МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ИС

КУРСОВАЯ РАБОТА

по дисциплине «Архитектура информационных систем»

Тема: проектирование сервиса каршеринга

Студенты гр. 2375 _____ Усачев Л.Е.
Шитов Б.А.

Преподаватель _____ Водяхо А.И.

Санкт-Петербург
2024

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студенты: Усачев Л.Е., Шитов Б.А.

Группа: 2375

Тема работы: проектирование сервиса каршеринга

Исходные данные: средствами ПО Enterprise Architect спроектировать сервис каршеринга. Сформировать технические требования, архитектурное описание и список тестов для проекта.

Содержание пояснительной записки: введение, требования, архитектурное описание, архитектурное обоснование, модели, UML описание, use case, классы, активности, развертка, тестирование, заключение, список литературы.

Предполагаемый объем пояснительной записки: не менее 25 страниц

Дата выдачи задания: 5.02.2024

Дата сдачи отчёта:

Студенты гр. 2375 _____ Усачев Л.Е.
Шитов Б.А.

Преподаватель _____ Водяхо А.И.

АННОТАЦИЯ

Содержание курсовой работы заключается в проектировании сервиса каршеринга с помощью ПО Enterprise Architect. Были сформированы технические требования, архитектурное описание, диаграммы и список тестов для проекта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: проектирование сервиса каршеринга.

Форма выполнения работы: курсовая работа.

Задание: Разработка базы данных для сервиса каршеринга, в которой хранится информация о автомобилях, их местоположении, состоянии, доступности, а также о клиентах и их арендных операциях. Для каждого автомобиля фиксируется: марка, модель, год выпуска, расход топлива, тип кузова, цвет, текущее местоположение, статус доступности. По каждому клиенту сохраняется следующая информация: ФИО, контактные данные, водительское удостоверение, история аренд. Аренда автомобилей планируется по дням и включает в себя список доступных автомобилей и продолжительность аренды каждого автомобиля. В сервисе каршеринга предусмотрена одна парковка. Поступление новых автомобилей на парковку отслеживается через акты приема-передачи. Каждый акт приема-передачи содержит следующие атрибуты: номер документа, дата документа, парковка, на которую были доставлены автомобили, поставщик, марка и модель автомобиля, количество каждой модели, стоимость аренды за единицу времени и общая сумма за все автомобили.

1.Требования

1.1. Глоссарий

База данных - Набор постоянно хранимых данных, используемых прикладными программными системами.

Система управления базами данных (СУБД) - Программный комплекс, обеспечивающий централизированное хранения данных и предоставляющий приложениям услуги по обработке данных.

Фреймворк - Программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

Use case - Описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды.

Автомобиль - Транспортное средство, доступное для аренды в сервисе каршеринга. Атрибуты автомобиля включают марку, модель, год выпуска, расход топлива, тип кузова, цвет, текущее местоположение и статус доступности.

Клиент - Человек, использующий услуги каршеринга. Данные о клиенте содержат ФИО, контактные данные, водительское удостоверение и историю аренд.

Аренда - Действие или период времени, в течение которого клиент использует автомобиль. Включает в себя продолжительность аренды и список выбранных автомобилей.

Парковка - Место, где хранятся автомобили сервиса каршеринга, когда они не используются. В данной системе предусмотрена одна парковка.

Акт приема-передачи - Документ, фиксирующий поступление новых автомобилей на парковку. Содержит номер документа, дату документа, информацию о парковке, поставщике, марке и модели автомобиля, количество каждой модели, стоимость аренды за единицу времени и общую сумму за все автомобили.

Местоположение автомобиля - Текущее расположение автомобиля, которое может быть использовано для определения его доступности для клиентов.

Статус доступности - Индикатор, показывающий, доступен ли автомобиль для аренды в данный момент.

История аренд - Записи о всех прошлых арендах, совершенных клиентом. Включает информацию о дате аренды, продолжительности и типе арендуемого автомобиля.

Расход топлива - Количество топлива, которое автомобиль потребляет на определенное расстояние (обычно измеряется в литрах на 100 км).

Тип кузова - Категория или стиль кузова автомобиля, например, седан, хэтчбек или внедорожник.

Водительское удостоверение - Официальный документ, подтверждающий право клиента управлять автомобилем.

Поставщик - Компания или индивидуум, предоставляющий автомобили для сервиса каршеринга.

Стоимость аренды за единицу времени - Цена использования автомобиля за определенный период времени (час, день и т.д.).

1.2 Бизнес-требования

1. Система должна обеспечивать управление поставками автомобилей на парковку.
2. Система должна позволять отслеживать наличие автомобилей на парковке.
3. Система должна предоставлять возможность рассчитывать стоимость аренды автомобиля в зависимости от времени использования и километража.
4. Система должна автоматически предлагать доступные автомобили для аренды, исходя из текущих запросов пользователей.

1.3 Пользовательские требования

1. Интерфейс сервиса должен быть интуитивно понятным и удобным в использовании.
2. Доступ к системе разрешен только после прохождения авторизации.
3. Доступ к системе должен быть возможен через интернет, чтобы обеспечить доступность с любого устройства.
4. Пользователи должны иметь возможность изменять и удалять данные своих бронирований.

1.4 Системные требования

1. Система должна поддерживать функции создания, редактирования и удаления данных о бронированиях и автомобилях.
2. База данных должна быть доступна для работы через интернет.

3. Доступ к базе данных должен быть строго регламентирован с разделением прав пользователей.
4. Необходим выделенный сервер для хранения и обработки данных.
5. Система должна гарантировать сохранность данных в случае технических сбоев.

1.5 Функциональные требования

1. Учет базы данных автомобилей, клиентов и парковок.
2. Автоматическое предложение доступных автомобилей для аренды в зависимости от запросов пользователей.
3. Расчет стоимости аренды автомобиля на основе времени и пройденного расстояния.
4. Ввод данных о новых поставках автомобилей на парковку.
5. Мониторинг наличия автомобилей на парковке и оповещение о необходимости пополнения парка.
6. Возможность авторизации администратора для управления системой.
7. Ведение базы данных поставщиков автомобилей.
8. Создание актов приема-передачи при поставке новых автомобилей на парковку.
9. Регистрация новых пользователей в системе.

2. Архитектурное описание

2.1 Архитектурное обоснование

Для разработки нашего веб-приложения мы решили использовать Java и Spring. Java - это высокоуровневый язык программирования, который обладает простым и интуитивно понятным синтаксисом, а также множеством библиотек и фреймворков, что делает его очень удобным для разработки приложений. Spring - это фреймворк для разработки приложений на языке Java, предоставляющий инструменты и библиотеки для упрощения разработки, улучшения производительности и обеспечения безопасности.

Для хранения и обработки информации мы выбрали реляционную систему управления базами данных PostgreSQL. Она обладает высокой производительностью и масштабируемостью, а также имеет множество инструментов для обработки и анализа данных.

Для обеспечения безопасности и надежности при проведении платежей мы выбрали платежный шлюз системы Robokassa. Это позволит нам обеспечить высокий уровень защиты данных и обеспечить безопасность при проведении платежей. Платежный шлюз Robokassa поддерживает множество способов оплаты, что позволит нашим пользователям выбирать наиболее удобный для них способ оплаты.

Наше веб-приложение будет создано с использованием клиент-серверной архитектуры. Клиентская часть приложения будет работать на стороне пользователя и обеспечивать интерфейс для взаимодействия с приложением. Серверная часть приложения будет обрабатывать запросы от клиентской части и взаимодействовать с базой данных и платежным шлюзом. Для взаимодействия между клиентской и серверной частями приложения мы будем использовать протокол HTTPS.

В целом, наше веб-приложение будет обладать высокой производительностью, масштабируемостью и безопасностью, что позволит нам обеспечить нашим пользователям удобный и безопасный опыт использования приложения.

2.2 Модели

В таблице 1. представлены данные и методы работы с ними.

Таблица 1. Данные и методы

Объект	**Методы**	**Свойства**
User	Регистрация, аренда автомобиля, возврат автомобиля, просмотр информации о доступных автомобилях, просмотр истории поездок	ID, ФИО, номер телефона, логин и пароль, статус аренды, история поездок, документы
Car	Отслеживание местоположения автомобиля, бронирование автомобиля, завершение аренды	ID автомобиля, марка и модель автомобиля, географические координаты, статус доступности
RentalHistory	Просмотр истории аренды автомобилей, расчет общей стоимости аренды	ID аренды, ID пользователя, ID автомобиля, координаты начала и окончания поездки, дата начала и окончания аренды, стоимость каждого дня аренды

2.3 UML описание

2.3.1 Use case

Пользователи делятся на два вида:

Администратор. Возможности: проверка документов новых пользователей, редактирование базы данных и состояний пользователей и доступных машин, круглосуточная поддержка пользователей.

Обычный пользователь. Возможности: бронирование доступных для поездки автомобилей, оплата поездок, загрузка документов, возможность связи с технической поддержкой.

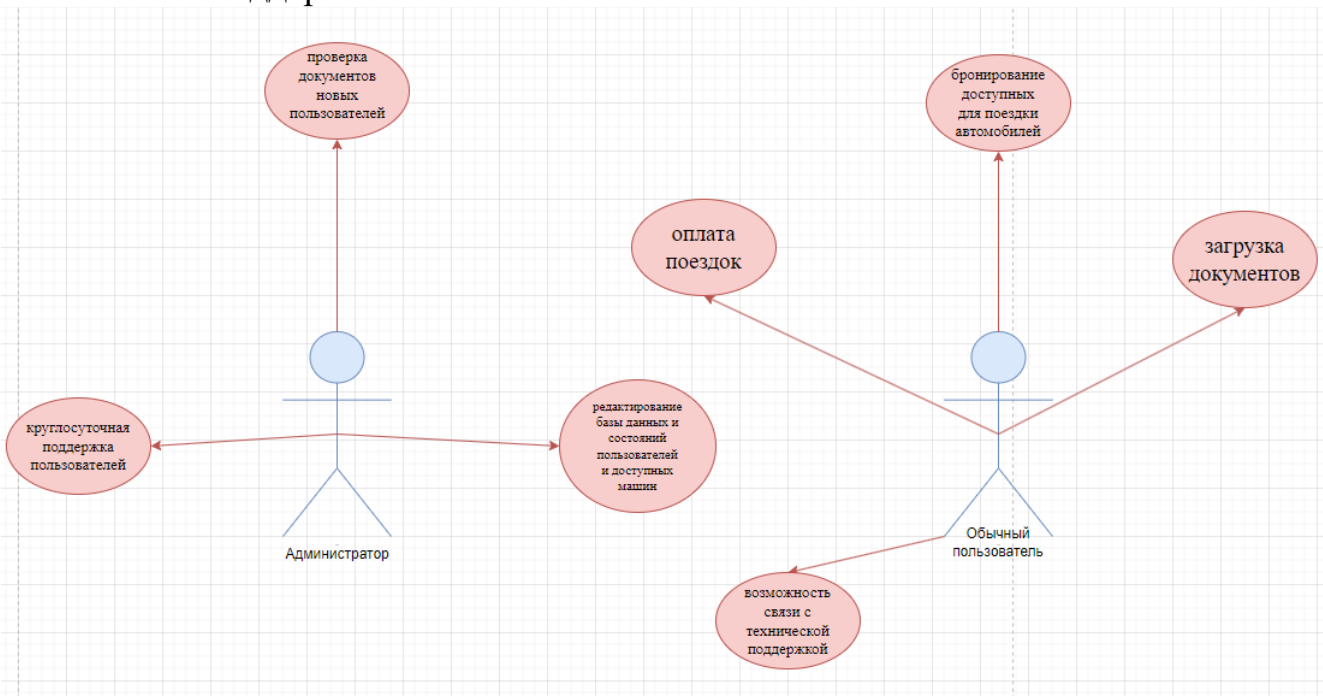

Рисунок 1. Use case диаграмма проекта

Описание основных действий Use case диаграммы:

Описание сценария выполнения варианта использования «Проверка документов новых пользователей» приведено в таблице 2.3.1.1.

Таблица 2.3.1.1. *Проверка документов новых пользователей*

Вариант использования	Проверка документов новых пользователей
Актеры	Администратор
Цель	Проверка документов новых пользователей
Краткое описание	Проверка подлинности документов, предоставляемых пользователями, а также их данных, которые должны отвечать требованиям компании.
Тип	Базовый
Ссылки на другие варианты использования	Отсутствуют

В таблице 2.3.1.2. описывается последовательность действий, приводящая к успешному выполнению варианта использования - Проверка документов новых пользователей

Таблица 2.3.1.2. *Ход действий Проверки документов новых пользователей*

Действия актеров	**Отклик системы**
1. Администратор открывает страницу пользователя с загруженными документами.	2. Система открывает страницу пользователя.
3. Администратор внимательно проверяем все необходимые данные и ставит подтверждения.	4. Система оценивает количество подтвержденных данных и на их основе либо предлагает отказать в регистрации, либо принять документы.
5. Администратор перепроверяет результат и выбирает доступный.	6. Система сохраняет результаты проверки.

Описание сценария выполнения варианта использования «Авторизация пользователя» приведено в таблице 2.3.1.3.

Таблица 2.3.1.3. *Авторизация пользователя*

Вариант использования	Авторизация пользователя
Актеры	Неавторизированный пользователь
Цель	Авторизация неавторизованных пользователей
Краткое описание	Неавторизованный пользователь авторизируется в системе, система, в свою очередь, проверяет корректность введенных данных и сравнивает с базой данных
Тип	Базовый
Ссылки на другие варианты использования	Отсутствуют

В таблице 2.3.1.4. описывается последовательность действий, приводящая к успешному выполнению варианта использования – Авторизации пользователя.

Таблица 2.3.1.4. *Ход действий для авторизации пользователя*

Действия актеров	**Отклик системы**
1. Пользователь вводит логин и пароль, почту, а также отправляет фото документов.	2. Система проверяет, корректны ли введенные данные. Данные сопоставляются с данными в базе, а документы отправляются администратору на проверку. В случае положительного результат пользователь входит в систему

Описание сценария выполнения варианта использования «Редактирование баз данных состояния пользователей и машин» приведено в таблице 2.3.1.5.

Таблица 2.3.1.5. Редактирование баз данных состояния пользователей и машин

Вариант использования	Редактирование баз данных состояния пользователей и машин
Актеры	Администратор
Цель	Необходимо контролировать неактивных пользователей или пользователей, которые нарушаю правила компании и блокировать их соответственно. Также важно по отчетам пользователей менять состояние машины, чтобы в случае поломки она была недоступна для других водителей.
Краткое описание	Редактирование баз данных состояния пользователей и машин
Тип	Базовый
Ссылки на другие варианты использования	Отсутствуют

В таблице 2.3.1.6. описывается последовательность действий, приводящая к успешному выполнению варианта использования – Редактирование баз данных состояния пользователей и машин

Таблица 2.3.1.6. Редактирование баз данных состояния пользователей и машин

Действия актеров	**Отклик системы**
1. Администратор выбирает опцию «Редактирование баз данных» в главном меню	2. Система отображает исходную страницу сервиса и дает выбрать тип данных: «Машины», «Пользователи».
3. Менеджер выбирает один из типов данных	4. Система отображает информацию по выбранному типу данных
5. Менеджер добавляет или изменяет информацию в выбранном типе данных и сохраняет	6. Система сохраняет измененные данные в БД

Описание сценария выполнения варианта использования «Оплата поездок» приведено в таблице 2.3.1.7.

Таблица 2.3.1.7. Оплата поездок.

Вариант использования	Оплата поездок
Актеры	Авторизированный пользователь
Цель	Предоставить пользователю возможность оплатить аренду автомобиля.
Краткое описание	Оплатить поездку по завершении
Тип	Базовый
Ссылки на другие варианты использования	Отсутствует

В таблице 2.3.1.8. описывается последовательность действий, приводящая к успешному выполнению варианта использования – Оплата поездок

Таблица 2.3.1.8. *Ход действий для оплаты поездок*

Действия актеров	Отклик системы
1. Авторизованный пользователь открывает страницу и выбирает опцию «Оплата поездки» в главном меню	2. Система отображает страницу со счетчиком, ценой, а также ссылкой на оплату.
3.Пользователь переходит по ссылке и оплачивает поездку. Затем нажимает кнопку проверка оплаты.	4. Система дожидается ответа от сервиса оплаты и подтверждает его пользователю.

Описание сценария выполнения варианта использования «Внесение данных о поставке продуктов на склад» приведено в таблице 2.3.1.9

2.3.1.9. *Внесение данных о поставке продуктов на склад*

Вариант использования	Внесение данных о поставке продуктов на склад
Актеры	Администратор
Цель	Заполнение и сохранение данных о поставках продуктов в БД
Краткое описание	Внесение администратором данных о совершенных поставках продуктов на склад в БД
Тип	Базовый
Ссылки на другие варианты использования	Отсутствуют

В таблице 2.3.1.10 описывается последовательность действий, приводящая к успешному выполнению варианта использования – Внесение данных о поставке продуктов на склад.

Таблица 2.3.1.10. *Ход действий для внесения данных о поставке*

продуктов на склад

Действия актеров	**Отклик системы**
1. Администратор через личный кабинет выбирает опцию «Поставка продуктов на склад»	2. Система спрашивает какие продукты необходимо доставить на склад
3. Администратор вводит данные о поставке продуктов	4. Система сохраняет данные о поставке продуктов в БД

Описание сценария выполнения варианта использования «Бронирование доступных для поездки автомобилей» приведено в таблице 2.3.1.11.

Таблица 2.3.1.11. *Бронирование доступных для поездки автомобилей*

Вариант использования	Бронирование доступных для поездки автомобилей
Актеры	Авторизованный пользователь
Цель	Дать пользователям возможность бронировать доступные для поездки автомобили
Краткое описание	Бронирование доступных для поездки автомобилей
Тип	Базовый
Ссылки на другие варианты использования	Отсутствует

В таблице 2.3.1.12. описывается последовательность действий, приводящая к успешному выполнению варианта использования – Бронирование доступных для поездки автомобилей

Таблица 2.3.1.12. *Ход действий для бронирования доступных для поездки автомобилей*

Действия актеров	**Отклик системы**
1. Пользователь открывает главную вкладку приложения, на которой отображается карта с его местоположением и местоположением доступных автомобилей.	2. Система делает запрос в базу данных, получает список автомобилей, готовых к поездке, и их координаты. Загружает местоположение пользователя, а затем все полученные данные расставляет на карте.
3. Пользователь выбирает понравившийся автомобиль	4. Система выводит всю информацию об автомобиле, а также предлагает забронировать его.
5. Пользователь нажимает забронировать.	6. Система меняет состояние автомобиля в базе данных, и теперь его не видят другие пользователи, также бронь действует некоторое время, поэтому таймер система также запустила.

Описание сценария выполнения варианта использования «Общение с тех-поддержкой» приведено в таблице 2.3.1.13.

Таблица 2.3.1.13 Общение с тех-поддержкой

Вариант использования	Общение с тех-поддержкой
Актеры	Администратор, пользователь
Цель	Предоставить канал связи между сотрудником компании и пользователем.
Краткое описание	Общение с тех-поддержкой
Тип	Базовый
Ссылки на другие варианты использования	Отсутствует

В таблице 2.3.1.14. описывается последовательность действий, приводящая к успешному выполнению варианта использования Общение с тех-поддержкой

Таблица 2.3.1.14. *Ход действий для общения с тех-поддержкой*

Действия актеров	**Отклик системы**
1. Пользователь открывает вкладку с тех-поддержкой.	2. Система отображает вкладку диалога, а также информирует администратора о пользователе.
3. Пользователь отправляет вопрос	4. Система перенаправляет вопрос администратору, который отправляет ответ. Система обрабатывает ответ и выдает его пользователю.
5. Пользователь получает ответ и оставляет отзыв.	6. Система перенаправляет отзыв администратору.

2.3.2 Диаграмма классов

Описание диаграммы классов в данном проекте основными классами являются классы Car, User, Rental, Admin.

Представление класса **Car** (описание автомобиля) в базе данных:

1. id – уникальный индентификатор машины
2. brand – марка автомобиля
3. model – модель автомобиля
4. year – год выпуска автомобиля
5. color – цвет автомобиля
6. fuelType – тип топлива автомобиля
7. pricePerHour – стоимость аренды автомобиля за час
8. isAvailable – доступность автомобиля для аренды
9. coordinates – текущие координаты.

Методы класса **Car**:

1. getCar() – вывод информации об автомобиле
2. getBrand() – вывод марки автомобиля
3. getModel() – вывод модели автомобиля
4. getYear() – вывод года выпуска автомобиля
5. getColor() – вывод цвета автомобиля
6. getFuelType() – вывод типа топлива автомобиля
7. getPricePerHour() – вывод стоимости аренды за час
8. getAvailability() – вывод доступности автомобиля
10. setCar() – изменение информации об автомобиле
11. addCar() – добавление нового автомобиля
12. removeCar() – удаление существующего автомобиля
13. setCoordinates() – установка новых координат.
14. getCoordinates() – получение текущих координат.

Представление класса **User** (описание пользователя) в базе данных:

1. id - уникальный индентификатор пользователя

2. name – имя пользователя

3. email – адрес электронной почты пользователя

4. phone – номер телефона пользователя

5. drivingLicense – номер водительского удостоверения пользователя

6. coordinates – текущие координаты.

Методы класса **User**:

1. getUser() – вывод информации о пользователе

2. getName() – вывод имени пользователя

3. getEmail() – вывод адреса электронной почты пользователя

4. getPhone() – вывод номера телефона пользователя

5. getDrivingLicense() – вывод номера водительского удостоверения пользователя

6. setUser() – изменение информации о пользователе

7. addUser() – добавление нового пользователя

8. removeUser() – удаление существующего пользователя

9. setCoordinates() – установка новых координат.

10. getCoordinates() – получение текущих координат.

Представление класса **Rental** (описание аренды) в базе данных:

1. id - уникальный индентификатор аренды

2. carId – идентификатор арендованного автомобиля

3. userId – идентификатор арендующего пользователя

4. startDate – дата начала аренды

5. endDate – дата окончания аренды

6. totalCost – общая стоимость аренды

Методы класса **Rental**:

1. getRental() – вывод информации об аренде

2. getCarId() – вывод идентификатора арендованного автомобиля

3. getUserId() – вывод идентификатора арендующего пользователя

4. getStartDate() – вывод даты начала аренды

5. getEndDate() – вывод даты окончания аренды

6. getTotalCost() – вывод общей стоимости аренды

7. getRentalsList() – вывод списка всех арендованных автомобилей

8. setRental() – изменение информации об аренде

9. addRental() – добавление новой аренды

10. removeRental() – удаление информации об аренде

Представление класса **Admin** (описание администратора) в базе данных:

1. id - уникальный индентификатор администратора

2. username – имя пользователя администратора

3. password – пароль администратора

Методы класса **Admin**:

1. getAdmin() – вывод информации об администраторе

2. getUsername() – вывод имени пользователя администратора

3. getPassword() – вывод пароля администратора

4. getAdminsList() – вывод списка всех зарегистрированных администраторов

5. setAdmin() – изменение информации об администраторе

6. addAdmin() – добавление нового администратора

7. removeAdmin() – удаление существующего администратора

Между классами Admin и User отношение агрегации1 к 1..*, так как один администратор может управлять несколькими пользователями, но один пользователь обычно имеет дело только с одним администратором. Удаление пользователя не влияет на администратора.

Между классами Admin и Car отношение агрегации1 к 1..*, так как один администратор может управлять несколькими автомобилями, но каждый автомобиль обычно имеет дело только с одним администратором.

Между классами Car и Rental отношение композиции 1 к 1, так как каждая аренда привязана к конкретному автомобилю, и удаление автомобиля приведет к удалению связанных с ним аренд.

Между классами User и Rental отношение агрегации 1 ко многим 1..*, так как один пользователь может арендовать несколько автомобилей, но каждая аренда обычно связана только с одним пользователем.

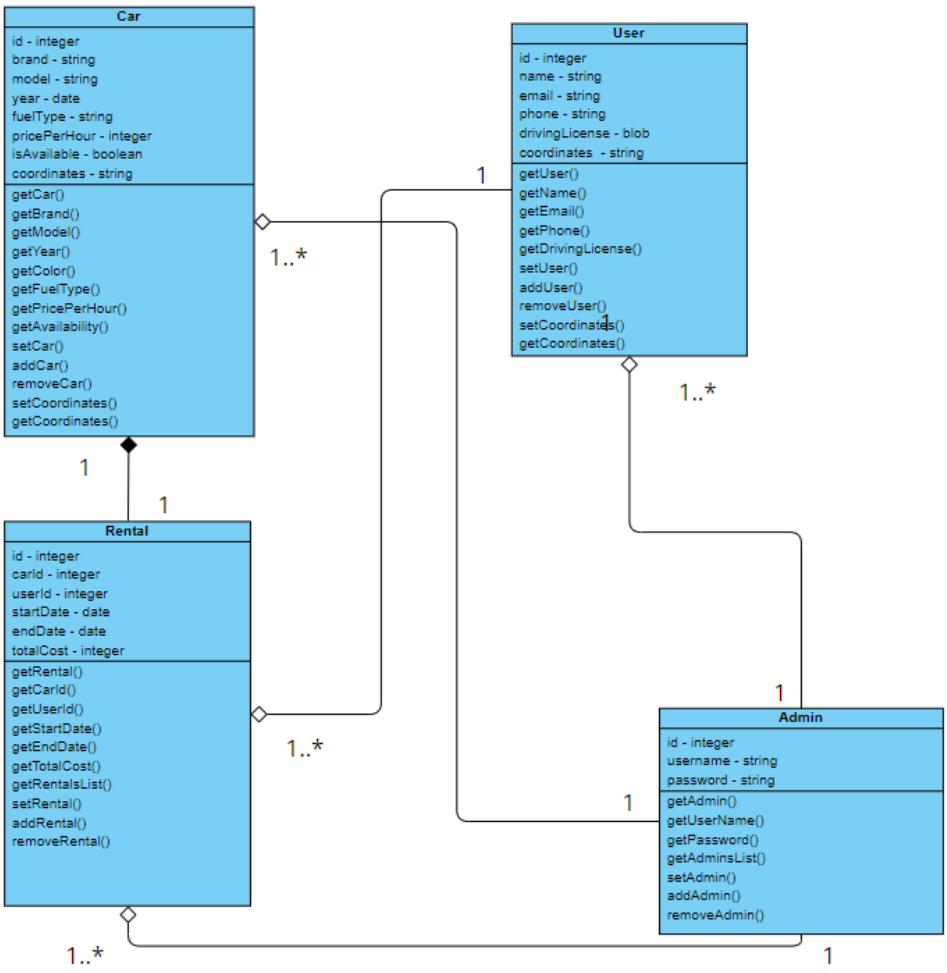

Рисунок 2 - Диаграмма классов

2.3.3 Диаграмма активности.

Описание диаграммы:

Диаграмма активности описывает процессы, которые могут проходить в системе управления продуктами и склада кафе. Она начинается с процесса авторизации или регистрации для незарегистрированных пользователей, которую осуществляет администратор.

При регистрации в системе администратор должен ввести данные о новом пользователе, включая логин, пароль и адрес электронной почты, необходимые документы. После успешной регистрации пользователь может авторизоваться в системе и получить доступ к функциям, предоставляемым системой.

Авторизованные пользователи имеют возможность бронировать автомобили и использовать их, совершать оплату, писать в тех-поддержку.

Администраторы могут поддерживать и решать проблемы пользователей, изменять состояние машин и пользователей, а также проверять документы.

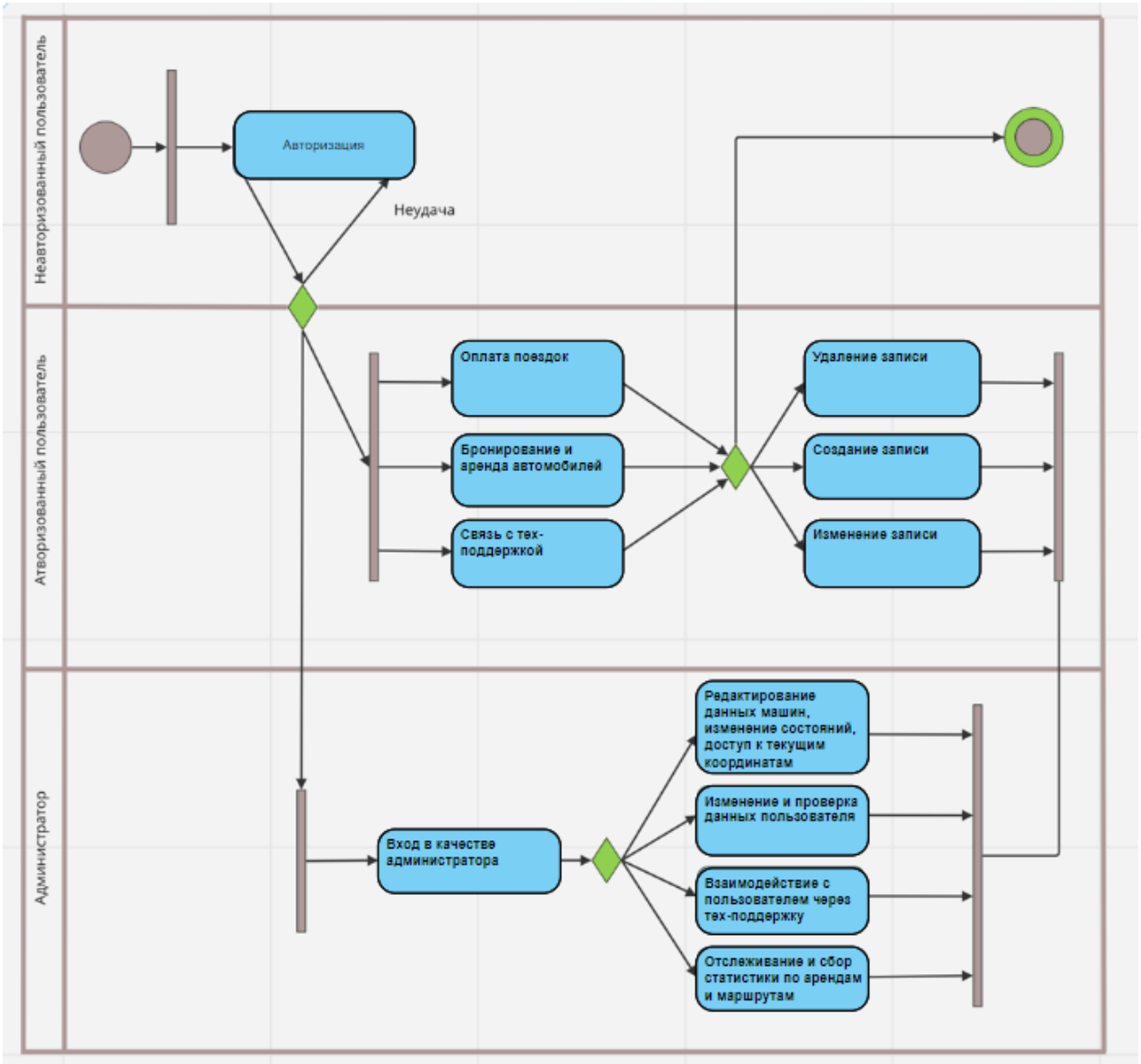

Рисунок 3 – Диаграмма активности.

2.3.4 Диаграмма последовательностей

Для сервиса каршеринга можно построить UML диаграмму последовательности, которая покажет взаимодействие между пользователями, автомобилями и администраторами при выполнении определенных операций. Ниже приведен пример описания для такой диаграммы:

На диаграмме последовательности для сервиса каршеринга можно увидеть следующие актеры: пользователь и администратор. Предполагается, что пользователь может забронировать автомобиль через приложение, выбрав нужное время и место аренды. Но прежде администратор подтверждает его аутентификацию.

Затем пользователь выбирает машину, и происходит проверка ее доступности, а также получение необходимых данных по машине. После чего пользователь заключает договор и начинает поездку. Статус машины меняется – теперь она занята. По окончании поездки пользователь завершает аренду, машина снова становится свободной, а пользователь получает данные, связанные с оплатой. Также администратор получает данные, связанные с окончанием поездки.

Таким образом, UML диаграмма последовательности для сервиса каршеринга поможет описать взаимодействие между пользователями, автомобилями и администраторами при выполнении различных операций.

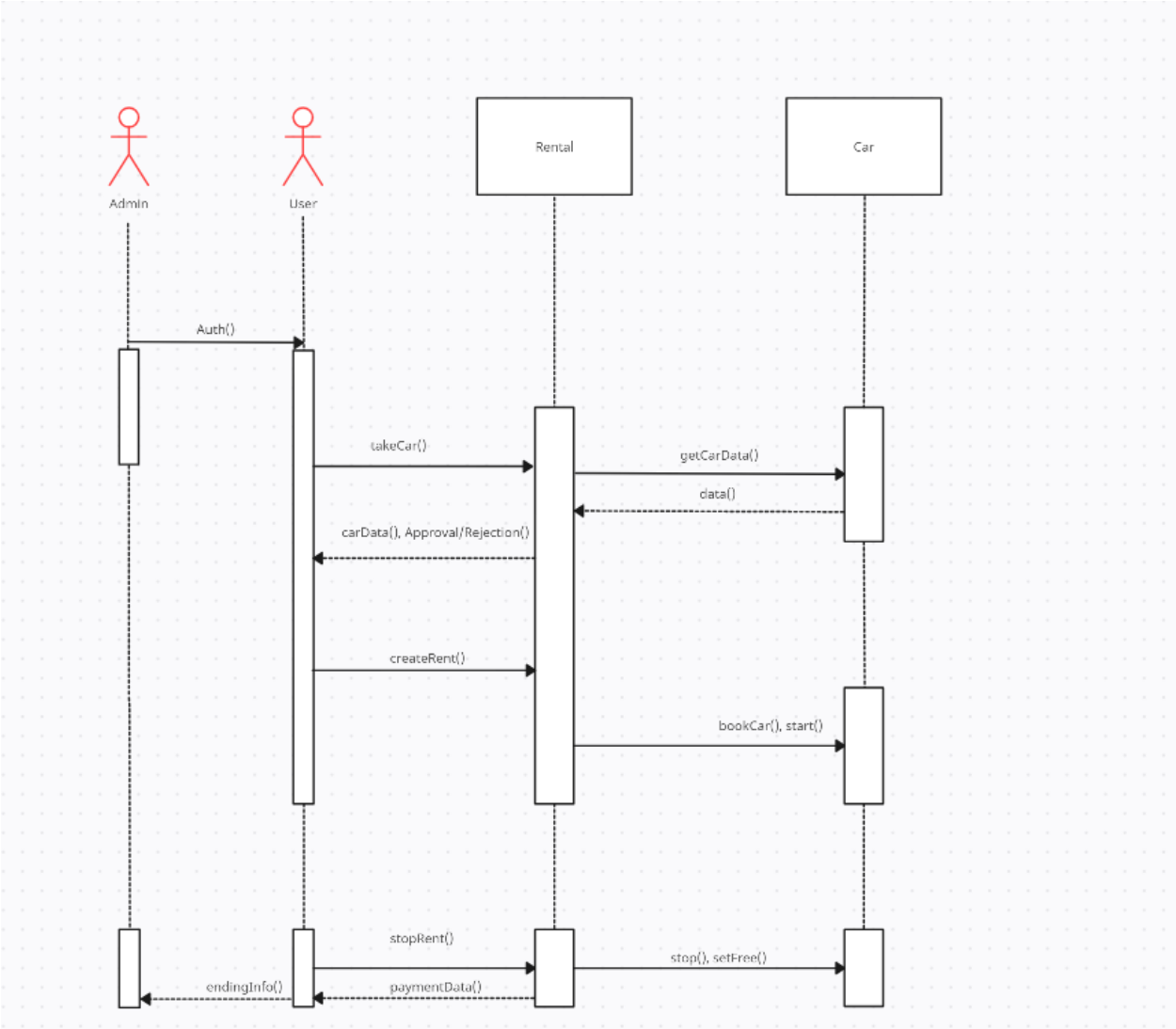

Рисунок 4 – Диаграмма последовательности

3. Тестирование

Функциональность готового сервера должна быть проверена с помощью тестирования следующих компонентов системы:

1. Проверка БД на ведении данных об автомобилях, пользователях и бронированиях
2. Правильность расчета стоимости аренды и доступности автомобилей
3. Проверка правильности и возможности внесения данных о техническом состоянии автомобилей
4. Работоспособность контроля доступности автомобилей и работы выдачи предупреждений о неисправностях
5. Правильность работы ведения БД о партнерах и автопарке
6. Возможность регистрации и авторизации пользователей
7. Корректность создания договоров аренды автомобилей

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы был разработан сервис для компании каршеринга. Были решены следующие задачи: изучены современные технологии разработки серверов для предприятий на примере различных CMS; принято решение использовать архитектуру MVC, что способствовало созданию гибкого веб-приложения; определены бизнес-требования, пользовательские, системные и функциональные требования; разработаны критерии тестирования сервера, описаны возможности пользователей и администраторов, построены диаграммы и тесты для оценки качества и эффективности работы сервера.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кораблев Ю. А., Сирота Д. Д., Архитектура информационных систем. – СПб: Издательство СПбГЭТУ «ЛЭТИ», 2016. – 58 с.
2. Водяхо А.И., Выговский Л. С, Дубенецкий В.А. Цехановский В.В., Архитектурные решения информационных систем. - СПб.: Издательство «Лань»,20 с.
3. Учебник для вузов по направлению подготовки 230400 «Информационные системы и технологии». – Москва: Издательство «Академия», 2012. – 283 с.