# 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.

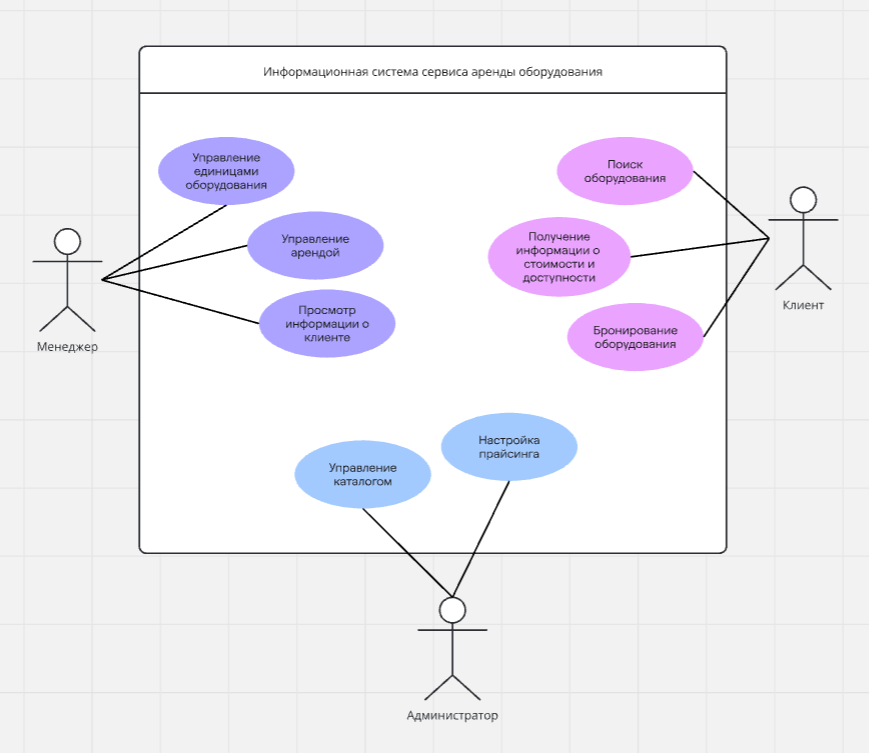
## Полный цикл арендования оборудования

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 переоформляется на новые даты.
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 возможные повреждения.
   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.

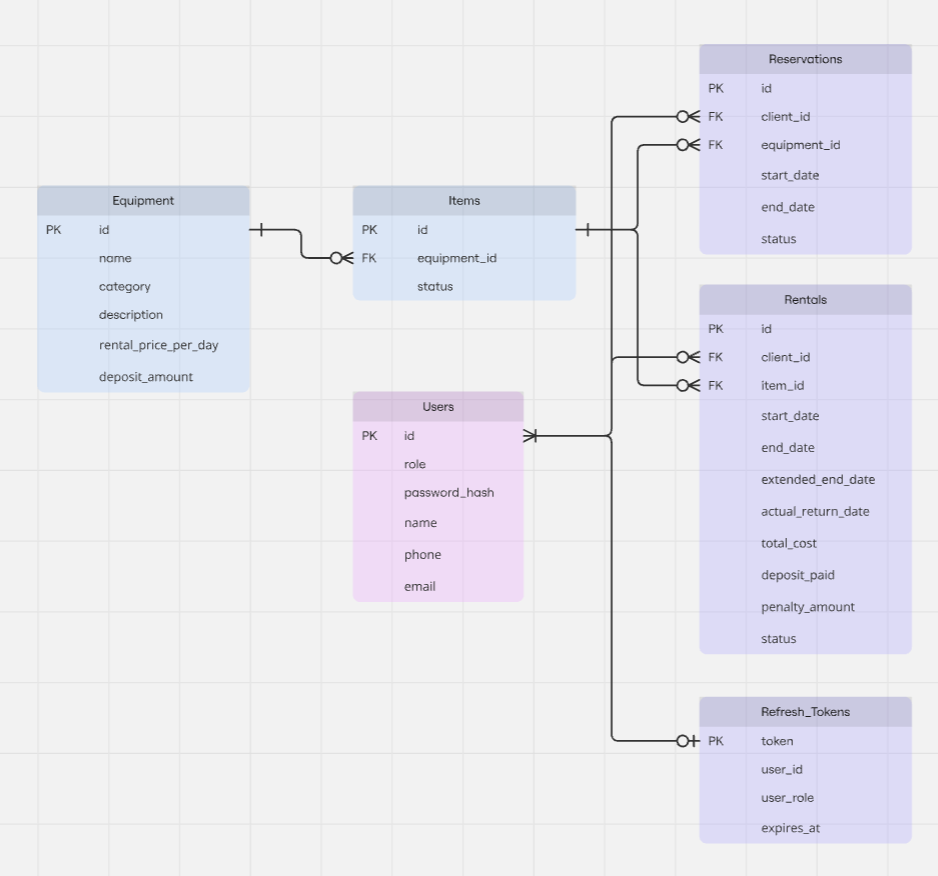
**Клиенты**:

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

## Информационные потоки для каждого типа пользователей

Система пользователей иерархическая:  
Клиент < Менеджер < Администратор < Разработчик  
То есть, например, всё, что видит менеджер на сайте, видит и администратор с разработчиком; всё, что может менеджер может и администратор с разработчиком. В контекстной диаграмме это явно не указано, но стоит это учитывать.

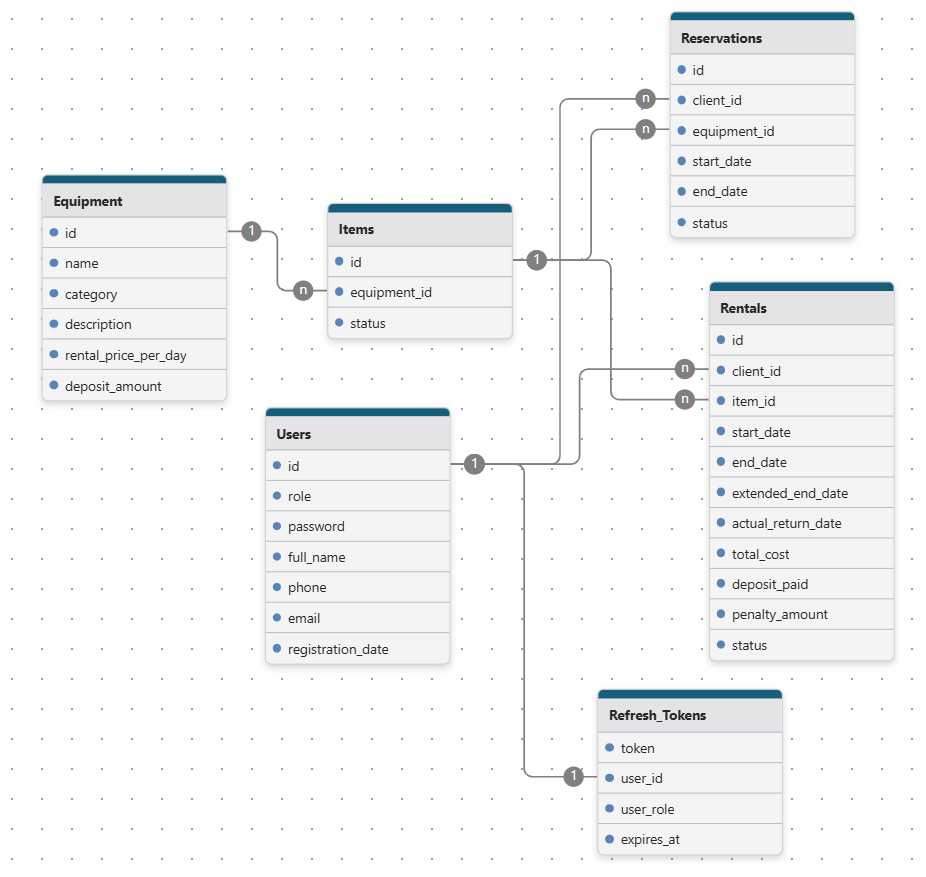
## Основные информационные объекты и связи



Сущности:

* Equipment - оборудование (например, “Перфоратор Bosch”)
* Items - единицы обрудования. Представляют собой конкретные физические вещи определённого оборудования (например, перфораторов Bosch может быть несколько в обороте, и каждый из них можно отдельно арендовать)
* Users - пользователи (клиенты, менеджеры, админы)
* Reservations - брони
* Rentals - аренды
* Refresh\_Tokent – Refresh токены (нужны для авторизации)

## 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)**:
   * id (уникальный идентификатор среди всех оборудований)
   * equipment\_id (внешний ключ на таблицу Equipment)
   * status (статус: доступно, забронировано, в аренде, на обслуживании, списано) - данное поле меняется автоматически с помощью триггеров
2. **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)
3. **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)
4. **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)
   * status (статус: активна, завершена)
5. **Refresh Токены (Refresh\_Tokens)**:
   * token (токен)
   * user\_id (внешний ключ на таблицу Users)
   * user\_role (клиент, менеджер или админ)
   * expires\_at (дата истечения токена)

## Скрипт schema.sql для создания схемы БД

## Ограничения

1. **Целостность данных**:
   1. Каждая единица оборудования должна быть связана с оборудованием.
   2. Каждая бронь и аренда должна быть связана с существующим клиентом и оборудованием.
   3. Каждый Refresh токен должен быть связан с пользователем.
2. **Уникальность данных**:
   1. Каждая единица оборудования должна имеет уникальный идентификатор (не только среди единиц определённого оборудования, но и среди всех единиц оборудования).
   2. Пользователи должны быть уникально идентифицированы (по имени, email или телефону).
3. **Ограничения на ввод данных**:
   1. При бронировании дата возврата аренды не может быть раньше даты начала, а также дата начала аренды не может быть раньше сегодняшнего дня.
   2. Стоимость аренды и залог не могут быть отрицательными.
4. **Бизнес-правила**:
   1. Оборудование может быть забронировано только если есть хотя бы одна единица оборудования со статусом "доступно".
   2. Стоимость аренды рассчитываются автоматически на основе тарифов, даты начала аренды и текущей даты (даты завершения аренды).
   3. Нельзя удалить и изменить статус забронированной или арендованой едииницы оборудования.
   4. Нельзя убрать из каталога оборудование, некоторые единицы которого находятся в брони или аренде.

## Стек технологий

## 

### Фронтенд (Клиентская часть)

* **Языки:** JavaScript (де-факто лучший язык для фронтенда. Работает во всех браузерах без плагинов и компиляции, поддерживает асинхронность, удобен для разработки), CSS (отвечает за внешний вид сайта), HTML (каркас для всего веб приложения).
* **Библиотека:** React — используется для построения интерфейса на основе компонентов. React работает с Virtual DOM, что обеспечивает эффективность, поддерживает хуки (управление состоянием, отклик на изменения в компоненте, эффекты)
* **Сборщик и дев-сервер:** Vite — обеспечивает быструю сборку и горячую перезагрузку во время разработки.

### Бэкенд (Серверная часть)

* **Язык программирования:** Python - удобный язык для быстрой разработки.
* **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.

### Дополнительно

* Возможность контейнеризации с использованием Docker.

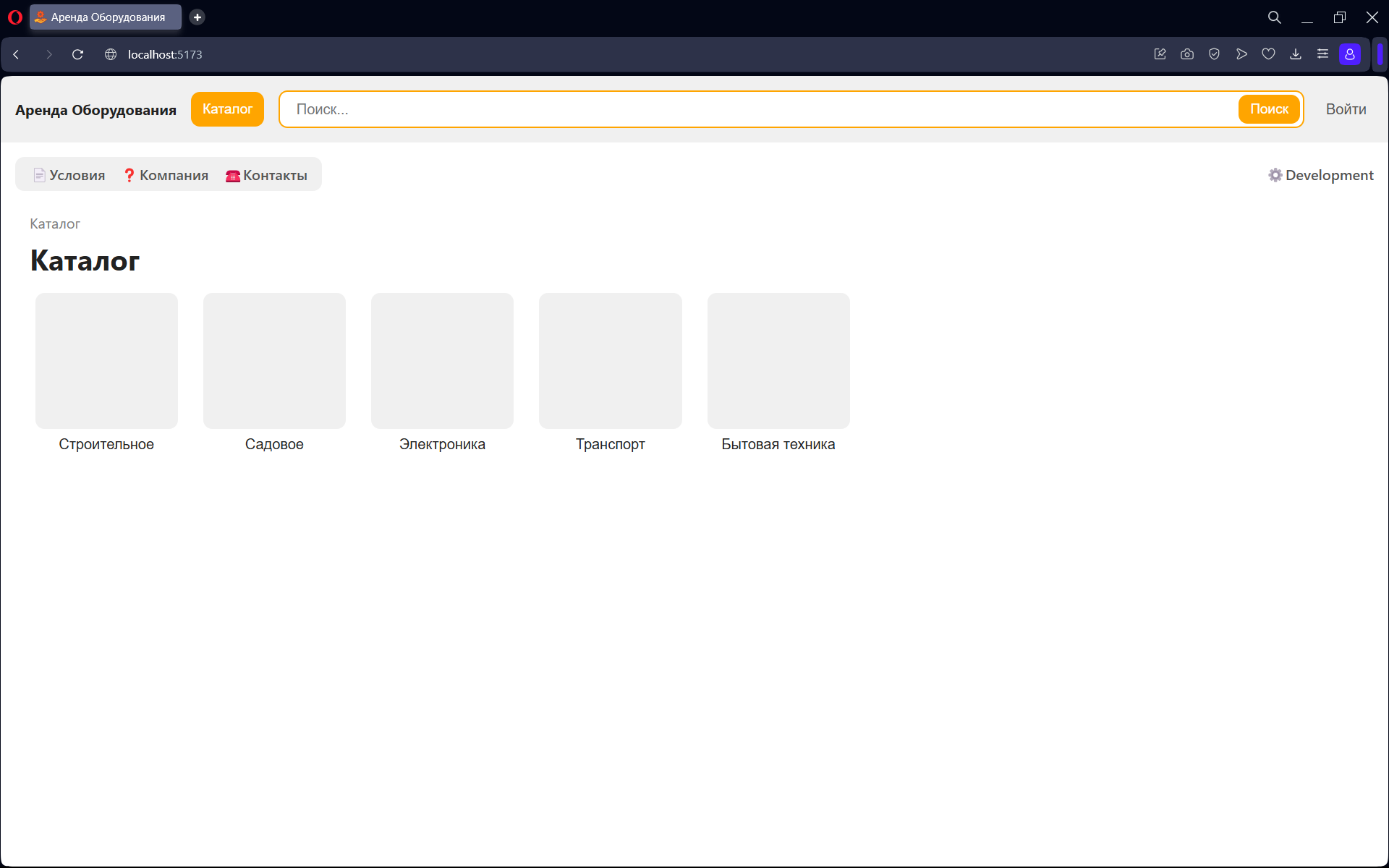
## Реализация

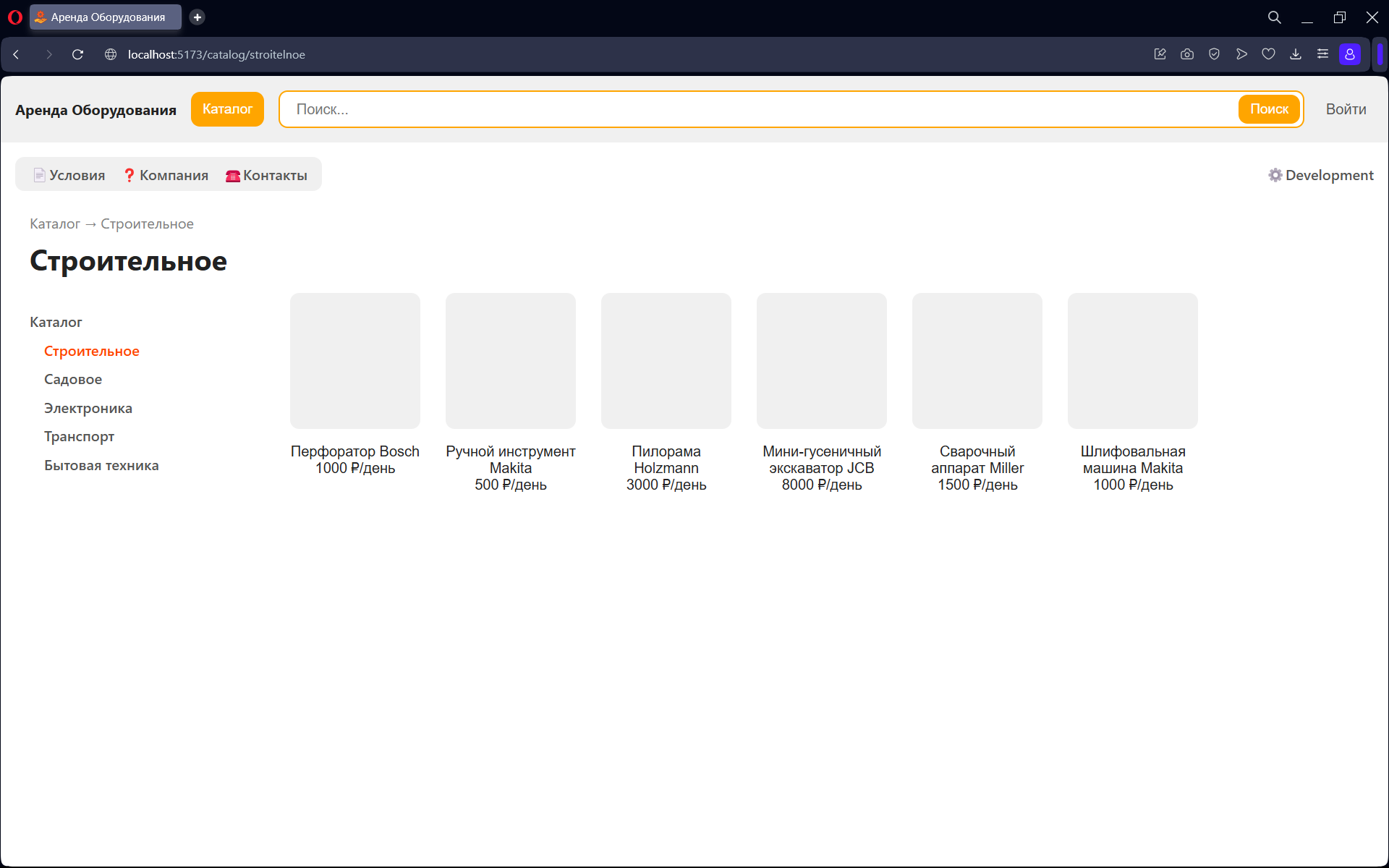
[Репозиторий с исходниками](https://github.com/SolomennikovKolya/nsu-course-3/tree/main/db/project_rental) - здесь же описано, как развернуть приложение, а также объяснено назначение всех файлов (для понимания).

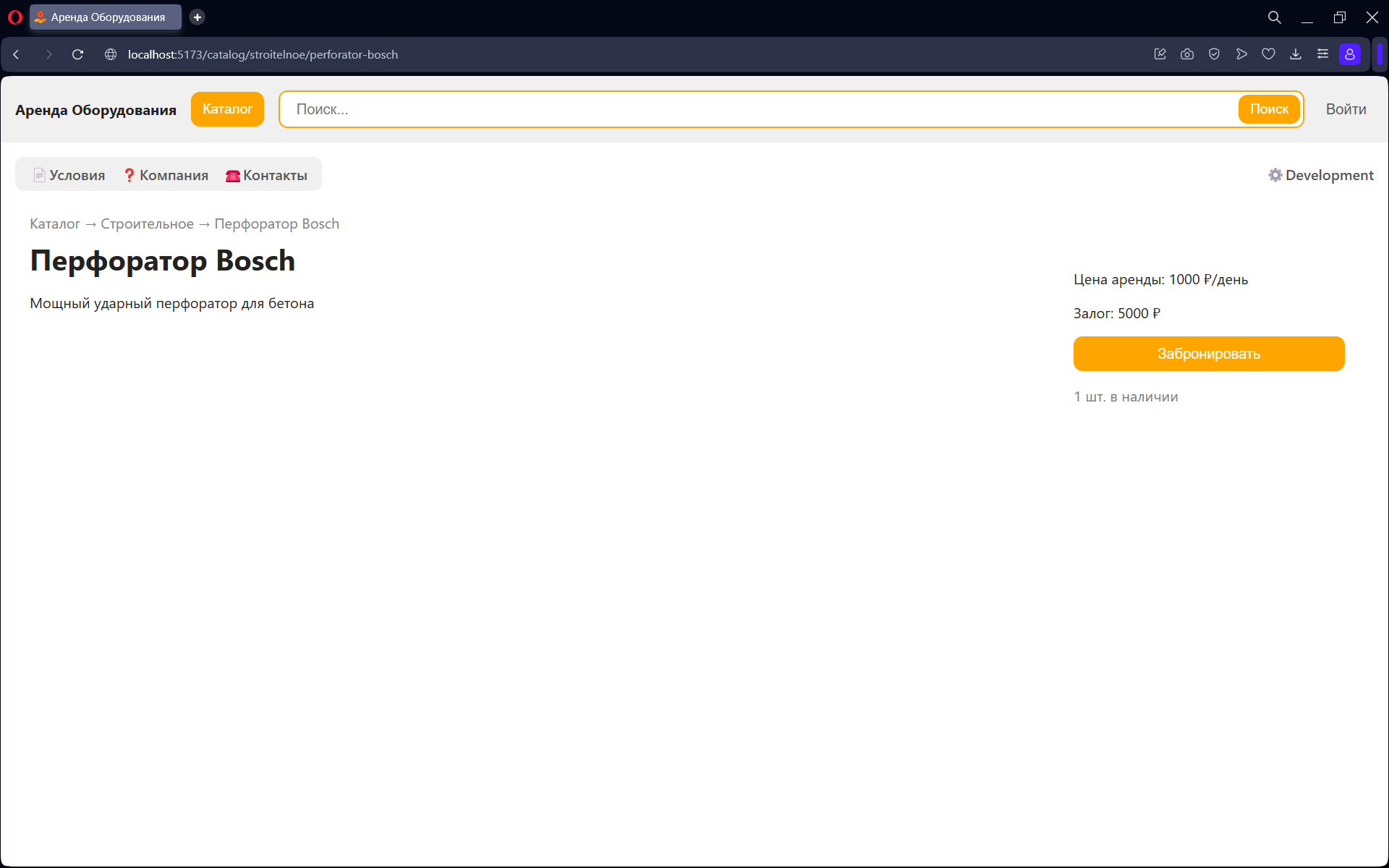
Бэкенд реализован на языке Python с библиотекой Flask. Запросы, относящиеся к разным ролям пользователя (клиент, менеджер, администратор), вынесены в разные модули. Также реализована своя обёртка для удобного взаимодействия с базой данных. Вынесены специальные команды для разработки (создание, очистка, заполнение, удаление, сохранение БД), которые доступны в терминале и 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). Дополнительную безопасность на уровне СУБД, обеспечивают роли БД (при выполнении команд, предназначенных для определённого пользователя, подключение к БД происходит с соответствующей ролью).

Фронтенд написан на JavaScript (библиотека React). Всё разбито на страницы, доступ к которым проверяется с помощью данных авторизации. В интерфейсе отображаются только компоненты, доступные текущему пользователю (зависит от роли пользователя). На каждой странице есть верхняя панель для перехода не другие страницы, что обеспечивает удобные переходы. Реализовано это с помощью компоненты Layout, который содержит компонент NavbarTop (верхняя панель) и дочерний элемент (вложенные компоненты). Для стилизации используется CSS.

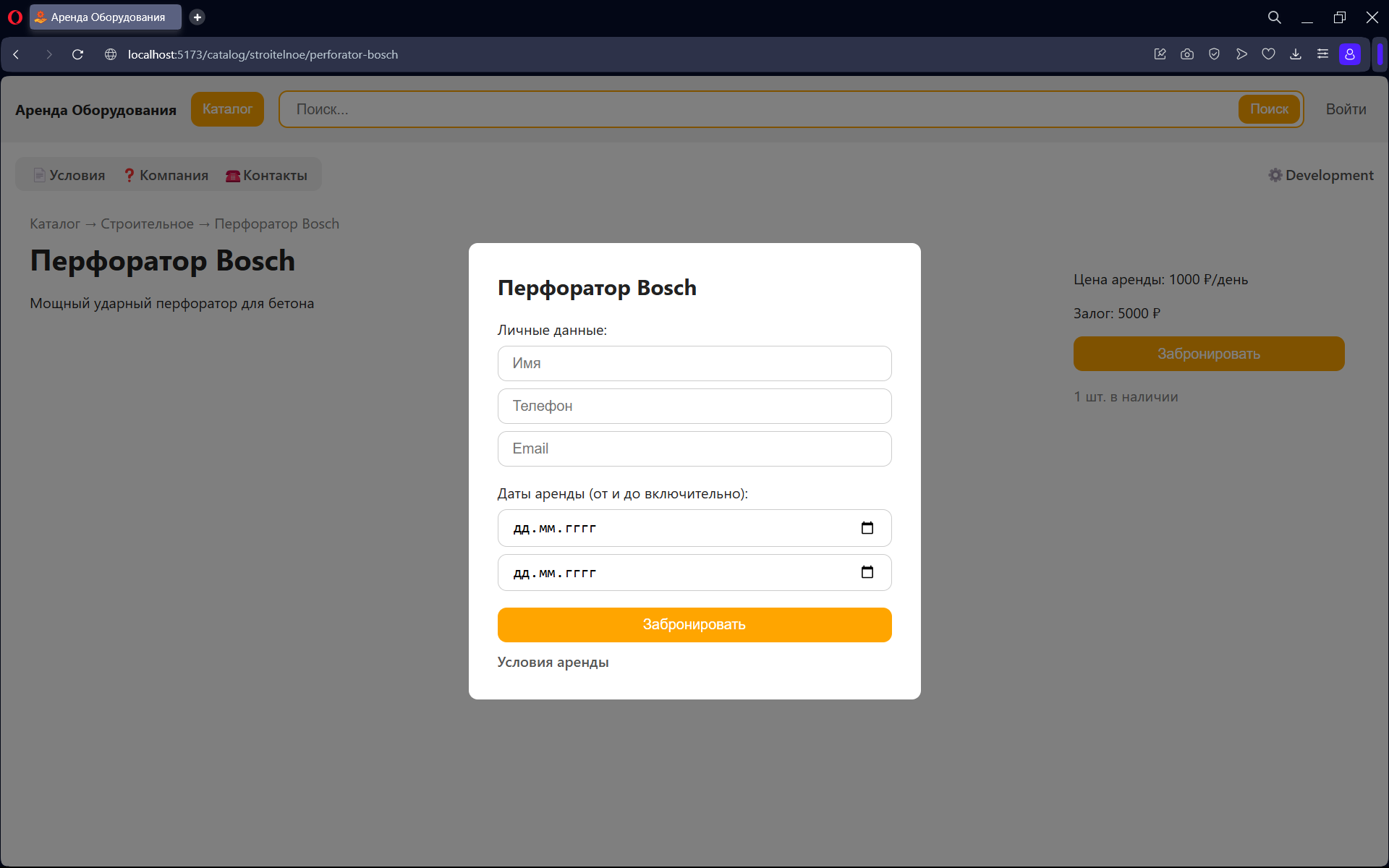
## Клиентский интерфейс

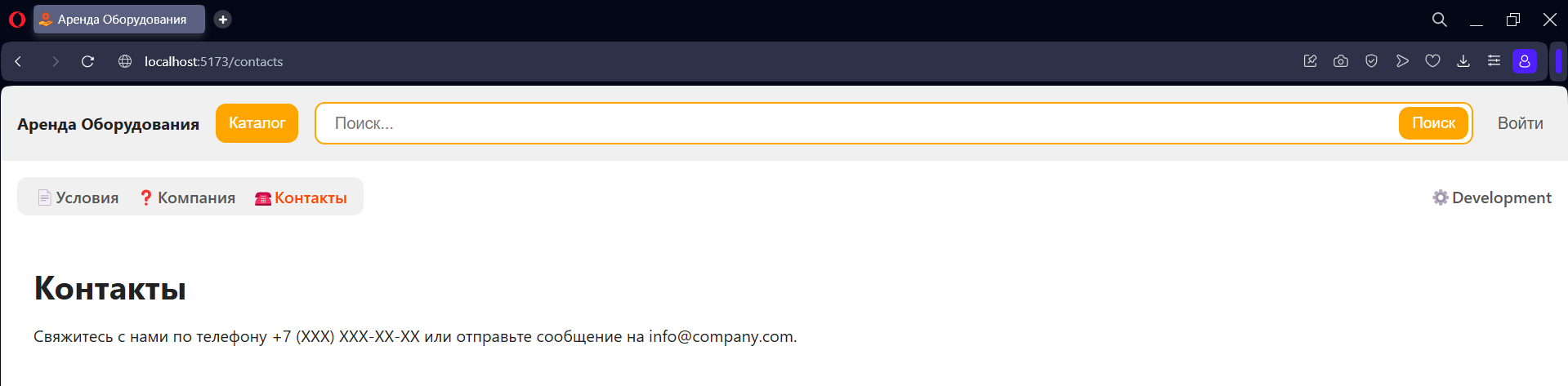
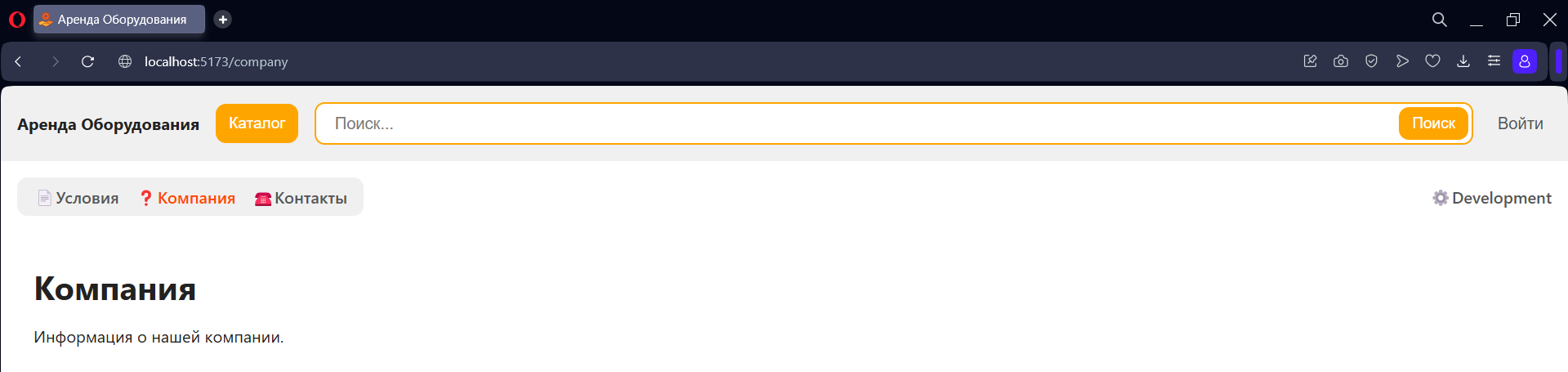
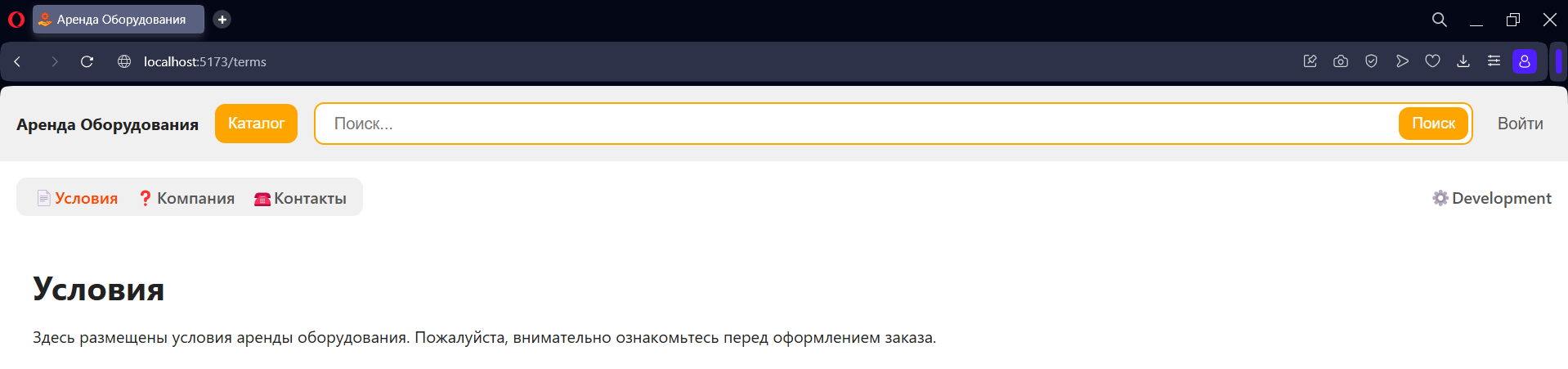
При заходе не сайт, клиент сразу попадает на страницу “Каталог” (потому что маршрутизация настроена таким образом, что каталог отображается и по пути “/catalog” и просто “/”). Сверху - верхняя панель, которая содержит: название компании, кнопку “Каталог”, строку поиска (исключительно косметическую), кнопку “Войти”, ссылки на другие страницы (“Условия”, “Компания”, “Контакты”) и кнопку “Development” (нужна только при разработке). На странице каталога отображаются все категории оборудования (серые квадраты - якобы картинки).

На странице определённой категории показывается оборудование соответствующей категории. Для удобства здесь же показывается цена аренды. Также слева есть навигационная панель, чтобы удобно перемещаться между категориями.

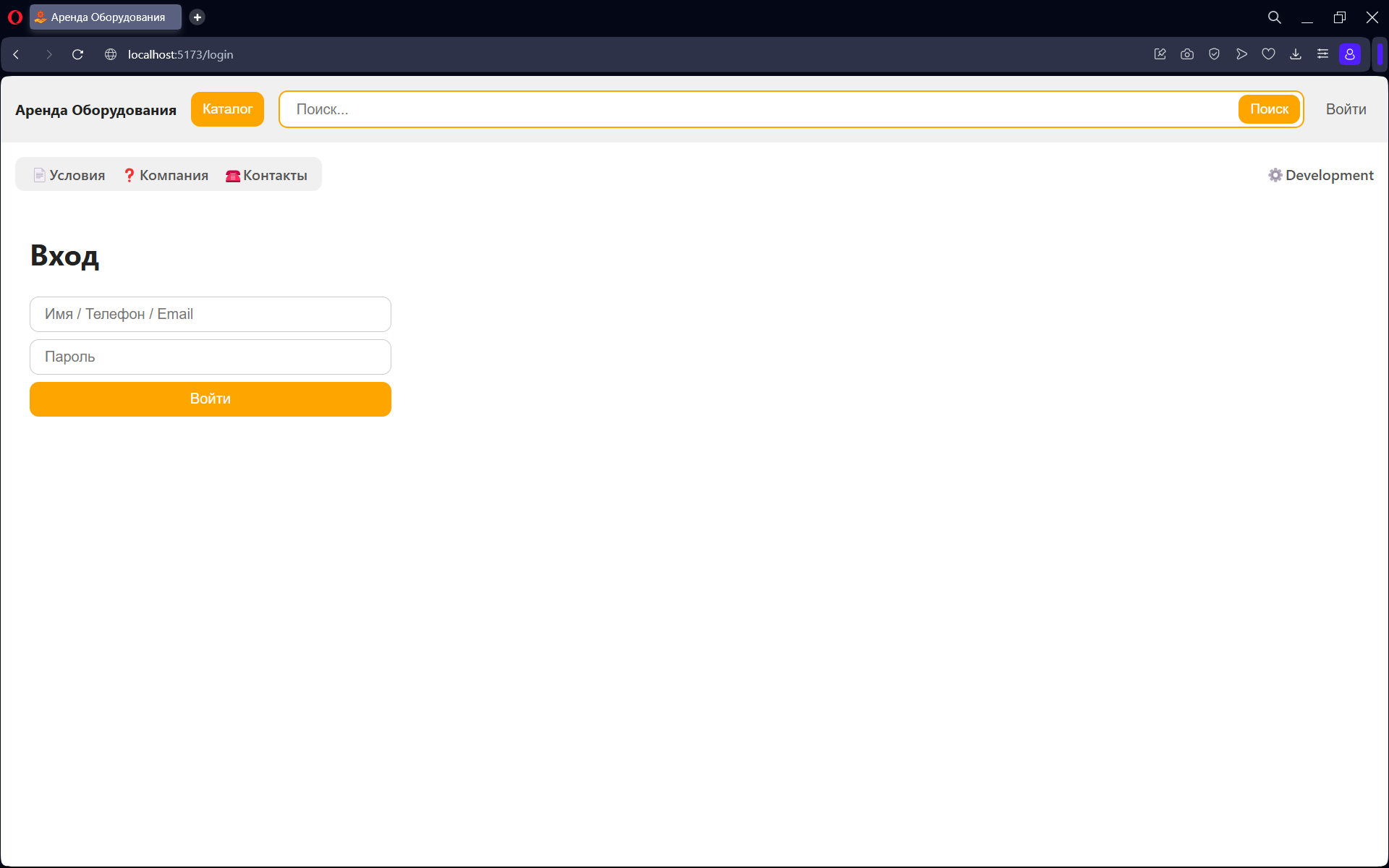
При переходе на страницу определённого оборудования показывается карточка, содержащая название оборудования, описание, цену аренды, сумму залога, количество доступных единиц оборудования и кнопку “Забронировать”. Также над названием оборудования можно заметить светло серым текстом путь по каталоги (Каталог -> Строительное -> Перфоратор Bosch). А в URL написан тот же путь только английским транслитом. Это тонкий момент, потому что в URL нельзя писать русский текст и другие спец символы. Реализовано это с помощью преобразования в SLUG формат. Маршрутизация устроена так:

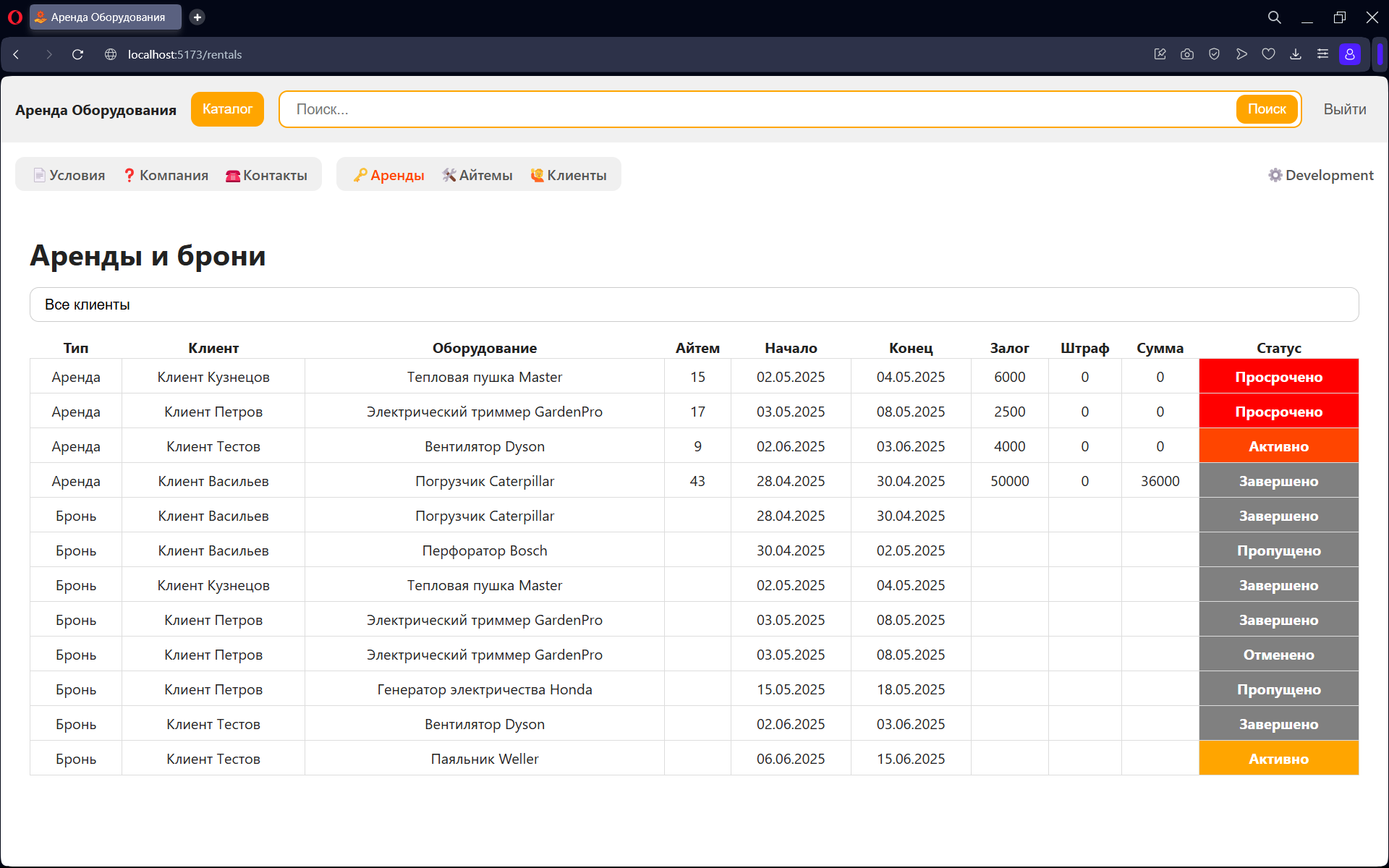


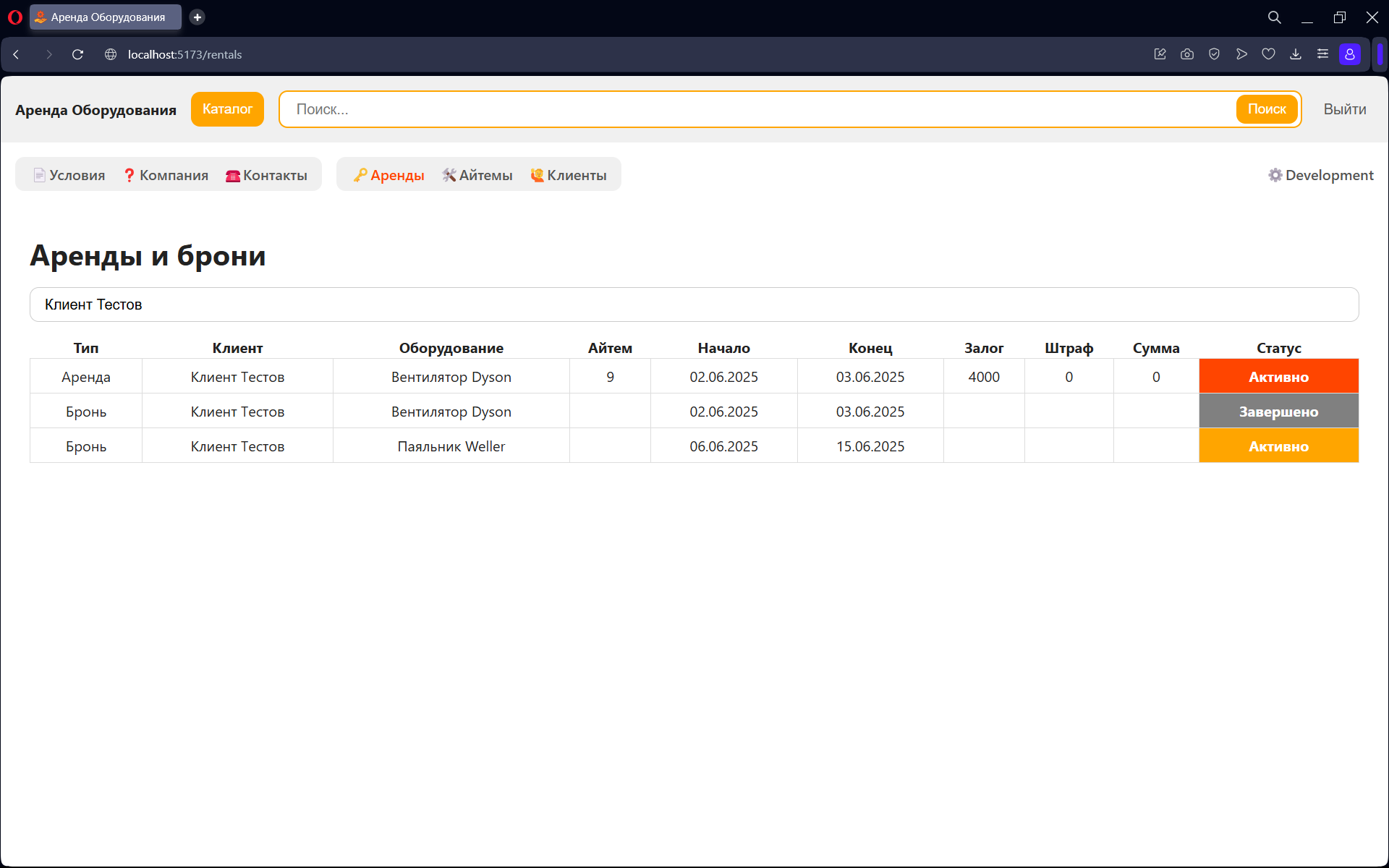
При нажатии на “Забронировать” появляется модальное окно. Здесь указывается имя, телефон, email клиента, даты аренды (когда клиент собирается использовать оборудование). Если данный пользователь ещё не делал заказов, в базе появляется новая запись с данными этого клиента. Дату начала брони нельзя указать раньше сегодняшнего для, а дату окончания аренды нельзя указать раньше даты начала. Также есть ссылка на условия аренды, чтобы клиент точно не забыл с ними ознакомиться.

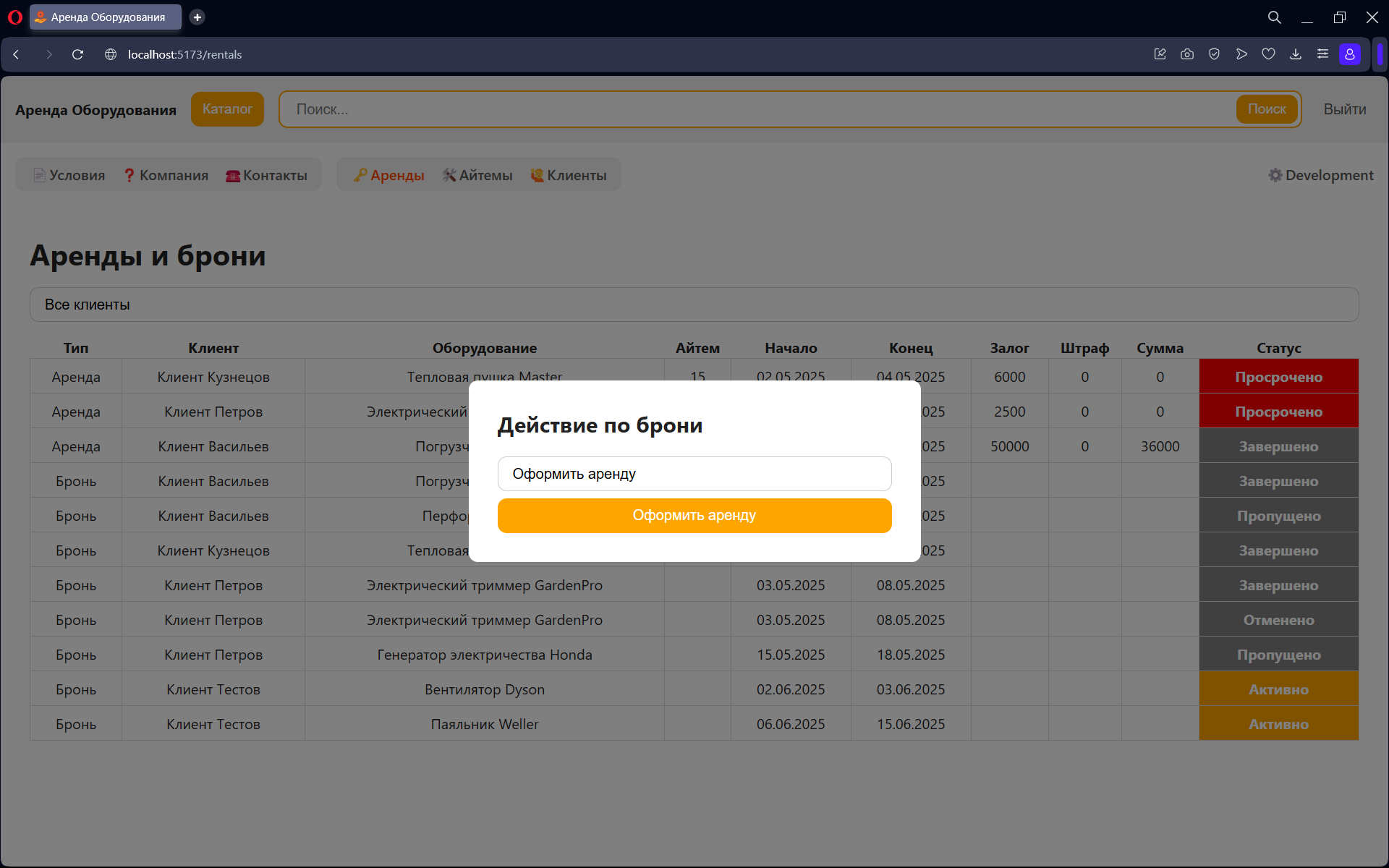
Условия, Компания и Контакты - простые информационные страницы с текстом.

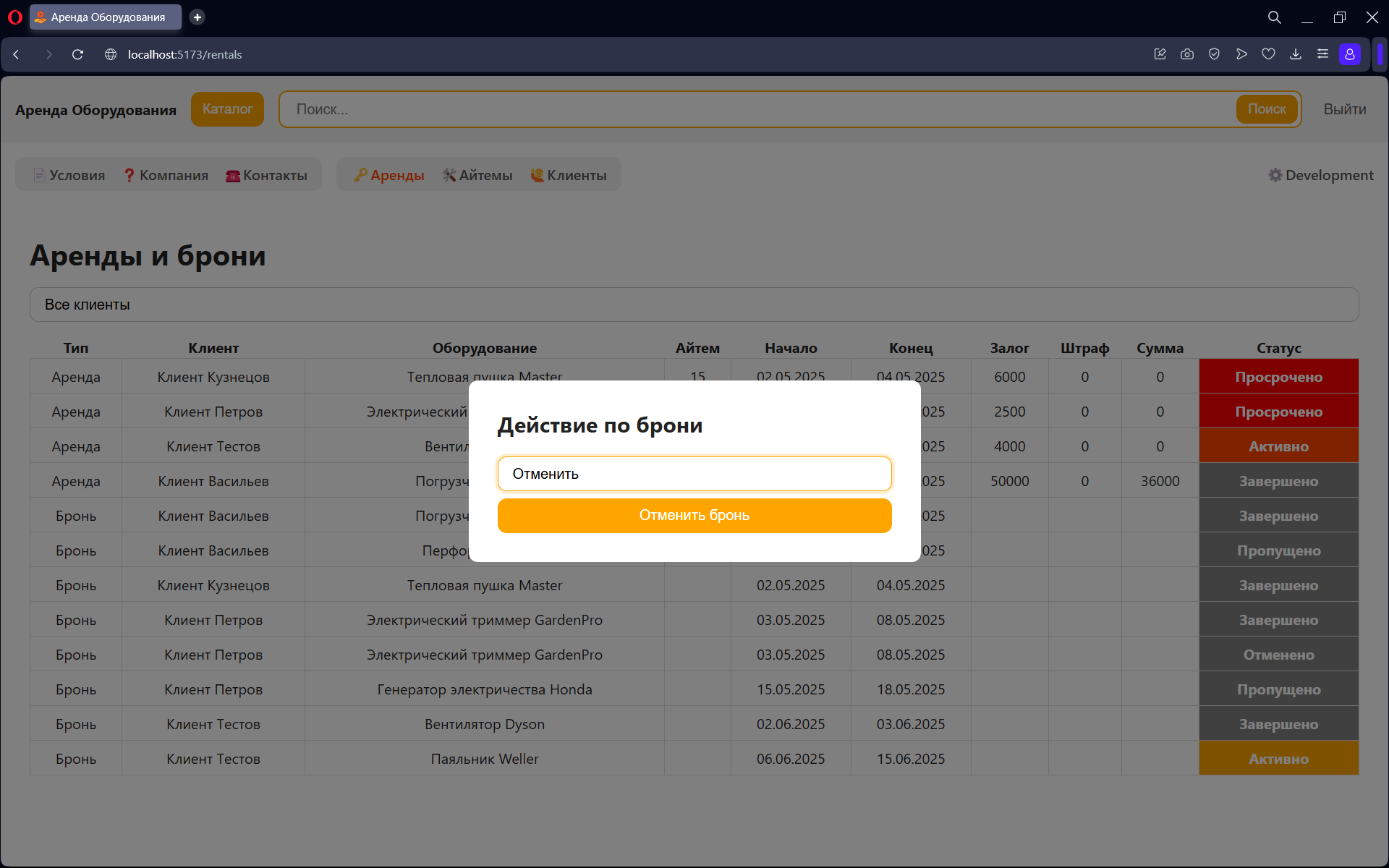
## Менеджерский интерфейс

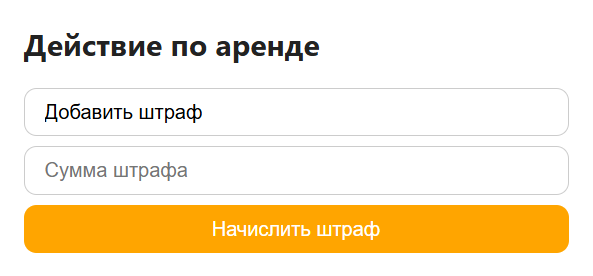
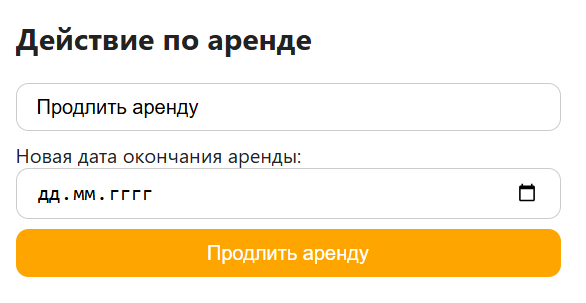
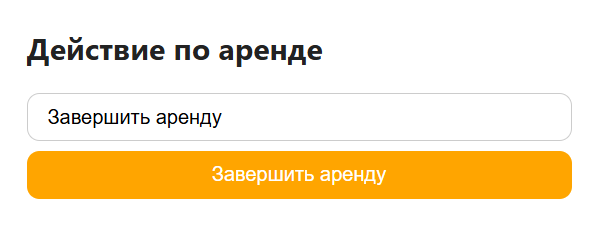
Для использования прав менеджера надо авторизоваться. Можно войти по имени, телефону или email и паролю, если в базе есть соответствующие данные (предполагается, что при добавлении нового сотрудника в штат, админ должен создать учётную запись для него). Та же логика работает при авторизации с ролью администратора. После успешного входа сервер присылает JWT (токен), который хранится в браузере и отправляется с каждым запросом на сервер, чтобы подтвердить аутентификацию. На сервере JWT декодируется с помощью ключа, который был использован при его создании, после чего проверяются роль (она вложена в JWT) и даётся доступ. Но срок действия JWT мал (около 15 минут), и поэтому существует Refresh токен. Он также отправляется сервером при авторизации. Клиент должен каждый раз отправлять его вместе с куки, чтобы в случае истечения JWT, выдался новый токен. Если и Refresh токен истёк, клиенту придётся войти заново.

Менеджеру доступна страница со всеми бронями и арендами. Здесь показывается, кто что заказал / забронировал, даты начала, конца аренды, а также залог, штраф и суммарная заплаченная цена аренды, статус. Сумма считается автоматически при завершении аренды (считается, что клиент оплачивает аренду после того, как он вернул оборудование, поэтому пока аренда активна, сумма = 0). Статус аренды “Просрочено” явно не хранится в БД. Он учитывается как статус “Активно”, но, если дата конца аренды меньше текущей даты. Это нужно, чтобы менеджер видел, каких клиентов предупредить о надобности вернуть оборудование. Если клиент вернул товар позже, чем надо, то он просто должен будет оплатить аренду за все дни, пока оборудование было у него на руках. Айтем (единица оборудования) указывается только для аренд, так как бронируется оборудование, а не конкретный айтем. Для броней предусмотрены статусы: “Активно”, “Завершено”, “Пропущено”, “Отменено”. Пропущено — значит, что бронь активна, но дата начала предполагаемой аренды больше текущей даты. Есть возможность отсортировать таблицу по каждому столбцу.

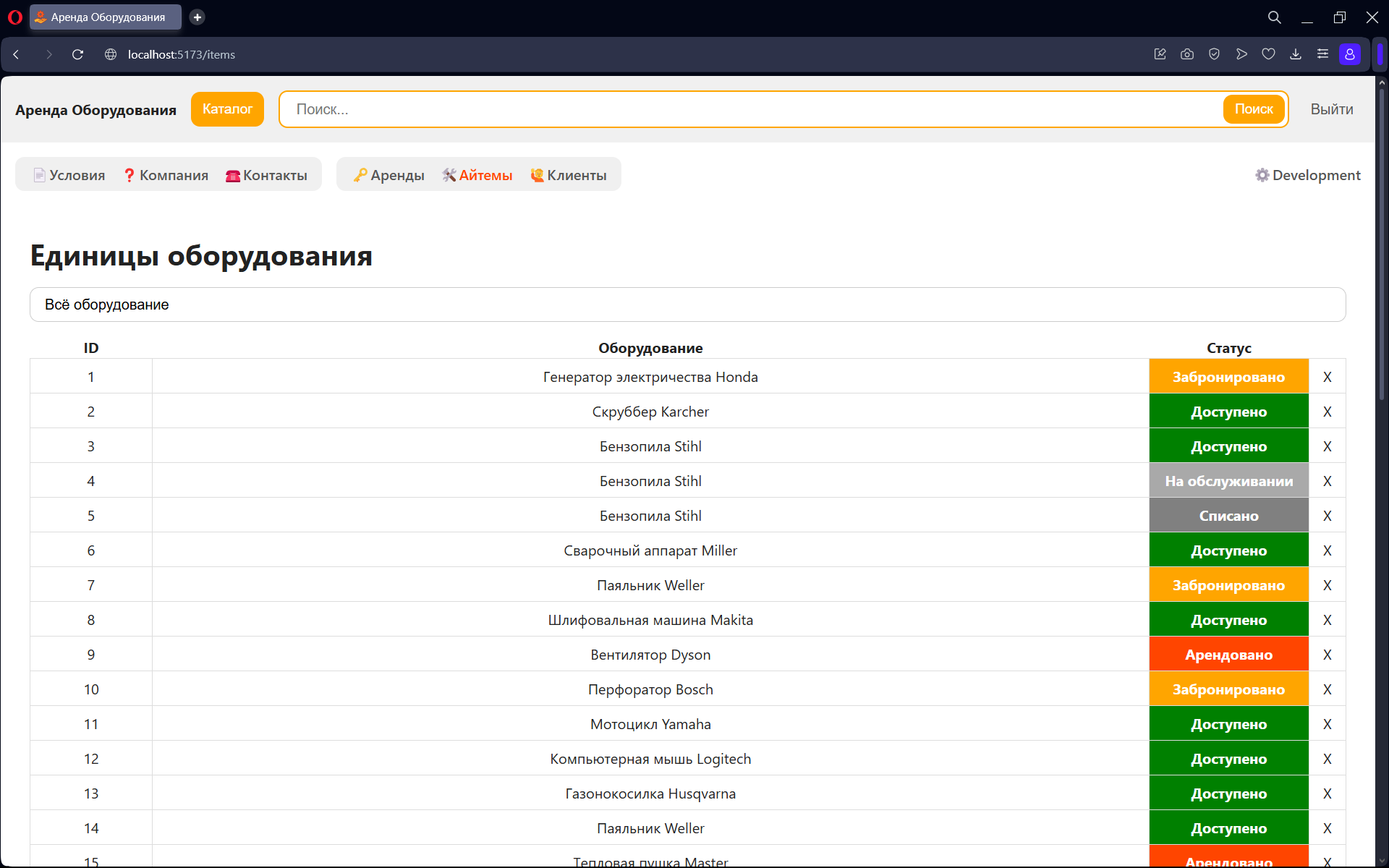
Можно выбрать аренды/брони только одного определённого клиента.

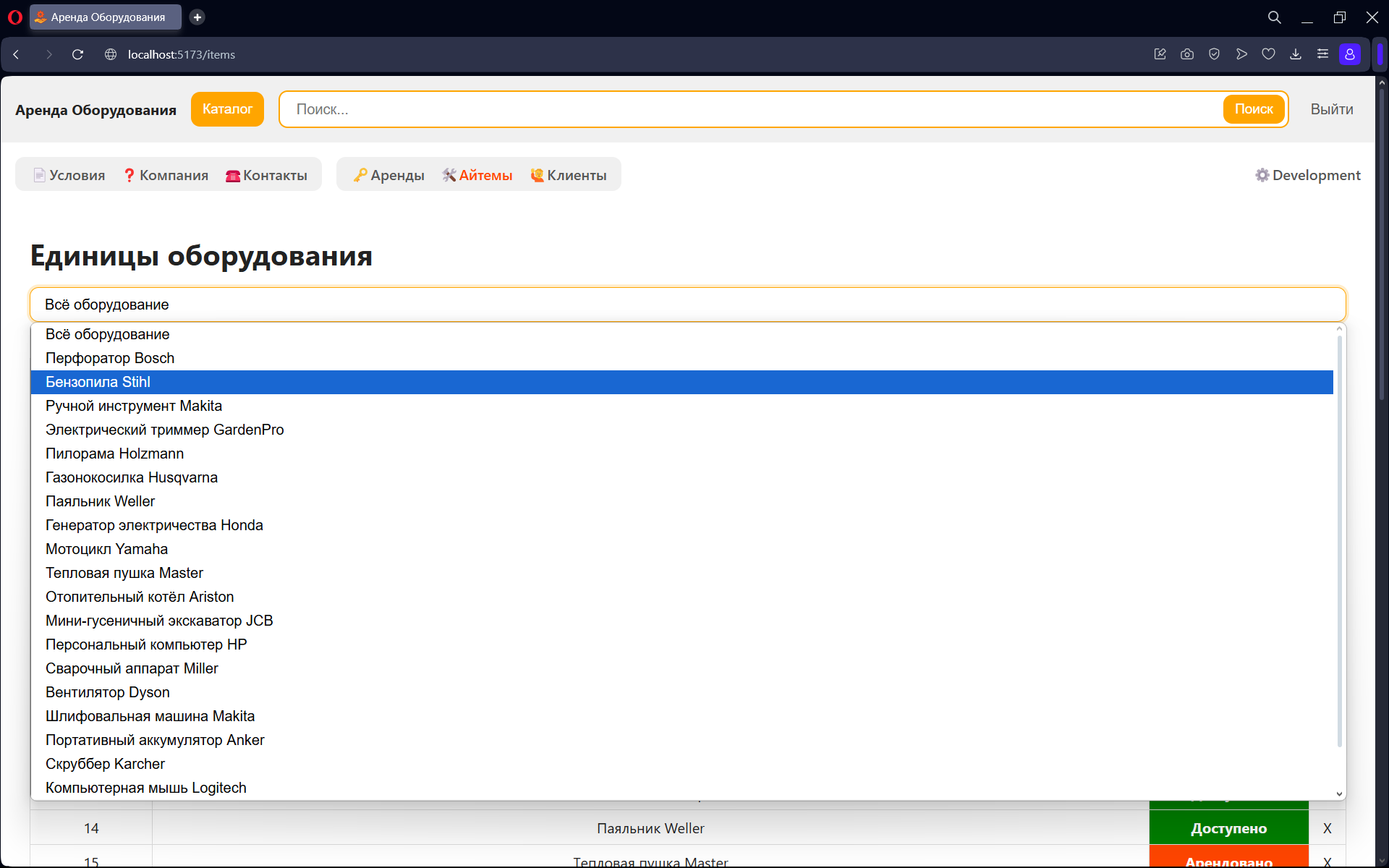
Если кликнуть по строчке с бронью, появится модальное окно действий по брони. Бронь можно перевести в аренду. В этом случае бронь станет со статусом “Завершено” и создастся новая аренда с данными из брони.

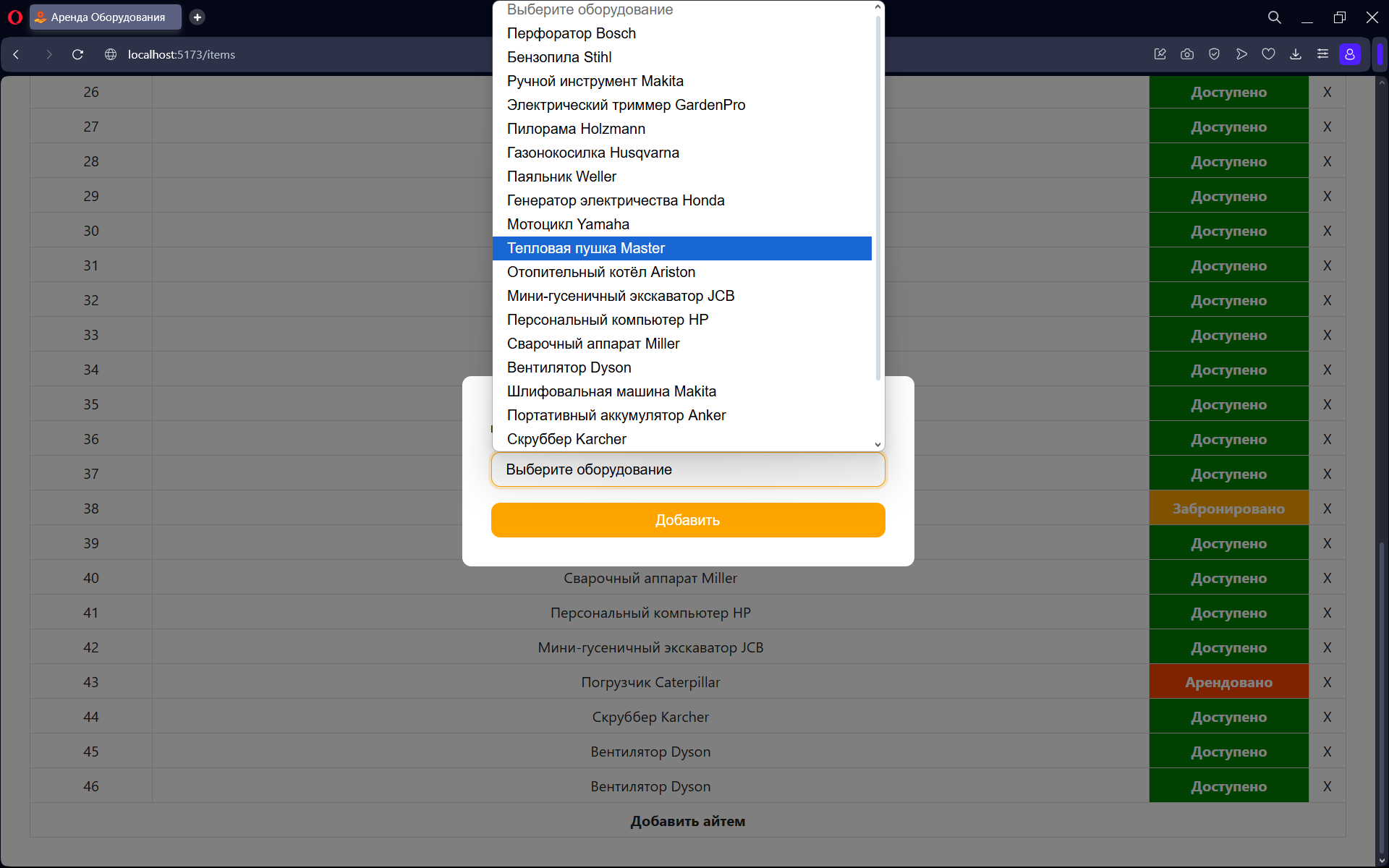
Также бронь можно просто отменить.

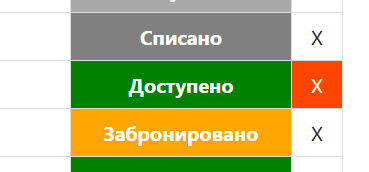


Если кликнуть по строчке с записью аренды, то откроется модельное окно с действиями по аренде. Можно завершить аренду, продлить её или начислить штраф.

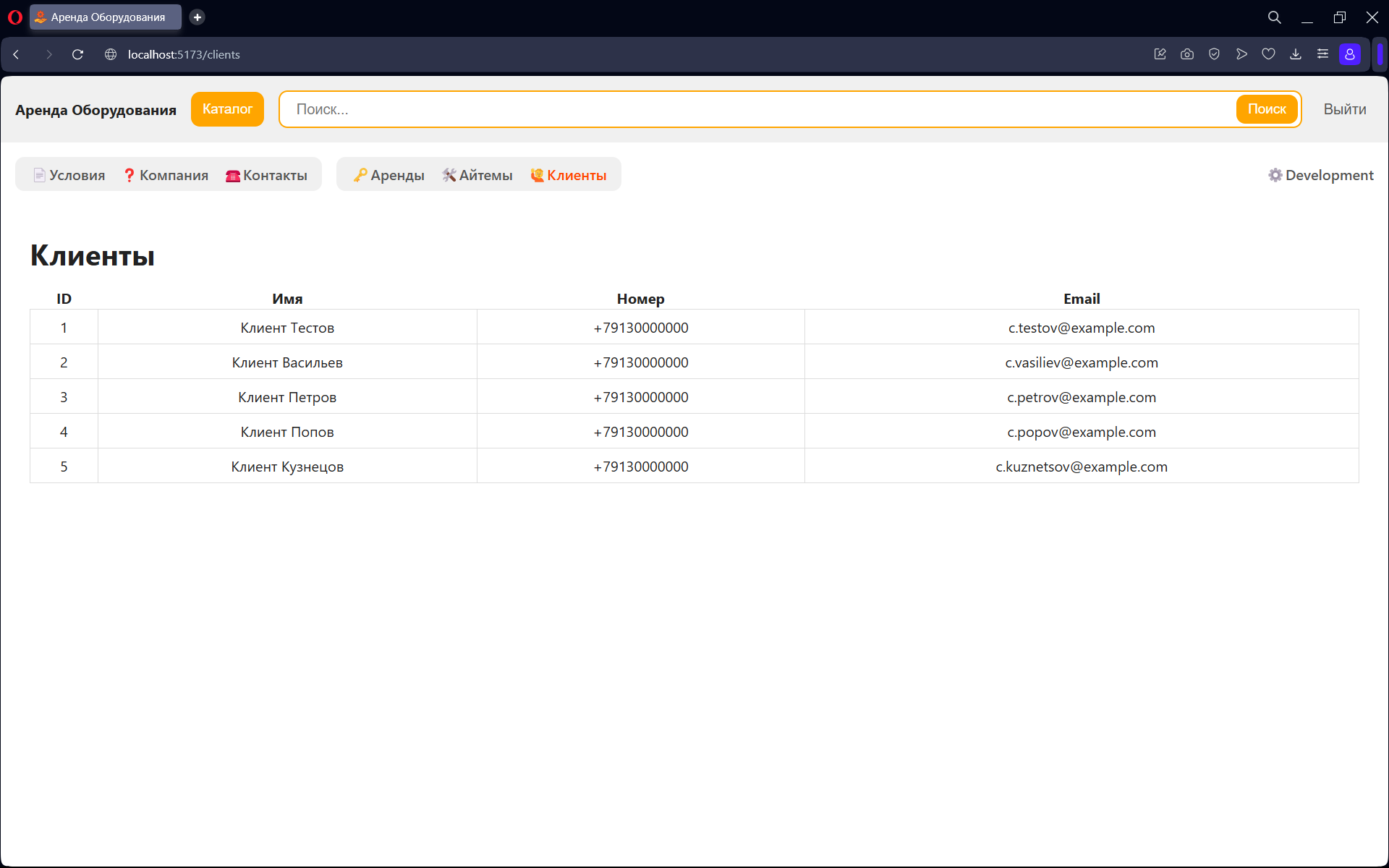
Также менеджеру доступна страница управления айтемами.

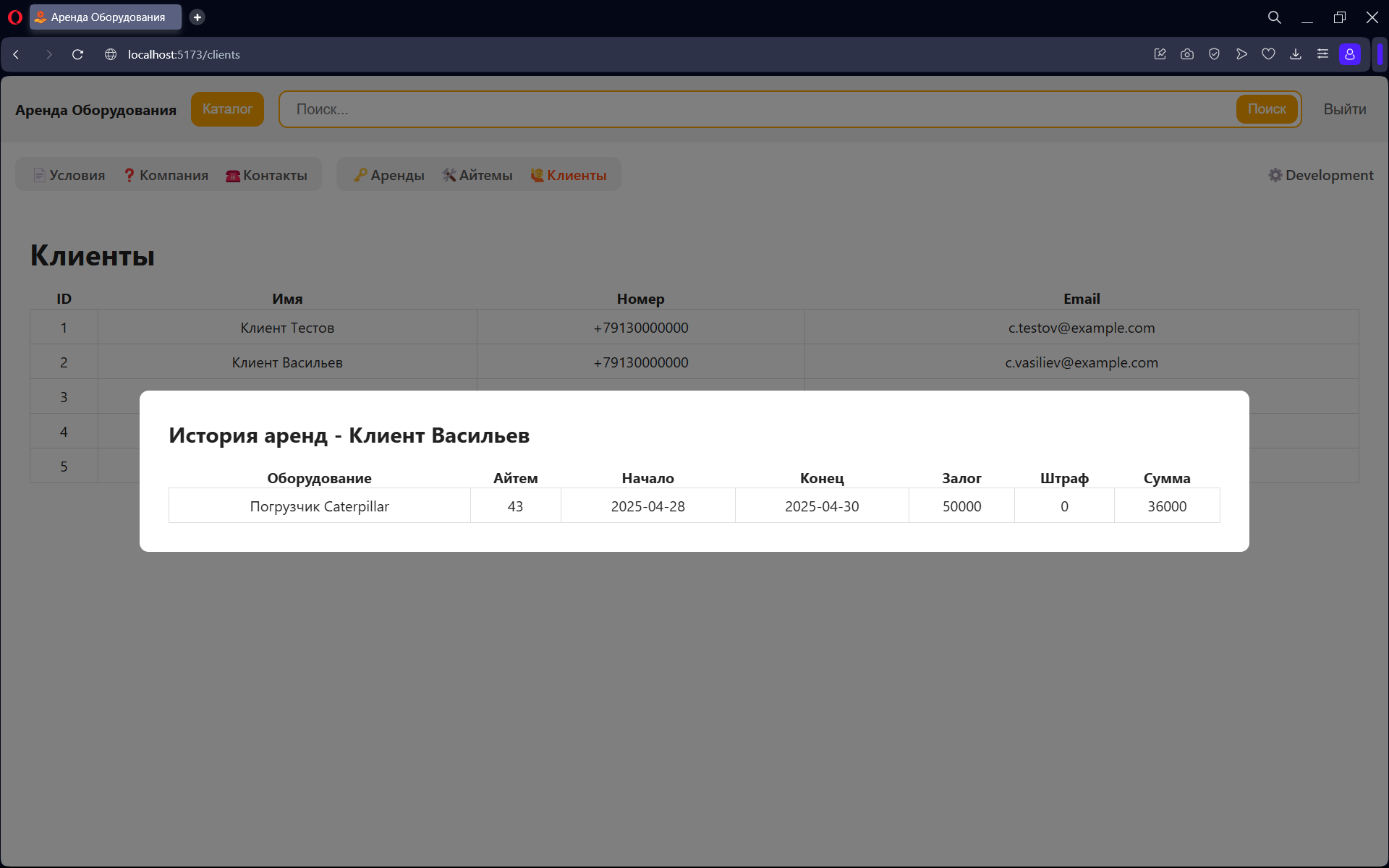
Можно выбрать оборудование, чтобы в списке показывались только айтемы этого оборудования. Также можно отсортировать таблицу по каждой колонке.

При нажатии на кнопку “Добавить айтем” снизу страницы, можно добавить новый айтем определённого оборудования.

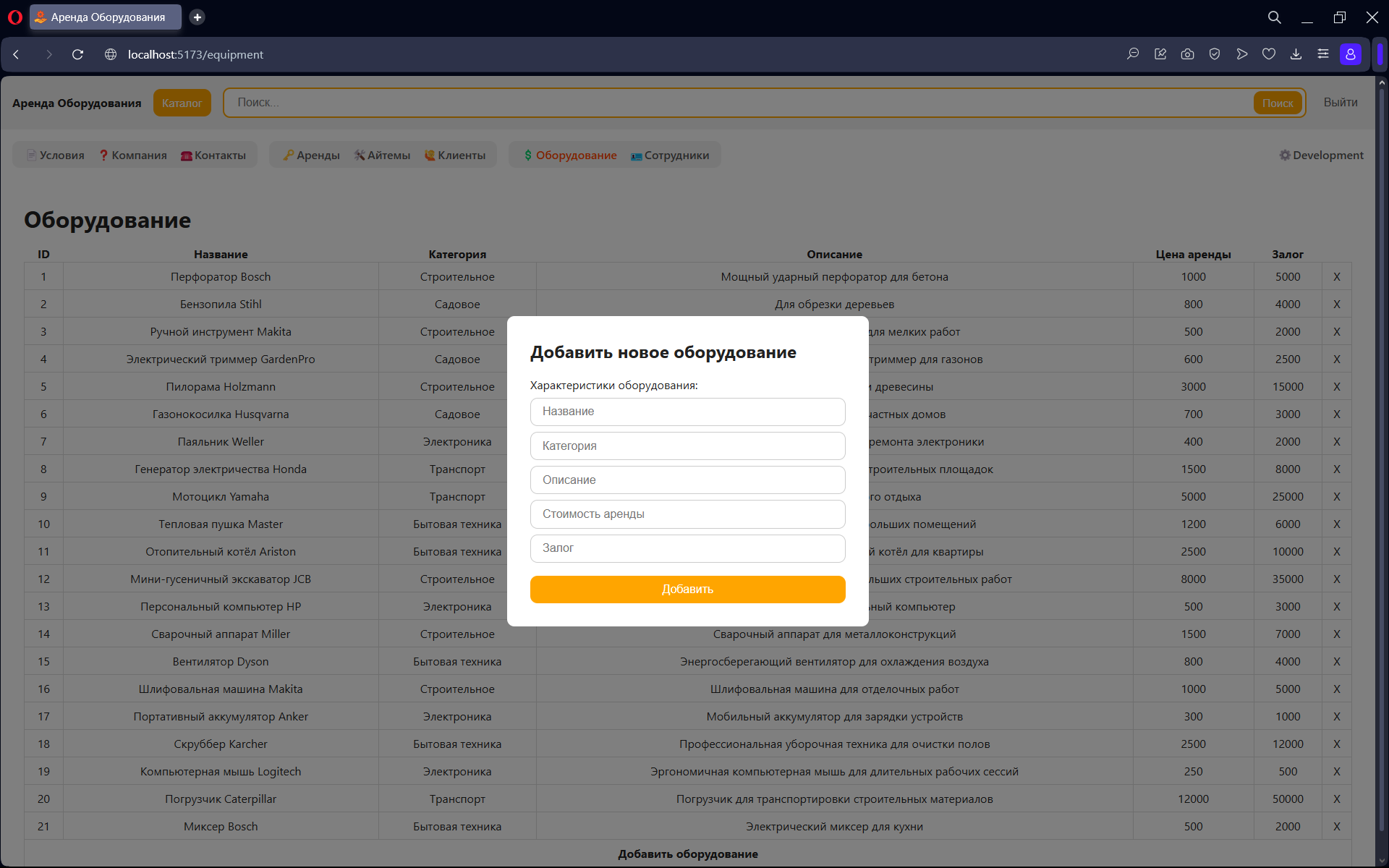


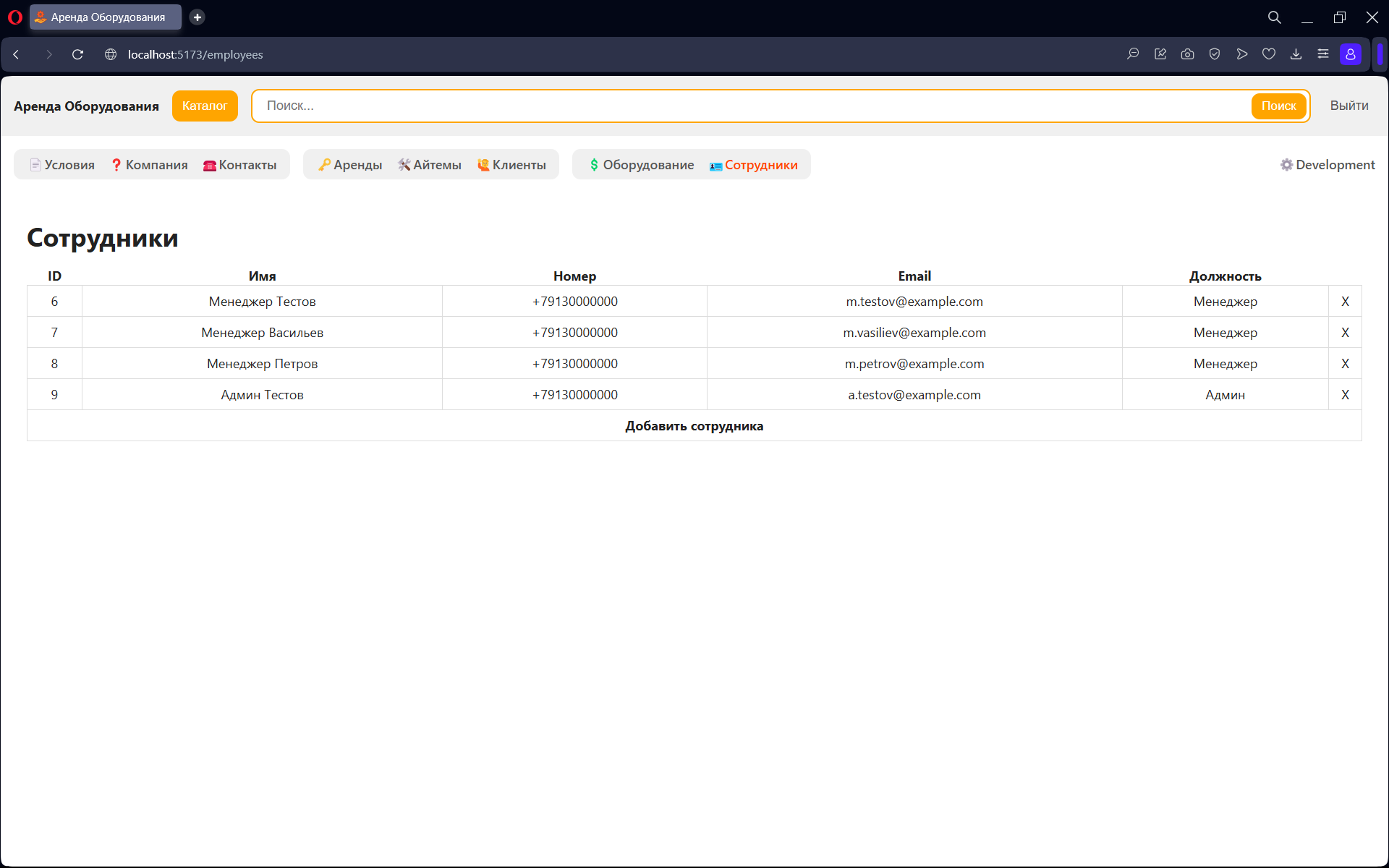
Если айтем не забронирован и не арендован, его можно удалить, нажав на крестик или сменить статус, нажав на статус (статус меняется по кругу: Доступно - На обслуживании - Списано).

Ещё менеджеру доступна страница со всеми клиентами, которые когда-либо что-то арендовали. Можно отсортировать таблицу по любому столбцу.

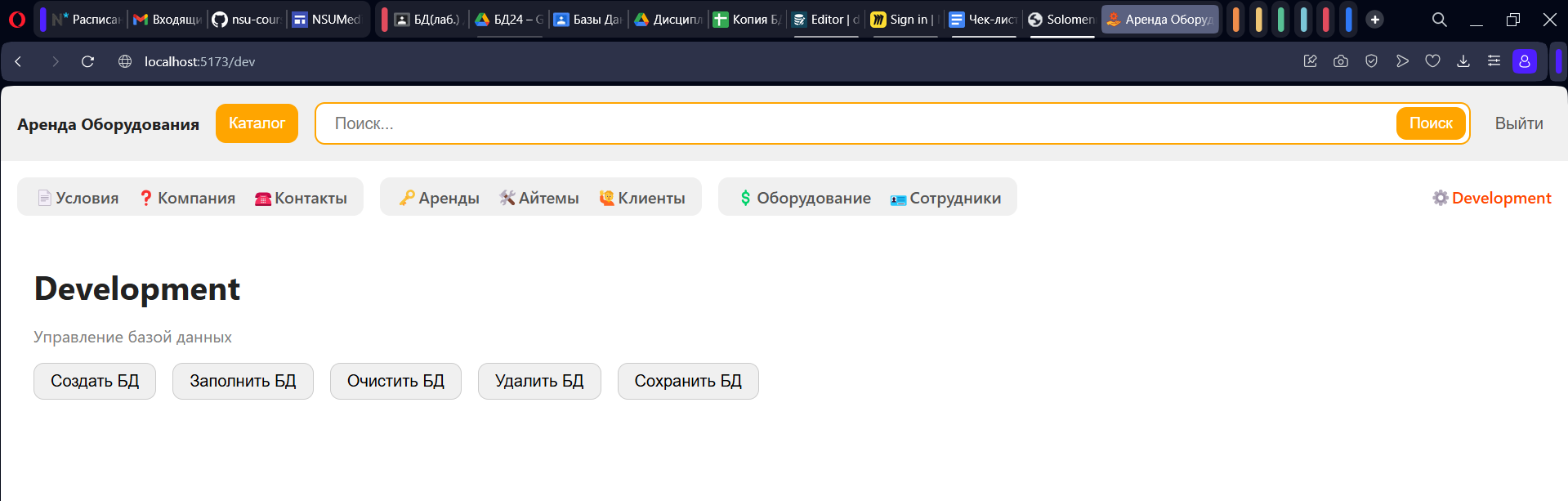
Если кликнуть по клиенту, появился история всех его завершённых аренд.

## Админский интерфейс

Администратор может управлять каталогом и прайсингом. Он может добавить оборудование и удалить его, если оно не участвует в активных бронях или арендах. Чтобы изменить данные оборудования, надо добавить оборудование с таким же названием, но с изменёнными характеристиками. И, конечно, можно сортировать таблицу по любому столбцу.

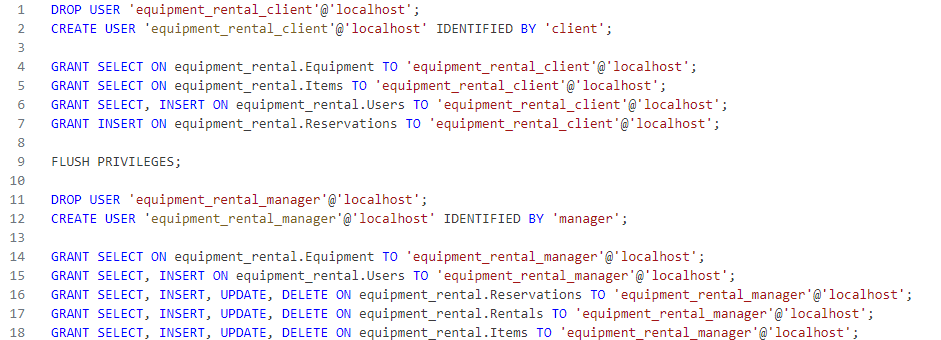
Также администратор может управлять учётными записями сотрудником.

## Интерфейс разработчика

Для удобной разработки был сделан интерфейс для управления БД. Можно создать, заполнить, очистить, удалить и сохранить БД. Сохранение БД работает так, что создаётся sql скрипт для заполнения БД по текущим данным. Это позволяет не терять данные даже при полном падении БД.

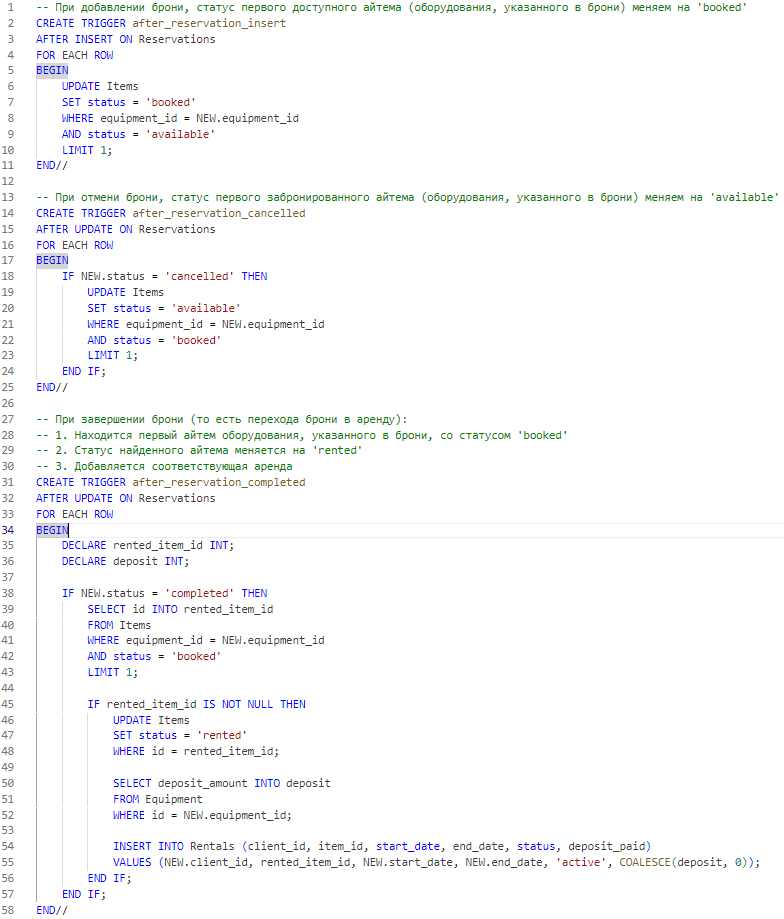
## Безопасность

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. Каждому пользователю даны ровно только нужные ему права:



## Триггеры

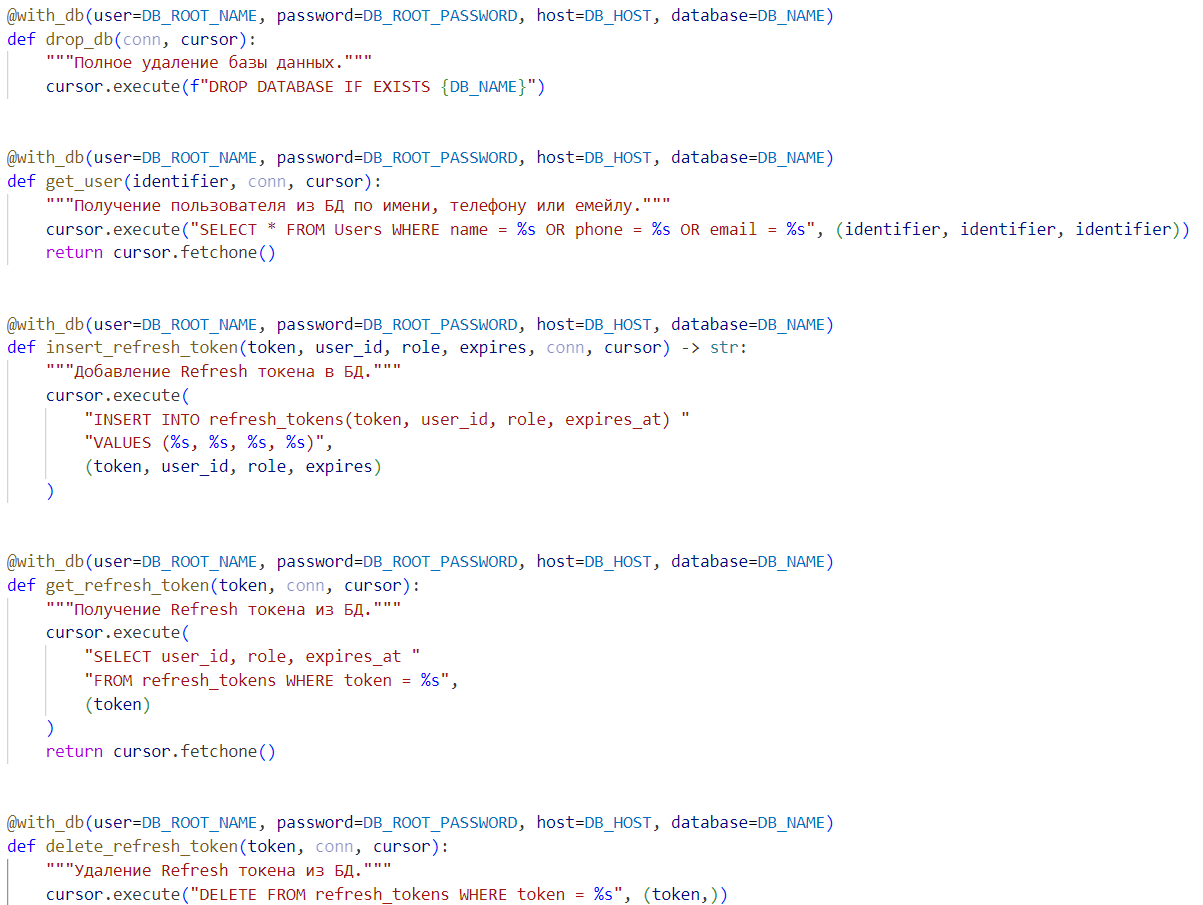
Для удобной работы с бронями, были реализованы триггеры:



## 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, поэтому поддерживать несколько соединений для каждого пользователя может быть ресурсозатратно. Пример кода:



## Тестирование

API тестировался с помощью заполнения БД тестовыми данными и работу через веб интерфейс.