

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной
математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Базы данных»
Тема: «Сеть оффлайн магазинов для продажи бытовой техники»

Студент: А.Д. Волков
Преподаватель: А.В. Малахов
Группа: М8О-306Б-22
Дата: 18.12.2024
Оценка:
Подпись:

Москва, 2024

Содержание

Формальные требования к курсовому проекту	2
Схема базы данных	4
Описание базы данных	4
Архитектура веб-приложения	13
Стек технологий	13
Контекстная диаграмма	14
Структура веб-приложения	15
Сервисы приложения	16
Пример сессии использования приложения	17
Пример использования приложения обычным пользователем	17
Пример использования приложения администратором	19
Выводы	21
QR-код репозитория	22

Формальные требования к курсовому проекту

1. При реализации курсового проекта допускается только использование СУБД PostgreSQL.
2. Необходимо выбрать предметную область для создания базы данных. Выбранная предметная область должна быть уникальной для всего потока, а не только в рамках учебной группы.
3. Необходимо описать модели предметной области и уровня инфраструктуры и их назначение в рамках реализуемого проекта (минимальное количество моделей предметной области и уровня инфраструктуры - 5). Также необходимо выполнить проектирование логической структуры базы данных. Все таблицы, связанные с описанными моделями предметной области, должны находиться в 3NF или выше. База данных должна иметь минимум 7 таблиц.
4. Необходимо разработать клиентское приложение (B2C) для доступа к информации из базы данных. Реализованное приложение должно быть понятно в плане использования (“приложение для домохозяек”). Выбор языков программирования и технологий разработки клиентского приложения произволен: C/C++, python, perl, ruby, JavaScript, php, swift, Java и др.
5. Необходимо организовать различные роли пользователей и права доступа к данным (например: администратор, редактор, рядовой пользователь). Клиентское приложение, взаимодействующее с базой данных, должно предоставлять функционал для авторизации пользователя по логину и паролю (хранение непосредственно пароля в базе данных запрещено).
6. Необходимо реализовать возможность создания администратором архивных копий базы данных и восстановления данных из клиентского приложения.
7. При разработке функционала базы данных следует организовать логику обработки данных не на стороне клиента, а на стороне серверного приложения (API), при этом клиентские приложения служат только для представления данных, выполнения запросов к данным и тривиальной обработки данных. Запросы к данным из базы данных должны выполняться асинхронно как для клиентского, так и для серверного приложения.
8. На стороне базы данных необходимо определить представления, триггеры, функции и хранимые процедуры, причем все эти объекты должны быть осмыслены, а их использование оправдано.
9. При демонстрации Вашего проекта необходимо уметь демонстрировать реализованный функционал уровня БД, уровня серверного приложения, уровня

клиентского приложения; уметь демонстрировать подключение к базе данных, основные режимы работы с данными (просмотр, редактирование, обновление и т. д.). Также необходимо подготовить скрипт SQL для инициализации структуры базы данных.

10. Необходимо реализовать корректную обработку различного рода ошибок, которые могут возникать при работе с базой данных.

Схема базы данных

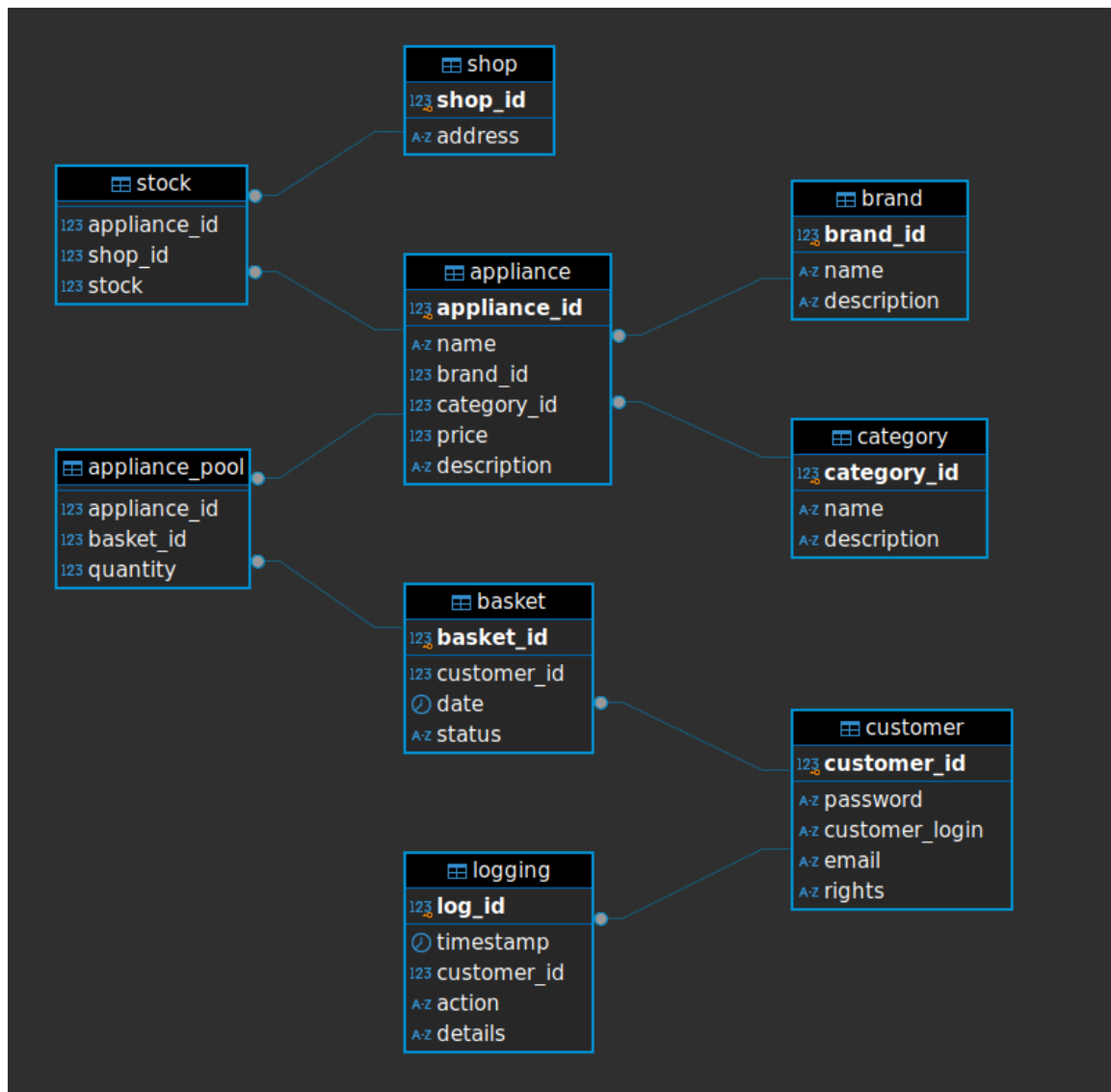


Рис. 1: Схема базы данных

Описание базы данных

Для начала опишем представленные таблицы:

1. **appliance** - таблица, в которой содержится вся техника, которая может быть представлена в магазинах сети. Рассмотрим поля в данной таблице:

- **appliance_id** - уникальный идентификатор техники, тип: **BIGINT**;
- **name** - наименование техники, тип: **VARCHAR()**;
- **brand_id** - уникальный идентификатор бренда техники, внешний ключ на таблицу **brand**, тип: **BIGINT**;
- **category_id** - уникальный идентификатор категории техники, внешний ключ на таблицу **category**, тип: **BIGINT**;
- **price** - цена техники за штуку, тип **FLOAT**;
- **description** - описание техники, тип **TEXT**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS appliance (
    appliance_id bigint generated always as identity
        primary key,
    name varchar(100) not null,
    brand_id bigint not null references brand(
        brand_id),
    category_id bigint not null references category(
        category_id),
    price float not null,
    description text
);
```

2. **customer** - таблица с данными заказчика. Рассмотрим поля в данной таблице:

- **customer_id** - уникальный идентификатор заказчика, тип: **BIGINT**;
- **password** - захешированный пароль заказчика, тип: **VARCHAR()**;
- **customer_login** - уникальный логин заказчика, тип: **VARCHAR()**;
- **email** - уникальная почта заказчика, тип: **VARCHAR()**;
- **rights** - права заказчика, тип: **VARCHAR()**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS customer (
    customer_id bigint generated always as identity
        primary key,
    password varchar(100) not null check(length(
        password) >= 3),
    customer_login varchar(150) not null check(length(
        customer_login) >= 3),
```

```

        email varchar(100) ,
        rights varchar(10) not null
    );

```

3. **shop** - таблица с магазинами сети. Рассмотрим поля в данной таблице:

- **shop_id** - уникальный идентификатор магазина, тип: **BIGINT**;
- **address** - адрес магазина, тип: **TEXT**;

Скрипт для создания таблицы

```

CREATE TABLE IF NOT EXISTS shop (
    shop_id bigint generated always as identity
        primary key,
    address text not null
);

```

4. **stock** - таблица, определяющая наличие техники в магазине. Рассмотрим поля в данной таблице:

- **appliance_id** - уникальный идентификатор техники, внешний ключ на таблицу **appliance**, тип: **BIGINT**;
- **shop_id** - уникальный идентификатор магазина, внешний ключ на таблицу **shop**, тип: **BIGINT**;
- **stock** - количество товара в магазине, тип: **BIGINT**;

Скрипт для создания таблицы

```

CREATE TABLE IF NOT EXISTS stock (
    appliance_id bigint not null references appliance
        (appliance_id),
    shop_id bigint not null references shop(shop_id),
    stock bigint not null
);

```

5. **brand** - таблица, в которой перечислены все возможные бренды. Рассмотрим поля в данной таблице:

- **brand_id** - уникальный идентификатор бренда, тип: **BIGINT**;
- **name** - название бренда, тип: **VARCHAR()**;
- **description** - описание бренда, тип: **TEXT**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS brand (
    brand_id bigint generated always as identity
    primary key,
    name varchar(100) not null,
    description text
);
```

6. **category** - таблица, в которой перечислены все возможные категории. Рассмотрим поля в данной таблице:

- **category_id** - уникальный идентификатор категории, тип: **BIGINT**;
- **name** - название категории, тип: **VARCHAR()**;
- **description** - описание категории, тип: **TEXT**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS category (
    category_id bigint generated always as identity
    primary key,
    name varchar(100) not null,
    description text
);
```

7. **basket** - таблица с корзинами пользователя. Рассмотрим поля в данной таблице:

- **basket_id** - уникальный идентификатор корзины, тип: **BIGINT**;
- **customer_id** - уникальный идентификатор заказчика, внешний ключ на таблицу **customer**, тип: **BIGINT**;
- **date** - дата создания корзины, тип: **DATE**;
- **status** - статус корзины, тип: **VARCHAR()**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS basket (
    basket_id bigint generated always as identity
    primary key,
    customer_id bigint not null references
    customer(customer_id),
    date date not null,
    status varchar(10) not null
);
```


8. **appliance_pool** - таблица с наборами одинаковой техники. Рассмотрим поля в данной таблице:

- **appliance_id** - уникальный идентификатор техники, внешний ключ на таблицу **appliance**, тип: **BIGINT**;
- **basket_id** - уникальный идентификатор корзины, внешний ключ на таблицу **basket**, тип: **BIGINT**;
- **quantity** - количество определенной техники в корзине, тип: **BIGINT**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS appliance_pool (  
    appliance_id bigint not null references  
        appliance(appliance_id),  
    basket_id bigint not null references basket(  
        basket_id),  
    quantity bigint not null  
);
```

9. **logging** - таблица для ведения логов корзины пользователя. Заполняется полностью автоматически с помощью триггеров. Рассмотрим поля в данной таблице:

- **log_id** - уникальный идентификатор лога, тип: **BIGINT**;
- **timestamp** - время лога, тип: **TIMESTAMP**;
- **customer_id** - уникальный идентификатор заказчика, от которого пришел лог, внешний ключ на таблицу **customer**, тип: **BIGINT**;
- **action** - действие заказчика, которое было залогировано, тип: **VARCHAR()**;
- **details** - краткое описание лога, тип: **TEXT**;

Скрипт для создания таблицы

```
CREATE TABLE IF NOT EXISTS logging (  
    log_id bigint generated always as identity  
        primary key,  
    timestamp timestamp not null default  
        current_timestamp,  
    customer_id bigint not null references  
        customer(customer_id),  
    action varchar(50) not null,  
    details text  
);
```

Теперь опишем представления, которые есть в базе данных.

1. **appliance_in_basket** - представление, которое собирает полную информацию и всей технике, которая находится в корзинах у пользователя. Такое представление было нужно для того, чтобы легко вычислять итоговую сумму для оплаты корзины пользователя.

Скрипт, который создает представление

```
create or replace view appliance_in_basket
as select customer_id, appliance_pool.basket_id,
    appliance_pool.appliance_id, appliance.name
    appliance_name, brand_id, brand.name
    brand_name, price, quantity, status
from appliance_pool join basket using(basket_id)
join appliance using(appliance_id)
join brand using(brand_id);
```

2. **appliance_with_shop** - представление, которое собирает полную информацию, о всех товарах во всех магазинах. С помощью этого представления будет удобно выводить товары на главной странице.

Скрипт, который создает представление

```
create or replace view appliance_with_shop
as select appliance.appliance_id, appliance.name
    appliance_name, brand.name brand_name,
    category.name category_name, appliance.price,
    stock.stock, stock.shop_id, shop.address,
    appliance.description appliance_description,
    brand.description brand_description, category.
    description category_description
from appliance join brand using(brand_id)
join category using(category_id)
join stock using(appliance_id)
join shop using(shop_id);
```

Теперь рассмотрим функции и триггеры, которые есть в базе данных.

1. **get_basket_price(basket_id integer)** - функция, вычисляющая итоговую стоимость корзины.

Скрипт который создает функцию

```
CREATE OR REPLACE FUNCTION get_basket_price(  
    basket_id integer) RETURNS FLOAT AS $$  
    select sum(price * quantity) total  
    from appliance_in_basket  
    where appliance_in_basket.basket_id =  
        basket_id  
    group by basket_id  
$$ LANGUAGE SQL;
```

2. **log_basket_changes()** - триггерная функция, для отслеживания изменений в таблице **basket**, для вставки их в таблицу **logging**.

Скрипт для создания функции и триггера

```
CREATE OR REPLACE FUNCTION log_basket_changes()  
    RETURNS TRIGGER AS $$  
BEGIN  
    IF TG_OP = 'INSERT' THEN  
        INSERT INTO logging (customer_id, action,  
            details)  
        VALUES (NEW.customer_id, 'CREATE_BASKET',  
            'Created a new basket');  
        RETURN NEW;  
    END IF;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER basket_create_trigger  
AFTER INSERT ON basket  
FOR EACH ROW EXECUTE FUNCTION log_basket_changes  
();
```

3. **log_appliance_pool_changes()** - триггерная функция, для отслеживания изменений в таблице **appliance_pool**, для вставки их в таблицу **logging**.

Скрипт для создания функции и триггера

```

CREATE OR REPLACE FUNCTION
    log_appliance_pool_changes()
RETURNS TRIGGER AS $$
declare
    user_id BIGINT;
BEGIN
    select customer_id into user_id from basket
        where basket_id = new.basket_id and status =
            'open';
    IF TG_OP = 'INSERT' then
        INSERT INTO logging (customer_id, action,
            details)
        VALUES (user_id, 'ADD_TO_BASKET', 'Added
            appliance with ID: ' || NEW.
            appliance_id || ' to basket with ID: '
            || NEW.basket_id || ' (Quantity: ' ||
            NEW.quantity || ')');
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO logging (customer_id, action,
            details)
        VALUES (user_id, 'UPDATE_BASKET', 'Added
            appliance with ID: ' || NEW.
            appliance_id || ' in basket with ID: '
            || NEW.basket_id || ' (Added quantity
            : ' || NEW.quantity || ')');
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER appliance_pool_add_trigger
AFTER INSERT OR UPDATE ON appliance_pool
FOR EACH ROW EXECUTE FUNCTION
    log_appliance_pool_changes();

```

Организация доступа данным в базе данных.

Для организации доступа был создан новый пользователь в базе данных, имеющий доступ только на запрос **SELECT**.

Скрипт для создания пользователя

```
CREATE ROLE slave_user WITH LOGIN PASSWORD 'qwerty';

GRANT SELECT ON appliance TO slave_user;
GRANT SELECT ON appliance_pool TO slave_user;
GRANT SELECT ON basket TO slave_user;
GRANT SELECT ON brand TO slave_user;
GRANT SELECT ON category TO slave_user;
GRANT SELECT ON customer TO slave_user;
GRANT SELECT ON shop TO slave_user;
GRANT SELECT ON stock TO slave_user;

GRANT SELECT ON appliance_in_basket TO slave_user;
GRANT SELECT ON appliance_with_shop TO slave_user;

GRANT EXECUTE ON FUNCTION get_basket_price TO slave_user;
```

Архитектура веб-приложения

Стек технологий

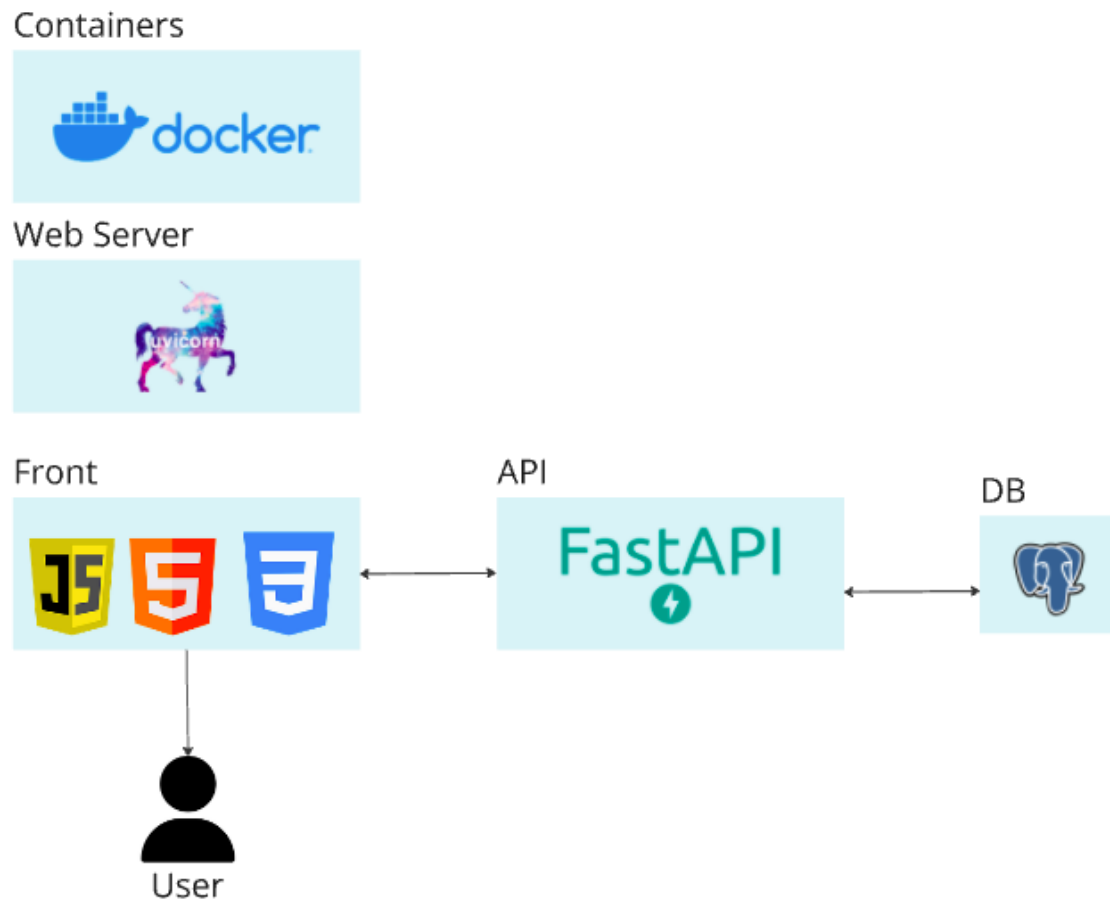


Рис. 2: Стек технологий веб-приложения

Приложение состоит из 2-х docker-контейнеров, один контейнер с СУБД PostgreSQL, второй контейнер с API на FastAPI. В качестве веб-сервера был выбран Unicorn.

Контекстная диаграмма

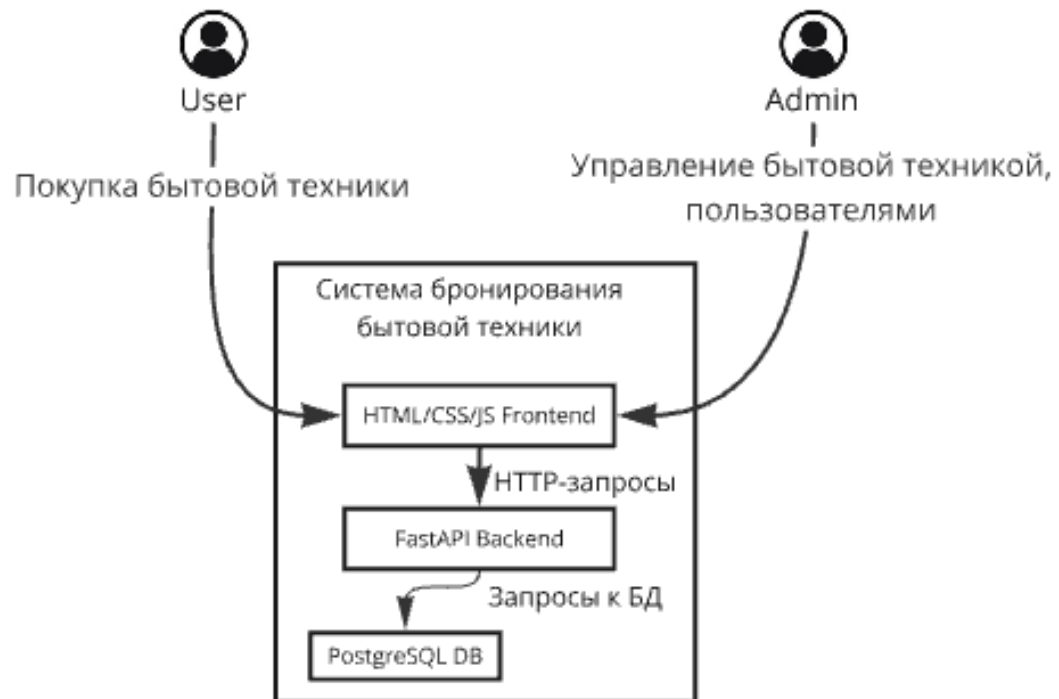


Рис. 3: Контекстная диаграмма

В данной контекстной диаграмме представлено взаимодействие разных типов пользователей с веб-сервисом.

Структура веб-приложения

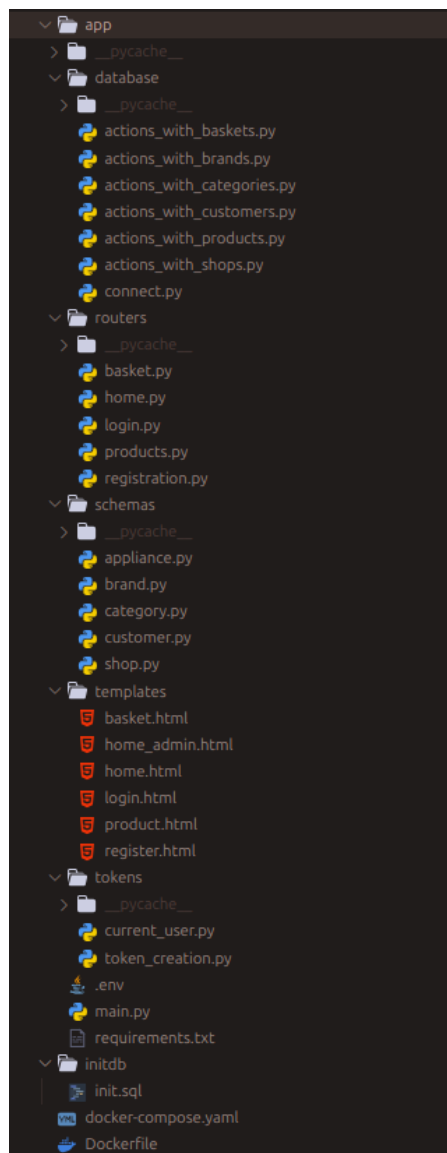


Рис. 4: Структура приложения

Сервисы приложения

```
services:
  web:
    build: .
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      DB_NAME: ${DB_NAME}
      DB_USER: ${DB_USER}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_SLAVE_USER: ${DB_SLAVE_USER}
      DB_SLAVE_PASSWORD: ${DB_SLAVE_PASSWORD}
      DB_HOST: ${DB_HOST}
      DB_PORT: ${DB_PORT}
      SECRET_KEY: ${SECRET_KEY}
      ALGORITHM: ${ALGORITHM}

  db:
    image: postgres:latest
    restart: always
    environment:
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
      POSTGRES_DB: ${DB_NAME}
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./initdb:/docker-entrypoint-initdb.d

volumes:
  postgres_data:
```

Рис. 5: Сервисы приложения

Пример сессии использования приложения

Пример использования приложения обычным пользователем

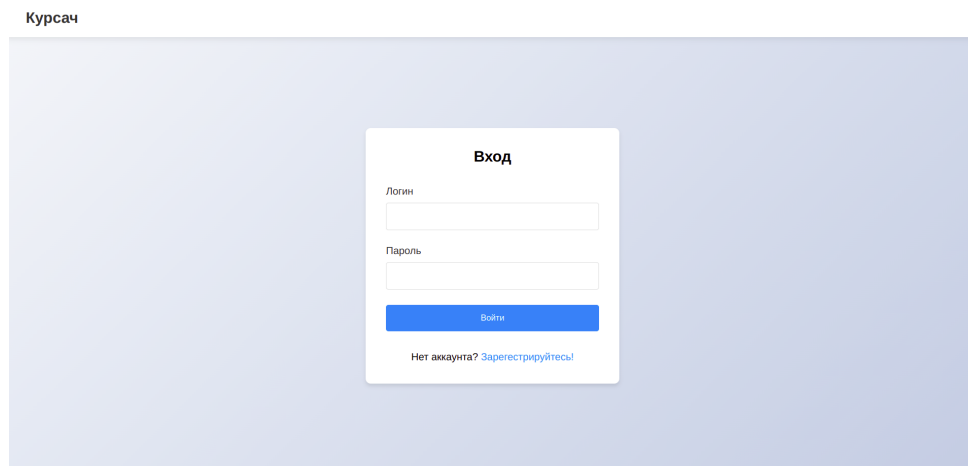


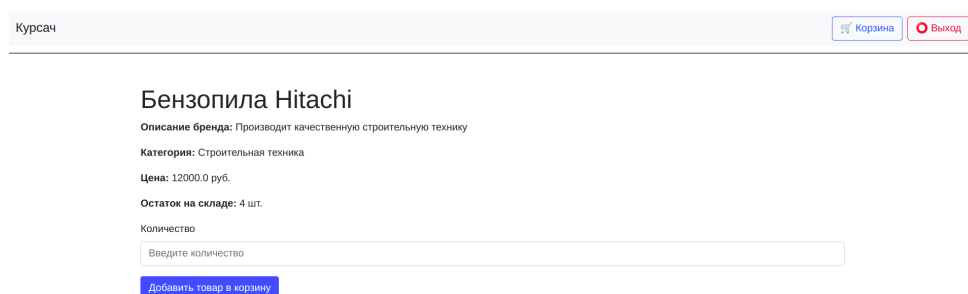
Рис. 6: Страница входа в аккаунт

Пользователь заходит в свой аккаунт и попадает на главную страницу со всеми доступными товарами.



Рис. 7: Домашняя страница с продуктами

Далее пользователь может выбрать товар, который хочет приобрести и ввести количество, которое он хочет приобрести.



Курсач

Корзина Выход

Бензопила Hitachi

Описание бренда: Производит качественную строительную технику

Категория: Строительная техника

Цена: 12000.0 руб.

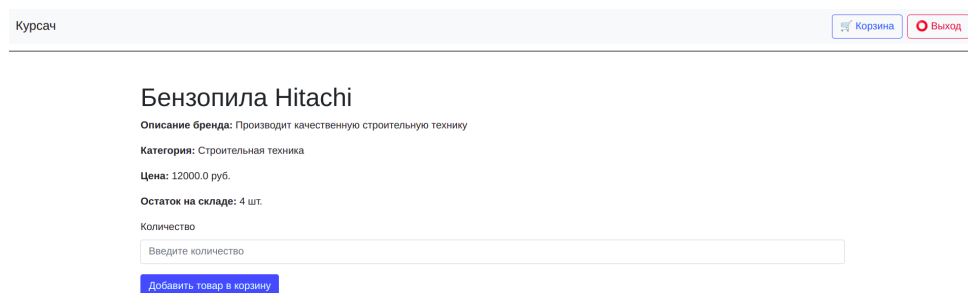
Остаток на складе: 4 шт.

Количество

Добавить товар в корзину

Рис. 8: Страница продукта

Потом пользователь может перейти в корзину, где ему сводная таблица по его корзине, итоговая стоимость корзины и кнопка, для того чтобы сделать заказ. После того как пользователь сделает заказ, товары удалятся из корзины и пойдут уже в сторонний сервис для оплаты заказов, но это уже не в нашей зоне ответственности.



Курсач

Корзина Выход

Бензопила Hitachi

Описание бренда: Производит качественную строительную технику

Категория: Строительная техника

Цена: 12000.0 руб.

Остаток на складе: 4 шт.

Количество

Добавить товар в корзину

Рис. 9: Страница корзины

Пример использования приложения администратором

Вместо обычной домашней страницы, администратору покажется администраторская панель, где можно будет управлять базой данных, а именно все техникой, магазинами, категориями, брендами и пользователями.

Управление базой данных

Добавить новую технику

Наименование товара

Бренды

Категории

Цена

Количество

Описание товара

Магазины

Удалить технику

Рис. 10: Панель для добавления новой техники

Описание товара

Магазины

Удалить технику

Название	Бренд	Магазин	Удалить
Бензопила	Samsung	Ул. Тверская д. 1	<input type="button" value="Удалить технику"/>
Бензопила	Hitachi	Ул. Тверская д. 1	<input type="button" value="Удалить технику"/>
Микроволновка	Samsung	Ул. Череповецкая д. 1	<input type="button" value="Удалить технику"/>

Добавить новый магазин

Адрес магазина

Рис. 11: Панель для удаления техники и добавления нового магазина

Удалить магазин

Адрес	Удалить
Ул. Тверская д. 1	Удалить магазин
Ул. Череповецкая д. 1	Удалить магазин

Добавить новый бренд

Название бренда

Описание бренда

[Добавить бренд](#)

Добавить новую категорию

Название категории

Описание категории

[Добавить категорию](#)

Рис. 12: Панель для удаления магазинов и добавления новых категорий и брендов

Добавить новый бренд

Название бренда

Описание бренда

[Добавить бренд](#)

Добавить новую категорию

Название категории

Описание категории

[Добавить категорию](#)

Сделать пользователя администратором

Имя пользователя	Назначить администратором
testuser	Назначить администратором

Рис. 13: Панель для назначения пользователей администратором

Выводы

В ходе выполнения данной курсовой работы, я ознакомился с набором технологий для взаимодействия с СУБД. Также подтянул свои знания в веб-разработке. Это поможет мне в будущем, так как умения работать с базами данных и писать простейшие SQL-запросы, это необходимые умения для любого программиста.

QR-код репозитория



Рис. 14: QR-код, ведущий в репозиторий проекта