МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (МОСКОВСКИЙ ПОЛИТЕХ)**

Факультет информационных технологий

Кафедра «Инфокогнитивные технологии»

**КУРСОВОЙ ПРОЕКТ**

на тему: *«Веб-приложение онлайн-аукцион»*

Направление подготовки 09.03.03 «Прикладная информатика»

Профиль «Корпоративные информационные системы»

**Выполнил:**

студент группы 211-362

Шарков Иван Александрович

30.06.2023

(подпись)

Москва 2023

**СОДЕРЖАНИЕ**

[1 ОБЩИЕ СВЕДЕНИЯ 3](#_Toc138850967)

[2 ТЕХНОЛОГИЧЕСКИЙ СТЕК 4](#_Toc138850968)

[3 ОПИСАНИЕ КЛАССОВ 6](#_Toc138850969)

[4 НАСТРОЙКА И ЗАПУСК 8](#_Toc138850970)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](#_Toc138850971)

# ОБЩИЕ СВЕДЕНИЯ

Целью проекта является создание удобного веб-приложения, позволяющего продавать ценные и раритетные вещи в формате аукциона в режиме онлайн, а также изучение фреймворка Django языка программирования Python.

Проект «Онлайн-аукцион» реализован в формате веб-приложения. Пользователи приложения имеют возможность регистрироваться, размешать свои собственные лоты на доступных и открытых аукционах, а также создавать свои тематические аукционы.

Для создания лота или аукциона необходимо ввести: название, описание, дату начала и дату окончания продажи. Лот считается проданным, когда заканчивается время на таймере в соответствии с выбранным временем окончания продажи. Создатель лота также имеет возможность завершить продажу досрочно по соответствующей кнопке на странице выбранного лота, но только при наличии хотя бы одной ставки. Аукцион считается закрытым, если вышло время на таймере в соответствии с датой закрытия аукциона.

Авторизированные пользователи имеют свой профиль, в котором представлена общая информация об аккаунте, такая как логин и почта, а также таблицы созданных пользователем аукционов и лотов.

# ТЕХНОЛОГИЧЕСКИЙ СТЕК

Клиентская и серверная часть приложения разделены на отдельные серверы. Общение между частями приложения происходит посредством клиент-серверного взаимодействия с использованием протокола REST. Клиентская и серверная часть расположены на сервисе контроля версий GitHub в отдельных репозиториях.

Серверная часть реализована с помощью языка программирования Python. Для реализации приложения в стиле REST API используется фреймворк Django 4.2.2 версии с библиотекой rest-framework. В качестве базы данных используется встроенная в Django база данных SQLite.

Архитектура приложения реализована самим фреймворком Django с добавлением частично разделенной на изолированные слои «Onion» архитектурой:

* слой адаптеров (routers) – используется для определения конечных api-маршрутов;
* слой контроллеров (controllers) – используется для обработки отдельных HTTP-запросов;
* слой сервисов (services) – используется для обработки бизнес-логики приложения, принимая обработанные запросы из соответствующих контроллеров.

Клиентская часть веб-приложения «Онлайн-аукцион» реализована с помощью языка Typescript, а именно с использованием библиотеки React 18 версии. Для хранения глобальный состояний приложения используется библиотека Redux с популярной сборкой Redux Toolkit. Также в react-приложении используются линтеры и форматеры коды, такие как ESLint и Prettier.

Архитектура приложения построена в нотации Feature-Sliced Design. Приложение разделено на логические функциональные блоки, такие как:

* app (глобальные настройки приложения) – настройки, стили и провайдеры для всего приложения;
* processes (процессы) – сложные сценарии, покрывающие несколько страниц;
* pages (страницы) – композиционный слой для сборки полноценных страниц из сущностей, фич и виджетов;
* widgets (виджеты) — композиционный слой для соединения сущностей и фич в самостоятельные блоки;
* features (фичи) — взаимодействия с пользователем, действия, которые несут бизнес-ценность для пользователя;
* entities (сущности) — бизнес-сущности;
* shared (переиспользуемые) — переиспользуемый код, не имеющий отношения к специфике приложения/бизнеса.

# ОПИСАНИЕ КЛАССОВ

Серверное Django-приложение содержит следующие классы контроллеры:

* AuctionController – класс, отвечающий за обработку пользовательских HTTP-запросов, связанных с какими-либо изменениями сущности аукциона. Класс содержит стандартные CRUD (Create, read, update, delete) методы, а также метод «close» для закрытия определённого аукциона и «search» для поиска аукционов по заданной строке;
* AuthController – класс, отвечающий за регистрацию и авторизацию пользователей. Содержит такие методы, как: «register», «login», «getSessionUserData» и «logout» для взаимодействия с сессионной системой авторизации, реализованной в Django;
* BidContoller – класс, отвечающий за обработку HTTP-запросов, связанных с сущностью ставки. Класс содержит стандартные CRUD методы, а также метод «getUserBidByLotId» для предоставления данных о ставке пользователя в конкретном лоте;
* LotController – класс, отвечающий за обработку HTTP-запросов, связанных с сущностью лота. Класс содержит стандартные CRUD методы, а также метод «finish» для завершения аукциона и объявления победителя;
* UserController – класс, отвечающий за обработку HTTP-запросов, связанных с сущностью пользователя. Класс содержит стандартные CRUD методы.

Слой сервисов содержит следующие классы:

* AuctionService – класс, отвечающий за обработку соответствующих запросов из класса AuctionController и выполняющий бизнес-логику и прямое взаимодействие с базой данных аукционов;
* AuthService – класс, отвечающий за обработку соответствующих запросов из класса AuthController и выполняющий бизнес-логику и прямое взаимодействие с базой данных пользователей;
* BidService – класс, отвечающий за обработку соответствующих запросов из класса BidController и выполняющий бизнес-логику и прямое взаимодействие с базой данных ставок;
* LotService – класс, отвечающий за обработку соответствующих запросов из класса LotController и выполняющий бизнес-логику и прямое взаимодействие с базой данных лотов;
* UserService – класс, отвечающий за обработку соответствующих запросов из класса UserController и выполняющий бизнес-логику и прямое взаимодействие с базой данных пользователей;

Конечные api-маршруты описаны в файле urls.py. Настройка приложения осуществляется путем настройки файла settings.py.

Клиентское react-приложение содержит компоненты формата .tsx, а также redux-файлы формата .ts для описания логики взаимодействия с сервером и хранения данных в состоянии. Общение с сервером происходит путем отправки HTTP-запросов при помощи библиотеки axios на конечные маршруты.

Все переменные конфигураций находятся в защищенных файлах .env, содержащие данные для различных настроек, таких как для CORS политики.

# НАСТРОЙКА И ЗАПУСК

Примеры настроек и запуска частей приложения продемонстрировано с использованием редактора «VSCode» и его встроенного терминала.

Чтобы запустить клиентское react-приложение, необходимо:

1. клонировать репозиторий в любую папку (URL: https://github.com/scharkoff/auction-react);
2. открыть терминал редактор;
3. скачать все необходимые зависимости командой «npm install»;
4. запустить приложение командой «npm run start»;
5. открыть приложение по адресу localhost:3000.

Чтобы запустить серверное Django приложение, необходимо:

1. клонировать репозиторий в любую папку (URL: https://github.com/scharkoff/auction-api);
2. перейти в корневую папку «src» командой «cd ./src»;
3. скачать все необходимые зависимости командой «pip install -r requirements.txt»;
4. запустить приложение командой «python manage.py runserver».

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация по фреймворку и библиотекам Django. [Электронный ресурс] – URL: https://django.fun/ru/docs/ (дата обращения: 30.06.2023).
2. Документация по библиотеке rest-framework django. [Электронный ресурс] – URL: https://django.fun/ru/docs/django-rest-framework/3.12/ (дата обращения: 30.06.2023).
3. Документация по библиотеке React. [Электронный ресурс] – URL: https://reactdev.ru/learn/ (дата обращения: 30.06.2023).
4. Документация по библиотеке Redux. [Электронный ресурс] – URL: https://rajdee.gitbooks.io/redux-in-russian/content/ (дата обращения: 30.06.2023).
5. Документация по библиотеке Redux Toolkit. [Электронный ресурс] – URL: https://reactdev.ru/libs/redux-toolkit/ (дата обращения: 30.06.2023).
6. Обзор архитектуры Feature-Sliced Design. [Электронный ресурс] – URL: https://feature-sliced.design/ru/docs/get-started/overview (дата обращения: 30.06.2023).