Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Жизненный цикл разработки программного обеспечения

Лабораторная работа №3

Студенты: Е.В. Ткачев

Т.Я. Умаров

Преподаватель: Е.В. Богдан

МИНСК 2024

**Тип приложения**

Веб-приложение, используя архитектурный подход Rest API (клиент-серверная модель; клиент и сервер общаются посредством HTTP-запросов различных видов (GET, POST, PUT, DELETE) и передают данные в каком-либо формате, например JSON).

**Стратегия развёртывания**

Нераспределенное развертывание (то есть представление, данные и логика хранятся на одном сервере).

**Выбранные технологии**

Для написание backend’а используется язык программирования Python и фреймворк Django Rest Framework. Для frontend’а – Java Script и React. Язык программирования Python Framework был выбран в связи с простой написания кода и наличию удобного инструментария (IDE). Django Rest Framework был выбран в связи с удобностью построения Rest API и наличию мощной ORM, позволяющей работать с СУБД (PostgreSQL) без использования SQL-запросов.

Для написания frontend составляющей используется React.js. React имеет несколько преимуществ над другими фреймворками для создания web-приложение. Например: react обладает виртуальным DOM и имеет эффективный рендеринг; у react есть расширения языка Java Script - JSX; react обладает односторонним потоком данным и управления состоянием; и конечно же, react имеет огромное сообщество разработчиков и активную экосистему инструментов и библиотек.

**Основные показатели качества**

К основным показателям качества можно отнести: удобство и простота обслуживания (способность системы изменяться для реализации нового функционала), возможность взаимодействия (способность системы взаимодействовать с внешними системами), производительность (быстродействие системы), удобство и простота использования (определяет, насколько пользователю удобно пользоваться приложением).

**Пути реализации сквозной функциональности**

Протоколирование (логирование), авторизация и аутентификация (при создании и входе в аккаунт пользователю раскрывается полный функционал системы), управления исключениями.

Диаграмма компонентов приведена на рисунке 1.

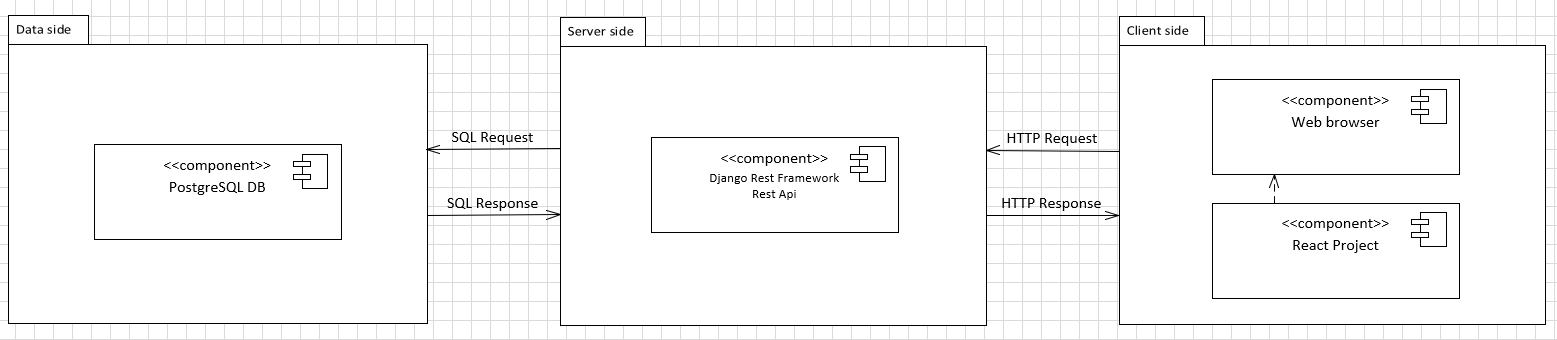


Рисунок 1 – диаграмма компонентов.

Диаграмма последовательностей приведена на рисунке 2.

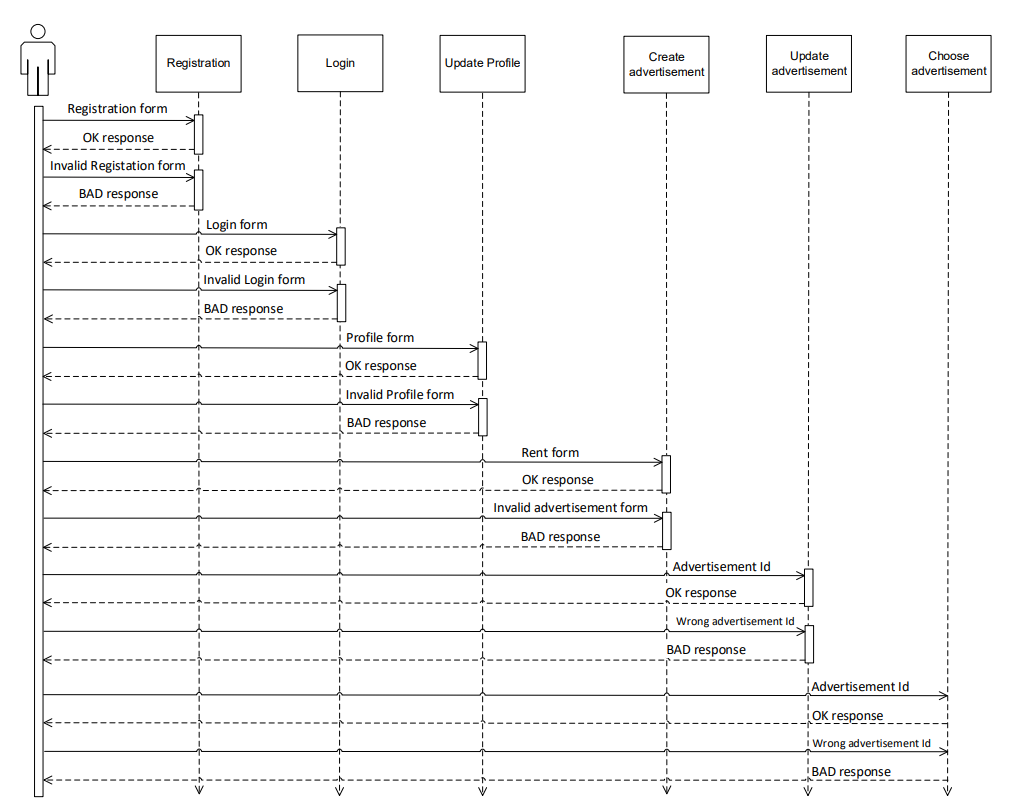


Рисунок 2 – Диаграмма последовательностей

Диаграмма классов реализованного функционала приведена на рисунке 3. На ней приведена иерархия реализованных моделей базы данных (пользователи, объявления, избранные объявления). На рисунке 4 приведена диаграмма классов frontend состовляющей.

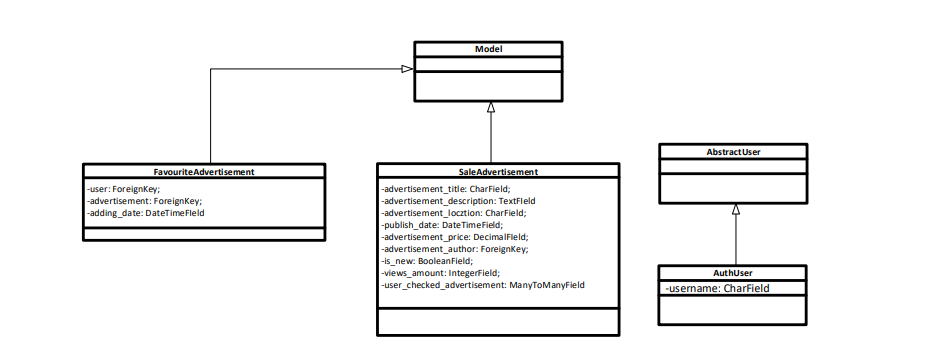


Рисунок 3 – Диаграмма классов backend

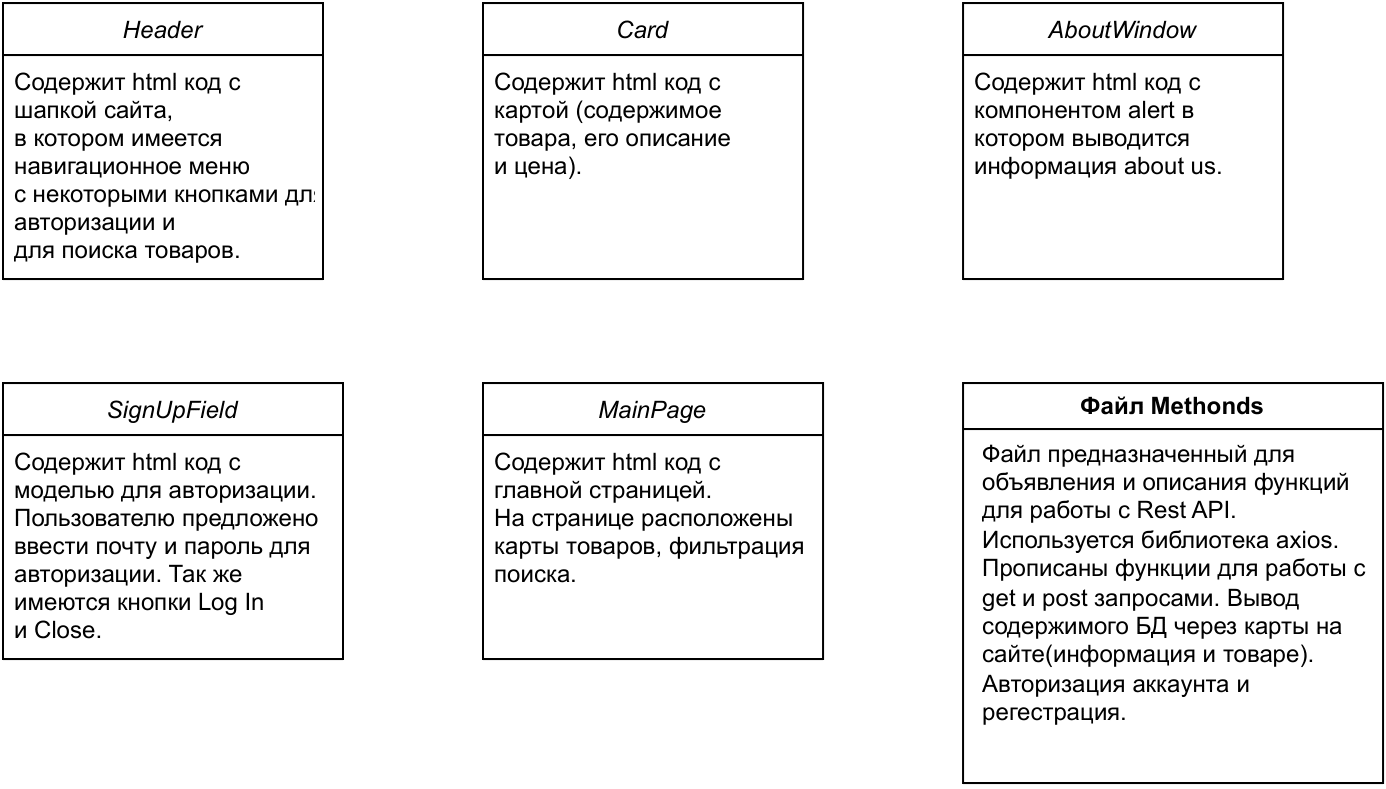
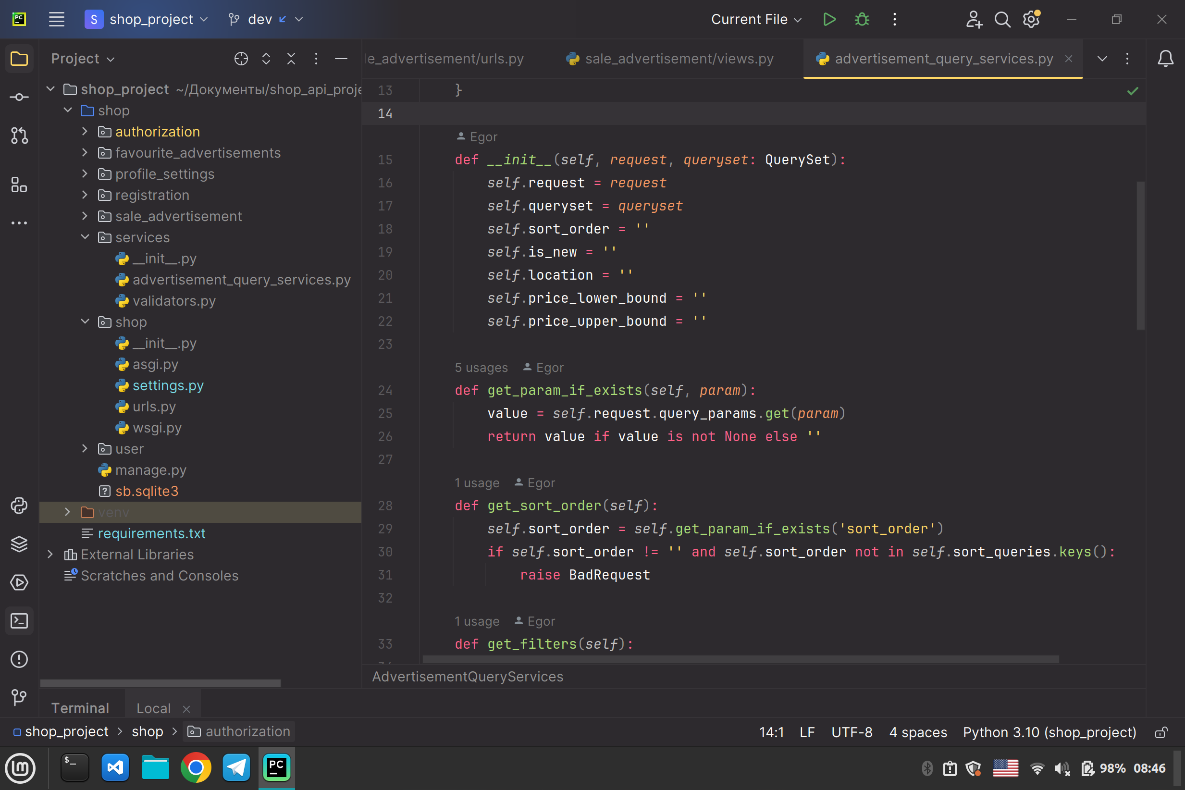


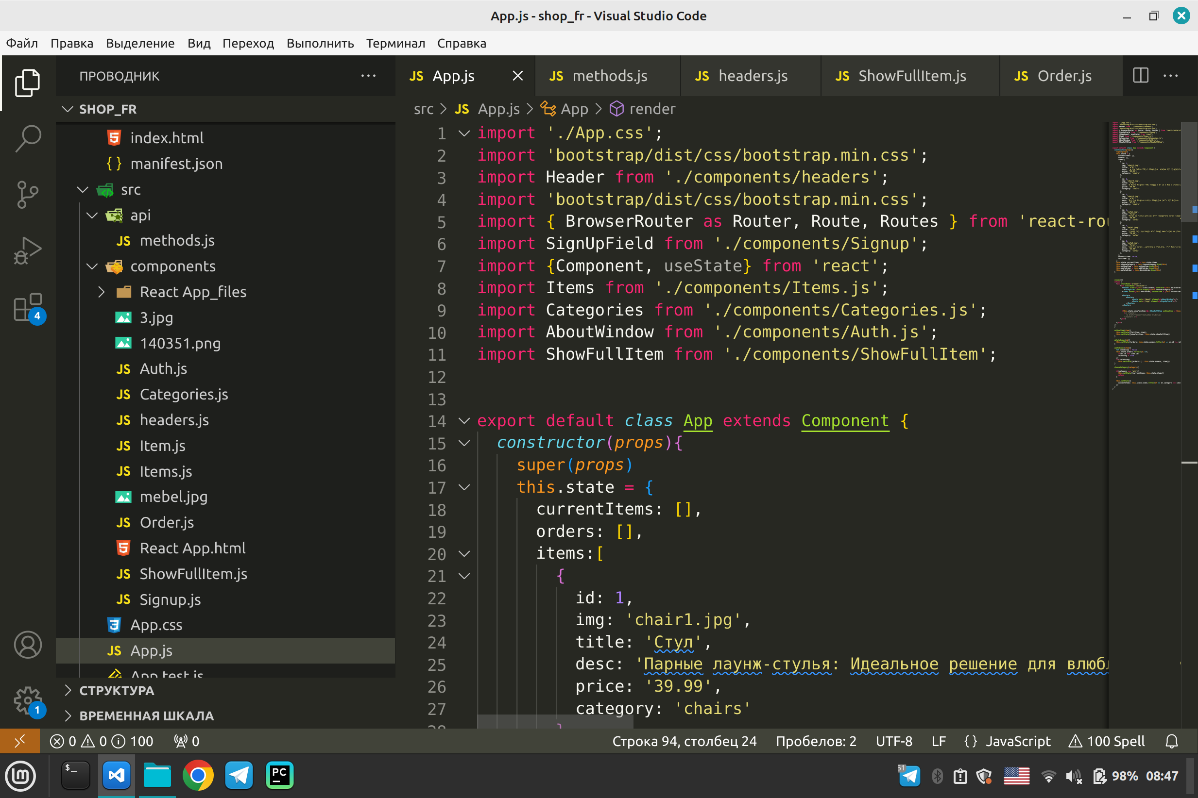
Рисунок 4 – Диаграмма классов frontend

**Сравнение архитектур**

Полученные представления, концепции и диаграммы классов позволяют на провести анализ и сравнение архитектуры «to be» и «as is». Далее на картинках 1 и 2 представлены участки кода и существующие компоненты.



Картинка 1 – Участок кода и компоненты (модели) backend составляющей



Картинка 2 – Участок кода и компоненты frontend составляющей

Исходя из этих картинок можно сделать вывод о проделанной работе. Заметим, что в указанных выше диаграммах классов и компонентов (рисунки 1 и 3) и картинками 1 и 2 просматривается соответствие планов и реальности. Были реализованы описанные компоненты для frontend и backend. Сделаем вывод, что была проведена успешная реализация планов и минимальное количество проблем при переходе от каждого состояния приложения к текущему и последующим состояниям. Имеем ясность и прозрачность в процессе анализа и проектирования. Любые изменения архитектуры основаны на четком понимании текущего состояния приложения и его будущему состоянию.