и технологии позволят создать надежное и функциональное программное средство, способное удовлетворить потребности пользователей и обеспечить успешное функционирование парикмахерского салона.

### **2.3 Архитектурные решения**

Архитектурные решения играют важную роль в создании качественного программного продукта, который соответствует требованиям заказчика и потребностям пользователей. В данной подглаве будут рассмотрены основные архитектурные решения, выбранные для реализации разрабатываемого программного средства.

В данном курсовом проекте использовалась архитектура «Клиент-Сервер», в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически, клиент и сервер – это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Диаграмма классов серверной части приложения представлена на рисунке 2.1.

Клиенты – это так называемые объекты, которые запрашивают необходимую им информацию у серверов. Прикладной программный

интерфейс – это набор всевозможных функций и программ, которые обеспечивают взаимодействие клиентов и сервера.

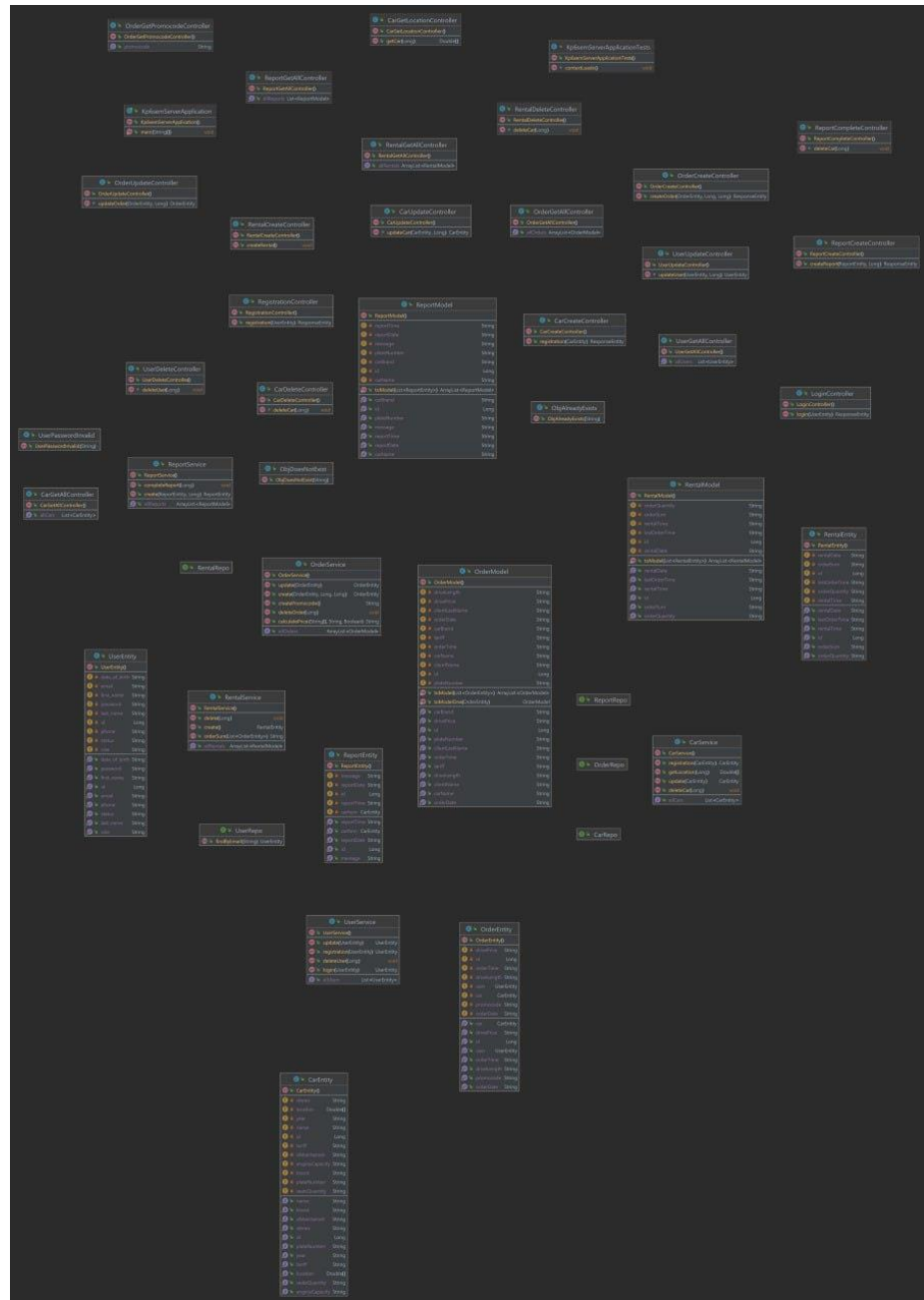


Рисунок 2.1 – Диаграмма классов серверной части приложения

## 2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства



Для того чтобы реализовать необходимую бизнес-логику, прежде всего необходимо составить алгоритмы работы главных бизнес-процессов, которые определяют порядок работы приложения и позволяют достичь поставленных целей. Далее будут описаны алгоритмы, реализующие ключевую бизнес-логику разрабатываемого программного средства.

После запуска появляется окно входа, где необходимо ввести почту и пароль. После входа данные отправляются на сервер, где и обрабатываются. Если все данные введены корректно, то происходит вход.

В зависимости от выбранной роли, введенного пароля, сервер, после получения введенных данных, будет обращаться к таблице базы данных для проверки корректности введенных данных. Если сравнение прошло успешно, и пользователь с такими данными найден, то открывается соответствующее окно: если была выбрана роль администратора, то меню администратора, если же роль пользователя – окно клиента, если же роль саппорта – окно саппорта. Алгоритм авторизации в приложении представлен на рисунке 2.2.

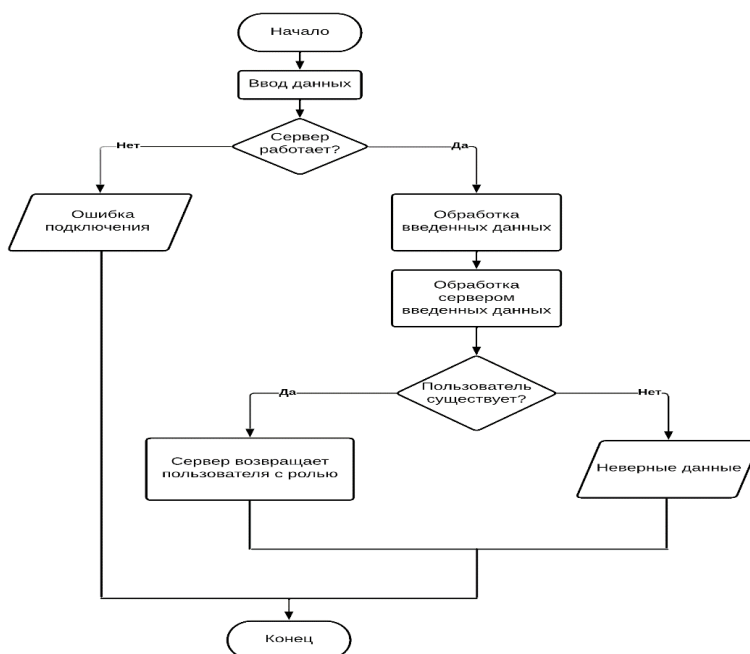


Рисунок 2.2 – Схема алгоритма работы авторизации приложения

При входе в приложение под ролью Пользователя предоставлена возможность сформировать заказ, согласно тарифу выбранного автомобиля, длительности поездки и формирования промокода (рисунок 2.3).

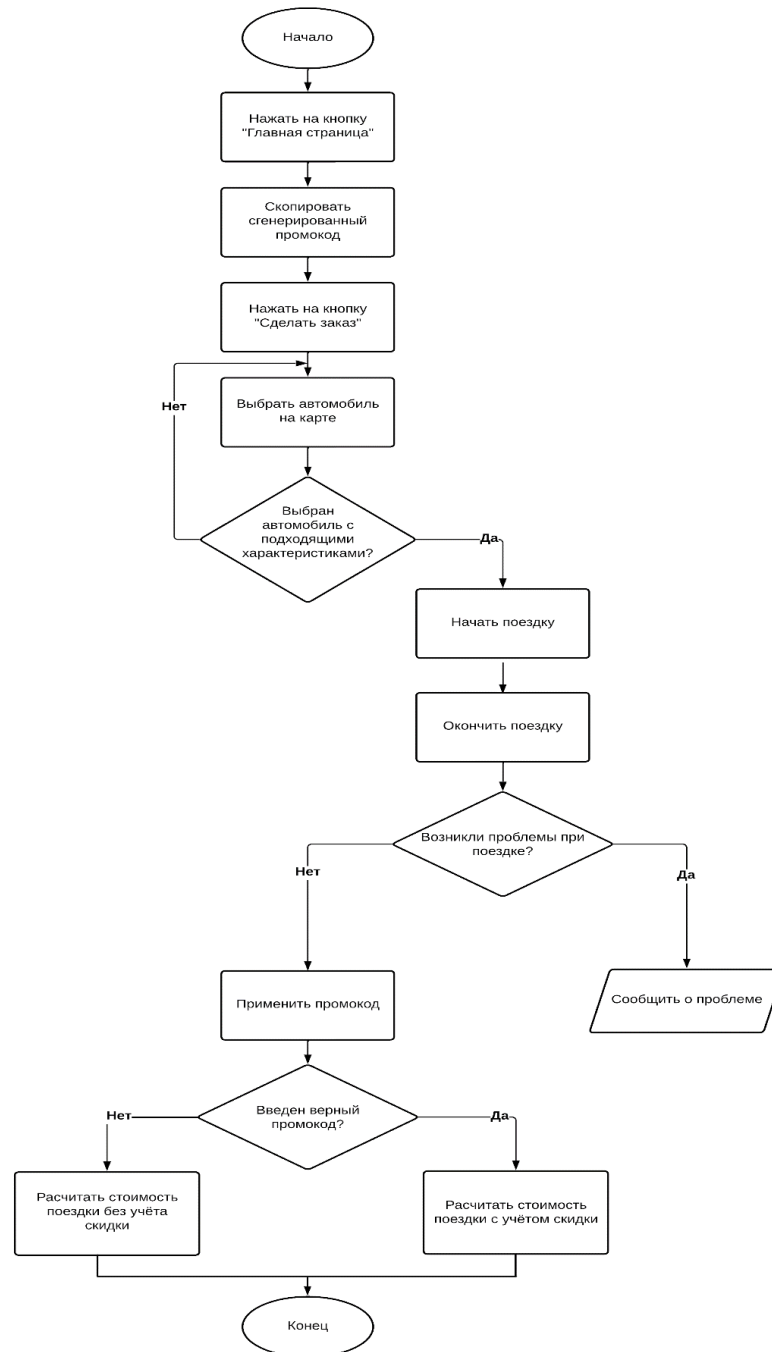


Рисунок 2.3 – Схема алгоритма формирования заказа

## 2.5 Проектирование пользовательского интерфейса

Интерфейс – это вся видимая пользователю часть сервиса, с которой он взаимодействует, решая свои задачи.

Проектирование интерфейсов – это всегда поиск наиболее эффективного решения, которое основано на понимании задач, мотиваций и обстоятельств пользователей и в то же время учитывает цели, возможности и ограничения со стороны бизнеса и технологий.

Одним из требований к хорошему графическому интерфейсу программной системы является концепция «предсказуемости», чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

При запуске веб-сайта от лица Пользователя мы попадаем на главную страницу (рисунок 2.4).

Здесь расположены такие кнопки как:

- Главная: - Это главная страница сайта, на которой генерируется промокод;
- Мой профиль: - На этой странице пользователь может просмотреть информацию о личных данных и отредактировать их либо удалить аккаунт;
- Сделать заказ: - На этой странице представлена информация о формировании пользователем заказа;
- Выйти: Эта кнопка позволяет выйти из учетной записи пользователя. После выхода пользователь будет перенаправлен на страницу авторизации.

Главная	Мой профиль	Сделать заказ	Выйти
<b>Личный кабинет</b>			
Имя user			
Фамилия user			
Дата рождения 2000-10-10			
Номер телефона +375255008904			
Электронная почта user@mail.ru			
Пароль user			
<button>Сохранить</button>			
<button>Удалить аккаунт</button>			

Рисунок 2.4 – Главная страница Пользователя

Все компоненты взаимодействуют друг с другом и связаны. Так же могут работать по отдельности. Разработанный интерфейс интуитивно понятен и примитивен.

## **2.6 Методы и средства, используемые для обеспечения безопасности данных**

В наше время компьютерных технологий, иметь доступ к практически любой информации - дело обыденное. Однако, за некоторой информацией стоит тщательно охраняемый доступ. Владельцы салонов проката автомобилей знают об этом особенно хорошо. Ведь даже кратковременный доступ к индивидуальным данным клиентов может иметь далеко идущие последствия. Единственный выход в этом случае - обеспечить безопасность хранения этих данных, и используемый алгоритм «Железнодорожная изгородь» может быть одним из наиболее эффективных методов защиты.

Алгоритм «Железнодорожная изгородь» создан специально для обеспечения высокой степени безопасности данных, особенно в случаях, когда данные передаются через ограниченные каналы связи. Этот алгоритм применяет метод шифрования с открытым ключом, который защищает данные от несанкционированного доступа.

Для обеспечения безопасности данных в салонах проката автомобилей, владельцы могут использовать услуги специализированных компаний, которые предоставляют полный комплекс защиты данных в соответствии с требованиями ISO/IEC 27001. Эти компании предоставляют различные опции, такие как защита сетевой инфраструктуры, аудит безопасности, управление доступом, мониторинг событий безопасности и многие другие, что помогает обеспечить прочную защиту данных.

В заключение, обеспечение безопасности данных в салоне проката автомобилей - это вопрос первостепенной важности, и требует тщательного подхода. Использование алгоритма «Железнодорожная изгородь» и других методов, таких как услуги специализированных компаний, создание резервных копий данных и дополнительные проверки - обеспечат безопасность данных клиентов в салонах проката автомобилей. Конечно, никакой метод не может гарантировать абсолютную безопасность, и поэтому владельцы салонов проката автомобилей должны постоянно совершенствовать защиту данных и никогда не снижать бдительность.

### 3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование – процесс анализа программного средства и сопутствующей документации с целью поиска дефектов и повышения качества продукта [2].

Для проверки уровня базовых пользовательских требований будем использовать тестирование на основе тест-кейсов.

Тест-кейс – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Оформленные основные тест-кейсы для данного программного приложения представлены в таблице 3.1.

Таблица 3.1 – Тест-кейсы для проверки базовых пользовательских требований

№	Заглавие тест-кейса	Шаги тест-кейса	Ожидаемый результат	Статус тест-кейса
1.	Регистрация нового пользователя в системе	1. Открыть форму регистрации. 2. Заполнить поля формы необходимыми данными	1. Форма регистрации открылась, все поля доступны для ввода. 2. При вводе корректных данных пользователь успешно зарегистрирован. При неправильном вводе выводится сообщение об ошибке.	Passed
2.	Авторизация пользователя в системе	1. Открыть форму авторизации. 2. Ввести данные.	1. Форма авторизации открылась, все поля доступны. 2. Вход в аккаунт. При неправильном вводе данных вывод на экран сообщения об ошибке.	Passed

№	Заглавие тест-кейса	Шаги тест-кейса	Ожидаемый результат	Статус тест-кейса
3.	Формирование заказа	1. Открыть страницу формирования заказа. 2. Пройти все шаги формирования заказа.	1. Шаги формирования заказа идут в правильной последовательности.	Passed
4.	Фильтрация пользователей	1. Открыть страницу со всеми пользователями. 2. Заполнить поля фильтрации по условию необходимыми данными. 3. Дождаться вывода результата.	1. Форма фильтрации пользователей открылась. 2. Все поля для ввода доступны. 3. Пользователь видит отфильтрованные по условию записи.	Passed
5.	Формирование отчета о всех заказах и общей выручки	1. Нажать кнопку «Сформировать отчёт».	1. Появилась новая запись, содержащая общую сумму выручки, количество заказов, дату и время формирования отчёта, номер последнего заказа.	Passed
6.	Редактирование профиля пользователя	1. Открыть форму редактирования профиля. 2. Ввести новые данные. 3. Подтвердить их.	1. Форма редактирования профиля открылась, все поля доступны для ввода. 2. При верном вводе данных информация о профиле успешно	Passed

			отредактирована. При неправильном вводе выводится сообщение об ошибке.	
--	--	--	--	--

В данной главе было описано тестирование программного приложения с помощью тест-кейсов. Все тест-кейсы были выполнены со статусом «Успешно». Таким образом, разработанные тест-кейсы помогли избежать потери данных и некорректной работы приложения при возникновении исключительных ситуаций.

## 4 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

### 4.1 Руководство по установке (развертыванию) программного средства

Для развертывания данного приложения необходимо иметь ПК с доступом к интернету, в веб-браузере включить поддержку JavaScript, а также скачать репозиторий с клиентским интерфейсом, который содержит все необходимые файлы для запуска приложения.

После скачивания репозитория следует в командной строке клиентского приложения прописать “npm start”, после чего дождаться сборки клиентской части приложения. Далее необходимо открыть серверную часть приложения и запустить её, дождавшись её полного запуска можно начинать взаимодействовать с клиентской частью приложения. Первой видимой страницей будет страница авторизации (рисунок 4.1), где пользователь может авторизоваться, заполнив корректно предложенные поля.



Рисунок 4.1 – Главная страница сайта

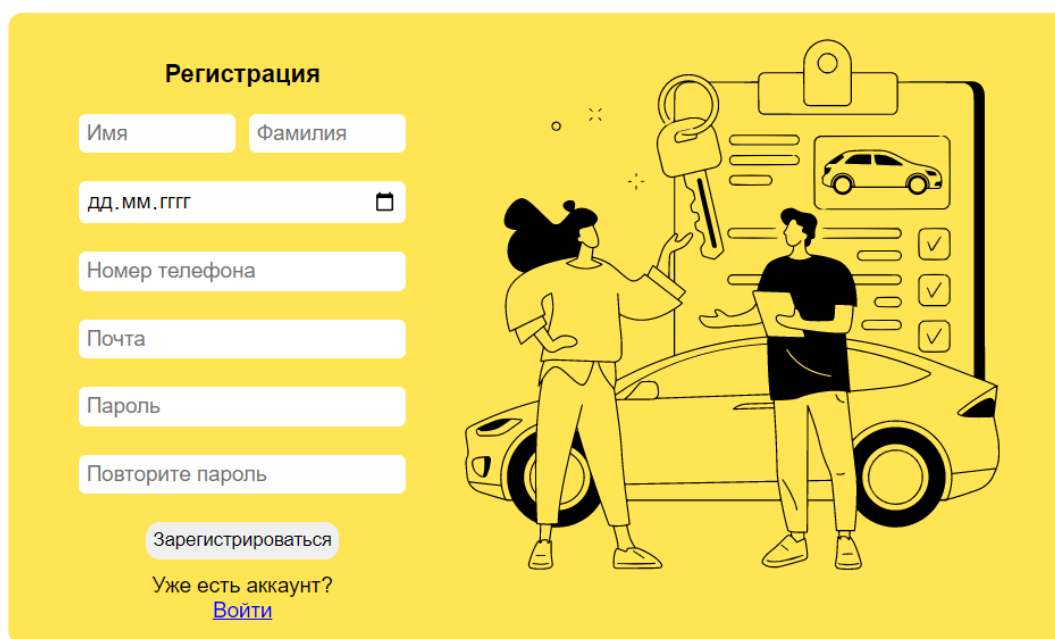


Благодаря простоте установки и запуска приложения, пользователь может легко запустить данное программное приложение на любом устройстве, которое удовлетворяет требованиям к работе с веб-приложениями. Это позволяет обеспечить максимальную доступность и удобство использования приложения для пользователей.

## 4.2 Руководство пользователя

Руководство пользователя является важной частью программного продукта и предназначено для того, чтобы помочь пользователям освоиться с интерфейсом и функционалом программы. В данном курсовом проекте руководство пользователя описывает основные возможности салона проката автомобилей и дает инструкции по их использованию.

Если пользователь на сайте впервые, то он должен зарегистрироваться на сайте (рисунок 4.2), если же пользователь уже имеет свой собственный аккаунт, ему следует авторизоваться (рисунок 4.3).



**Регистрация**

Имя  Фамилия

дд.мм.гггг

Номер телефона

Почта

Пароль

Повторите пароль

Уже есть аккаунт? [Войти](#)

Рисунок 4.2 – Страница регистрации

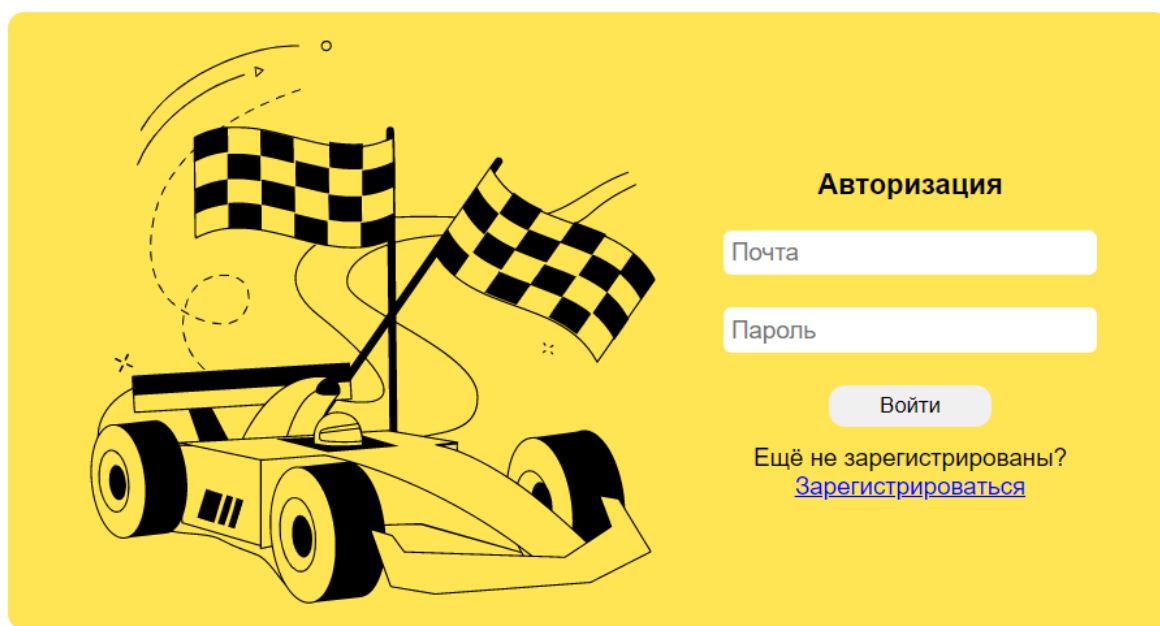








Рисунок 4.3 – Страница авторизации

После авторизации пользователь попадает в свой личный кабинет (рисунок 4.4), в котором он может редактировать свои личные данные или удалить свой аккаунт.

### Личный кабинет

Имя	
Валерий	
Фамилия	
Жмышенко	
Дата рождения	
1965-08-29	
Номер телефона	
+375298542275	
Электронная почта	
zhmih@mail.ru	
Пароль	
valera	

Сохранить

Удалить аккаунт

Рисунок 4.4 – Личный кабинет

Если перейти на главную страницу, то у нас сгенерируется промокод, который, в последствие, можно применить при оплате заказа (рисунок 4.5).

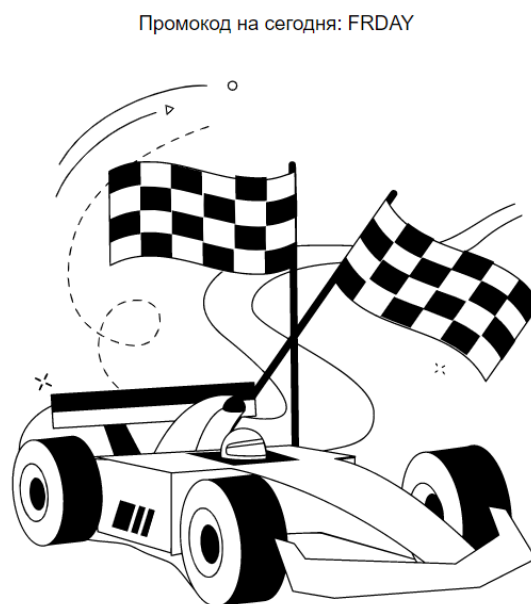


Рисунок 4.5 – Главная страница

Когда пользователь хочет сделать заказ, открывается карта с расположенными на ней автомобилями, пользователь выбирает автомобиль (рисунок 4.6) и начинает поездку (рисунок 4.7).

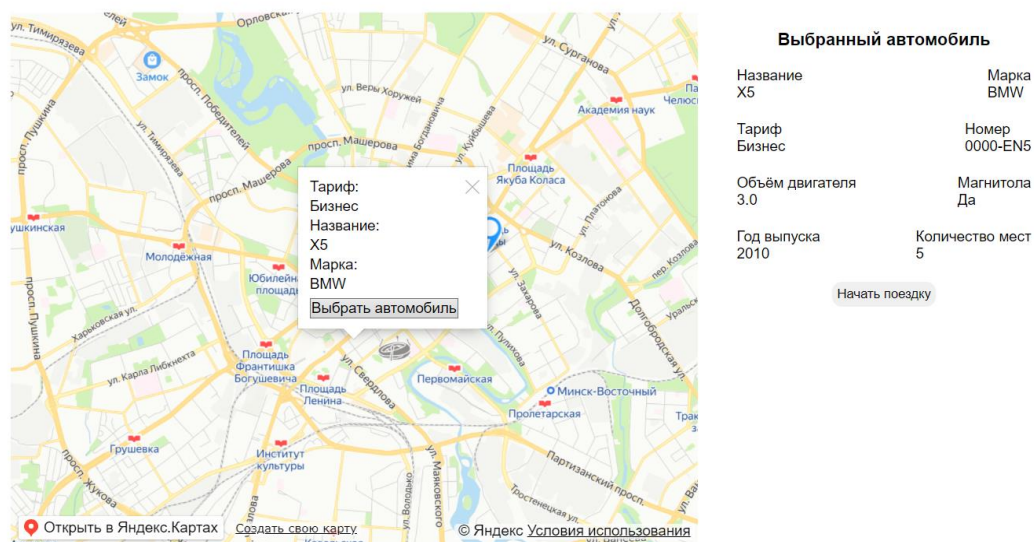


Рисунок 4.6 – Выбранный пользователем автомобиль

## Выбранный автомобиль

Название	Марка
X5	BMW
Тариф	Номер
Бизнес	0000-EN5
Объем двигателя	Магнитола
3.0	Да
Год выпуска	Количество мест
2010	5

0:0:13

Завершить поездку

Рисунок 4.7 – Пользователь начал поездку

Когда пользователь проедет необходимое ему расстояние и завершит поездку, он попадет на страницу заказа (рисунок 4.8).

## Данные о заказе

Время поездки: 0:1:10

Марка автомобиля: BMW

Название автомобиля: X5

Номер автомобиля: 0000-EN5

Тариф: Бизнес

Расчитать стоимость поездки

Сообщить о проблеме

Рисунок 4.8 – Страница заказа

Здесь пользователь может сообщить о возникшей в ходе поездки проблеме (рисунок 4.9), которую в дальнейшем решит сотрудник салона. Также, при вводе правильного промокода стоимость поездки снизится на 30 % (рисунок 4.10), если же введенный промокод некорректен или отсутствует, то скидка к оплате поездки применена не будет (рисунок 4.11).

### Сообщить о проблеме

Грязный салон

Отправить

Рисунок 4.9 – Сообщить сотрудникам салона о проблеме

### Данные о заказе

Время поездки: 0:1:10

Марка автомобиля: BMW

Название автомобиля: X5

Номер автомобиля: 0000-EN5

Тариф: Бизнес

FRDAY

Стоимость поездки: 1.8725 р.

Оплатить

Рисунок 4.10 – Расчёт поездки с учётом скидки

### Данные о заказе

Время поездки: 0:1:10

Марка автомобиля: BMW

Название автомобиля: X5

Номер автомобиля: 0000-EN5

Тариф: Бизнес

Промокод

Стоимость поездки: 2.675 р.

Оплатить

Рисунок 4.11 – Расчёт поездки без учёта скидки

Если же при авторизации зайти под ролью Администратора, то мы всё также попадём в личный кабинет, но функционал будет отличен от функционала пользователя.

Таким образом, рассмотрим всех зарегистрированных в системе пользователей (рисунок 4.12). По нажатию на кнопку «Добавить пользователя» мы можем добавлять новых пользователей в систему (рисунок 4.13), удалять пользователей по нажатию на кнопку «Удалить пользователя»,

а также редактировать пользователей при помощи двойного нажатия левой кнопки мыши на необходимого пользователя (рисунок 4.14).

Добавить пользователя

Удалить пользователя

ID	Имя	Фамилия	Дата рождения	Номер телефона	Почта	Пароль	Роль
2	Дмитрий	Жмышенко	2002-02-07	+375849304572	test@mail.ru	karasik1	user
3	admin	admin	2002-01-10	+375285948341	admin@mail.ru	admin	admin
5	user	user	2000-10-10	+375255008904	user@mail.ru	user	user
7	Валерий	Жмышенко	1965-08-29	+375298542275	zhmih@mail.ru	valera	user

Рисунок 4.12 – Зарегистрированные в системе пользователи

Добавить пользователя

Имя

Фамилия

ДД.ММ.ГГГГ

Номер телефона

Почта

Пароль

Роль

Добавить

Рисунок 4.13 – Добавить пользователя в систему

ID	Имя	Фамилия	Дата рождения	Номер телефона	Почта	Пароль	Роль
2	Дмитрий	Жмышенко	2002-02-07	+375849304572	test@mail.ru	karasik1	user
3	admin	admin	2002-01-10	+375285948341	admin@mail.ru	admin	admin
5	user	user	2000-10-10	+375255008904	user@mail.ru	user	user
7	Валерий	Жмышенко	1965-08-29	+375298542275	zhmih@mail.ru	valera	user

Рисунок 4.14 – Редактировать пользователя

Рассмотрим все зарегистрированные в системе автомобили (рисунок 4.15), при нажатии на кнопку «Добавить автомобиль» мы можем вносить в систему новые автомобили (рисунок 4.16), при нажатии на кнопку «Удалить автомобиль» можно удалить автомобиль из системы, автомобили можно редактировать таким же способом, как и пользователей.

Добавить автомобиль    Удалить автомобиль

ID	Название	Марка	Год выпуска	Номер	Объем двигателя	Мест	Магнитола
1	X5	BMW	2010	0000-EN5	3.0	5	Да
3	S5	Mercedes	2010	00001-AZ5	3.0	5	Да

Рисунок 4.15 – Зарегистрированные в системе автомобили

### Данные автомобиля

Тариф

☒ Эконом   
 ☐ Комфорт   
 ☐ Бизнес

☒ Готов к работе   
 ☐ Нет

☒ Магнитола в наличии   
 ☐ Нет

### Месторасположение автомобиля

Рисунок 4.16 – Внести автомобиль в систему

Также, для всех страниц Администратора с выводом данных для каждого столбца сделаны идентичные сортировки (рисунок 4.17) и фильтрации (4.18).



Марка ↓

Mercedes

BMW

Рисунок 4.17 – Сортировка по возрастанию

Марка ▼

BMW

Год выпуска

Contains ▼

BMW

☐ AND ☒ OR

Blank ▼

Рисунок 4.18 – Фильтрация по условию

При выборе пункта меню «Общая сводка» у Администратора появляется возможность составлять отчёты с учётом текущих заказов (рисунок 4.19).

Провести учёт   Удалить запись   Отобразить на графике

ID	Количество заказов	Сумма заказов	Дата учёта	Время учёта	Время последнего заказа
1	1	1.7622499999999999	2023-05-04	19:11	19:10
3	2	4.31225	2023-05-05	12:49	12:47
4	2	4.31225	2023-05-11	18:43	12:47

Рисунок 4.19 – Отчёты по заказам в салоне

При нажатии на кнопку «Провести учёт» количество записей учётов увеличится на 1, при нажатии на кнопку «Удалить запись» количество записей уменьшится на 1, при нажатии на кнопку «Отобразить на графике» будет отображена текущая статистика отчётов по сумме заказов и их количеству (рисунок 4.20).

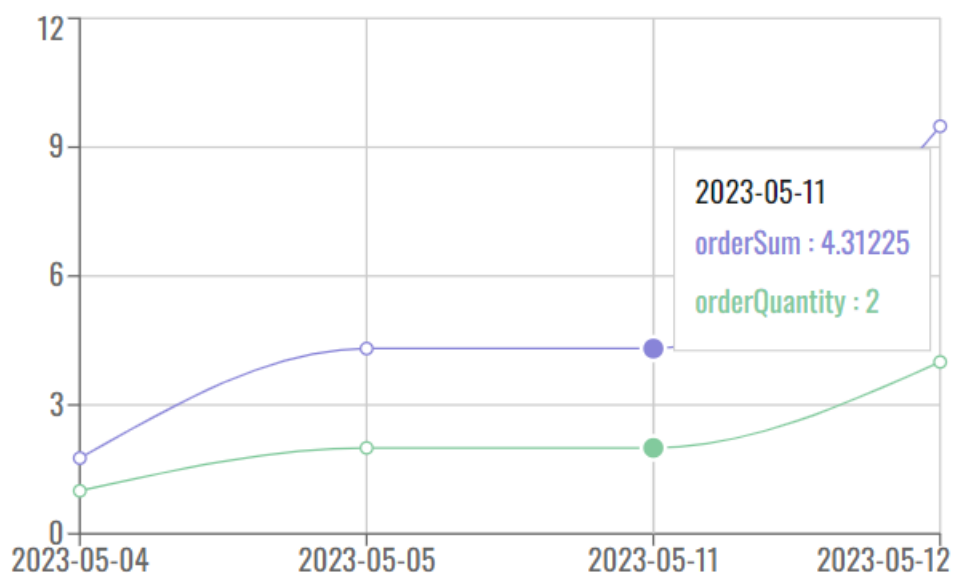


Рисунок 4.20 – Отображение статистики отчётов на графике

Администратор также может банить пользователей, и, если забаненный пользователь попытается войти в приложение дальнейший его функционал будет приостановлен (рисунок 4.21).

## Доступ ограничен

Вы были забанены администратором сервиса.



Рисунок 4.21 – Попытка входа забаненного пользователя

При входе в приложение под ролью Саппорта будет доступна лишь одна функция – обработка жалоб пользователей (рисунок 4.22). Здесь Саппорт может лишь исправлять жалобы пользователей.

ID	Название	Марка	Номер	Время	Дата	Сообщение
1	X5	BMW	0000-EN5	12:42	2023-05-05	Грязный салон
3	X5	BMW	0000-EN5	16:47	2023-05-12	Грязный салон

Рисунок 4.22 – Жалобы пользователей

В целом, функциональные возможности и инструкции по установке и использованию программного средства позволяют пользователям быстро овладеть его функционалом и эффективно работать с программным

средством, что делает его более доступным и привлекательным для использования.

## ЗАКЛЮЧЕНИЕ

В ходе данного курсового проекта были достигнуты поставленные цели и реализованы необходимые задачи. Так, например, была изучена область деятельности салона проката автомобилей. А результатом выполнения проекта стало программное средство, которое позволяет данной компании автоматизировать большое количество процессов, увеличить производительность и др.

Кроме создания приложения можно отметить, что результатом изучения процесса деятельности салона проката автомобилей стала диаграмма IDEF0, состоящая из контекстной диаграммы и декомпозиций 4 уровней.

Изученные деятельность организации и процесс принятия решения позволили разработать базу данных, хранящую данные о пользователях, автомобилях, заказах и др.

Реализованы проверки на различные виды ввода, и при неправильном вводе пользователь получает оповещение-подсказку, помогающую ему сориентироваться и исправить ввод.

Для разработки программного средства были смоделированы алгоритмы, реализующие бизнес-логику программного средства. Для улучшения взаимодействия пользователя с программным средством были предоставлены руководства по развёртыванию программного средства и его работе.

Программа оснащена понятным и удобным интерфейсом, а также простой системой навигации и необходимыми функциями для решения основной задачи курсового проекта. В данном приложении реализованы базовые принципы объектно-ориентированного программирования, использованы стандартные и пользовательские функции, была реализована работа с базами данных, взаимодействие между сервером и клиентом.

Для улучшения взаимодействия пользователя с программным средством были предоставлены руководства по развёртыванию программного средства и его работе.

По завершении разработки было проведено полное тестирование, в результате которого не было выявлено критических сбоев в работе программного средства, что подтверждает работоспособность программы.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Прохоров Ю.К., Фролов В. В. Управленческие решения: Учебное пособие. – 2-е изд., испр. и доп. – СПб: СПбГУ ИТМО, 2011 – 138 с.
- [2] Диаграмма вариантов использования [Электронный ресурс]. – Режим доступа: <http://www.informicus.ru/default.aspx>.
- [3] JavaRush [Электронный ресурс]. – Режим доступа <http://www.javarush.ru/samouchitel.html>.
- [4] Программирование на языке Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/>.
- [5] Java Learning [Электронный ресурс]. – Режим доступа: [программирование на языке Java .pdf](#).
- [6] Полное руководство по обучению программирования на Java | Java [Электронный ресурс] – Режим доступа: <http://www.internet-technologies.ru/articles/kak-nauchitsya-programmirovat-na-java.html>.
- [7] Java SoloLearn [Мобильное приложение] – Режим доступа: Play Market
- [8] Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. – 2-е изд. – СПб.: Питер, 2006. – 500с.
- [9] Гома, Х. UML Проектирование систем реального времени, параллельных и распределенных приложений/ Слинкин, А.– М. Издательство ДМК Пресс, 2016. – 480 с.
- [10] Скот, К. UML. Основы / С. Орлов. – Третье изд. – СПб: Питер, 2016–736 с.

# ПРИЛОЖЕНИЕ А

## (обязательное)

### Отчет о проверке на заимствования в системе «Антиплагиат»



#### Отчет о проверке на заимствования №1



Автор: ID: 10370422  
Проверяющий:

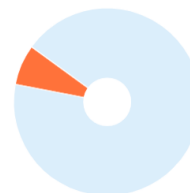
Отчет предоставлен сервисом «Антиплагиат» - <http://users.antiplagiat.ru>

#### ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 2  
Начало загрузки: 12.05.2023 21:21:15  
Длительность загрузки: 00:00:01  
Имя исходного файла: Курсач.pdf  
Название документа: Курсач  
Размер текста: 56 кБ  
Символов в тексте: 57588  
Слов в тексте: 6721  
Число предложений: 437

#### ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 12.05.2023 18:21:17  
Длительность проверки: 00:00:04  
Комментарии: не указано  
Модули поиска: Интернет Free



СОВПАДЕНИЯ

6,53%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

0%

ОРИГИНАЛЬНОСТЬ

93,47%

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

**ПРИЛОЖЕНИЕ Б**  
**(обязательное)**  
**Листинг скрипта генерации базы данных**

```
CREATE SCHEMA IF NOT EXISTS `car-rent` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci ;

USE `car-rent` ;

CREATE TABLE IF NOT EXISTS `car-rent`.`car` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `brand` VARCHAR(255) NULL DEFAULT NULL,
  `engine_capacity` VARCHAR(255) NULL DEFAULT NULL,
  `is_maintained` VARCHAR(255) NULL DEFAULT NULL,
  `location` VARBINARY(255) NULL DEFAULT NULL,
  `name` VARCHAR(255) NULL DEFAULT NULL,
  `plate_number` VARCHAR(255) NULL DEFAULT NULL,
  `seats_quantity` VARCHAR(255) NULL DEFAULT NULL,
  `stereo` VARCHAR(255) NULL DEFAULT NULL,
  `tariff` VARCHAR(255) NULL DEFAULT NULL,
  `year` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `car-rent`.`rental` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `last_order_time` VARCHAR(255) NULL DEFAULT NULL,
  `order_quantity` VARCHAR(255) NULL DEFAULT NULL,
  `order_sum` VARCHAR(255) NULL DEFAULT NULL,
  `rental_date` VARCHAR(255) NULL DEFAULT NULL,
  `rental_time` VARCHAR(255) NULL DEFAULT NULL,
```



```

    PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO_INCREMENT = 4

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `car-rent`.`user` (
    `id` BIGINT NOT NULL AUTO_INCREMENT,
    `date_of_birth` VARCHAR(255) NULL DEFAULT NULL,
    `email` VARCHAR(255) NULL DEFAULT NULL,
    `first_name` VARCHAR(255) NULL DEFAULT NULL,
    `last_name` VARCHAR(255) NULL DEFAULT NULL,
    `password` VARCHAR(255) NULL DEFAULT NULL,
    `phone` VARCHAR(255) NULL DEFAULT NULL,
    `role` VARCHAR(255) NULL DEFAULT NULL,
    `status` VARCHAR(255) NULL DEFAULT NULL,
    PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO_INCREMENT = 6

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `car-rent`.`orders` (
    `id` BIGINT NOT NULL AUTO_INCREMENT,
    `drive_length` VARCHAR(255) NULL DEFAULT NULL,
    `drive_price` VARCHAR(255) NULL DEFAULT NULL,
    `order_date` VARCHAR(255) NULL DEFAULT NULL,
    `order_time` VARCHAR(255) NULL DEFAULT NULL,
    `promocode` VARCHAR(255) NULL DEFAULT NULL,
    `car_id` BIGINT NULL DEFAULT NULL,
    `order_id` BIGINT NULL DEFAULT NULL,
    `user_id` BIGINT NULL DEFAULT NULL,
    PRIMARY KEY (`id`),

```

```

INDEX `FK8ptpdp8fxan4etjis2wkqglea` (`car_id` ASC) VISIBLE,
INDEX `FKbt2nbmdu9pi6afpuptj347c5` (`order_id` ASC) VISIBLE,
INDEX `FKel9kyl84ego2otj2accfd8mr7` (`user_id` ASC) VISIBLE,
CONSTRAINT `FK8ptpdp8fxan4etjis2wkqglea`
    FOREIGN KEY (`car_id`)
    REFERENCES `car-rent`.`car` (`id`),
CONSTRAINT `FKbt2nbmdu9pi6afpuptj347c5`
    FOREIGN KEY (`order_id`)
    REFERENCES `car-rent`.`rental` (`id`),
CONSTRAINT `FKel9kyl84ego2otj2accfd8mr7`
    FOREIGN KEY (`user_id`)
    REFERENCES `car-rent`.`user` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 3
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `car-rent`.`report` (
    `id` BIGINT NOT NULL AUTO_INCREMENT,
    `message` VARCHAR(255) NULL DEFAULT NULL,
    `report_date` VARCHAR(255) NULL DEFAULT NULL,
    `report_time` VARCHAR(255) NULL DEFAULT NULL,
    `car_id` BIGINT NULL DEFAULT NULL,
    PRIMARY KEY (`id`),
    INDEX `FKtbqh5kpliq7t4tnbd176tlxjq` (`car_id` ASC) VISIBLE,
    CONSTRAINT `FKtbqh5kpliq7t4tnbd176tlxjq`
        FOREIGN KEY (`car_id`)
        REFERENCES `car-rent`.`car` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 2
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

## ПРИЛОЖЕНИЕ В

### (обязательное)

#### Листинг кода алгоритмов, реализующих основную бизнес-логику

```
@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/login")
public class LoginController {

    @Autowired
    private UserService userService;

    @PostMapping
    public ResponseEntity login(@RequestBody UserEntity user){
        try {
            return ResponseEntity.ok(userService.login(user));
        } catch(UserPasswordInvalid e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        } catch(ObjDoesNotExist e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        } catch(Exception e){
            return ResponseEntity.badRequest().body("Произошла ошибка: " +
e.getMessage());
        }
    }
}

@Service
public class CarService {
    @Autowired
    private CarRepo carRepo;

    public CarEntity registration (CarEntity car) { return carRepo.save(car);}

    public List<CarEntity> getAllCars() { return carRepo.findAll();}

    public Double[] getLocation (Long id) {
        Optional<CarEntity> optionalCar = carRepo.findById(id);
        CarEntity car = optionalCar.get();
        return car.getLocation();
    }

    public void deleteCar(Long id) { carRepo.deleteById(id);}

    public CarEntity update (CarEntity car) throws ObjDoesNotExist {
        return carRepo.findById(car.getId())
            .map(newCar -> {
                newCar.setBrand(car.getBrand());
                newCar.setEngineCapacity(car.getEngineCapacity());
            })
        .orElseThrow(ObjDoesNotExist::new);
    }
}
```

```

        newCar.setIsMaintained(car.getIsMaintained());
        newCar.setName(car.getName());
        newCar.setPlateNumber(car.getPlateNumber());
        newCar.setSeatsQuantity(car.getSeatsQuantity());
        newCar.setStereo(car.getStereo());
        newCar.setTariff(car.getTariff());
        newCar.setYear(car.getYear());
        newCar.setLocation(car.getLocation());
        return carRepo.save(car);
    }).orElseThrow(() -> new ObjDoesNotExist("Такого автомобиля в
системе не существует"));
    }
}

```

```

@Service
public class OrderService {
    @Autowired
    private OrderRepo orderRepo;
    @Autowired
    private UserRepo userRepo;
    @Autowired
    private CarRepo carRepo;

    public OrderEntity create(OrderEntity order, Long userID, Long carID) {
        UserEntity user = userRepo.findById(userID).get();
        CarEntity car = carRepo.findById(carID).get();

        Date date = new Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("yyyy-MM-dd");
        SimpleDateFormat formatTime = new SimpleDateFormat("kk:mm");
        String strDate = formatDate.format(date);
        String strTime = formatTime.format(date);

        String tariff = car.getTariff();
        String[] driveLengthDelimited = order.getDriveLength().split(":");

        OrderService orderService = new OrderService();
        String promocode = orderService.createPromocode();
        Boolean flag = false;
        if(promocode.equals(order.getPromocode())) flag = true;

        String drivePrice = calculatePrice(driveLengthDelimited, tariff, flag);

        order.setUser(user);
        order.setCar(car);
        order.setOrderDate(strDate);
        order.setOrderTime(strTime);
        order.setDrivePrice(drivePrice);

        return orderRepo.save(order);
    }

    public String createPromocode() {

```

```

        Date date = new Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("EEEE", Locale.US);
        String strDate = formatDate.format(date).trim();

        String promocode = "";

        if(strDate.equals("Monday")){promocode = "MNDAY";}
        if(strDate.equals("Tuesday")){promocode = "TSDAY";}
        if(strDate.equals("Wednesday")){promocode = "WSDAY";}
        if(strDate.equals("Thursday")){promocode = "THDAY";}
        if(strDate.equals("Friday")){promocode = "FRDAY";}
        if(strDate.equals("Saturday")){promocode = "STDAY";}
        if(strDate.equals("Sunday")){promocode = "SNDAY";}

        return promocode;
    }

    public String calculatePrice(String[] driveLength, String tariff, Boolean
flag) {
        double coef = 0;
        double startPrice = 2.5;

        if(tariff == "Комфорт"){
            startPrice *= 1.5;
            coef = 0.05;
        }
        if(tariff == "Бизнес"){
            startPrice *= 2;
            coef = 0.01;
        }
        else{
            coef = 0.0025;
        }

        Byte hB = Byte.valueOf(driveLength[0]);
        Byte mB = Byte.valueOf(driveLength[1]);
        Byte sB = Byte.valueOf(driveLength[2]);

        double h = (double) hB;
        double m = (double) mB;
        double s = (double) sB;

        double time = h * 3600 + m * 60 + s;

        if(flag) return String.valueOf((time * coef + startPrice) * 0.7);
        return String.valueOf(time * coef + startPrice);
    }

    public ArrayList<OrderModel> getAllOrders() { return
OrderModel.toModel(orderRepo.findAll());}

    public void deleteOrder(Long id) { orderRepo.deleteById(id);}

    public OrderEntity update(OrderEntity entity) throws ObjDoesNotExist {

```

```

        return orderRepo.findById(entity.getId())
            .map(newOrder -> {
                newOrder.setOrderDate(entity.getOrderDate());
                newOrder.setOrderTime(entity.getOrderTime());
                newOrder.setDrivePrice(entity.getDrivePrice());
                newOrder.setDriveLength(entity.getDriveLength());
                newOrder.setCar(entity.getCar());
                newOrder.setUser(entity.getUser());
                newOrder.setId(entity.getId());
                return orderRepo.save(entity);
            }).orElseThrow(() -> new ObjDoesNotExist("Заказа не
существует"));
    }

}

@Service
public class RentalService {

    @Autowired
    private RentalRepo rentalRepo;

    @Autowired
    private OrderRepo orderRepo;

    public RentalEntity create() {
        RentalEntity rental = new RentalEntity();
        List<OrderEntity> orders = orderRepo.findAll();
        String orderQuantity = String.valueOf(orders.size());
        String orderSum = orderSum(orders);
        String lastOrderTime = orders.get(orders.size() - 1).getOrderTime();

        Date date = new Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("yyyy-MM-dd");
        SimpleDateFormat formatTime = new SimpleDateFormat("kk:mm");
        String strDate = formatDate.format(date);
        String strTime = formatTime.format(date);

        rental.setRentalDate(strDate);
        rental.setRentalTime(strTime);
        rental.setOrderQuantity(orderQuantity);
        rental.setOrderSum(orderSum);
        rental.setLastOrderTime(lastOrderTime);

        return rentalRepo.save(rental);
    }

    public String orderSum(List<OrderEntity> orders){
        double sum = 0;

        for(OrderEntity entity : orders) {
            sum += Double.parseDouble(entity.getDrivePrice());
        }
    }
}

```

```

        return String.valueOf(sum);
    }

    public ArrayList<RentalModel> getAllRentals() { return
RentalModel.toModel(rentalRepo.findAll());}

    public void delete(Long id) { rentalRepo.deleteById(id);}

}

public class RentalModel {
    private Long id;
    private String orderQuantity;
    private String orderSum;
    private String rentalDate;
    private String rentalTime;
    private String lastOrderTime;

    public static ArrayList<RentalModel> toModel(List<RentalEntity> list) {
        ArrayList<RentalModel> resList = new ArrayList<RentalModel>();

        for(RentalEntity entity : list) {
            RentalModel model = new RentalModel();

            model.setId(entity.getId());
            model.setRentalDate(entity.getRentalDate());
            model.setRentalTime(entity.getRentalTime());
            model.setOrderQuantity(entity.getOrderQuantity());
            model.setOrderSum(entity.getOrderSum());
            model.setLastOrderTime(entity.getLastOrderTime());

            resList.add(model);
        }

        return resList;
    }

    public RentalModel() {
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}

```

```

public String getOrderQuantity() {
    return orderQuantity;
}

public void setOrderQuantity(String orderQuantity) {
    this.orderQuantity = orderQuantity;
}

public String getOrderSum() {
    return orderSum;
}

public void setOrderSum(String orderSum) {
    this.orderSum = orderSum;
}

public String getRentalDate() {
    return rentalDate;
}

public void setRentalDate(String rentalDate) {
    this.rentalDate = rentalDate;
}

public String getRentalTime() {
    return rentalTime;
}

public void setRentalTime(String rentalTime) {
    this.rentalTime = rentalTime;
}

public String getLastOrderTime() {
    return lastOrderTime;
}

public void setLastOrderTime(String lastOrderTime) {
    this.lastOrderTime = lastOrderTime;
}

```