УДК 004.774.4

Обобщение опыта проведения занятий по созданию API-приложения

Половикова О.Н., Михеева Т.В.
Алтайский государственный университет, г. Барнаул
ponOlgap@gmail.com, miheeva-t@yandex.ru

Аннотация

В данной работе представлен подход организации и проведения занятий по созданию web-приложений в рамках программы дополнительного образования со студентами естественно-научных институтов. Материалы работы будут полезны преподавателям, которые организуют и проводят учебные занятия по дисциплинам связанным с применением СУБД для построения готовых прикладных решений, в частности web-приложений.

Ключевые слова: Backend-разработка, API-приложение, база данных, контейнеризация, асинхронный доступ, переменные окружения, дополнительное образование, учебный курс, практические занятия

На базе цифровой кафедры в Алтайском государственном университете организованы курсы подготовки студентов по программам, связанным с информационными технологиями. В которых студенты могут повысить свою ИТ-квалификацию параллельно с освоением основной образовательной программы. Большинство кусов фокусируется на получение слушателями именно практического опыта решения задач в их профессиональной сфере и навыков, востребованных у будущих работодателей. Студенты сами выбирают направление подготовки, в зависимости от своих интересов, знаний и навыков. Используются как дистанционные, так и очные формы проведения занятий.

Востребованными направлениями рамках дополнительного образования для студентов естественно-научных институтов являются курсы, связанные с непосредственной прикладной разработкой готовых решений с использованием баз данных. Курс “Backend-разработка на языке python” в предполагает полный цикл теоретических и практических занятий для создания готового к использованию API-приложения на базе технологий FastAPI [1] + PostgreSQL [2] + SQLAlchemy [3].

Совместное использование PostgreSQL и FastAPI позволяет организовать ассинхронный доступ к данным, что обеспечивает более эффективное использование ресурсов по сравнению с синхронным подходом и является актуальным для повышения “отзывчивости” многопользовательских API-приложения. PostgreSQL+SQLAlchemy реализуют два основных подхода взаимодействия с реляционной СУБД:

- с использованием SQL-запросов и низкоуровневого интерфейса для подключения к реляционной СУБД;
- на основе технологии объектно-реляционного отображения (ORM), которая позволяет синхронизировать объекты с записями в реляционных базах данных.

Оба подхода рассматриваются в рамках данного курса и закрепляются конкретными практическими заданиями.

Программа курса включает три основных блока:

1. Основы баз данных.

2. Backend-разработка с использованием FastAPI+PostgreSQL+SQLAlchemy.

3. Развёртывание приложений.

Первый блок является “выравнивающим”: студенты обучаются в нескольких институтах по разным образовательным программам. Так как качество изучения материалов по основам работы с базой данных во многом определяет перспективность освоения всего курса, необходимо в этом блоке предусмотреть:

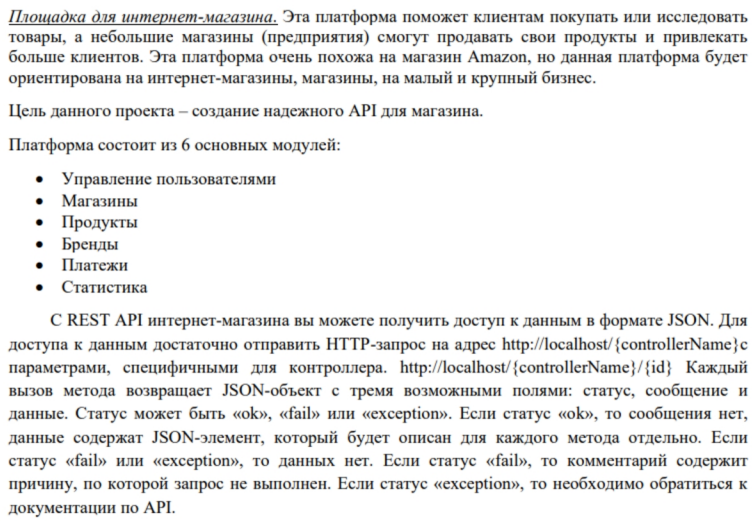
1) Несколько видов индивидуальных заданий.

2) Практическую работу с преподавателем в офлайн режиме малыми группами, желательно, с дифференциацией по начальному уровню подготовки.

3) Самостоятельную работу над несложными кейсами, но с необходимостью оценить ресурсозатратность выполнения запросов к базе данных (например, по времени). Такие задания необходимы для понимания необходимости детальной проработки всех индексов, связей и зависимостей при проектировании БД. Понимание временной сложности позволяет анализировать планы выполнения запроса для выявления узких мест и для написания более эффективного кода.

Во втором блоке уделяется внимание вопросам использования фреймворков и библиотек для профессиональной разработки backend-части web-приложения. В частности, на практике реализуются кейсы:

- Работа с реляционной СУБД PostgreSQL (два способа взаимодействия),
- Настройка и управлению миграциями (фреймворк Alembic [4]),
- Использование специальных моделей хранения неструктурированных данных: документно-ориентированные, графовые и т.д. Пример практического задания представлен на рисунке 1.
- Docker-контейнеризация готового приложения.



Площадка для интернет-магазина. Эта платформа поможет клиентам покупать или исследовать товары, а небольшие магазины (предприятия) смогут продавать свои продукты и привлекать больше клиентов. Эта платформа очень похожа на магазин Amazon, но данная платформа будет ориентирована на интернет-магазины, магазины, на малый и крупный бизнес.

Цель данного проекта – создание надежного API для магазина.

Платформа состоит из 6 основных модулей:

- Управление пользователями
- Магазины
- Продукты
- Бренды
- Платежи
- Статистика

С REST API интернет-магазина вы можете получить доступ к данным в формате JSON. Для доступа к данным достаточно отправить HTTP-запрос на адрес http://localhost/{controllerName}с параметрами, специфичными для контроллера. http://localhost/{controllerName}/{id} Каждый вызов метода возвращает JSON-объект с тремя возможными полями: статус, сообщение и данные. Статус может быть «ok», «fail» или «exception». Если статус «ok», то сообщения нет, данные содержат JSON-элемент, который будет описан для каждого метода отдельно. Если статус «fail» или «exception», то данных нет. Если статус «fail», то комментарий содержит причину, по которой запрос не выполнен. Если статус «exception», то необходимо обратиться к документации по API.

Рисунок 1. Пример практического задания

В рамках данного блока на практических занятия студенты создают Docker-образ с разработанным приложением и его зависимостями, а также подготавливают Docker-образ

для PostgreSQL. Затем настраивают взаимодействие созданных образов в файле *docker-compose.yml* для управления как приложением, так и базой данных. После подготовки необходимых образов и настройки их взаимодействия разворачивают созданные контейнеры на сервере.

В третьем блоке рассматривается процесс автоматизации всего цикла разработки программного обеспечения, начиная с интеграции кода и заканчивая его доставкой пользователям. Кроме трех основные блоков программа курса предусматривает выполнение итогового проекта. Примером итогового проекта является разработка системы "BioSense" [5] для хранения, обработки и управления климатическими данным Государственного природного заповедника "Тигирекский".

Так как основной акцент курса сделан на практическое использование современных инструментов для профессионального решения задач, особое внимание на занятиях уделяется надежности и безопасности подключения к СУБД PostgreSQL. В примере демонстрируется два способа формирования строки подключения: с использованием асинхронного клиента БД PostgreSQL и через специальный драйвер: синхронная работа.

```
# Для асинхронного подключения с asyncpg
str_connect = "postgresql+asyncpg://user:password@host:port/dbname"
# Для синхронного подключения с psycopg2
str_connect = "postgresql://user:password@host:port/dbname"
```

Можно заметить, что параметры подключения хранятся в "открытом виде" в коде проекта. Чтобы этого избежать, необходимо использовать переменные окружения для безопасного управления "секретными" данными. С помощью специальных библиотек имя пользователя, пароль и адрес базы данных извлекаются из файла с расширением "*.env" и используются в коде для формирования строки подключения к БД. Пример показывает основные шаги такого подхода:

```
# импортируем библиотеки os, dotenv
# Загружаем переменные из файла *.env: load_dotenv()
# формируем значения параметров для строки подключения из переменных окружения:
str_user = os.getenv("STR_USER")
str_password = os.getenv("STR _PASSWORD")
str_host = os.getenv("STR _HOST")
str_name = os.getenv("STR _NAME")
# Формируем строку подключения к базе данных
str_connect = f"postgresql://{str_user}:{str_password}@{str_host}/{str_name}"
```

Данный пример демонстрирует: как избежать хранения *секретных* учетных данных непосредственно в коде проекта. Конечно, это не единственная проблемная ситуация, которая возникает при реализации взаимодействия с базой данных и влияет на безопасную работу backend-части приложения. Примеры "подобных ситуаций" следует рассматривать и анализировать варианты их решения на очных практических занятиях.

Опыт проведения занятий в рамках данного курса показал заинтересованность студентов в изучении курса по созданию API-приложения на основе реляционной СУБД. Так же следует заметить, только дистанционные формы работы со студентами не позволяют передать глубину изложения материала, что особенно важно для занятий по изучению основ работы с реляционной базой данных. "Живое общение", особенно с представителями профессионального сообщества, способствует повышению мотивации к изучению нового, стимулируют развитие навыков, подталкивает к самообразованию. Без желания постоянно "самообразовываться" выпускнику сложно будет адаптироваться в новой среде после трудоустройства.

Список литературы

1. Documentation FastAPI. [Электронный ресурс]. — URL: https://fastapi.tiangolo.com. Дата обращения 02.11.2025.

2. Documentation PostgreSQL. [Электронный ресурс]. — URL: https://www.postgresql.org/. Дата обращения 02.11.2025.

3. Documentation SQLAlchemy. [Электронный ресурс]. — URL: https://docs.sqlalchemy.org/en/20/. Дата обращения 02.11.2025.

4. Documentation Alembic. [Электронный ресурс]. — URL: https://alembic.sqlalchemy.org/. Дата обращения 02.11.2025.

5. BioSense. [Электронный ресурс]. — URL: https://biosense.asu.ru. Дата обращения 02.11.2025.