**Дипломная работа по теме: “ Сравнение производительности и удобства использования различных ORM (Object-Relational Mapping) библиотек: Django ORM, SQLAlchemy и Tortoise ORM”**

Автор: Цесенко Максим Александрович

**Оглавление дипломной работы:**

1. **Введение**

**Обоснование выбора темы:**

1. Наиболее часто встречающимся способом хранения данных является базы данных, так как обеспечивают безопасность данных (операция либо выполняется полностью, либо не выполняется, то есть риск потери данных из-за неполностью выполненной операции минимален.), уже реализован основной функционал работы с данными (нет необходимости его разрабатывать самостоятельно, что снижает затраты на разработку проектов).
2. Потребности рынка: существует растущая потребность в специалистах, обладающих навыками применения различных баз данных.
3. Практическая значимость: Работа в области разработки веб приложения позволит применить полученные знания на практике и даст опыт использования различных ORM на практике.
4. Личный интерес и потенциальные перспективы: Использование различных инструментов и создание продуктов, сочетающих в себе разные технологии (например, веб-приложение в сочетании с управлением базой данных, а также авторизации и аутентификации) позволяет развить и отработать опыт создания интересных и разносторонних проектов. Кроме того, такие навыки востребованы на рынке труда, что открывает перспективы для карьерного роста и развития.

Таким образом, выбор темы " Сравнение производительности и удобства использования различных ORM (Object-Relational Mapping) библиотек: Django ORM, SQLAlchemy и Tortoise ORM." обусловлен ее актуальностью, потребностями рынка, практической значимостью и личным интересом, что делает эту тему подходящей для проведения дипломной работы.

**Определение цели и задач исследования:**

Цель исследования: разработать приложения решающие одинаковую задачу, выполнение базовых операций online-магазина, но с использованием разных ORM.

Задачи исследования:

1. Обзор фреймворков для написания веб приложений и ORM: Провести анализ существующих инструментов, выбрать наиболее подходящие под имеющуюся задачу.
2. Разработка сценариев сравнения: Разработать алгоритм сравнения безопасности и удобства использования, определить критерии сравнения и набрать минимальный набор данных для первичной проверки.
3. Создание платформы и проведение тестирования: реализовать платформу в формате веб-приложения, на можно будет оценить эффективность различных моделей, и провести их тестирование.
4. Написание дипломной работы: Составить дипломную работу, включающую в себя введение, обзор литературы, методологию и результаты исследования, анализ результатов, выводы и рекомендации.

Цели и задачи исследования направлены на получение практических результатов, которые позволят сформировать рейтинг ORM.

1. **Основные понятия и определения**

**Обзор основных понятий в области сравнения ORM:**

* 1. **Фреймворк (Framework)**: Программная платформа, которая предоставляет готовые компоненты и инструменты для разработки приложений. В контексте веб-приложений часто используются Django, FastAPI или Flask.
  2. **Веб-приложение (Web Application)**: Программное приложение, которое работает на веб-сервере и доступно через браузер. В контексте данного проекта, веб-приложение будет использоваться для анализа безопасности и удобства использования.
  3. **Объектно-реляционное отображение (ORM)** является ключевой концепцией в области **Систем управления базами данных (СУБД)** мост между подходом объектно-ориентированного программирования и реляционными базами данных. ORM имеет решающее значение для упрощения взаимодействия с данными, оптимизации кода и плавного объединения приложений и баз данных. Цель этой статьи - объяснить ORM, охватить его основные принципы, преимущества и важность в современной разработке программного обеспечения.

**4 Методы и подходы к разработке**

**Выбор и интеграция ORM с Веб - интерфейсом**

Выбор модели: Начнем с анализа задач, которые наше веб-приложение должно решать, и выберем подходящую ORM реализации небольшого проекта . Для этой цели подходят: Django ORM, SQLAlchemy и Tortoise ORM.

Интеграция ORM: После выбора библиотеки для взаимодействия с базой данных, необходимо интегрировать её в веб-приложение. Это может включать в себя: подключение базы данных к ORM, создание базы данных, обеспечение связи между ORM и интерфейсом пользователя

**Архитектура веб-приложения**

Frontend и Backend: фронтенд (интерфейс пользователя) и бэкенд (серверная логика). Фронтенд может быть разработан с использованием HTML, CSS и JavaScript, а бэкенд может быть реализован на Python с использованием фреймворков, таких как FastAPI или Django.

Для связи между фронтендом и бэкендом можно использовать REST API. Это позволит фронтенду отправлять изображения или другие данные на сервер для обработки моделью и получать результаты.

**Обеспечение безопасности**

Аутентификация и авторизация: Одной из важных задач при работе с базами данных – это разграничение пользователей по их действиям с данными. В дипломной работе представлены разграничения ролей пользователей. В дипломной работе пользователи разделены на три группы: покупатели, сотрудники и администратор.

**Обзор популярных инструментов для разработки веб-приложений на Python**

Основные – Django, Flask, FastAPI, ниже более подробное описание.

**Django**

Django —фреймворк для веб-разработки на Python.

**Основные возможности:**

* Встроенная страница администрирования для управления данными.
* ORM (Object-Relational Mapping) для работы с базами данных.
* Поддержка аутентификации пользователей и управления доступом.
* Мощная система маршрутизации URL.
* Шаблонизатор для создания HTML-шаблонов.

**Особенности:**

Идеально подходит для больших и сложных проектов благодаря своей полной экосистеме.

**Flask**

Это фреймворк для веб-разработки на Python. Обеспечивает простоту и гибкость.

**Основные возможности:**

* Минималистичный подход: только основные компоненты без лишних зависимостей.
* Высокая гибкость для добавления нужных библиотек и модулей.
* Jinja2 для шаблонов и Werkzeug для маршрутизации и обработки запросов.

**Особенности:**

Подходит для небольших и средних проектов, где требуется большая гибкость.

**FastAPI**

Это современный, высокопроизводительный фреймворк для создания API с Python 3.7+ на основе стандартов OpenAPI и JSON Schema.

**Основные возможности**:

* Высокая производительность благодаря использованию асинхронного программирования (async/await).
* Автоматическая генерация интерактивной документации API.
* Поддержка валидирования и аннотаций типов данных.
* Интеграция с различными библиотеками и инструментами, такими как Pydantic и SQLAlchemy.

**Особенности:**

Часто используется для создания API и микросервисов.

**5. Проектирование приложения**

**Планирование и анализ требований**

Выбор фреймворка и инструментов: Определение наиболее подходящего фреймворка (например, Django, Flask или FastAPI) и инструментов для разработки.

Определение структуры приложения: Разработка схемы архитектуры, включающей фронтенд, бэкенд и базу данных.

**Разработка приложения**

Реализация основного функционала: Создание базовой версии приложения с минимально необходимым функционалом для тестирования и демонстрации.

Разработка ключевого функционала

Развитие функционала: Добавление взаимосвязей между таблицами.

Создание API для взаимодействия с фронтендом: Разработка REST API для обработки запросов от пользовательского интерфейса.

Реализация пользовательского интерфейса: Создание форм и страниц для выполнения операций с товаром, управления пользователями.

**Основные требования**

Авторизация и аутентификация пользователей: Пользователь должен выполнять только разрешённые для него действия.

Выполнение операций с товаром: Добавление нового, изменение текущего состояния, выбор и удаление.

**Технические требования**

Фронтенд: использовать HTML, CSS, JavaScript, возможно Bootstrap

Бэкенд: Python с использованием фреймворков Django, FastAPI.

База данных: Использование базы данных (например,SQLLite) для хранения данных о пользователях, товаре и покупках.

API: Разработка API для взаимодействия между фронтендом и бэкендом.

Интеграция программной логики с базой данных с помощью ORM: Подключение ORM к базе данных и к пользовательскому интерфейсу

**6. Разработка в соответствии с созданной документацией**

**Планирование разработки**

Разработка была разделена на несколько основных этапов: проектирование интерфейса, реализация серверной логики и интеграция интерфейса с базой данных с помощью ORM.

**Разработка**

Фронтенд-разработка:

Создан пользовательский интерфейс с использованием Jinja2 для шаблонов и Bootstrap для стилизации.

Разработаны формы и страницы: для выполнения операций с товаром, управление категориями товара, управление магазинами и для работы с пользователями.

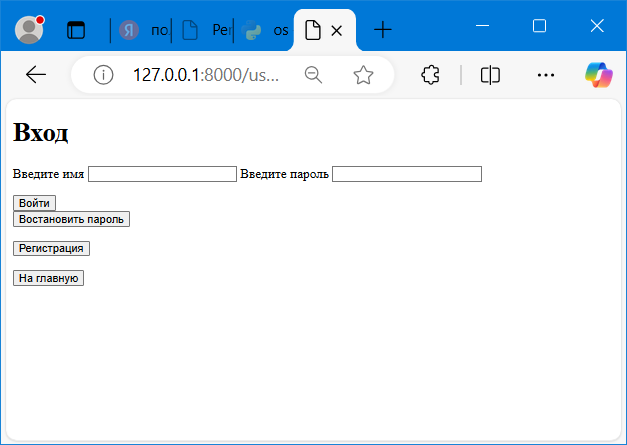


Рис. 1 Страница авторизации

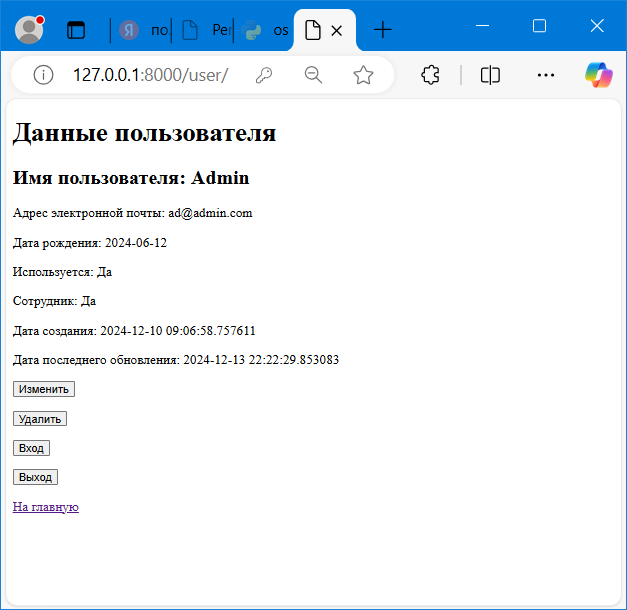


Рис. 2 Страница пользователя исполняющего роль сотрудника и администратора

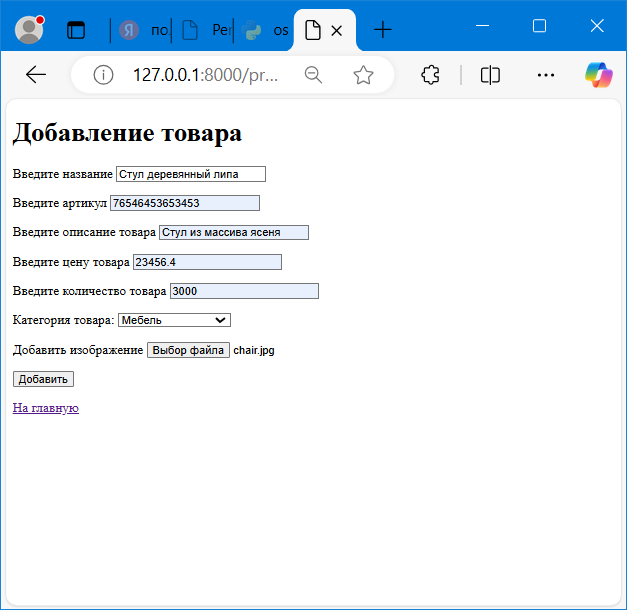


Рис. 3 Добавление нового товара

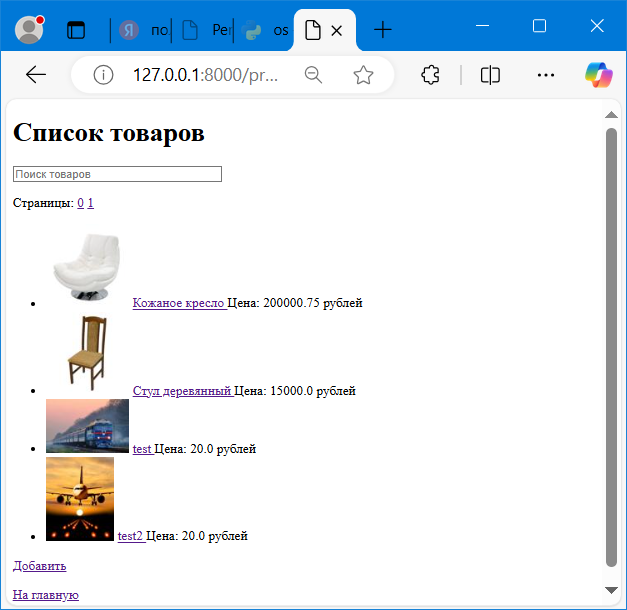


Рис. 4 список товаров

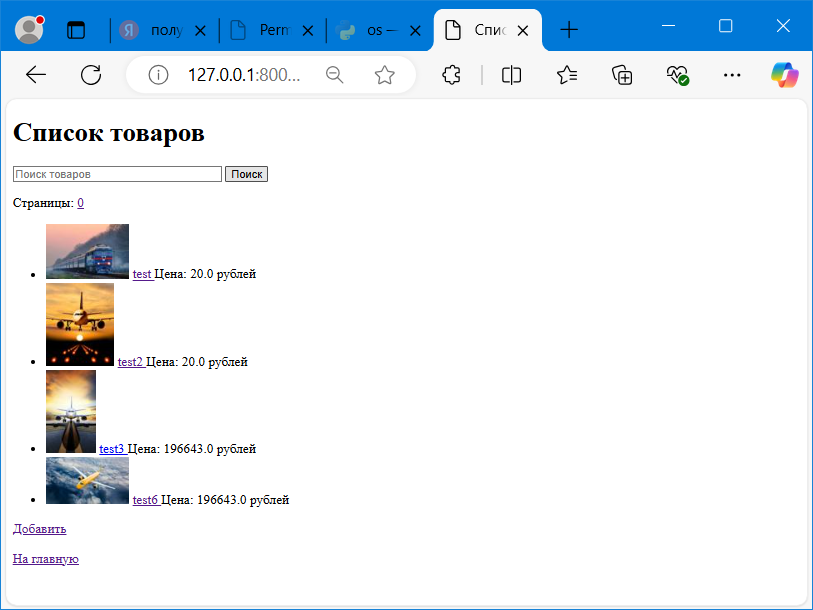


Рис. 5 Результат поиска по строке «test»

**Бэкенд-разработка:**

Реализована серверная логика с использованием библиотеки tortoise-ORM и Фреймворка FastAPI.

Настроены маршруты для обработки действий пользователей и отображение результата на фронтенд.

Интеграция ORM:

Зарегистрирована и настроена tortoise-ORM для взаимодействия между Python и SQLLite, для выполнения операций с базой данных. Результаты взаимодействия выводятся на экран.

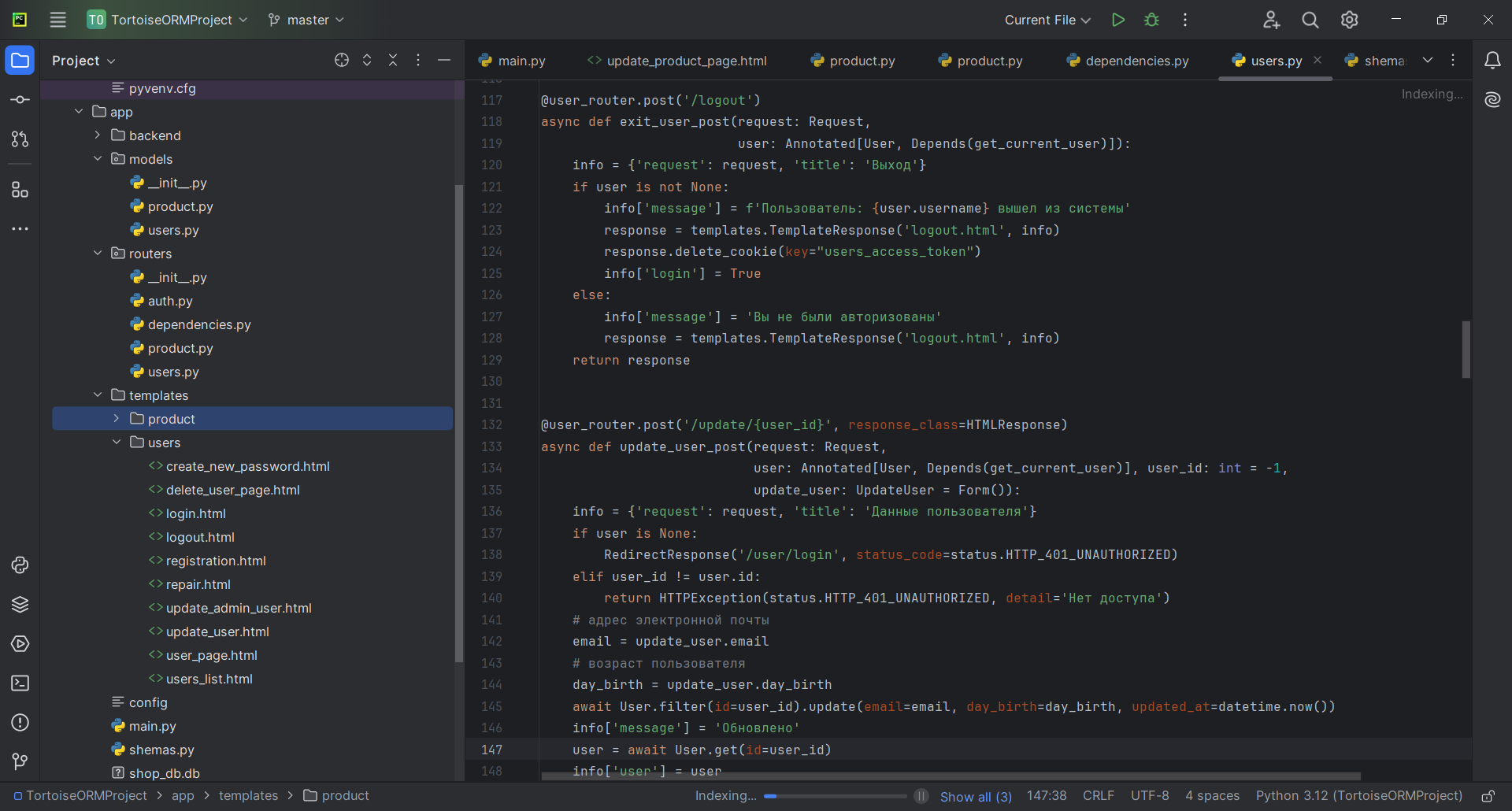


Рис. 6 Файловая структура проекта

**7. Анализ и интерпретация результатов**

**Сравнение ORM**

Были протестированы различные варианты ORM на выполнение различных операций над данными.

Основные метрики, такие как простота написания запроса для выполнения заданной операции, и возможность интеграции в приложение были использованы для оценки каждой ORM.

Интеграция ORM с фреймворками FastAPI и Django:

Django ORM – является частью фреймворка Django и за счёт этого получаем сразу взаимодействие с базой данных и инструмент для работы с фрондэндом.

SQLAlhemi – сопрягается и с Django и с FastAPI за счёт того, что все внутри библиотеки присутствует всё необходимые инструменты в том числе и подключение к базе данных, то есть контроль за подключением перекладывается на разработка приложения.

Tortoise ORM – интегрируется с FastAPI за счёт механизма регистрации в приложении, что избавляет от необходимости контроля подключений к базе данных.

Операции с данными:

При выполнении любых операций в Django необходимо указывать только модель данных и какую операция необходимо выполнить.

При выполнении любых операций в SQLAlhemi с данными необходимо явно указывать на подключение к базе данных. При выполнении операций: создания, изменения и удаления необходимо фиксировать данные изменения.

При выполнении любых операций в Tortoise ORM необходимо указывать модель данных, какую операция необходимо выполнить, а также указание дождаться выполнения операции.

**Интерпретация результатов**

SQLAlchemi показала свою независимость от других фреймворков, но написание запросов сложнее и есть необходимость контроля за подключением

Tortoise ORM – требовательность к окружающим фреймворкам, относительная простота написания запросов и нет необходимости контроля за подключением.

Django orm – простота запросов и нет необходимости контроля за подключением

**Рекомендации по выбору ORM в зависимости от проекта**

Для монолитных проектов лучше использовать Django. Так как вы одновременно получаете ORM и бэкхэнд.

Если необходимо встраивать ORM и базу данных, то предпочтительней SQLAlchemi.

**8. Заключение**

**Обзор выполненной работы**

Проектирование и разработка веб-приложения для online- магазина были успешно завершены в соответствии с изначально созданной документацией. Приложение включает функционал авторизация и аутентификация пользователей, операции над товаром (создание, изменение, покупка, удаление).

Реализованное приложение соответствует требованиям и демонстрирует возможности ORM.

Tortoise-orm показала простоту разработки и требования для интеграции с другими фреймворками.

SQLAlchimi – небольшую сложность с запросами, необходимость контроля за подключением и хорошую интеграцию с другими фреймворками.

Django-ORM - показала простоту разработки/

**Дальнейшие планы**

Будет полезным создание сводной статистики по всем ORM, со всеми метриками эффективности, чтобы разработчики смогли на практике оценить плюсы и минусы каждого в отдельности.