# Информационная система сервиса аренды оборудования

Соломенников Николай 22204

## Введение

Сервис аренды оборудования позволяет клиентам временно пользоваться инструментами, техникой или устройствами без необходимости их покупки. Это удобно для тех, кому оборудование нужно на короткий срок.

Клиент ищет оборудование, бронирует его на нужные даты и забирает в пункте выдачи. Менеджер оформляет аренду, выдает оборудование, контролирует его состояние при возврате и взаимодействует с клиентами. Администратор управляет каталогом оборудования, настраивает тарифы и штрафы, а также анализирует отчеты по доходам и популярности оборудования.

## Назначение приложения

Приложение нужно для автоматизации процессов управления арендой оборудования. Оно поможет компании, занимающейся арендой, эффективно управлять запасами оборудования, оформлять аренду, отслеживать возвраты и состояние оборудования, а также вести финансовую отчетность.

**Целевые пользователи:**

* **Администраторы**: управление каталогом оборудования, регулировка цен.
* **Менеджеры по аренде**: взаимодействие с клиентами, оформление аренды, выдача оборудования, контроль сроков возврата.
* **Клиенты**: получение информации о наличии и стоимости оборудования, бронирование.

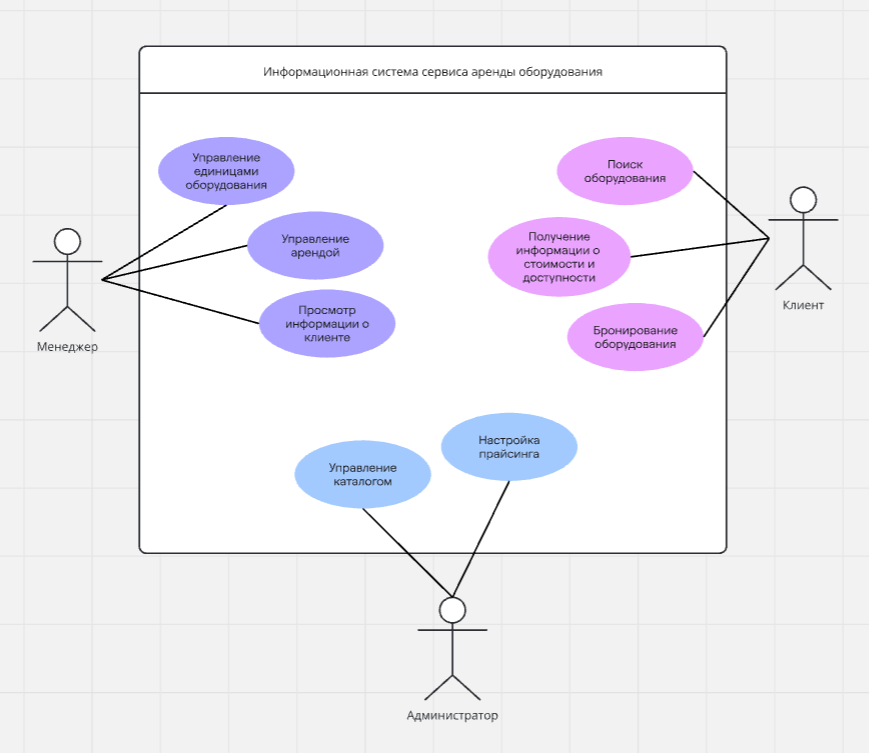
## Полный цикл арендования оборудования

1. **Бронирование оборудования**:
   1. Клиент выбирает оборудование на сайте.
   2. Указывает: ФИО, контактный телефон и email, желаемые даты аренды (начало и конец).
   3. Система резервирует оборудование на указанный срок.
2. **Подготовка к выдаче оборудования**:
   1. Менеджер в пункте выдачи видит информацию о новом бронировании.
   2. Оборудование подготавливается к выдаче (проверяется его состояние, комплектность).
   3. Если клиент не появляется в назначенное время, менеджер связывается с ним для уточнения:
      1. Если клиент отказывается от аренды, бронь отменяется, оборудование снова становится доступным.
      2. Если клиент подтверждает задержку, бронь сохраняется.
3. **Оформление аренды**:
   1. Клиент приходит в пункт выдачи.
   2. Менеджер проверяет документы клиента.
   3. Клиент подписывает договор аренды, в котором указаны: сроки аренды, стоимость аренды, размер залога, условия возврата и штрафы за повреждения или просрочку.
   4. Клиент вносит залог.
   5. Менеджер выдает оборудование клиенту.
4. **Использование оборудования**
   1. Клиент использует оборудование в течение оговоренного срока.
   2. Если клиенту требуется продлить аренду, он может связаться с менеджером для продления сроков возврата.
5. **Возврат оборудования**:
   1. Клиент возвращает оборудование в пункт выдачи и оплачивает аренду за столько дней, сколько он пользовался оборудованием.
   2. Менеджер осматривает оборудование на предмет повреждений:
      1. Если повреждений нет, залог возвращается клиенту.
      2. Если есть повреждения, менеджер оценивает их стоимость и вычитает ее из залога. Остаток залога возвращается клиенту.
   3. Если клиент не вернул оборудование в срок, менеджер связывается с ним для напоминания.
6. **Завершение аренды**:
   1. Менеджер закрывает аренду в системе:
      1. Фиксирует штрафы, если они были.
      2. Обновляет состояние оборудования (например, отправляет его на обслуживание, если это необходимо).
   2. Клиенту предоставляется чек или отчет о завершении аренды.

## Бизнес-процессы системы

* **Оформление аренды**:
  + Поиск доступного оборудования.
  + Оформление договора аренды.
  + Оплата и внесение залога.
  + Получение оборудования.
* **Возврат оборудования**:
  + Проверка состояния оборудования.
  + Начисление штрафов за повреждения.
  + Возврат залога.
  + Обновление статуса оборудования.
* **Взаимодействие с клиентами**:
  + Ответ на вопросы клиентов.
  + Оформление аренды менеджером.
  + Напоминание о сроках возврата.
* **Учет оборудования**:
  + Добавление, обновление и удаление информации об оборудовании.
  + Учет состояния оборудования (доступно, забронировано, в аренде, на обслуживании, списано).
  + Учет технического обслуживания и ремонта.

## Бизнес-процессы для каждого типа пользователей

**Администраторы**:

* Управление каталогом оборудования (добавление / удаление карточек оборудования, редактирование описания).
* Настройка прайсинга (тарифы, залог).

**Менеджеры по аренде**:

* Управление конкретными единицами оборудования (добавление / удаление, изменение статуса (доступно, забронировано, в аренде, на обслуживании, списано)).
* Управление арендой (снятие брони, подтверждение аренды, продление аренды, начисление штрафов)
* Просмотр информации о клиенте, его истории аренд.

**Клиенты**:

* Поиск оборудования.
* Получение информации о его стоимости и доступности.
* Бронирование оборудования.

## Информационные потоки для каждого типа пользователей

**Администраторы**:

* Входящие данные: описание оборудования, тарифы.
* Исходящие данные: истории оборота оборудованием, история аренд.

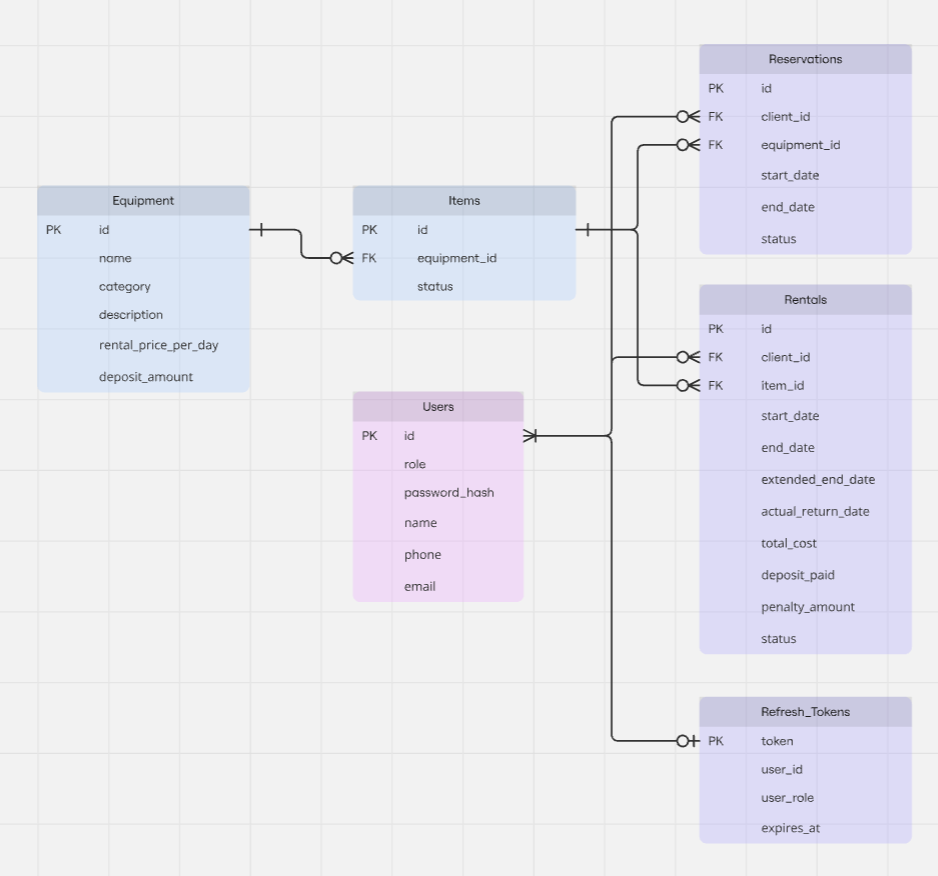
**Менеджеры по аренде**:

* Входящие данные: данные о конкретных единицах оборудования и об арендах.
* Исходящие данные: информация о клиентах, уведомления.
* Отчеты: список активных аренд, просроченные аренды.

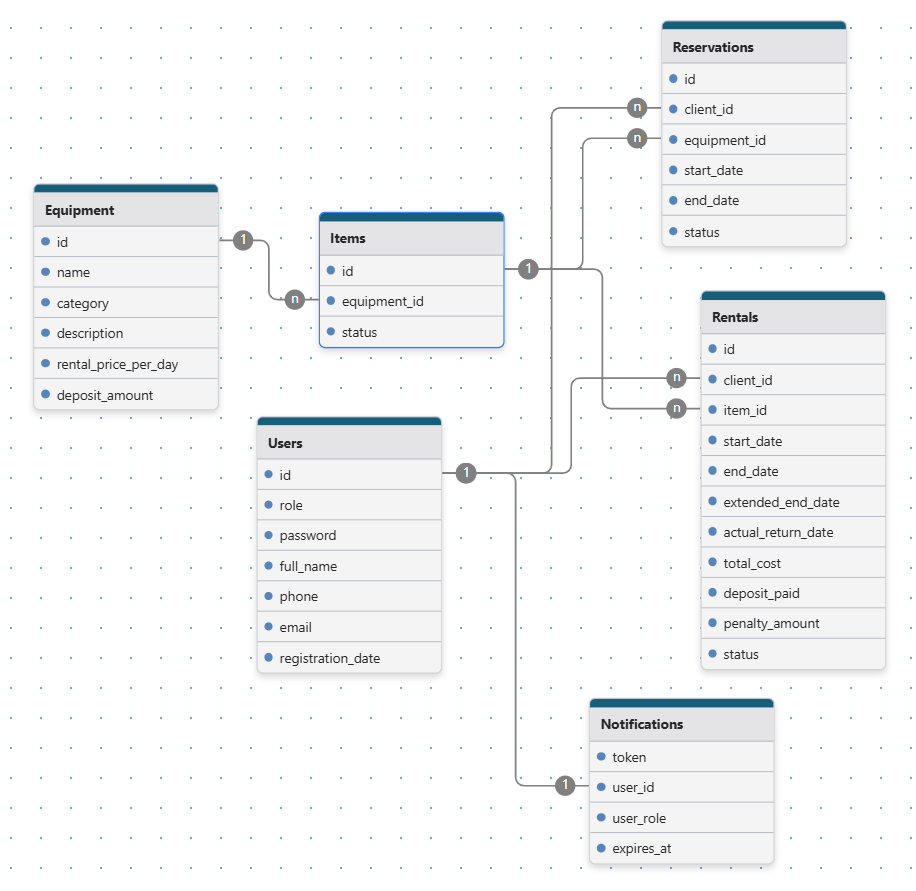
**Клиенты:**

* Входящие данные: личные данные, запросы на аренду.
* Исходящие данные: поиск оборудования, расчет стоимости, уведомления о сроках возврата.

## Основные информационные объекты и связи



## Схема базы данных



1. **Оборудование (Equipment)**:

* id (уникальный идентификатор)
* name (название оборудования)
* category (категория оборудования)
* description (описание)
* rental\_price\_per\_day (стоимость аренды за день)
* deposit\_amount (сумма залога)

1. **Единицы оборудования (Items)**:

(конкретные физические объекты оборудования)

* + id (уникальный идентификатор среди всех оборудований)
  + equipment\_id (внешний ключ на таблицу Equipment)
  + status (статус: доступно, забронировано, в аренде, на обслуживании, списано)

1. **Пользователи (Users)**:
   * id (уникальный идентификатор)
   * User\_role (роль: администратор, менеджер, клиент)
   * Password\_hash (хэш пароля)
   * name (ФИО или просто имя)
   * phone (контактный телефон)
   * email (электронная почта)
2. **Бронирования (Reservations)**:
   * id (уникальный идентификатор)
   * client\_id (внешний ключ на таблицу Users)
   * equipment\_id (внешний ключ на таблицу Equipment)
   * start\_date (планируемая дата начала аренды)
   * end\_date (планируемая дата окончания аренды)
   * status (статус: активна, отменена, завершена)
3. **Аренда (Rentals)**:
   * id (уникальный идентификатор)
   * client\_id (внешний ключ на таблицу Users)
   * item\_id (внешний ключ на таблицу Items)
   * start\_date (дата начала аренды)
   * end\_date (дата окончания аренды)
   * Extended\_end\_date (продлённая дата окончания)
   * actual\_return\_date (фактическая дата возврата)
   * total\_cost (сумма, заплаченная за аренду)
   * deposit\_paid (сумма залога, внесенная клиентом)
   * penalty\_amount (штраф за повреждения или просрочку)
   * status (статус: активна, завершена)
4. **Refresh Токены (Refresh\_Tokens)**:
   * token (токен)
   * user\_id (внешний ключ на таблицу Users)
   * User\_role (клиент, менеджер или админ)
   * Expires\_at (дата истечения токена)

## Основные ограничения

1. **Целостность данных**:
   1. Каждая аренда должна быть связана с существующим клиентом и оборудованием.
   2. Оборудование может быть арендовано только если его статус "доступно".
2. **Уникальность данных**:
   1. Каждая единица оборудования должна иметь уникальный идентификатор (например, серийный номер).
   2. Клиенты должны быть уникально идентифицированы (например, по email или телефону).
3. **Ограничения на ввод данных**:
   1. Дата возврата аренды не может быть раньше даты начала.
   2. Стоимость аренды не может быть отрицательной.
4. **Бизнес-правила**:
   1. Стоимость аренды рассчитываются автоматически на основе тарифов, даты начала аренды и текущей даты.

## Стек технологий

## 

## Фронтенд (Клиентская часть)

* **Язык программирования:** JavaScript.
* **Библиотека:** React — используется для построения интерфейса на основе компонентов.
* **Сборщик и дев-сервер:** Vite — обеспечивает быструю сборку и горячую перезагрузку во время разработки.

### Бэкенд (Серверная часть)

* **Язык программирования:** Python.
* **Фреймворк:** Flask — лёгкий веб-фреймворк, предназначенный для создания REST API и быстрой обработки HTTP-запросов.

### База данных

* **СУБД:** MySQL
* **Связь с бэкендом:** Через Python-библиотеку mysql-connector.

### Взаимодействие компонентов

* **Фронтенд ↔ Бэкенд:**
  + Используются HTTP-запросы к REST API, реализованному во Flask.
  + Данные передаются в формате JSON.
* **Бэкенд ↔ База данных:**
  + Взаимодействие через SQL-запросы или ORM.
  + Обработка, сохранение и извлечение данных из MySQL.

### Дополнительно

* Возможность контейнеризации с использованием Docker.

## Реализация

[Репозиторий с исходниками](https://github.com/SolomennikovKolya/nsu-course-3/tree/main/db/project_rental)

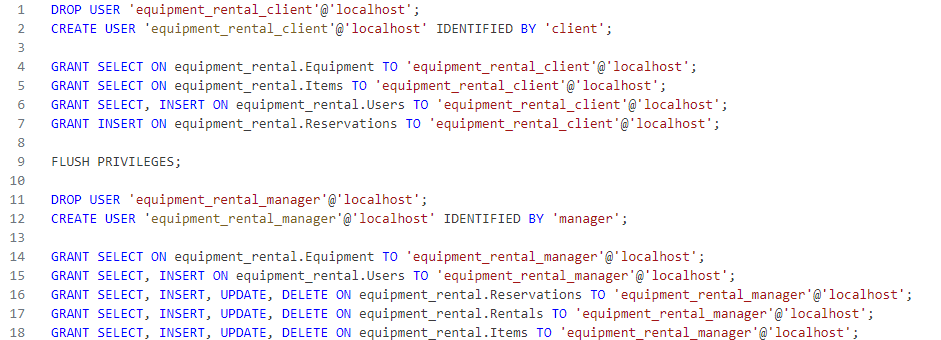
Бэкенд реализован на языке Python с библиотекой Flask. Запросы, относящиеся к разным ролям пользователя (клиент, менеджер, администратор), вынесены в разные модули. Также реализована своя обёртка для удобного взаимодействия с базой данных. Вынесены специальные команды для разработки (создание, очистка, заполнение, удаление, сохранение БД), которые также доступны на сайте в пользовательском интерфейсе. Доступ к API, предназначенным только для менеджеров или админов, проверяется через JWT токены (то есть для доступа, пользователь должен быть авторизован с соответствующей ролью).

Фронтенд написан на JavaScript (библиотека React). Всё разбито на страницы, доступ к которым проверяется с помощью данных авторизации. Также использовались css стили.

## Безопасность

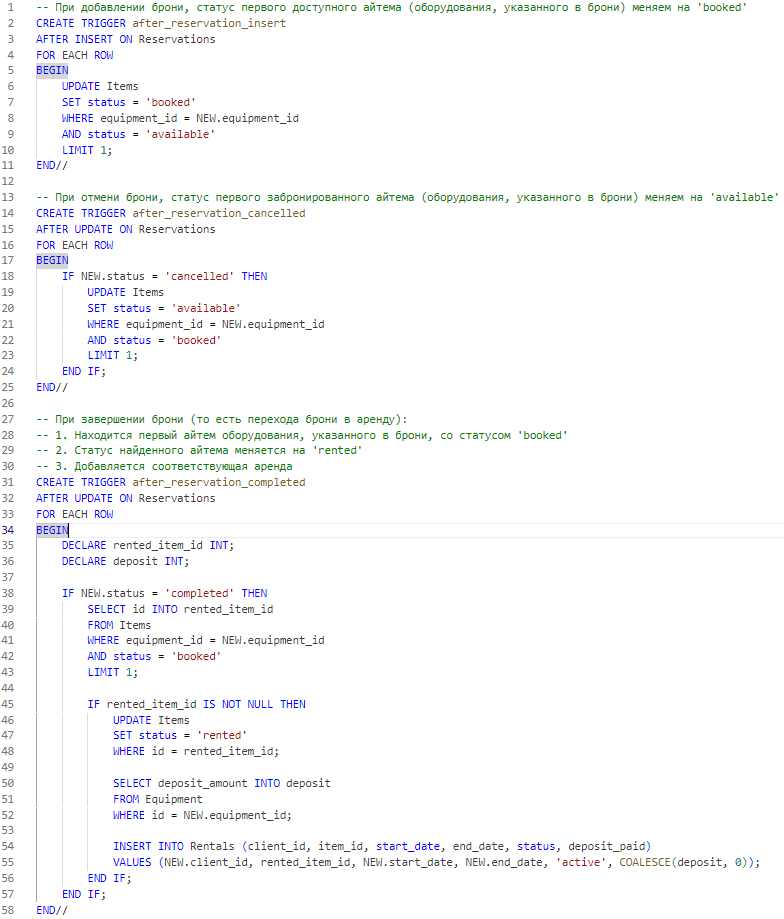
Система авторизации: на сайте можно зайти как менеджер или администратор, после чего откроются специальные возможности. Данные аккаунтов, в которые можно зайти фиксированы и добавляются другими менеджерами. То есть, чтобы новый сотрудник мог зайти как менеджер или администратор, другой администратор должен добавить его данные в систему. Авторизация реализована через JWT и Refresh токены. JWT хранятся у пользователя, а Refresh токены - в базе данных на сервере. На сайте роль пользователя используется, чтобы скрывать недоступные элементы, а на сервере роль проверяется, чтобы выполнить или отклонить запрос.

Разные действия, реализующие бизнес логику, выполняются из-под разных пользователей базы данных, у которых есть доступ только к необходимым возможностям. Также есть поддержка .env файлов для хранения паролей, ключей и других приватных данных.



## Триггеры

Для удобной работы с бронями, были реализованы триггеры:



## SQL запросы

[Основные SQL запросы](https://github.com/SolomennikovKolya/nsu-course-3/tree/main/db/project_rental/backend/src/db/queries)

Остальные SQL запросы представлены в коде и обёрнуты в функции, причём у этих функций указан собственный декоратор @with\_db. Это означает, что при выполнениях каждой из этих функций будет производиться подключение к базе данных с заданными параметрами входа, а при завершении функции - автоматическое освобождение всех ресурсов и отключение от базы данных. Решение не поддерживать соединение, а переподключаться к базе данных каждый раз заново, было принято специально, так как запросы от клиентов приходят не часто, а пользователей может быть много. Пример кода:

