МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Разработка серверных частей интернет-ресурсов
по профилю: Разработка и дизайн компьютерных игр и мультимедийных приложений
направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть веб-приложения “Новостной агрегатор”

Студент: Рассадин Глеб Андреевич
Группа: ИКБО-33-22
Работа представлена к защите 10.12.2024 (дата)_____/Рассадин Г.А./
(подпись и ф.и.о. студента)

Руководитель: Беляев Павел Вячеславович, доцент

Работа допущена к защите_____(дата)_____/_____/
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: _____
_____ / _____ /
_____ / _____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА. 2024 г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Разработка серверных частей интернет-ресурсов
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Рассадин Глеб Андреевич
Группа: ИКБО-33-22
Срок представления к защите: 10.12.2024
Руководитель: Беляев Павел Вячеславович, доцент

Тема: Серверная часть веб-приложения «Новостной агрегатор»

Исходные данные: используемые технологии: HTML5, CSS3, JavaScript, Visual Studio Code, Python, FastApi, PostgreSQL, наличие: межстраничной навигации, внешнего вида страниц, соответствующего современным стандартам веб-разработки, использование паттерна проектирования MVC. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

Перечень вопросов, подлежащих разработке, и обязательного графического материала: 1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Провести тестирование серверной части веб-приложения 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____/Р.Г. Болбаков/, «7» ноября 2024 г.

Задание на КР выдал: _____/П.В. Беляев/, «7» ноября 2024 г.

Задание на КР получил: _____/Г.А. Рассадин/, «7» ноября 2024 г.

УДК 004.4
Руководитель курсовой работы: доцент, Беляев П.В.

РЕФЕРАТ

Отчёт 31 с., 22 рис., 1 табл., 20 источн.

Целью работы является создание серверной части веб-приложения, обеспечивающей функционал новостного агрегатора. В рамках работы проведен анализ предметной области и обоснован выбор технологий (HTML5, CSS3, JavaScript, Python, FastAPI, PostgreSQL), разработана архитектура приложения на основе паттерна проектирования MVT, реализована серверная логика с использованием выбранных технологий, создана база данных с необходимыми связями для функционирования приложения, а также настроена межстраничная навигация и внешний вид страниц в соответствии с современными стандартами веб-разработки.

REPORT

Report: 31 pages, 22 figures, 1 table, 20 sources.
Keywords: server-side logic, web application, MVT pattern, FastAPI, server technologies.

The aim of the work is to create the server-side part of the web application to provide functionality for a news aggregator. The work includes domain analysis and justification for technology choices (HTML5, CSS3, JavaScript, Python, FastAPI, PostgreSQL), the development of the application architecture based on the MVT design pattern, the implementation of server-side logic using the selected technologies, the creation of a database with necessary relationships for the application to function, and the setup of interpage navigation and page appearance according to modern web development standards.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе доступ к актуальным новостям становится все более востребованным. Пользователи стремятся получать своевременные и разнообразные сведения из различных источников, что обуславливает необходимость создания эффективных инструментов для агрегирования информации. Одним из таких инструментов является новостной агрегатор, который собирает, систематизирует и предоставляет пользователям новости из множества источников в одном месте.

Целью данной курсовой работы является разработка серверной части веб-приложения новостного агрегатора, обеспечивающей функционал сбора, обработки и предоставления новостных данных. В рамках работы планируется реализовать механизмы интеграции новостей, хранение данных в базе данных, обеспечение безопасности и надежности сервера, а также создание интерфейсов для взаимодействия с клиентской частью приложения.

Таким образом, данная работа направлена на создание функционального и масштабируемого серверного решения для новостного агрегатора, что позволит пользователям получать актуальные и разнообразные новости в удобном формате. Реализация поставленных задач способствует углублению знаний в области веб-разработки, серверных технологий, что имеет важное практическое значение в условиях стремительного развития информационных технологий.

Курсовая работа состоит из введения, основной части, заключение и приложения. Основная часть содержит анализ предметной области, выбор технологий, проектирование, создание базы данных и серверной логики. В заключении подводятся итоги проделанной работы, а в приложениях представлены дополнительные материалы, включая схемы, таблицы и фрагменты исходного кода.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В этой главе проводится исследование предметной области разрабатываемого программного продукта — новостного агрегатора. Рассматриваются текущие решения, представленные на российском рынке, их функционал, преимущества и недостатки. На основании проведенного анализа определяются требования к создаваемому приложению и выделяются его уникальные особенности.

1.1 Обзор существующих решений

Для проведения анализа были выбраны три популярных российских новостных портала: Лента.ру (https://lenta.ru/), РИА Новости (https://ria.ru/) и Газета.ру (https://www.gazeta.ru/). Ниже представлен их краткий обзор.

Лента.ру

Лента.ру — одна из крупнейших российских новостных платформ. Портал освещает широкий спектр тем: от политики и экономики до спорта и культуры. Пользователи могут сортировать новости по категориям, просматривать самые популярные материалы и получать доступ к архивам. Особенностью является возможность подписки на обновления через электронную почту и пуш-уведомления.

РИА

РИА Новости — федеральный портал, предоставляющий не только текстовые новости, но и мультимедийный контент, включая фото, видео и инфографику. Платформа имеет мощную систему поиска и возможность детализированного фильтра контента. Интеграция с социальными сетями позволяет пользователям делиться материалами напрямую.

Газета.ру

Газета.ру — информационно-аналитический портал, специализирующийся на новостях, обзорах и мнениях экспертов. Удобный пользовательский интерфейс обеспечивает быстрый доступ к последним новостям и аналитическим статьям. Сайт предлагает уникальную возможность

читать материалы в тематических подборках, которые обновляются ежедневно.

Таблица 1 – Сравнение интернет-ресурсов в предметной области

Свойство	**Лента.ру**	**Риа Новости**	**Газета.ру**	**Разрабатываемое приложение**
Категории новостей	+	+	+	+
Фильтрация по темам	+	+	-	+
Дата публикации	+	+	+	+
Ролевая система	-	+	-	+
Курсы валют	-	-	-	+

1.2 Основной функционал готовых решений

Анализ существующих платформ выявил следующий базовый функционал:

- Отображение новостной ленты с сортировкой по категориям.
- Возможность фильтрации контента по интересам.
- Доступ к актуальным курсам валют.
- Администрирование.

1.3 Требования к программному обеспечению

На основе анализа выделены требования, которые планируется внедрить в разрабатываемом приложении:

- Роль администраторов: только администраторы смогут добавлять, редактировать и удалять новости.
- Удобство пользования: простой и понятный интерфейс.
- Персонализация контента: пользователи смогут выбирать интересующие их категории и получать новости в соответствующем формате.

- Актуальность контента: пользователи получают свежие новости из популярных источников.
- Безопасность: обеспечение безопасности доступа администратора для защиты контента.
- Доступность: система доступна, минимальное кол-во ошибок.

1.4 Результаты анализа

Проведенный анализ показал, что разрабатываемый новостной агрегатор будет сочетать основные функциональные возможности существующих решений, при этом предлагая уникальные функции, такие как персонализация новостей и управление ролями. Эти особенности позволят приложению выделиться на фоне конкурентов и удовлетворить требования различных категорий пользователей.

2 ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ

2.1 Обзор и анализ технологий

В данной главе проводится исследование технологий и архитектурных паттернов, которые могут быть использованы для разработки системы новостного агрегатора. Выбор инструментов и подходов основывается на их функциональных возможностях, совместимости и эффективности для достижения поставленных целей.

Ключевые компоненты системы

– Для реализации системы были выделены основные компоненты, которые требуется разработать:

– Фронтенд: интуитивно понятный пользовательский интерфейс.

– Бэкенд: серверная часть для обработки запросов и отправки ответов.

– База данных: надежное хранилище новостей.

– Архитектура: логическая организация взаимодействия всех компонентов.

1. Фронтенд

– HTML и CSS: основные технологии для создания структуры и оформления пользовательского интерфейса. Эти инструменты широко используются, легко осваиваются и обеспечивают кроссбраузерную совместимость.

– JavaScript: добавляет интерактивность и динамическую функциональность в интерфейс.

– Bootstrap: CSS-фреймворк, упрощающий создание адаптивных и эстетически приятных интерфейсов, особенно для мобильных устройств.

2. Бэкенд

– Django (Python): мощный фреймворк, подходящий для масштабируемых и безопасных приложений. Однако его избыточность может усложнить процесс разработки небольших проектов.

– FastAPI (Python): современный фреймворк для создания высокопроизводительных API. FastAPI поддерживает асинхронную обработку запросов, автоматическую генерацию документации и строгую типизацию данных. Это делает его удобным для быстрой разработки и исключает большое количество ошибок.

– Flask (Python): минималистичный фреймворк, предоставляющий высокую гибкость при настройке, но требующий больше усилий для реализации базовых функций, таких как аутентификация или маршрутизация.

3. База данных

– PostgreSQL: мощная реляционная СУБД, поддерживающая сложные запросы, транзакции и обработку больших объемов данных. PostgreSQL также предоставляет средства для масштабирования и высокой надежности.

– MySQL: легкая в освоении база данных, эффективная для структурированных данных, но несколько уступающая PostgreSQL в плане гибкости и возможностей.

– Redis: in-memory база данных, идеально подходящая для кэширования и быстрого доступа к временным данным.

4. Архитектурные подходы

– MVC (Model-View-Controller): шаблон проектирования, разделяющий логику приложения, пользовательский интерфейс и управление данными. Это делает приложение более структурированным и легким в сопровождении.

– MVT (Model-View-Template): схож с MVC, но включает серверные шаблоны для генерации пользовательского интерфейса. Этот подход часто используется в Django.

– DDD (Domain-Driven Design): подход, ориентированный на моделирование бизнес-логики, разделяющий проект на домены, что упрощает реализацию сложных систем.

Вывод

Анализ технологий показал, что использование FastAPI для бэкенда и PostgreSQL для работы с данными является наиболее подходящим решением для данной системы. Асинхронные возможности FastAPI обеспечат высокую производительность, а PostgreSQL предоставит надежное и гибкое хранилище данных.

HTML, CSS и JavaScript в сочетании с Bootstrap упростят разработку удобного интерфейса. Выбор архитектуры MVC обеспечит разделение логики приложения, что упростит разработку и поддержку системы. Такой набор технологий полностью соответствует требованиям, предъявляемым к системе.

2.2 Обоснование и выбор технологий

Для разработки системы новостного агрегатора для пользователей были выбраны следующие технологии:

– FastAPI (Python): данный фреймворк обеспечивает удобство и скорость разработки API благодаря асинхронной обработке запросов. FastAPI также автоматически генерирует документацию, что ускоряет процесс интеграции и тестирования.

– PostgreSQL: реляционная база данных, которая гарантирует надежное хранение новостей. PostgreSQL отличается высокой производительностью и поддержкой сложных запросов, что делает ее подходящей для аналитики и обработки больших объемов данных.

– HTML, CSS: используются для создания пользовательского интерфейса, предоставляющего интуитивно понятный и визуально приятный опыт взаимодействия с системой.

– Uvicorn: высокопроизводительный ASGI-сервер, применяемый для запуска FastAPI. Он поддерживает асинхронные запросы, что позволяет обеспечить устойчивую работу сервера даже при высоких нагрузках.

– MVC (Model-View-Controller): архитектурный паттерн, разделяющий логику приложения, представление и управление данными. Это повышает читаемость кода, упрощает поддержку и облегчает расширение функционала системы.

Для запуска проекта используется команда запуска сервера через Uvicorn. Такой подход позволяет быстро развернуть приложение в локальной или серверной среде.

2.3 Вывод

В данной главе выполнен выбор технологий и архитектурного подхода для реализации системы новостного агрегатора. Решение использовать стек из FastAPI, PostgreSQL, HTML, CSS и Uvicorn обусловлено необходимостью обеспечения высокой производительности, надежного хранения данных и удобства в использовании.

Асинхронные возможности FastAPI в сочетании с производительностью Uvicorn делают систему устойчивой к нагрузкам, а PostgreSQL позволяет эффективно обрабатывать данные и выполнять сложные запросы. Применение паттерна MVC упрощает поддержку кода и способствует более структурированному подходу к разработке. Такой набор технологий полностью соответствует требованиям, предъявляемым к разработке системы.

3 РАЗРАБОТКА АРХИТЕКИТУРЫ ПРИЛОЖЕНИЙ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА

В данном разделе рассмотрено применение паттерна проектирования MVC для создания архитектуры системы новостного агрегатора. Использование MVC обеспечивает структурированность приложения, разделение ответственности между компонентами и упрощает разработку и поддержку системы.

Основные бизнес-правила и требования формируют структуру приложения. На рисунке 1 представлена диаграмма вариантов использования, иллюстрирующая основные сценарии взаимодействия пользователей с системой новостного агрегатора.

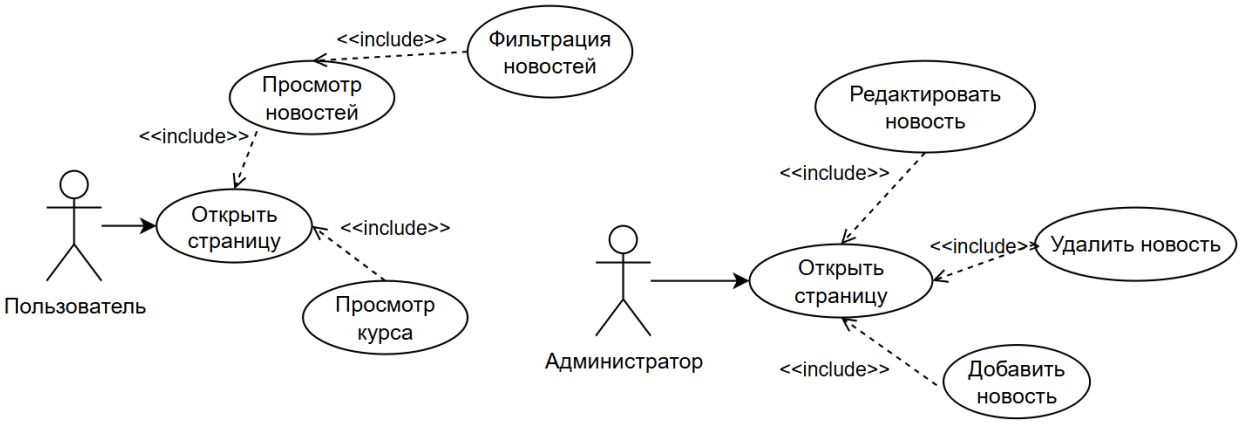

Рисунок 1 - Диаграмма вариантов использования.

3.1 Применение паттерна MVC

Архитектура приложения построена на паттерне MVC, разделяющем систему на три ключевых компонента:

Модель (Model):

Отвечает за управление данными, взаимодействие с базой данных (PostgreSQL) и обработку бизнес-логики. Например, содержит классы для работы с новостями (News) и пользователями (User).

Представление (View):

Обеспечивает отображение данных пользователю через статические файлы HTML, CSS и JavaScript. Представление формирует пользовательский интерфейс для управления новостями и просмотра информации.

Контроллер (Controller):

Обрабатывает запросы пользователей, связывая модель и представление. Контроллеры реализуют такие функции, как создание, редактирование и удаление новостей, а также обработку аутентификации пользователей.

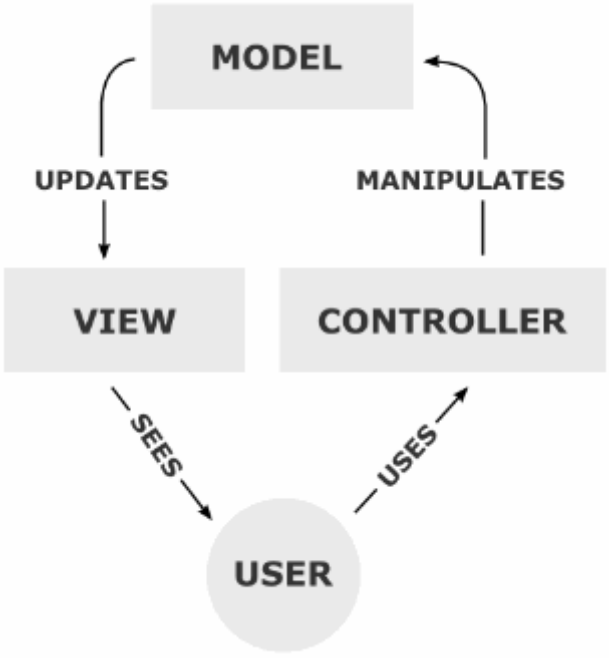

Рисунок 2 - Схема работы MVC-паттерна.

3.2 Архитектура приложения

Основные компоненты системы новостного агрегатора:

Модели: Управляют взаимодействием с базой данных через ORM (SQLAlchemy). Например, классы News и User представляют таблицы базы данных и управляют их состоянием.

Представления: Реализуют интерфейс пользователя через HTML/CSS и обрабатывают ответы от контроллеров.

Контроллеры: Управляют логикой приложения, связывая пользовательский интерфейс и модели. Реализованы с использованием FastAPI.

Пример сценария взаимодействия

Пользователь отправляет запрос через веб-интерфейс, например, чтобы просмотреть список новостей. Контроллер принимает запрос, вызывает соответствующий метод модели для получения данных из базы, после чего результат передается в представление, которое формирует ответ и отправляет его пользователю.

3.3 Диаграммы архитектуры

На рисунках 3 и 4 представлены диаграммы для системы:

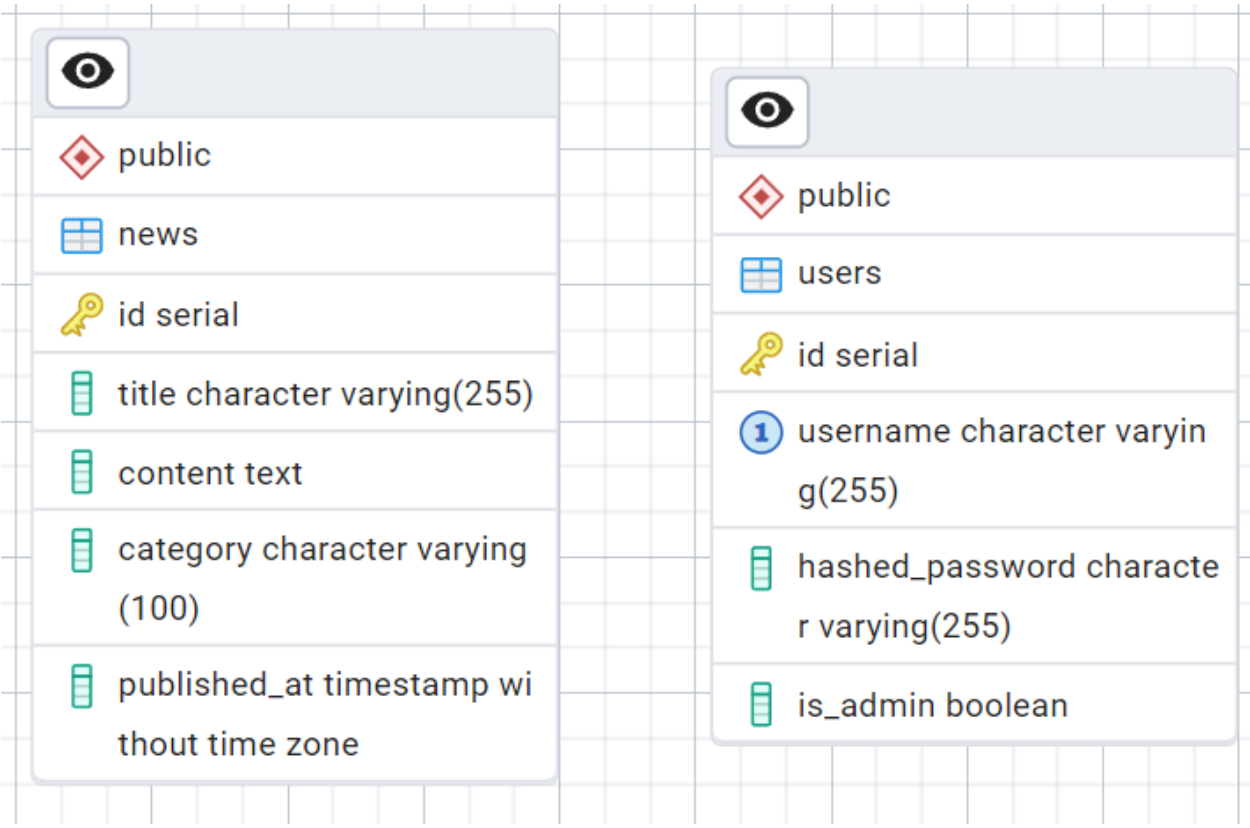

Рисунок 3 – ER-диаграмма базы данных в PostgreSQL.

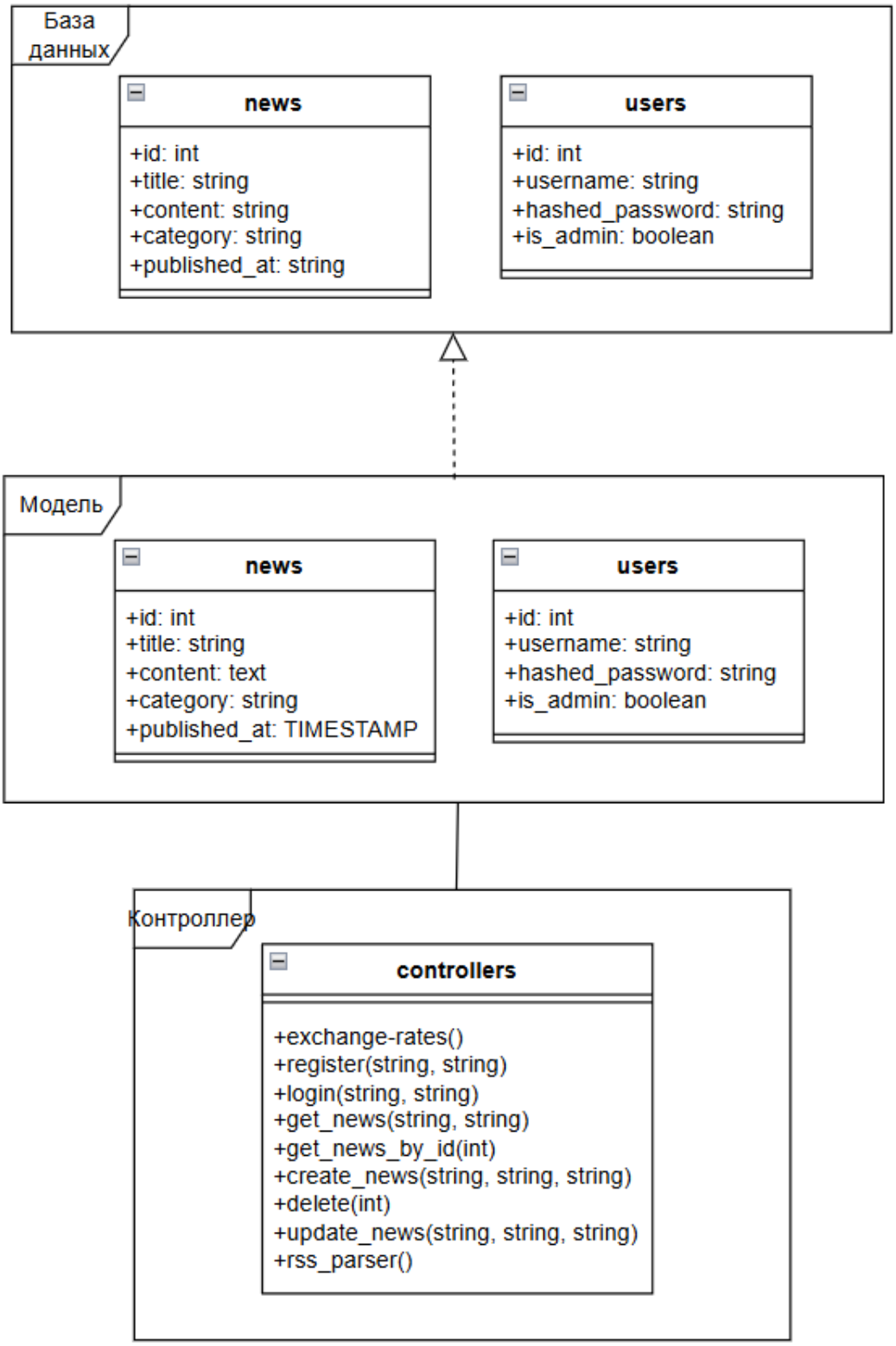

Рисунок 4 - Диаграмма классов.

3.4 Реализация ключевых функций

Добавление новости (только для администраторов):

- Контроллер обрабатывает запрос на добавление новости.
- Модель обновляет таблицу новостей в базе данных.
- Представление возвращает подтверждение успешного добавления новости.

Просмотр новостей:

- Контроллер обрабатывает запрос на получение списка новостей.
- Модель извлекает данные из базы.
- Представление формирует страницу с перечнем новостей.

Изменение новости (только для администраторов):

- Контроллер обрабатывает запрос на редактирование новости.
- Модель обновляет запись в базе данных.
- Представление возвращает обновленные данные.

3.5 Вывод

Архитектура приложения построена с использованием паттерна MVC, который обеспечивает логическое разделение на уровни представления, обработки данных и бизнес-логики. Это позволяет улучшить структурированность кода, упростить разработку, поддержку и масштабирование системы.

4. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА

Разработка серверной части системы новостной агрегатор основана на использовании языка программирования Python и фреймворка FastAPI с применением паттерна MVC. В данной главе подробно рассматриваются ключевые аспекты реализации серверной части: обработка запросов, взаимодействие с базой данных PostgreSQ. Также уделяется внимание тестированию системы и обеспечению её стабильной работы.

4.1 Работа с базой данных

Данные хранятся в реляционной базе данных PostgreSQL, база данных представлена в виде диаграммы на рисунке 3.

```
news_db=# \l
                                                                 List of datab
ases
   Name    |  Owner   | Encoding | Locale Provider |       Collate       |
        Ctype         | Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+---------------------+
---------------------+--------+-----------+-----------------------
 news_db   | postgres | UTF8     | libc            | Russian_Russia.1251 |
 Russian_Russia.1251 |        |           |
 postgres  | postgres | UTF8     | libc            | Russian_Russia.1251 |
 Russian_Russia.1251 |        |           |
 template0 | postgres | UTF8     | libc            | Russian_Russia.1251 |
 Russian_Russia.1251 |        |           | =c/postgres          +
           |          |          |                 |                     |
                      |        |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | Russian_Russia.1251 |
 Russian_Russia.1251 |        |           | =c/postgres          +
           |          |          |                 |                     |
                      |        |           | postgres=CTc/postgres
(4 rows)
```

Рисунок 5 – Созданная база данных.

```python
database.py > ...
from sqlalchemy.ext.asyncio import AsyncSession, create_async_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import text


SQLALCHEMY_DATABASE_URL = "postgresql+asyncpg://postgres:23052004*xX@localhost/news_db"
DATABASE_URL = "postgresql+asyncpg://<user>:<password>@<host>:<port>/<database>"


engine = create_async_engine(SQLALCHEMY_DATABASE_URL, echo=True)

async_session = sessionmaker(
    engine, class_=AsyncSession, expire_on_commit=False
)

async def get_db():
    async with async_session() as session:
        await session.execute(text("SET client_encoding = 'UTF8';"))
        yield session
```

Рисунок 6 – Подключение к базе данных.

4.2 Реализация моделей

С помощью моделей предоставляется доступ к базе данных. Их примеры отображены на рисунке 7.

```python
from sqlalchemy import Column, Integer, String, Text, TIMESTAMP, Boolean
from sqlalchemy.ext.declarative import declarative_base
import datetime

Base = declarative_base()

class News(Base):
    __tablename__ = "news"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, index=True)
    content = Column(Text,nullable=False)
    category = Column(String, nullable=True)
    published_at = Column(TIMESTAMP, default=datetime.datetime.utcnow)

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True, nullable=False)
    hashed_password = Column(String, nullable=False)
    is_admin = Column(Boolean, default=False)
```

Рисунок 7 – Пример модели проекта.

4.3 Реализация контроллеров

За вызов моделей и обработку запросов отвечают контроллеры. Пример контроллера для поиска новостей показан на рисунке 8.

```python
async def get_news(
    title: str | None = None,
    category: str | None = None,
    db: AsyncSession = Depends(get_db)
):
    query = select(News)

    if title:
        query = query.filter(News.title.ilike(f"%{title}%"))
    if category:
        query = query.filter(News.category == category)
    query = query.order_by(News.published_at.desc())

    result = await db.execute(query)
    news_list = result.scalars().all()
    return [convert_news_to_response(news) for news in news_list]

@app.get("/news/{news_id}", response_model=NewsResponse)
async def get_news_by_id(news_id: int, db: AsyncSession = Depends(get_db)):
    result = await db.execute(select(News).filter(News.id == news_id))
    news = result.scalar_one_or_none()
    if news is None:
        raise HTTPException(status_code=404, detail="News not found")
    return convert_news_to_response(news)
```

Рисунок 8 – Пример контроллера.

4.4 Получившееся приложение.

Рассмотрим получившееся приложение. На последующих рисунках показана главная страница с демонстрацией функционала новостного агрегатора. Она имеет новостную ленту меня поиска и авторизации. Авторизация необходима лишь администраторам.

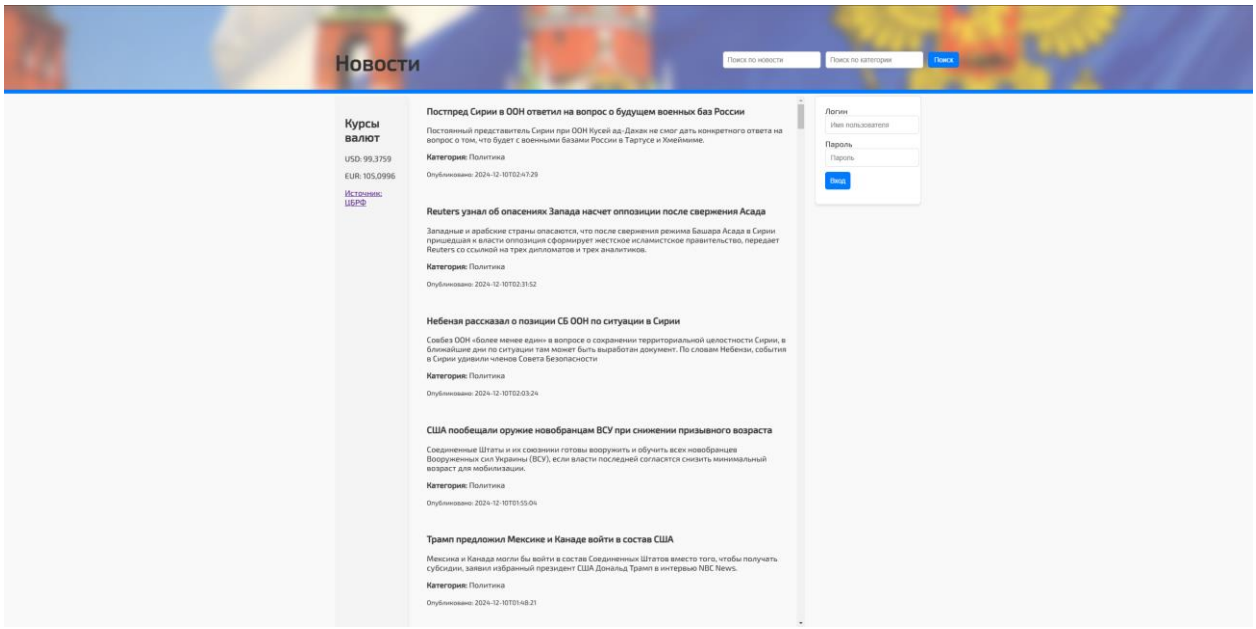

Рисунок 9 – Общий вид страницы.

Рассмотрим тестирование функции поиска на рисунках 10-11. Поиск работает как, по ключевым словам в новости, так и по категориям.

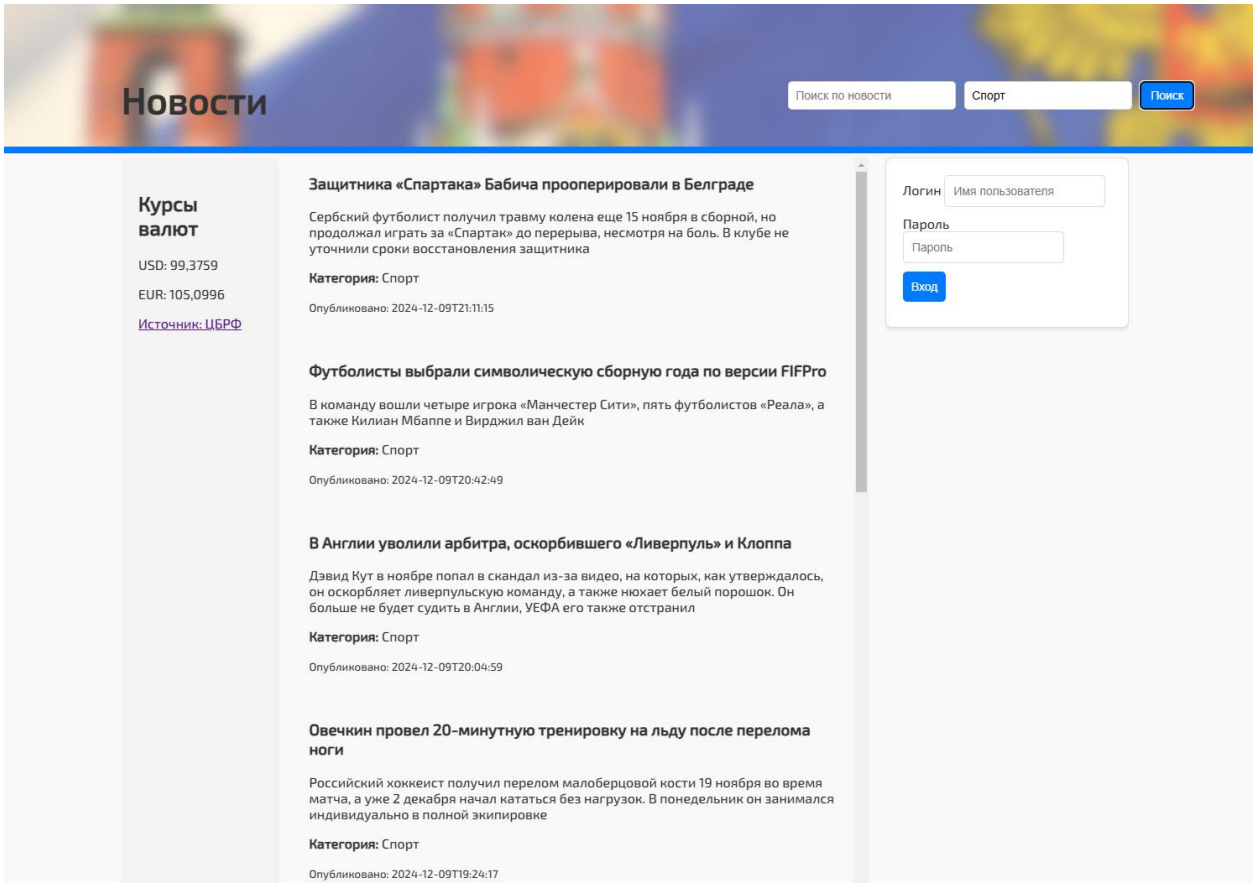

Рисунок 10 – Поиск по категории.

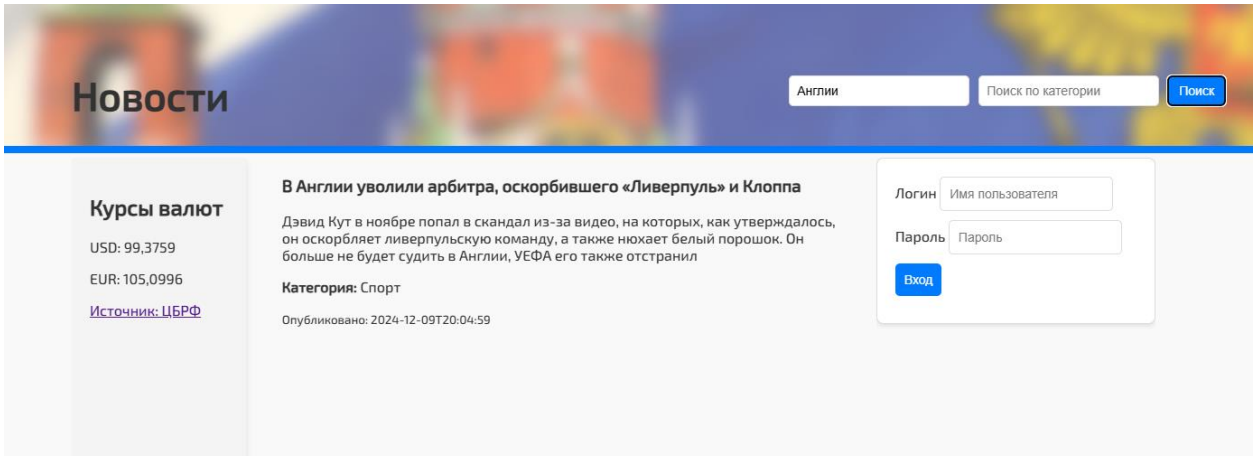

Рисунок 11 – Поиск по новости.

Подробнее изучим функцию авторизации администрации. Администратор имеет права удалять, добавлять и редактировать новости. Тестирование данных функций можно наблюдать на рисунках 13-16. Так же присутствует уведомление об успешной авторизации на рисунке 12.

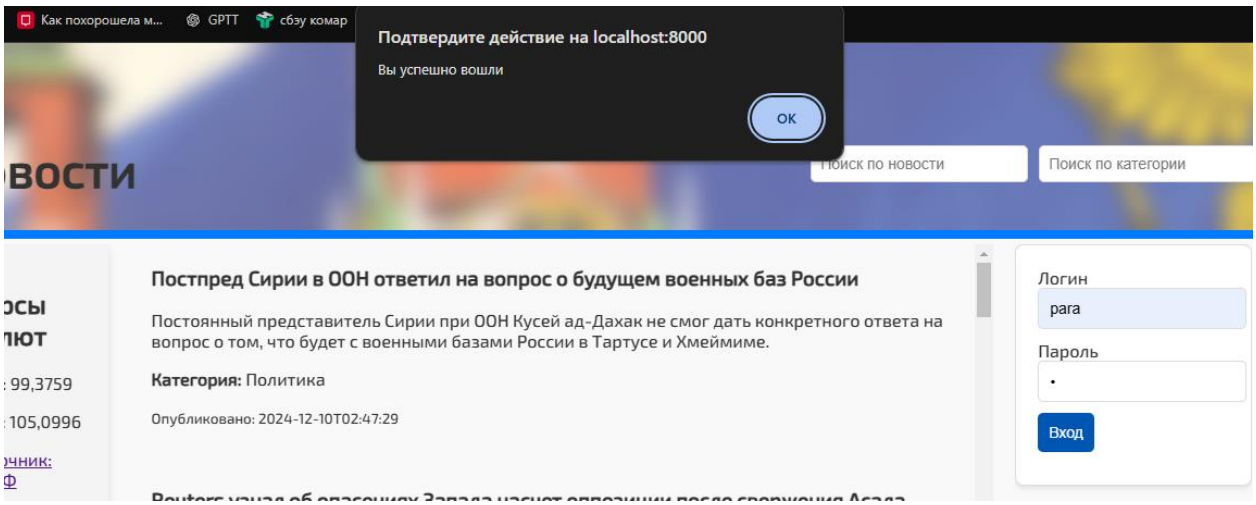

Рисунок 12 – Уведомление об авторизации.

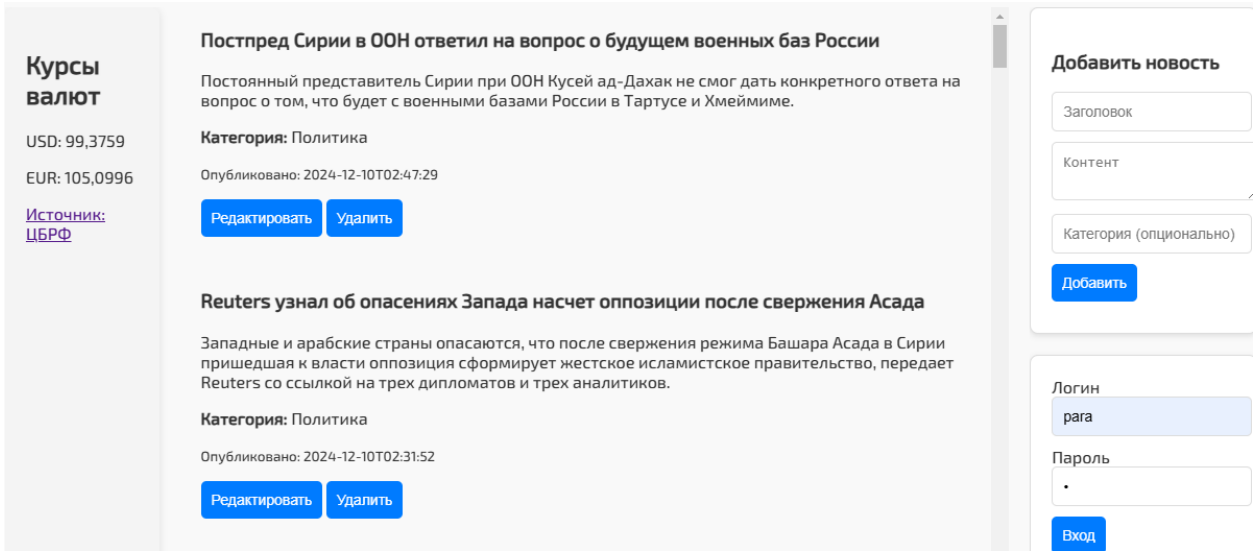

Рисунок 13 – Функции администратора.

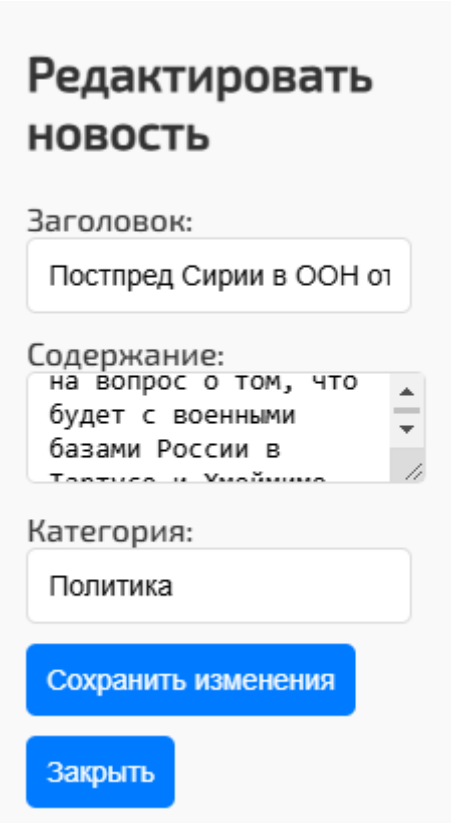

Рисунок 14 – Функция редактирования.

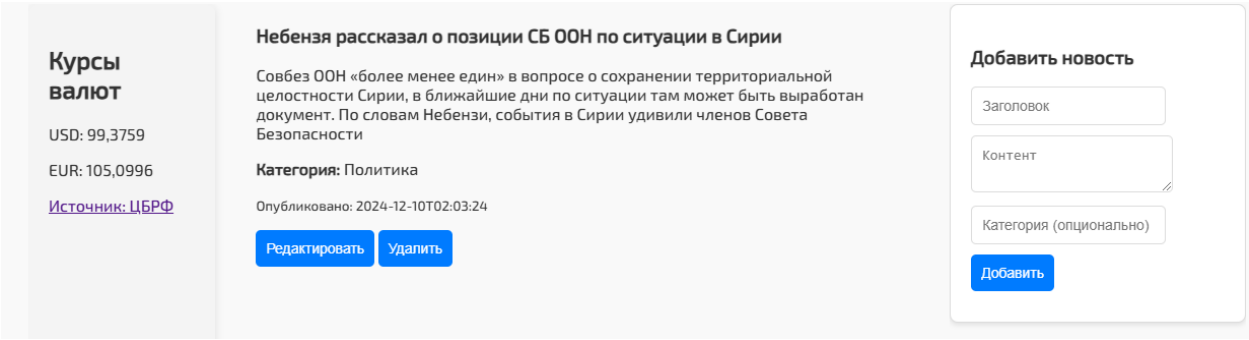

Рисунок 15 – Тест удаления.

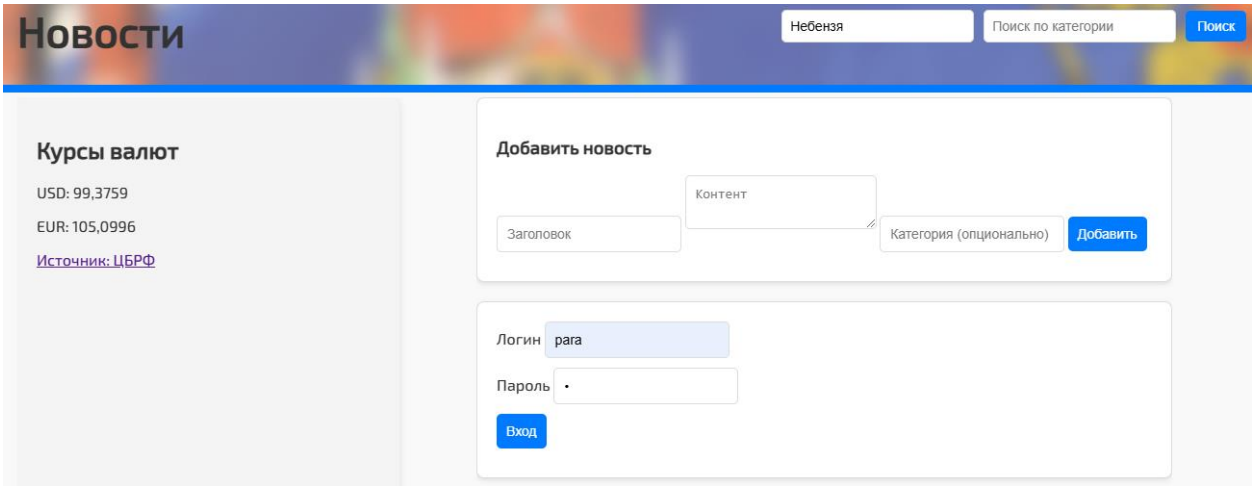

Рисунок 16 – Тест удаления.

Как можно заметить из тестов на рисунках, функции на сайте отрабатывают корректно.

Теперь подробно рассмотрим тестирование запросов через панель FastApi которая доступна по ссылке http://localhost:8000/docs.

Рисунок 17 – Тест регистрации.

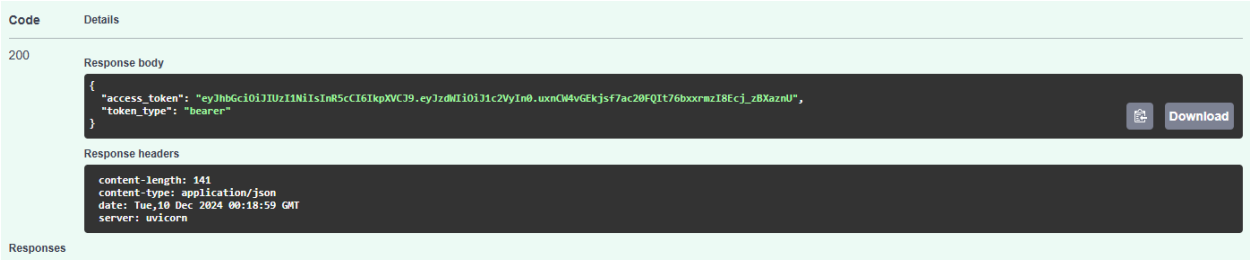

Рисунок 18 – Тест авторизации.

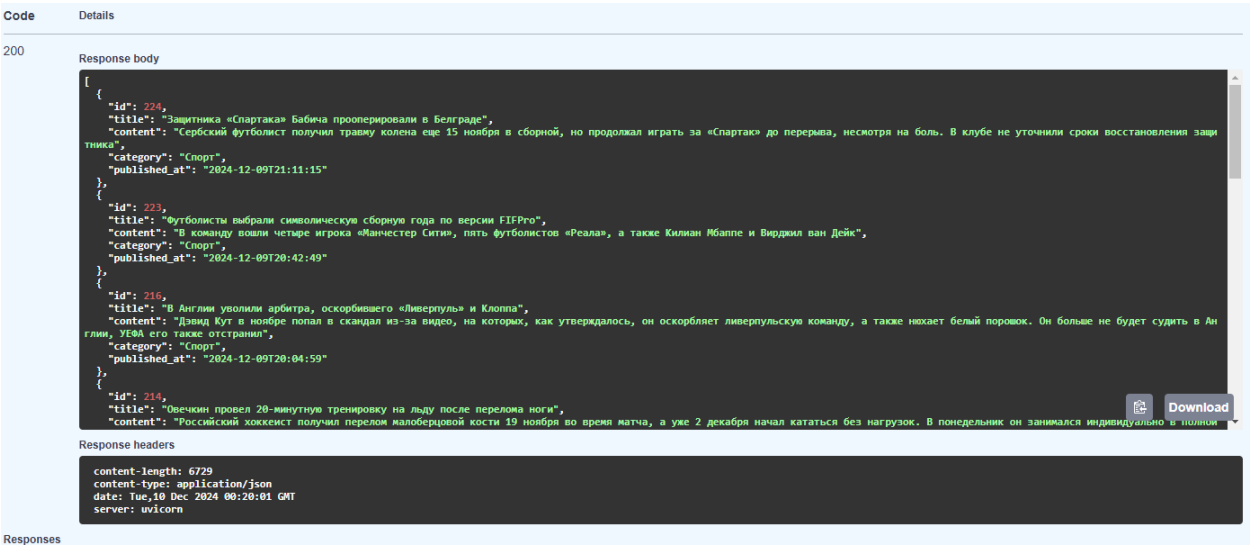

Рисунок 19 – Тест поиска.

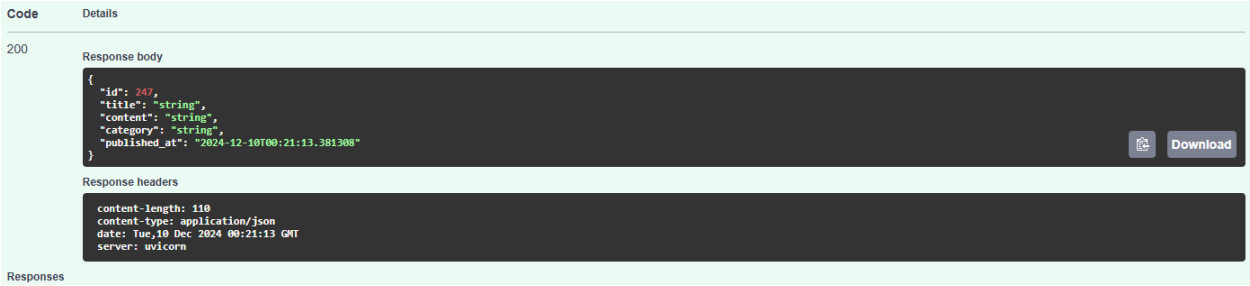

Рисунок 20 – Тест создания.

Рисунок 21 – Тест удаления.

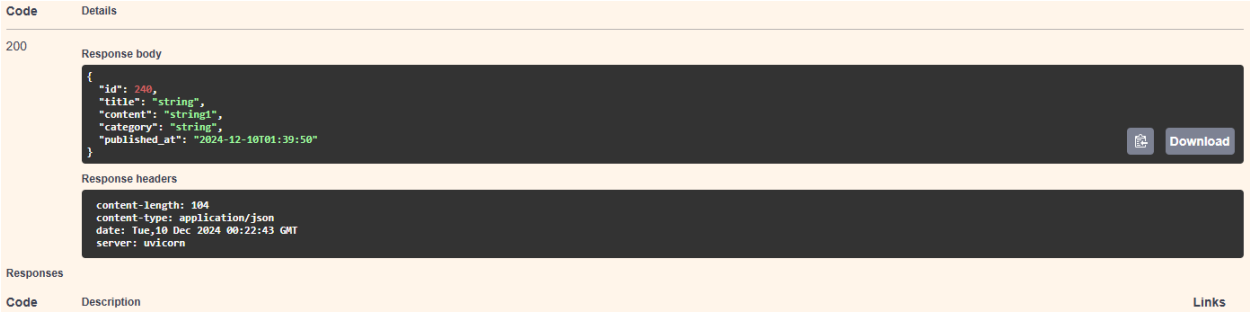

Рисунок 22 – Тест редактирования.

Вывод

Тестирование является неотъемлемой частью разработки и позволяет обеспечить надежность приложения. В данном проекте было использовано несколько видов тестов, что позволило покрыть все основные компоненты приложения: от моделей и представлений до API. Все тесты были успешно пройдены, что подтверждает правильность работы системы. В будущем тесты имеет возможность быть дополненными для проверки новых функциональностей.

ЗАКЛЮЧЕНИЕ

В рамках выполнения курсовой работы успешно достигнуты цели разработки системы новостного агрегатора, соответствующей современным стандартам в сфере информационных ресурсов. Основная задача состояла в создании веб-сервиса, который существенно упрощает поиск и управление новостями благодаря удобной системе фильтрации, добавления, редактирования и удаления данных.

Для реализации поставленных задач был проведен всесторонний анализ предметной области. Изучены существующие программные решения, их функциональные особенности, преимущества и недостатки. Это позволило сформулировать ключевые требования к системе и создать функционал, который выгодно выделяет разработанный сервис среди аналогов.

На основании результатов аналитической работы выбран архитектурный паттерн MVC, обеспечивающий четкое разделение логики приложения, упрощение разработки и возможность масштабирования. В процессе выполнения проекта были освоены и применены современные технологии, такие как PostgreSQL, HTML/CSS, Uvicorn и FastAPI.

Серверная часть системы реализована с использованием FastAPI (Python) и базы данных PostgreSQL, спроектированной с учетом специфических потребностей новостных ресурсов для эффективного хранения и управления данными.

В ходе тестирования подтверждена корректная работа всех модулей системы. Разработанный функционал предоставляет следующие возможности:

- добавление новых записей в базу данных;
- получение актуального контента;
- удаление и редактирование существующих записей.

СПИСОК ЛИТЕРАТУРЫ И ИСПОЛЬЗУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Васильев Н. П., Заяц А. М. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и node.js. Лань — 2021. — 120
2. Хоффман Эндрю X85 Безопасность веб-приложений. — СПб.: Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O’Reilly») (дата обращения: 24.11.2024).
3. Меджуи М., Уайлд Э., Митра Р., Амундсен М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. - СПб.: Питер, 2020. (дата обращения: 24.11.2024).
4. Диков А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 188 с. – Режим доступа: https://e.lanbook.com/book/122174 (дата обращения: 24.11.2024).
5. Документация по архитектурному паттерну MVC. URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (дата обращения: 13.11.2024).
6. Документация FastAPI. URL: https://fastapi.tiangolo.com/ (дата обращения: 02.12.2024).
7. Документация Uvicorn. URL: https://www.uvicorn.org/ (дата обращения: 02.12.2024).
8. Github: https://github.com/paraumir/news_aggregator
9. Habr. " Почему Вы должны попробовать FastAPI?" — URL: https://habr.com/ru/articles/478620/ (дата обращения: 01.12.2024).
10. Habr. " Создание собственного API на Python (FastAPI): Знакомство и первые функции" — URL https://habr.com/ru/companies/amvera/articles/826196/ (дата обращения: 02.12.2024).

11. Real Python. "FastAPI: создание и развертывание REST API" — URL: https://realpython.com/fastapi-python-web-apis/ (дата обращения: 22.11.2024).

12. MDN Web Docs. "HTML: Hypertext Markup Language" — URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата обращения: 02.12.2024).

13. Дакетт Д. Javascript и jQuery. Интерактивная веб-разработка Эксмо — 2017. — 640 с.

14. CSS-Tricks. "A Complete Guide to Grid" — URL: https://css-tricks.com/snippets/css/complete-guide-grid/ (дата обращения: 02.12.2024).

15. Uvicorn Docs. "Uvicorn CLI Options" — URL: https://www.uvicorn.org/#cli-options (дата обращения: 06.12.2024).

16. ГОСТ Р 57193-2016. Национальный стандарт Российской Федерации. Системная и программная инженерия. Процессы жизненного цикла систем (введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 31.10.2016 № 1538-ст). — М.: Стандартинформ, 2016. — 45 с

17. Reasoning on UML class diagrams [Электронный ресурс] URL: https://www.sciencedirect.com/science/article/pii/S0004370205000792

18. A Comparative Analysis of Entity-Relationship Diagrams [Электронный ресурс]URL:https://www.cin.ufpe.br/~in1008/aulas/A%20Comparative%20Analysis%20of%20Entity-Relationship%20Diagrams.pdf

19. ГОСТ 7.32—2017 Система стандартов по информации, библиотечному и издательскому делу ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ Структура и правила оформления — М.:Стандартинформ, 2017. — 32 с.

20. ГОСТ 7.1–2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Общие требования и правила составления. — М.: Стандартинформ, 2003. — 28 с