ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Журнал практики

Институт № 8 «Компьютерные науки и прикладная математика»

Кафедра 806 Учебная группа М8О-312Б-22

ФИО обучающегося Андрюшин Лев Дмитриевич

Направление подготовки/ специальность 01.03.02 Прикладная математика и информатика

шифр, наименование направления подготовки/специальности

Вид практики Научно-исследовательская работа (получение первичных навыков научно-исследовательской работы)

учебная, производственная, преддипломная или другой вид практики

Оценка за практику _____ Крылов С.С.

Москва

2025

1. **Место и сроки проведения практики:**

Наименование организации: Кафедра 806

Сроки проведения практики

дата начала практики: 10.02.2025

дата окончания практики: 08.06.2025

2. **Инструктаж по технике безопасности:**

_____ /____Крылов С.С._/ ___10 _февраля___ 2025г.

подпись проводившего *расшифровка подписи* *дата проведения*

3. **Индивидуальное задание обучающегося:**

Разработка онлайн-магазина с возможностью поиска и просмотра товаров, продавцов, анализом пользовательской активности и комментариев. Писал функционал для неавторизованных пользователей, общий функционал для авторизованных пользователей всех типов, а так же для клиентов и большую часть функционала админа.

4. **План выполнения индивидуального задания обучающегося:**

№ п/п	Место проведения	Тема	Период выполнения
1	Кафедра 806	Инструктаж.	10.02.2025 – 10.02.2025
2	Кафедра 806	Ознакомление с архитектурами и выбор стека проекта, изучение стека	11.02.2025 – 25.02.2025
3	Кафедра 806	Работа над проектом	26.02.2025 – 16.05.2025
4	Кафедра 806	Объединение проекта в общее приложение	17.05.2025 – 01.06.2025
5	Кафедра 806	Оформление отчета. Подведение итогов.	02.06.2025 – 08.06.2025

Утверждаю

_____ /____Крылов С.С._/ ___10 _февраля__ 2025г.

подпись руководителя от МАИ *расшифровка подписи* *дата утверждения*

_____ /____Крылов С.С._/ ___10 _февраля__ 2025г.

подпись руководителя от организации/предприятия *расшифровка подписи* *дата утверждения*

Ознакомлен

_____ /___Андрюшин Л.Д.___/ ___10 _февраля__ 2025г.

подпись обучающегося *расшифровка подписи* *дата ознакомления*

5. Отзыв руководителя практики от организации/предприятия:

Обучающийся группы М8О-312Б-22 Андрюшин Л. Д. проходил практику на кафедре 806. Прослушаны установочные лекции. В течение практики были осуществлены поставленные задачи в полном объеме, реализованы необходимые эндпоинты в соответствии с поставленными целями, а также реализованы итеграционные тесты для всех реализованных эндпоинтов. За время прохождения практики, практикант показал необходимый уровень развития практических навыков и компетенций в процессе выполнения индивидуального задания. Задание практики выполнено. Рекомендую оценку _____. Материалы, изложенные в отчете обучающегося, полностью соответствуют индивидуальному заданию.

_____ /____Крылов С.С._ / 8 июня 2025 г.

_

подпись руководителя от организации/предприятии | *расшифровка подписи* | *дата*

6. Отчет обучающего по практике:

В рамках данного проекта я решал задачу разработки "Разработка онлайн-магазин". Основной целью было организация полноценной системы для просмотра и поиска товаров, продавцов и анализа комментариев с поддержкой поискового движка, и дашборда аналитики просмотров и пользовательской активности. Общей задачей было разрработать архитектуру приложения и обосновать ее выбор. Архитектура представлена на рисунке 1.

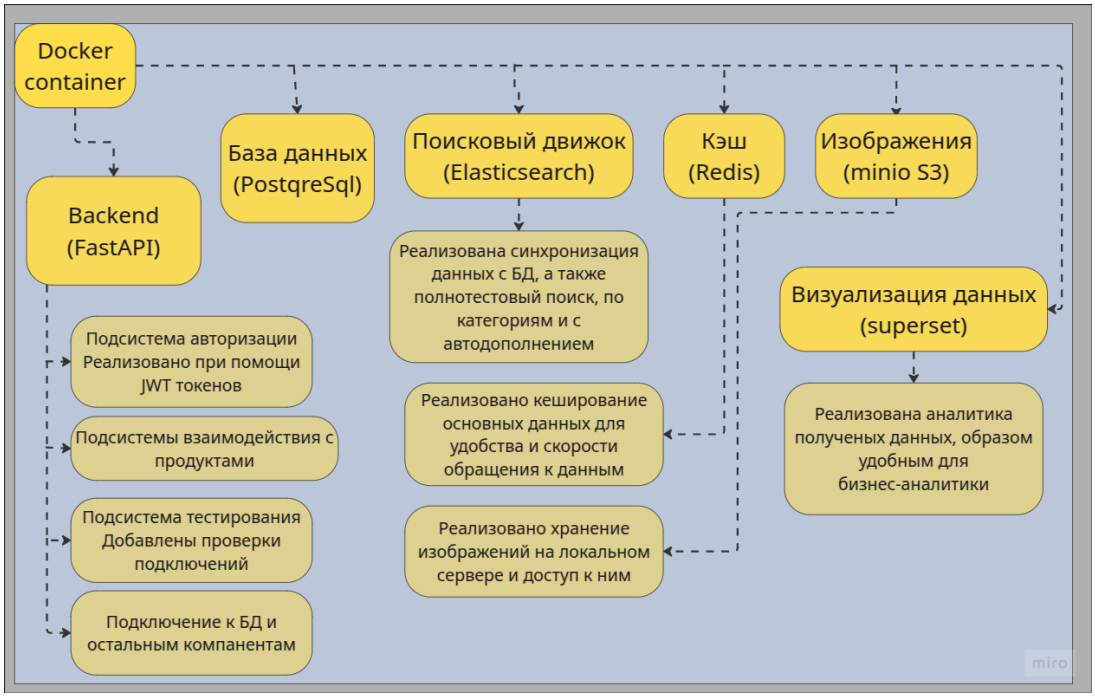

Рисунок 1 – Архитектура приложения

Мы остановились на монолитной архитектуре в силу её ключевых преимуществ, которые оптимально соответствуют требованиям нашего проекта.:

1. Простота разработки и развертывания

Использование FastAPI в качестве единого ядра приложения значительно упрощает процесс разработки API. Вся бизнес-логика (аутентификация, управление товарами, обработка заказов) сосредоточена в одном месте, что минимизирует сложность взаимодействия между компонентами и ускоряет внесение изменений.

2. Высокая производительность и минимальные задержки

Монолитная архитектура особенно эффективна для систем с тесной связностью компонентов, таких как онлайн-магазин. В нашем случае каталог товаров, корзина и система заказов взаимодействуют напрямую, а обработка запросов в рамках одного процесса исключает накладные расходы, характерные для микросервисов (сетевые вызовы, сериализация данных).

3. Лёгкость управления транзакциями и согласованностью данных

Поскольку основная бизнес-логика работает с одной базой данных (PostgreSQL), проще обеспечивать ACID-транзакции. Например, при оформлении заказа можно

атомарно обновить баланс пользователя, уменьшить количество товара и создать запись в истории заказов.

На рисунке 2 представлена структура нашего проекта.

```
backend/
├── auth/
├── catalog/
│   ├── admin/
│   ├── basic/
│   ├── client/
│   ├── basic_authorization/
│   ├── seller/
│   └── search/
├── elastic/
├── debug/
├── db.py
└── main.py
```

Рисунок 2 – Структура проекта

Описание структуры проекта

В папке auth реализованы регистрация обычных пользователей, авторизация, выход, регистрация продавцов/админов через подтверждение админа.

В папке elastic различные для поискового движка

В папке catalog/basic реализован весь доступный функционал для неавторизованных пользователей.

В папке catalog/basic_authorization реализован общий функционал доступный для всех авторизованных пользователей, а так же пару функкий реализованных для админа

В папке catalog/client реализован функционал для пользователей только типа «user» (взаимодействие с корзиной и покупка)

В папке catalog/admin реализованы функционал администратора для бана/разбана пользователей и товаров.

В папке catalog/search реализан функционал поискового движка

В папке catalog/seller реализованы функции для взаимодействия продавцов со своими товарами, а так же ETL пайплайн

Стек технологий

Для реализации проекта был выбран следующий стек технологий, каждая из которых играет роль в архитектуре системы:

Python + FastAPI: Python используется для разработки бекенд-сервисов, обеспечивающих API для работы с товарами, пользователями и комментариями. FastAPI позволяет быстро разворачивать RESTful-эндпойнты, обрабатывать запросы и подключаться к базе данных благодаря асинхронной обработке и большому количеству готовых расширений.

PostgreSQL: в качестве основной реляционной СУБД задействован PostgreSQL. Он хранит структуру каталога товаров, информацию о продавцах, пользователях и комментариях. PostgreSQL обеспечивает транзакционность (ACID), расширяемость (JSONB-поля, полнотекстовый поиск) и легко масштабируется при росте нагрузки.

Elasticsearch: используется в роли поискового движка для быстрого полнотекстового поиска товаров и продавцов. Elasticsearch индексирует описания и метаданные, что позволяет выполнять операции поиска по ключевым словам, фильтрации и ранжирования с низкой задержкой.

Apache Superset: предназначен для визуализации аналитики просмотров и пользовательской активности. Superset развёрнут в контейнере и подключается к PostgreSQL для построения дашбордов, диаграмм и отчетов по метрикам и логам.

Promtail + Grafana Loki: Promtail собирает логи из контейнеров (бэкенд-сервисов, базы данных и т.д.) и отправляет их в Loki для индексирования. Loki хранит структурированные логи, что позволяет в Grafana быстро искать и фильтровать записи, строить оповещения (alerts) при аномалиях.

Docker Compose: оркестрация всех компонентов: бекенд, PostgreSQL, Elasticsearch, Superset, Promtail/Loki. Благодаря docker-compose.yml можно за несколько команд поднять локальную или тестовую среду со стэком из контейнеров, настроить сетевое взаимодействие между ними и задать общий механизм конфигурации.

Python scripts (ETL-пайплайн): отдельные скрипты на Python реализуют ETL-процессы для переноса описаний товаров в объектное хранилище (S3-совместимое или минIO), а также для сбора и обработки пользовательской активности (напрямую из логов или по событиям). Они запускаются по расписанию (cron) или через контейнеры в режиме batch, формируя готовые данные для аналитики и поиска.

Redis : может быть задействован как кэш для ускорения запросов к часто запрашиваемым товарам и снижения нагрузки на PostgreSQL — например, для хранения сессий пользователей или промежуточных результатов поиска.

Grafana: в связке с Loki используется для визуализации метрик и логов. Grafana строит дашборды по основным показателям производительности, позволяет отслеживать состояние сервисов, пиковые нагрузки и цепочки запросов.

Docker Registry : все образы сервисов (бэкенд, Superset, Promtail и т.д.) собираются и публикуются в приватный или публичный Docker Registry, что упрощает CI/CD и масштабирование.

В рамках проекта я выполнял роль тимлидера, организовывая работу в команде и эффективно распределеяя обязанности между участника. Как разработчик мною были реализован функционал для неавторизованных пользователей: просмотр списка товаров с различными критериями сортировки, просмотр всей информации об отдельном товаре, просмотр комментариев, просмотр категорий товаров и списка продавцов; общий функционал (доступный всем ролям) для авторизованных пользователей: редактирование профиля и написание комментариев (были реализованы «ветки» комментариев благодаря столбцы replay_to_comment_id в талице с комментариями, который принимал NULL, если это первый комментарий в ветке и id комментария, если это ответ на чей-то комментарий); функционал для клиентов: оценка товаров, возможость

просмотра своих заказов, возможность удалять собственные комментарии (сами комментарии сохраняются, чтобы не ломать ветки комментариев, но они перестают ссылаться на id пользователя, а текст комментария заменяется на «[удалено]»), возможность добавлять и удалять товар из корзины, а так же возможность покупать товары; часть функционала администратора: возможность банить пользователей не админов (реализовано через столбец is_active, который является булевым, если у какого-то пользователя (и клиента и продавца) он принимает значение false, то он теряет весь свой функционал и может только авторизоваться и выйти из своего аккаунта), возможность разбанивать пользователей, возможность запрещать/разрешать продажу отдельных товаров или всех товаров от определенного продавца, возможность удалять любые комментарии (и продавцов и клиентов и свои сосбтвенные, но не других админов), а так же я добавил возможность администратору получить список покупок любого пользователя. Помимо этого я в некоторых местах так же добавил кеширование с помощью Redis для данных, которые нам часто нужны, но моментельно отслеживать изменения которых нам не столь критично, например для средней оценки товара.

_____	/___Андрюшин Л.Д.___/	08 июня 2025 г.
подпись обучающегося	*расшифровка подписи*	*дата*