Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка базы данных сервиса краткосрочной аренды городских электросамокатов (кикшеринг)»

Студент ИУ7-65Б (Группа) _____ (Подпись, дата) М. Д. Котцов (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) М. В. Филиппов (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 40 с., 9 рис., 6 табл., 0 источн., 1 прил.
Ключевые слова: базы данных, SQL, REST API, PostgreSQL, TypeScript, городской транспорт, кикшеринг, электросамокаты.

Объектом разработки является база данных сервиса краткосрочной аренды городских электросамокатов (кикшеринг).

Цель работы — спроектировать и разработать базу данных для хранения данных сервиса кикшеринга.

Для достижения данной цели были решены следующие задачи:

- проанализированы варианты представления данных и выбран подходящий вариант для решения задачи;
- проанализированы системы управления базами данных и выбрана подходящая система для хранения данных;
- спроектирована база данных, описаны ее сущности и связи;
- реализован интерфейс для доступа к базе данных;
- реализовано программное обеспечение, позволяющее взаимодействовать со спроектированной базой данных.

В результате выполнения работы была спроектирована и разработана база данных для хранения данных сервиса по аренде электросамокатов. Также было разработано приложение для доступа к этим данным.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Бурный рост мегаполисов предъявляет все более строгие требования к городской мобильности. Ответом на современные вызовы стали сервисы кикшеринга (англ. kicksharing), в которых пользователи могут арендовать электрические самокаты, велосипеды или другие транспортные средства на короткий период времени.

Основной отличительной чертой кикшеринга является возможность аренды транспорта на несколько минут или часов, что делает его удобным для передвижения по городу на короткие расстояния. Кикшеринг становится все более популярным в городах по всему миру, так как он представляет собой экологически чистый и удобный способ передвижения.

Массовое использование электросамокатов в качестве городского транспорта началось сравнительно недавно: в России сервисы проката электросамокатов появились в 2018 году. Сегодня арендовать электросамокаты можно в более чем 100 городах страны [2].

В связи с растущей популярностью подобных сервисов [4] все чаще возникает проблема хранения операционных данных о поездках, пользователях, парке электросамокатов и многом другом.

Целью данной работы является проектирование и разработка базы данных для хранения данных сервиса краткосрочной аренды электросамокатов.

Для достижения данной цели необходимо решить следующие задачи:

- проанализировать варианты представления данных и выбрать подходящий вариант для решения задачи;
- проанализировать системы управления базами данных и выбрать подходящую систему для хранения данных;
- спроектировать базу данных, описать ее сущности и связи;
- реализовать интерфейс для доступа к базе данных;
- реализовать программное обеспечение, позволяющее взаимодействовать со спроектированной базой данных.

1 Аналитический раздел

В данном разделе будут выдвинуты требования к приложению, определены пользователи системы, формализованы хранимые сервисом данные. Также будет проведен анализ существующих решений и выбрана модель базы данных.

1.1 Анализ предметной области

Предметной областью поставленной задачи является операционная деятельность сервиса краткосрочной аренды электросамокатов в крупном российском городе.

Пользователями кикшеринга могут стать лица, достигшие возраста 18 лет [3][1][7]. Для регистрации необходим номер телефона и реквизиты банковской карты. Для отправки чеков может потребоваться указать адрес электронной почты.

Для поиска доступных для аренды электросамокатов пользователи используют специально разработанное программное обеспечение (в большинстве случаев это мобильное приложение), где на карте отмечены свободные самокаты. В этом же приложении пользователи кикшеринг-сервиса выбирают самокат и или бронируют его, или начинают на нем поездку.

Бронирование позволяет скрыть электросамокат с карты для других пользователей севриса на определенное время. В большинстве кикшеринг-сервисов бронирование бесплатно и предусматривает отмену без необходимости начинать аренду. При этом некоторые сервисы могут выдвигать дополнительные условия: например, находиться рядом с бронируемым самокатом.

Для начала аренды пользователи сканируют QR-код на руле электросамоката или вводят его номер в приложении.

Стоимость аренды складывается из двух составляющих: платы за старт и платы за каждую минуту поездки. Некоторые сервисы позволяют приобрести подписку, во время действия которой плата за начало поездок взиматься не будет. В зависимости от спроса на самокаты на конкретной парковке цены могут быть скорректированы как в бóльшую, так и в меньшую сторону. Перед началом аренды на банковской карте пользователя блокируется определенная сумма, которая возвращается после завершения поездки. По желанию пользователя и за дополнительную плату поездка может быть застрахована.

Во время поездки в мобильном приложении пользователю доступна информация о километраже, длительности и стоимости аренды. При попадании в зону с ограничением максимальной скорости движения электросамокат автоматически сбрасывает скорость и не позволяет превысить установленный лимит. В некоторых зонах движение электросамокатов может быть запрещено.

Пользователи также имеют возможность поставить аренду на паузу и заблокировать самокат. При этом во время простоя плата за аренду продолжает начисляться.

Завершение аренды возможно на специальных парковках, отмеченных на карте в мобильном приложении. По завершении аренды у пользователя может быть запрошена фотография припаркованного самоката.

Для автоматизации учета информации о пользователях, поездках, самокатах, тарифах и т. д. разрабатывается специализированная информационная система. Использование данной системы возможно пользователями с различными уровнями доступа.

1.2 Анализ существующих решений

Для проведения анализа существующих решений среди имеющихся на рынке сервисов кикшеринга были выбраны три наиболее популярных [5]: Whoosh [9], Юрент [8] и Яндекс Go [6]. Сравнение проводилось по следующим критериям.

1. Максимальное количество одновременных аренд на одном аккаунте.
2. Возможность забронировать электросамокат, находясь на любом расстоянии до него.
3. Необходимость делать фотографии припаркованного самоката после завершения поездки.
4. Возможность поставить аренду на паузу по более низкой цене в минуту, чем во время активной аренды.
5. Возможность посмотреть суммарный километраж всех поездок.

Результат анализа представлен в таблице 1.1.

Таблица 1.1 – Анализ существующих решений

	Whoosh	**Юрент**	**Яндекс Go**
Критерий №1	не более 3	не более 5	не более 3
Критерий №2	+	−	−
Критерий №3	−	+	+
Критерий №4	−	−	+
Критерий №5	+	−	−

Создаваемый в рамках курсовой работы сервис должен не просто быть на уровне с конкурентами, но и предоставлять пользователям дополнительный функционал. Конкурентными преимуществами станут:

- возможность взять в аренду до 10 самокатов одновременно;
- возможность бронировать самокат с любого расстояния;
- отсутствие необходимости делать фотографию после завершения аренды.

1.3 Требования к разрабатываемой базе данных и приложению

Предметная область поставленной задачи является обширной и включает в себя множество понятий и связей между ними, поэтому были сформулированы следующие требования к разрабатываемой в рамках курсовой работы программе.

1. Должен быть предоставлен функционал для регистрации и авторизации пользователей в системе.
2. После регистрации пользователя в системе для него должен быть создан личный кабинет с основной информацией.
3. Должен быть предоставлен функционал для получения списка парковок, зон ограничения максимальной скорости и доступных для аренды электросамокатов.

4. Должен быть предоставлен функционал для бронирования электросамокатов.

5. Должен быть предоставлен функционал для начала и завершения аренды, а также управления электросамокатом во время поездки.

6. Должен быть предоставлен функционал для ограничения скорости электросамокатов в определенных зонах города.

7. Должен быть предоставлен функционал для сохранения истории поездок с возможностью просмотра пользователем своих поездок.

Также были сформулированы следующие допущения:

1. Верификация номера телефона пользователя, равно как и отправка на него СМС-сообщений реализуется сторонним сервисом, предоставляющим программный интерфейс для взаимодействия (API). Реализация подобного функционала выходит за рамки курсовой работы по базам данных.

2. Верификация адреса электронной почты пользоователя путем отправки на него письма со ссылкой для подтверждения выходит за рамки курсовой работы по базам данных.

3. Оплата поездок производится сторонним платежным сервисом, предоставляющим программный интерфейс для взаимодействия (API). Реализация подобного функционала выходит за рамки курсовой работы по базам данных.

1.4 Формализация данных

На рисунке 1.1 представлена ER-диаграмма сущностей проектируемой базы данных в нотации Чена, описывающая объекты предметной области и их взаимодействие.

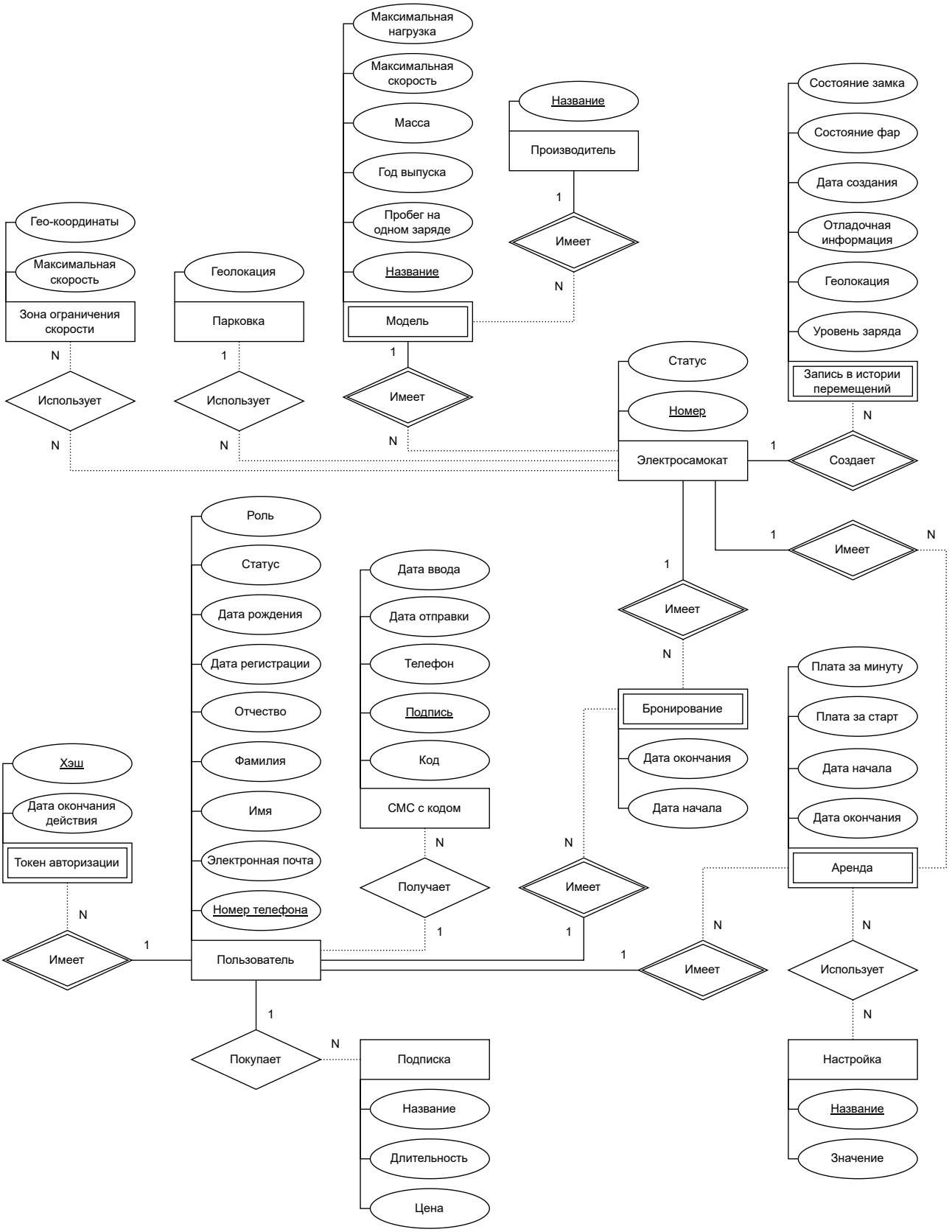

Рисунок 1.1 – ER-диаграмма сущностей проектируемой базы данных

База данных, проектируемая в ходе выполнения курсовой работы, включает в себя информацию о следующих объектах.

1. Пользователь — сущность, описывающая пользователя сервиса.
2. Парковка — сущность, описывающая городскую велопарковку или любое другое место, где можно завершить аренду.
3. Зона ограничения скорости — сущность, описывающая часть города, где максимальная скорость движения ограничена из соображений безопасности.
4. Электросамокат — сущность, хранящая информацию о самокате, который пользователи могут брать в аренду.
5. Запись в истории перемещений — сущность, хранящая информацию о положении электросамоката в пространстве в определенный момент времени.
6. Модель — сущность, описывающая модель электросамоката.
7. Производитель — сущность, описывающая производителя электросамоката.
8. Бронирование — сущность, описывающая процесс резервирования пользователем конкретного самоката на определенный промежуток времени.
9. Аренда — сущность, описывающая процесс проката пользователем электросамоката с фиксированным на момент начала поездки тарифом.
10. Настройка — сущность, хранящая соответствие имени переменной и некоторого значения (например, стоимости страховки).
11. Токен авторизации — сущность, которая используется для авторизации запросов от пользователя к серверу для исключения передачи паролей.
12. СМС с кодом — сущность, хранящая информацию об отправленном на номер телефона пользователя сообщении с коротким кодом авторизации.
13. Подписка — сущность, описывающая подписку, с которой пользователь может бесплатно начинать поездки.

1.5 Формализация пользователей приложения

Ролевая модель используется для реализации системы безопасности сервера базы данных и позволяет разрешать или запрещать тем или иным группам пользователей работу с объектами базы данных.

В рамках поставленной задачи были выделены следующие роли.

1. Администратор. Имеет максимальный уровень доступа ко всем данным сервиса.
2. Механик. Имеет доступ к списку самокатов и их местоположению.
3. Клиент. Имеет возможность бронировать самокаты, брать самокаты в аренду, просматривать историю поездок.
4. Электросамокат. Имеет возможность создавать записи в истории перемещений.

1.6 Диаграмма вариантов использования

На рисунках 1.2–1.7 представлены диаграммы вариантов использования системы в соответствии с выделенными типами пользователей.

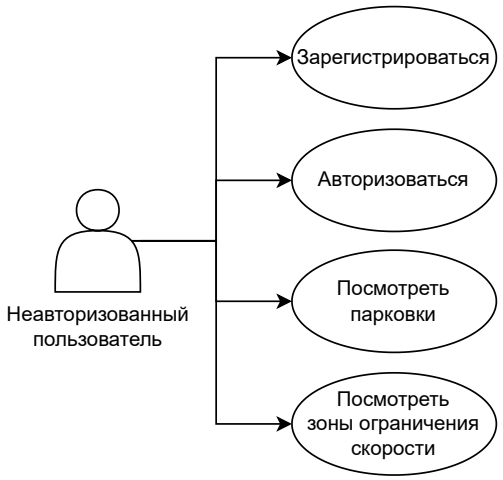

Рисунок 1.2 – Диаграмма вариантов использования системы неавторизованным пользователем

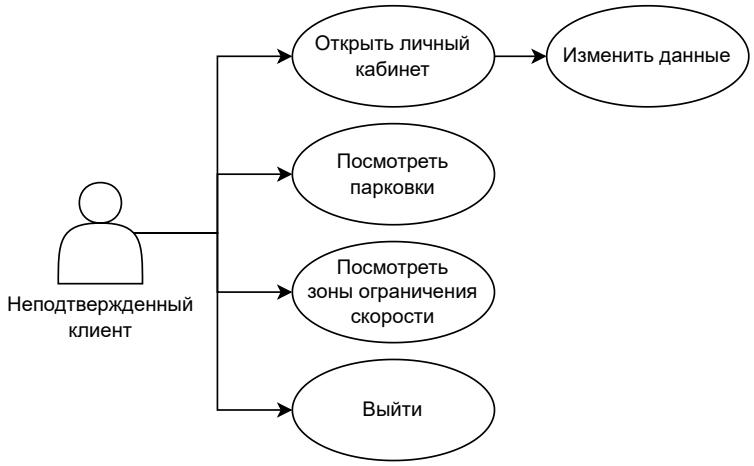

Рисунок 1.3 – Диаграмма вариантов использования системы не подтвердившим свой возраст клиентом

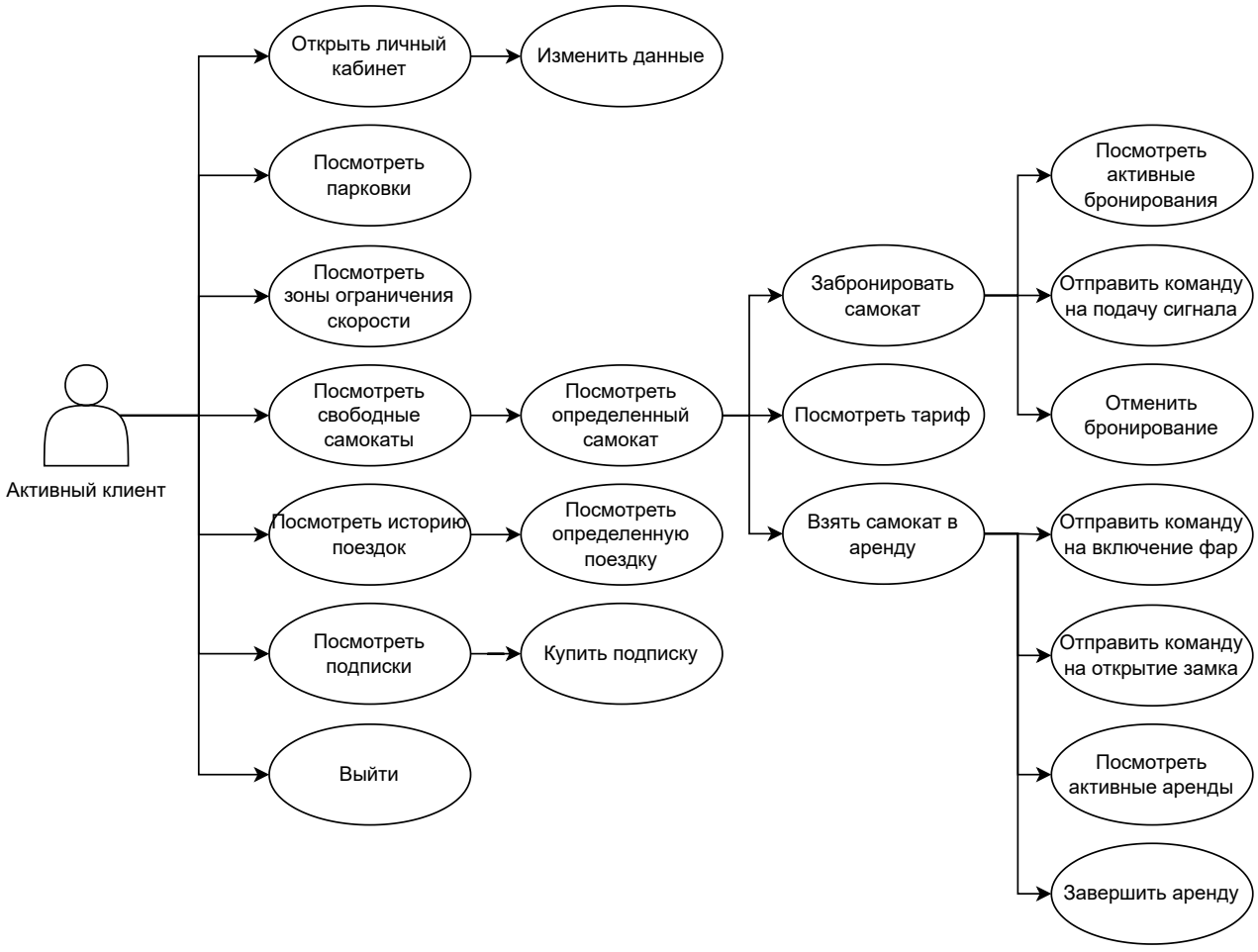

Рисунок 1.4 – Диаграмма вариантов использования системы активным клиентом

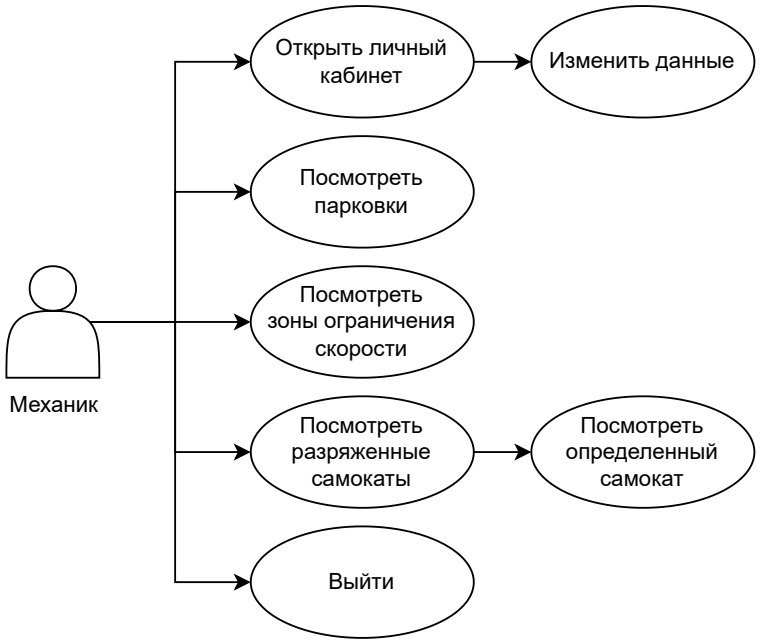

Рисунок 1.5 – Диаграмма вариантов использования системы механиком

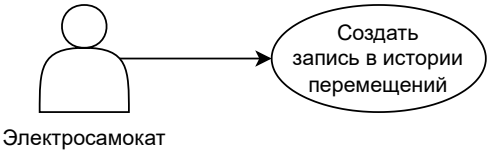

Рисунок 1.6 – Диаграмма вариантов использования системы электросамокатом

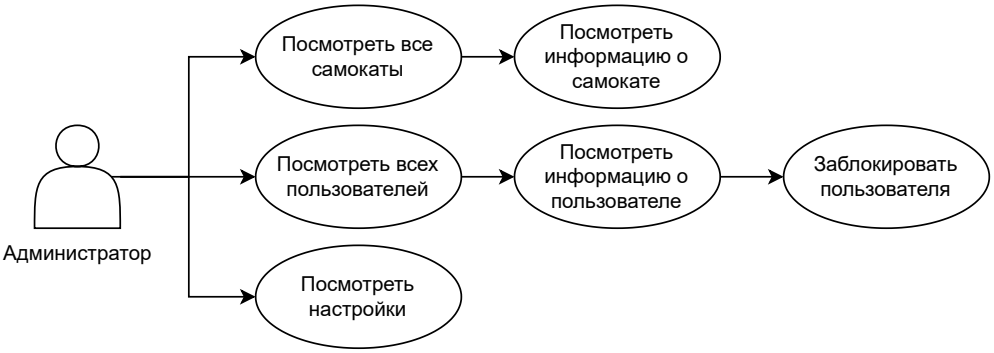

Рисунок 1.7 – Диаграмма вариантов использования системы администратором

1.7 Анализ существующих баз данных

По организации и способу хранения данных все базы данных можно разделить на две группы: реляционные и нереляционные. Реляционные базы данных в свою очередь делятся на строковые и колоночные, а нереляционные — на графовые, документные и базы данных типа «ключ-значение».

1.7.1 Реляционные базы данных

Реляционные базы данных основываются на реляционной модели данных. Данные в таких базах организованы в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. Такие базы данных удобно использовать для хорошо структурированных данных.

Строковые базы данных

Строковыми базами данных называются базы данных, записи которых в памяти представлены построчно. Строковые базы данных используются в транзакционных системах. Для таких систем характерно большое количество коротких транзакций с операциями вставки, обновления и удаления данных.

Колоночные базы данных

Колоночными базами данных называются базы данных, записи которых в памяти представляются по столбцам. Колоночные базы данных используются в аналитических системах. Такие системы характеризуются низким объемом транзакций, а запросы к ним зачастую сложны и включают в себя агрегацию.

1.7.2 Нереляционные базы данных

Нереляционная база данных — это база данных, в которой в отличие от большинства традиционных систем баз данных не используется табличная схема строк и столбцов. В этих базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных.

Базы данных «ключ-значение»

В базах данных «ключ-значение» данные хранятся как совокупность пар ключ-значение, где ключ служит уникальным идентификатором. Такие базы данных удобно использовать для хранения и обработки разных по типу и содержанию данных, их легко масштабировать. Однако такой тип баз данных не подходит для работы со сложными и связанными друг с другом данными.

Документные базы данных

Документные базы данных — это тип нереляционных баз данных, предназначенный для хранения и запроса данных в виде документов в формате, подобном JSON. Такие базы данных позволяют хранить и запрашивать данные в базе данных с помощью той же документной модели, которая используются в коде приложения. Документные базы данных хорошо подходят для быстрой разработки систем и сервисов, работающих с по-разному структурированными данными. Они легко масштабируются и меняют структуру при необходимости. Однако такие базы данных теряют свою эффективность при решении задач, в которых требуется работа с множеством связанных объектов.

Графовые базы данных

Графовые базы данных — это тип нереляционных баз данных, предназначенный для хранения взаимосвязей между сущностями и навигации по ним. Для хранения сущностей используются узлы, а для хранения их взаимосвязей — ребра. В таких базах отсутствуют ограничения на количество и тип взаимосвязей, которые может иметь узел. Графовые базы данных используются для решения задач, имеющих сложные взаимосвязи между данными. При незначительном количестве связей и больших объемах данных графовые базы данных демонстрируют значительно более низкую производительность.

Для решения поставленной задачи была выбрана реляционная база данных с построчным хранением данных, так как:

- задача предполагает хранение структурированных и связанных между собой данных;
- задача предполагает постоянное добавление и изменение данных;
- задача предполагает быструю отзывчивость на запросы пользователя;
- задача не предполагает выполнения сложных аналитических запросов.

Вывод

В данном разделе были выделены ролевые модели системы, конкретизированы хранимые данные и их связь между собой, построены соответствующие

диаграммы. Также был проведен анализ существующих на рынке решений, который позволил понять, какие особенности стоит добавить в разрабатываемый проект. Был осуществлен выбор модели базы данных.

2 Конструкторский раздел

В данном разделе будет спроектирована база данных, которая требуется для реализации поставленной задачи, и описаны её ограничения. Также будет описана используемая ролевая модель.

2.1 Диаграмма проектируемой базы данных

На рисунке 2.1 представлена диаграмма проектируемой базы данных.

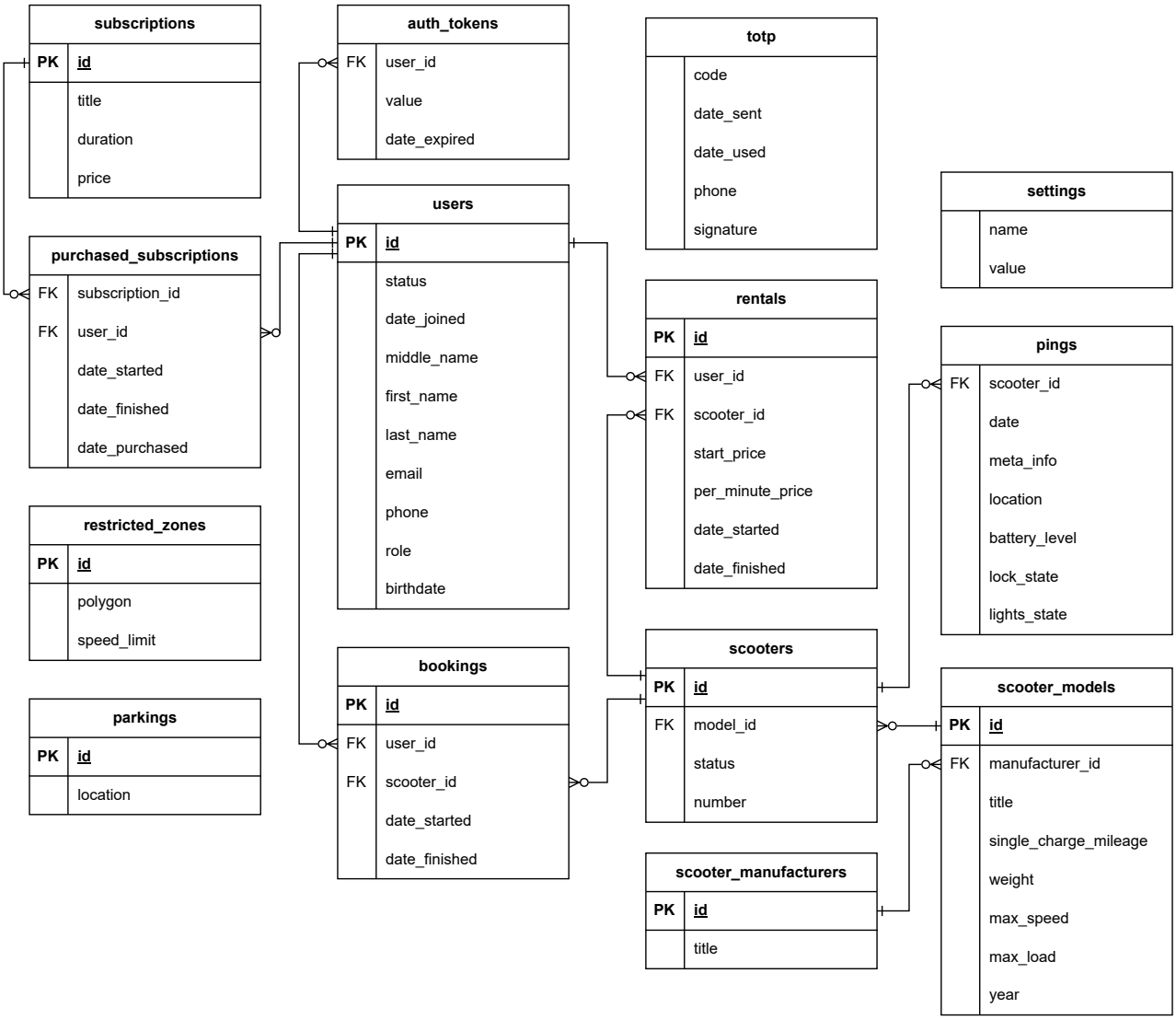

Рисунок 2.1 – Диаграмма проектируемой базы данных

2.2 Сущности проектируемой базы данных

2.2.1 Токен авторизации (таблица auth_tokens)

Сущность токена авторизации содержит следующие поля.

1. user_id — уникальный идентификатор пользователя. Тип: UUID.
2. value — уникальное значение токена. Тип: строка.
3. date_expired — дата окончания действия. Тип: дата со временем.

2.2.2 СМС с кодом (таблица totp)

Сущность СМС с кодом содержит следующие поля.

1. code — случайный код. Тип: целое число.
2. date_sent — дата отправки кода. Тип: дата со временем.
3. date_used — дата использования кода. Тип: дата со временем.
4. phone — номер телефона, на который был отправлен код. Тип: строка.
5. signature — уникальная подпись сообщения. Тип: строка.

2.2.3 Настройка (таблица settings)

Сущность настройки содержит следующие поля.

1. name — название. Тип: строка.
2. value — значение. Тип: текст.

2.2.4 Пользователь (таблица users)

Сущность пользователя содержит следующие поля.

1. id — уникальный идентификатор пользователя. Тип: UUID.
2. status — статус пользователя. Тип: строка. Может принимать одно из трёх возможных значений: «pending» (ожидающий: возраст или не указан, или имеет недопустимое значение), «active» (активный) и «blocked» (заблокированный).
3. date_joined — дата регистрации пользователя в системе. Тип: дата со временем.

4. middle_name — отчество. Тип: строка.
5. first_name — имя. Тип: строка.
6. last_name — фамилия. Тип: строка.
7. email — адрес электронной почты. Тип: регистронезависимая строка.
8. phone — номер телефона. Тип: строка.
9. birthdate — дата рождения. Тип: дата.
10. role — роль пользователя. Тип: строка. Может принимать одно из трех возможных значений: «customer» (клиент), «technician» (механик) и «admin» (администратор).

2.2.5 Аренда (таблица rentals)

Сущность аренды содержит следующие поля.

1. id — уникальный идентификатор аренды. Тип: UUID.
2. user_id — уникальный идентификатор пользователя, который начал аренду. Тип: UUID.
3. scooter_id — уникальный идентификатор арендованного электросамоката. Тип: UUID.
4. start_price — стоимость начала аренды в копейках. Тип: целое число.
5. per_minute_price — стоимость минуты аренды в копейках. Тип: целое число.
6. date_started — дата начала аренды. Тип: дата со временем.
7. date_finished — дата завершения аренды. Тип: дата со временем.

2.2.6 Запись в истории перемещений (таблица pings)

Сущность записи в истории перемещений содержит следующие поля.

1. scooter_id — уникальный идентификатор электросамоката, отправившего информацию. Тип: UUID.
2. date — дата создания записи. Тип: дата со временем.
3. meta_info — дополнительная информация о техническом состоянии электросамоката. Тип: JSON.
4. location — местоположение электросамоката. Тип: географические координаты.
5. battery_level — уровень заряда батареи в количестве процентов. Тип: целое число.
6. lock_state — состояние замка. Тип: строка. Может принимать одно из двух возможных значений: «locked» (закрыт) и «unlocked» (открыт).
7. lights_state — состояние фар. Тип: строка. Может принимать одно из двух возможных значений: «on» (включены) и «off» (выключены).

2.2.7 Зона ограничения скорости (таблица restricted_zones)

Сущность зоны ограничения скорости содержит следующие поля.

1. id — уникальный идентификатор зоны. Тип: UUID.
2. polygon — координаты зоны. Тип: массив географических координат.
3. speed_limit — максимальная скорость в километрах в час. Тип: целое число.

2.2.8 Парковки (таблица parkings)

Сущность парковки содержит следующие поля.

1. id — уникальный идентификатор парковки. Тип: UUID.
2. location — местоположение парковки. Тип: географические координаты.

2.2.9 Бронирование (таблица bookings)

Сущность бронирования содержит следующие поля.

1. id — уникальный идентификатор бронирования. Тип: UUID.
2. user_id — уникальный идентификатор пользователя, создавшего бронирование. Тип: UUID.
3. scooter_id — уникальный идентификатор забронированного электросамоката. Тип: UUID.
4. date_started — дата начала бронирования. Тип: дата со временем.
5. date_finished — дата окончания бронирования. Тип: дата со временем.

2.2.10 Электросамокат (таблица scooters)

Сущность электросамоката содержит следующие поля.

1. id — уникальный идентификатор самоката. Тип: UUID.
2. model_id — уникальный идентификатор модели электросамоката. Тип: UUID.
3. status — статус. Тип: строка. Может принимать одно из двух значений: «enabled» (активный) и «disabled» (деактивированный).
4. number — номер самоката. Тип: строка.

2.2.11 Модель электросамоката (таблица scooter_models)

Сущность модели электросамоката содержит следующие поля.

1. id — уникальный идентификатор модели. Тип: UUID.
2. manufacturer_id — уникальный идентификатор производителя модели. Тип: UUID.
3. title — название модели. Тип: строка.

4. single_charge_mileage — пробег на одном заряде в километрах. Тип: целое число.
5. weight — масса электросамоката в килограммах. Тип: целое число.
6. max_speed — максимальная скорость в километрах в час. Тип: целое число.
7. max_load — максимальная нагрузка в килограммах. Тип: целое число.
8. year — год выпуска. Тип: целое число.

2.2.12 Производитель электросамоката (таблица scooter_manufacturers)

Сущность производителя электросамоката содержит следующие поля.

1. id — уникальный идентификатор производителя. Тип: UUID.
2. title — название. Тип: строка.

2.2.13 Подписка (таблица subscriptions)

Сущность подписки содержит следующие поля.

1. id — уникальный идентификатор подписки. Тип: UUID.
2. title — название подписки. Тип: строка.
3. duration — длительность действия подписки в секундах. Тип: целое число.
4. price — стоимость подписки в копейках. Тип: целое число.

2.2.14 Купленная подписка (таблица purchased_subscriptions)

Сущность купленной подписки содержит следующие поля.

1. subscription_id — уникальный идентификатор купленной подписки. Тип: UUID.

2. user_id — уникальный идентификатор пользователя, купившего подписку. Тип: UUID.

3. date_started — дата начала действия подписки. Тип: дата со временем.

4. date_finished — дата окончания действия подписки. Тип: дата со временем.

5. date_purchased — дата покупки подписки. Тип: дата со временем.

2.3 Ограничения целостности базы данных

Для обеспечения целостности базы данных введены следующие ограничения.

2.3.1 Таблица auth_tokens

Значение столбца user_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы users.

Значение столбца value должно быть уникальным в пределах таблицы. Не может быть пустым.

Значение столбца date_expired не может быть пустым.

2.3.2 Таблица totp

Значение столбца code не может быть пустым.

Значение столбца date_sent не может быть пустым. По умолчанию заполняется отметкой текущего времени.

Значение столбца text не может быть пустым.

Значение столбца signature должно быть уникальным в пределах таблицы. Не может быть пустым.

Значение столбца date_used должно быть позже даты отправки (date_sent).

2.3.3 Таблица settings

Значение столбца name должно быть уникальным в пределах таблицы. Не может быть пустым.

Значение столбца value не может быть пустым.

2.3.4 Таблица users

Значение столбца id является первичным ключом. Не может быть пустым. По умолчанию заполняется случайным UUID.

Значение столбца status может принимать одно из трёх значений: «pending», «active», «blocked». Не может быть пустым. По умолчанию заполняется значением «pending».

Значение столбца date_joined не может быть пустым. По умолчанию заполняется отметкой текущего времени. Дата должна быть позже, чем дата рождения (birthdate).

Значение столбца phone должно быть уникальным в пределах таблицы. Не может быть пустым. Должно иметь формат 7NNNNNNNNN, где N — целое число от 0 до 9.

Значение столбца birthdate должно быть позже 1 января 1930 г.

Значение столбца role может принимать одно из трех возможных значений: «customer», «technician» и «admin». Не может быть пустым. По умолчанию заполняется значением «customer».

2.3.5 Таблица rentals

Значение столбца id является первичным ключом. По умолчанию заполняется случайным UUID. Не может быть пустым.

Значение столбца user_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы users.

Значение столбца scooter_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы scooters.

Значение столбца start_price не может быть пустым. Должно быть не меньше 0.

Значение столбца per_minute_price не может быть пустым. Должно быть не меньше 0.

Значение столбца date_started не может быть пустым. По умолчанию заполняется отметкой текущего времени.

Значение столбца date_finished должно быть позже, чем дата начала (date_started).

2.3.6 Таблица pings

Значение столбца scooter_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы scooters.

Значение столбца date не может бысть пустым. По умолчанию заполняется отметкой текущего времени.

Значение столбца location не может быть пустым.

Значение столбца battery_level не может быть пустым. Должно быть от 0 до 100 включительно.

Значение столбца lock_state может принимать одно из двух значений: «locked» и «unlocked». Не может быть пустым.

Значение столбца lights_state может принимать одно из двух значений «on» и «off». Не может быть пустым.

2.3.7 Таблица restricted_zones

Значение столбца id является первичным ключом. Не может быть пустым. По умолчанию заполняется случайным UUID.

Значение столбца polygon не может быть пустым.

Значение столбца speed_limit не может быть пустым. Должно быть не меньше 0.

2.3.8 Таблица parkings

Значение столбца id является первичным ключом. Не может быть пустым. По умолчанию заполняется случайным UUID.

Значение столбца location не может быть пустым.

2.3.9 Таблица bookings

Значение столбца id является первичным ключом. Не может быть пустым. По умолчанию заполняется случайным UUID.

Значение столбца user_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы users.

Значение столбца scooter_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы scooters.

Значение столбца date_started не может быть пустым. По умолчанию

заполняется отметкой текущего времени.

Значение столбца date_finished не может быть пустым. Дата завершения должна быть позже, чем дата начала (date_started).

2.3.10 Таблица scooters

Значение столбца id является первичным ключом. По умолчанию заполняется случайным UUID. Не может быть пустым.

Значение столбца model_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы scooter_models.

Значение столбца status может принимать одно из двух значений: «enabled» и «disabled». Не может быть пустым. По умолчанию заполняется значением «disabled».

Значение столбца number должно быть уникальным в пределах таблицы. Не может быть пустым.

2.3.11 Таблица scooter_models

Значение столбца id является первичным ключом. По умолчанию заполняется случайным UUID. Не может быть пустым.

Значение столбца manufacturer_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы scooter_manufacturers.

Значение столбца title не может быть пустым.

Значение столбца single_charge_mileage не может быть пустым. Должно быть положительным целым числом.

Значение столбца weight не может быть пустым. Должно быть положительным целым числом.

Значение столбца max_speed не может быть пустым. Должно быть положительным целым числом.

Значение столбца max_load не может быть пустым. Должно быть положительным целым числом.

Значение столбца year не может быть пустым. Должно быть больше 2000.

2.3.12 Таблица scooter_manufacturers

Значение столбца id является первичным ключом. По умолчанию заполняется случайным UUID. Не может быть пустым.

Значение столбца title не может быть пустым.

2.3.13 Таблица subscriptions

Значение столбца id является первичным ключом. По умолчанию заполняется случайным UUID. Не может быть пустым.

Значение столбца title не может быть пустым.

Значение столбца duration не может быть пустым. Должно быть положительным числом.

Значение столбца price не может быть пустым. Должно быть не меньше 0.

2.3.14 Таблица purchased_subscriptions

Значение столбца subscription_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы subscriptions.

Значение столбца user_id не может быть пустым. Является внешним ключом: ссылается на столбец id таблицы users.

Значение столбца date_started не может быть пустым. По умолчанию заполняется отметкой текущего времени.

Значение столбца date_finished не может быть пустым. Дата должна быть позже, чем дата начала (date_started).

Значение столбца date_purchased не может быть пустым. По умолчанию заполняется отметкой текущего времени.

2.4 Описание триггера

На рисунке 2.2 представлена схема алгоритма работы триггера, который не позволяет удалять пользователей, которые имеют активные (незавершенные) аренды.

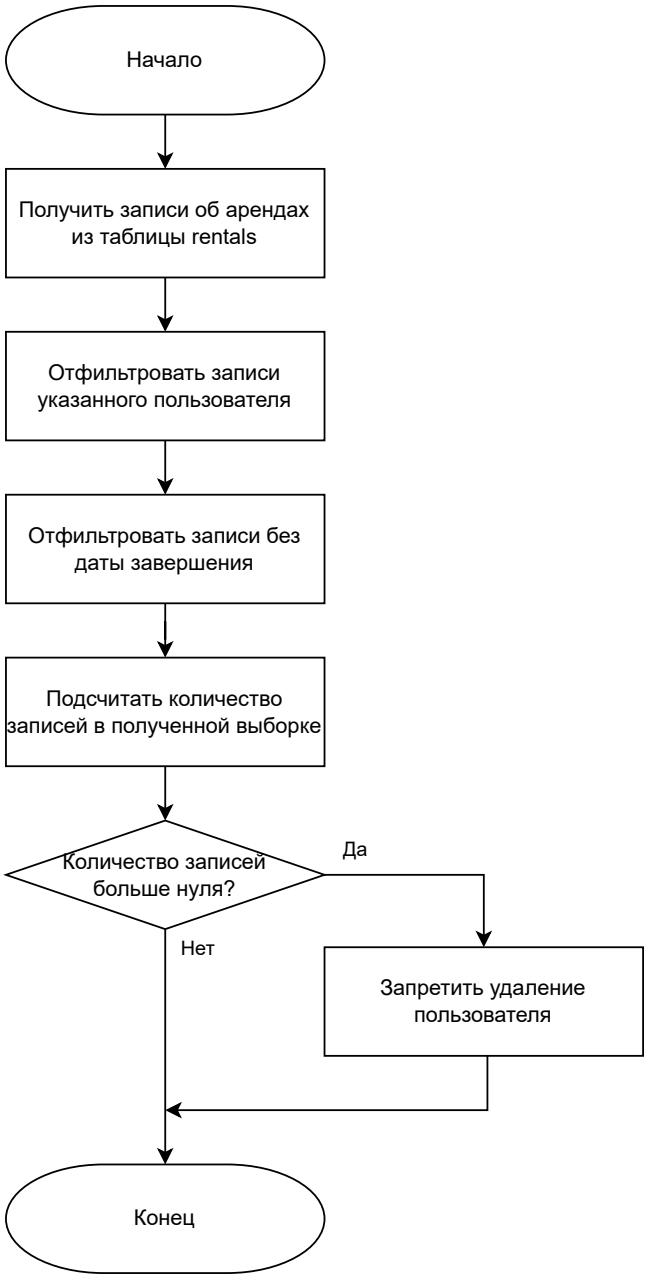

Рисунок 2.2 – Схема алгоритма работы триггера

2.5 Ролевая модель

Спроектированная в соответствии с формализацией пользователей сервиса ролевая модель представлена в таблицах 2.1-2.5.

Таблица 2.1 – Политика доступа не подтвердившего свой возраст клиента

	Чтение	**Вставка**	**Обновление**	**Удаление**
auth_tokens	–	–	–	–
bookings	–	–	–	–
parkings	+	–	–	–
pings	–	–	–	–
purchased_subscriptions	–	–	–	–
rentals	–	–	–	–
restricted_zones	+	–	–	–
scooter_manufacturers	–	–	–	–
scooter_models	–	–	–	–
scooters	–	–	–	–
settings	–	–	–	–
subscriptions	–	–	–	–
totp	–	–	–	–
users	+[1]	–	+[1]	–

[1] Операция доступна только для строки, хранящей информацию об этом клиенте.

Таблица 2.2 – Политика доступа активного клиента

	Чтение	**Вставка**	**Обновление**	**Удаление**
auth_tokens	–	–	–	–
bookings	+[1]	+	+[1]	–
parkings	+	–	–	–
pings	+[2]	–	–	–
purchased_subscriptions	+[3]	+	–	–
rentals	+[4]	+	+[4]	–
restricted_zones	+	–	–	–
scooter_manufacturers	+	–	–	–
scooter_models	+	–	–	–
scooters	+[5]	–	–	–
settings	–	–	–	–
subscriptions	+	–	–	–
totp	–	–	–	–
users	+[6]	–	+[6]	–

[1] Операция доступна только для бронирований, которые были созданы этим клиентом.

[2] Операция доступна только для последних записей электросамокатов, имеющих положительный уровень заряда батареи.

[3] Операция доступна только для подписок, купленных этим клиентом.

[4] Операция доступна только для аренд, которые были начаты этим клиентом.

[5] Операция доступна только для самокатов в статусе «enabled».

[6] Операция доступна только для строки, хранящей информацию об этом клиенте.

Таблица 2.3 – Политика доступа механика

	Чтение	**Вставка**	**Обновление**	**Удаление**
auth_tokens	–	–	–	–
bookings	–	–	–	–
parkings	+	–	–	–
pings	+[1]	–	–	–
purchased_subscriptions	–	–	–	–
rentals	–	–	–	–
restricted_zones	+	–	–	–
scooter_manufacturers	–	–	–	–
scooter_models	–	–	–	–
scooters	+	–	–	–
settings	–	–	–	–
subscriptions	–	–	–	–
totp	–	–	–	–
users	+[2]	–	+[2]	–

[1] Операция доступна только для последних записей электросамокатов, имеющих нулевой уровень заряда батареи (разряженных).

[2] Операция доступна только для строки, хранящей информацию об этом механике.

Таблица 2.4 – Политика доступа электросамоката

	Чтение	**Вставка**	**Обновление**	**Удаление**
auth_tokens	–	–	–	–
bookings	–	–	–	–
parkings	–	–	–	–
pings	–	+	–	–
purchased_subscriptions	–	–	–	–
rentals	–	–	–	–
restricted_zones	–	–	–	–
scooter_manufacturers	–	–	–	–
scooter_models	–	–	–	–
scooters	–	–	–	–
settings	–	–	–	–
subscriptions	–	–	–	–
totp	–	–	–	–
users	–	–	–	–

Таблица 2.5 – Политика доступа администратора

	Чтение	**Вставка**	**Обновление**	**Удаление**
auth_tokens	+	+	+	+
bookings	+	+	+	+
parkings	+	+	+	+
pings	+	+	+	+
purchased_subscriptions	+	+	+	+
rentals	+	+	+	+
restricted_zones	+	+	+	+
scooter_manufacturers	+	+	+	+
scooter_models	+	+	+	+
scooters	+	+	+	+
settings	+	+	+	+
subscriptions	+	+	+	+
totp	+	+	+	+
users	+	+	+	+

Вывод

В данном разделе была спроектирована база данных для разрабатываемого приложения, а так же рассмотрены способы обеспечения ее целостности и ролевая модель.

ПРИЛОЖЕНИЕ А
Создание таблиц базы данных

Листинг А.1 – SQL-скрипт создания таблиц и ограничений базы данных

```
CREATE EXTENSION IF NOT EXISTS citext;
CREATE EXTENSION IF NOT EXISTS postgis;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TYPE user_status AS ENUM ('pending', 'active', 'blocked');

CREATE TYPE user_role AS ENUM ('customer', 'technician',
   'admin');

CREATE DOMAIN phone_number AS TEXT CHECK (VALUE ~
   '^7[0-9]{10}$');

CREATE TABLE IF NOT EXISTS users (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    status user_status NOT NULL DEFAULT 'pending',
    date_joined timestamp with time zone NOT NULL DEFAULT now()
       CHECK (date_joined > birthdate),
    middle_name varchar(256),
    first_name varchar(256),
    last_name varchar(256),
    email citext,
    phone phone_number UNIQUE NOT NULL,
    birthdate date CHECK (birthdate > '1930-01-01'),
    role user_role NOT NULL DEFAULT 'customer'
);

CREATE TABLE IF NOT EXISTS scooter_manufacturers (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    title varchar(256) NOT NULL
);

CREATE TABLE IF NOT EXISTS scooter_models (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    manufacturer_id uuid NOT NULL REFERENCES
       scooter_manufacturers(id),
    title varchar(256) NOT NULL,
```

```
    single_charge_mileage integer NOT NULL CHECK
       (single_charge_mileage > 0),
    weight integer NOT NULL CHECK (weight > 0),
    max_speed integer NOT NULL CHECK (max_speed > 0),
    max_load integer NOT NULL CHECK (max_load > 0),
    year smallint NOT NULL CHECK (year > 2000)
);

CREATE TYPE scooter_status AS ENUM ('enabled', 'disabled');

CREATE TABLE IF NOT EXISTS scooters (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    model_id uuid NOT NULL REFERENCES scooter_models(id),
    status scooter_status NOT NULL DEFAULT 'disabled',
    number varchar(8) UNIQUE NOT NULL
);

CREATE TYPE scooter_lock_state AS ENUM ('locked', 'unlocked');
CREATE TYPE scooter_lights_state AS ENUM ('on', 'off');

CREATE TABLE IF NOT EXISTS pings (
    scooter_id uuid NOT NULL REFERENCES scooters(id),
    date timestamp with time zone NOT NULL DEFAULT now(),
    meta_info json,
    location geography NOT NULL,
    battery_level smallint NOT NULL CHECK (battery_level >= 0
       AND battery_level <= 100),
    lock_state scooter_lock_state NOT NULL,
    lights_state scooter_lights_state NOT NULL
);

CREATE TABLE IF NOT EXISTS rentals (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id uuid NOT NULL REFERENCES users(id),
    scooter_id uuid NOT NULL REFERENCES scooters(id),
    start_price integer NOT NULL CHECK (start_price >= 0),
    per_minute_price integer NOT NULL CHECK (per_minute_price >=
       0),
    date_started timestamp with time zone NOT NULL DEFAULT now(),
    date_finished timestamp with time zone CHECK (date_finished
       > date_started)
```

```
);

CREATE TABLE IF NOT EXISTS bookings (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id uuid NOT NULL REFERENCES users(id),
    scooter_id uuid NOT NULL REFERENCES scooters(id),
    date_started timestamp with time zone NOT NULL DEFAULT now(),
    date_finished timestamp with time zone NOT NULL CHECK
       (date_finished > date_started)
);

CREATE TABLE IF NOT EXISTS parkings (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    location geography NOT NULL
);

CREATE TABLE IF NOT EXISTS restricted_zones (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    polygon geometry NOT NULL,
    speed_limit smallint NOT NULL CHECK (speed_limit >= 0)
);

CREATE TABLE IF NOT EXISTS totp (
    code integer NOT NULL,
    date_sent timestamp with time zone NOT NULL DEFAULT now(),
    phone text NOT NULL,
    signature text UNIQUE NOT NULL,
    date_used timestamp with time zone CHECK (date_used >
       date_sent)
);

CREATE TABLE IF NOT EXISTS auth_tokens (
    user_id uuid NOT NULL REFERENCES users(id),
    value text UNIQUE NOT NULL,
    date_expired timestamp with time zone NOT NULL
);

CREATE TABLE IF NOT EXISTS settings (
    name varchar(64) UNIQUE NOT NULL,
    value text NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS subscriptions (
    id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
    title varchar(64) NOT NULL,
    duration integer NOT NULL CHECK (duration > 0),
    price integer NOT NULL CHECK (price >= 0)
);

CREATE TABLE IF NOT EXISTS purchased_subscriptions (
    subscription_id uuid NOT NULL REFERENCES subscriptions(id),
    user_id uuid NOT NULL REFERENCES users(id),
    date_started timestamp with time zone NOT NULL DEFAULT now(),
    date_finished timestamp with time zone NOT NULL CHECK
       (date_finished > date_started),
    date_purchased timestamp with time zone NOT NULL DEFAULT
       now()
);
```

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Договор о предоставлении права использования Сервиса Юрент — Шеринг самокатов и велосипедов Юрент // Режим доступа: https://urent.ru/docs/accession.html (дата обращения: 15.04.2023).

2. О компании — Шеринг самокатов и велосипедов Юрент // Режим доступа: https://urent.ru/about.html (дата обращения: 08.05.2023).

3. Ответы на вопросы — Шеринг самокатов Whoosh // Режим доступа:

4. Пологойко М. Д. Перспективы использования сервисов проката электросамокатов в повседневных перемещениях по городу // Скиф. Вопросы студенческой науки. — 2021. — №57 (5). — С. 315–319.

5. Прокат самокатов в Москве — Аренда самокатов // Режим доступа: https://arenda-samokatov.ru/prokat-samokatov-v-moskve/ (дата обращения: 19.03.2023). https://whoosh-bike.ru/faq (дата обращения: 25.03.2023).

6. Самокаты — Яндекс Go // Режим доступа: https://go.yandex/ru_ru/lp/rides/scooter (дата обращения: 26.03.2023).

7. Условия аренды электросамоката Яндекс Go — Яндекс Правовые документы // Режим доступа: https://yandex.ru/legal/samokaty/ (дата обращения: 15.04.2023).

8. Шеринг самокатов и велосипедов Юрент // Режим доступа: https://urent.ru/ (дата обращения: 26.03.2023).

9. Шеринг самокатов Whoosh // Режим доступа: https://whoosh-bike.ru/ (дата обращения: 26.03.2023).