Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 21.Б07-мм

Рефакторинг кода для сайта кафедры системного программирования

Колесников Михаил Олегович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ст. преподаватель кафедры ИАС К. К. Смирнов

Санкт-Петербург
2024

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Обзор **5**

2.1. Используемые технологии 5

2.2. Существующие решения 5

2.3. Обработка конфигураций 7

2.4. Архитектура проекта до рефакторинга 7

3. Реализация **8**

3.1. Создание экземпляра приложения 8

3.2. Конфигурации 8

3.3. Хранение данных 9

3.4. Чертежи 9

3.5. Иерархия проекта 9

Заключение **11**

Список литературы **12**

Введение

С ростом какого-либо программного продукта ключевыми компонентами дальнейшей разработки проекта, а также сопровождения и обслуживания, является структурирование кода и его архитектура. Иногда можно столкнуться с ситуацией, когда в начале разработки для ускорения процесса все компоненты или модули пишутся в один файл. Хотя такой подход может ускорить начальные этапы разработки, со временем это приводит к значительным проблемам. Такой код становится сложнее поддерживать, а добавление новой функциональности может привести к огромному количеству ошибок или большой сложности внедрения её в проект. Однако, с применением структурного подхода и правильного проектирования можно избежать большинства трудностей, возникающих при разработке.

Одним из таковых проектов является серверная часть [7] сайта кафедры системного программирования [5]. Он представляет собой веб приложение, а не набор статических страниц. Сайт разрабатывался большим количеством людей, каждый раз появлялись новые участники, вносящие свой вклад в проект, расширяя функциональность. Начиная с главного файла, сайт развился в большое, запутанное Flask-приложение. Для обеспечения успешного развития проекта было решено:

- Сделать приложение модульным. Разбить код на части, где каждая отвечает за конкретную функциональность. Разбиение позволит лучше организовывать разработку и обеспечивать четкую структуру проекта.

- Вынести конфигурационные файлы, что позволит управлять настройками приложения более гибко и создавать конфигурации для различных сценариев работы приложения.

1. Постановка задачи

Целью работы является рефакторинг кода сайта кафедры системного программирования. Для её выполнения были поставлены следующие задачи.

1. спроектировать архитектуру проекта.
2. разделить конфигурационные файлы от приложения.
3. разделить данные и приложение.

2. Обзор

2.1. Используемые технологии

Для выполнения задачи были задействованы следующие инструменты:

- Python [4] - высокруровневый язык программирования с динамической типизацией.
- Flask [3] — фреймворк для создания веб-приложений на языке программирование Python. Является легковесным и простым в использовании инструментом, предназначенным для разработки.

2.2. Существующие решения

2.2.1. Blueprints

Для разделения большого приложения на независимые модули в Flask используются чертежи или же Blueprints [2], представляющие собой эффективный механизм организации кода, позволяющий структурировать приложение на различные модули в соответствии с его функциональными компонентами. Создание чертежа начинается с импорта класса Blueprint, после этого создается его экземпляр с уникальным именем. Чтобы определить функцию связанную с url-маршрутом, используются специальные декораторы @bp.route().

Далее для использования написанного модуля чертеж необходимо зарегистрировать его в созданном приложении Flask. Это происходит с помощью метода register_blueprint().

С помощью такого подхода приложение получается модульным, разбитым на логические компоненты. Каждая из которых отвечает за свою функциональность. Это не только улучшает структуру кода, но и облегчает дальнейшую поддержку и расширения приложения. Использование чертежей не только способствует разбиению проект на логические части, но и обеспечивает более высокую изоляцию между различными

частями, что в свою очередь облегчает тестирование отдельной функциональности.

2.2.2. Фабрика приложений

Хорошей практикой написания Flask приложений является применение паттерна фабрика приложений [1] . Этот подход позволяет более гибко управлять созданием экземпляра приложения, что является полезным при тестировании или поддержке приложения с различными конфигурационными настройками. Суть паттерна заключается в том, что главный экземпляр приложения создается в отдельной функции. С помощью него появляется возможность тестировать приложение с различными настройками, также создание нескольких экземпляров приложения может позволить запускать различные версии приложения. Основными шагами использования фабрики являются:

- Создание экземпляра приложения. В фабричной функции создается Flask-приложение, обрабатывающее последующие входящие запросы.

- Добавление конфигураций. К созданному экземпляру приложения добавляются настройки, позволяющие задавать параметры, такие как настройки базы данных, секретные ключи и другие параметры.

- Регистрация расширений и чертежей. Чертежи регистрируются в приложении, чтобы определить URL-маршруты. Расширения, такие как SQLAlchemy или FLask_WTF, также регистрируются в приложении, если это необходимо.

Применение этого паттерна улучшает общую структуру кода. Обеспечивает четкое разделение ответственности между созданием приложения, добавление конфигов и добавлением функциональности с помощью чертежей.

2.3. Обработка конфигураций

Для подключения конфигурационных файлов в Flask существуют различные подходы:

- Конфиги из файлов PY. В данном случае конфигурационные настройки хранятся в Python-файлах, что обеспечивает удобство в использовании. Определение переменных происходит непосредственно в коде.

- Конфиги из файлов, например TOML [6] . Использование различных форматов, таких как TOML, позволяет вынести настройки в отдельный файл. Это позволяет изменять настройки приложения без вмешательства в код.

- Конфиги из переменных окружения. Может быть полезно для управления настройками напрямую через окружение.

Разнообразие подходов для взаимодействия с конфигурационными файлами в Flask предоставляет разработчикам выбор, наиболее подходящий для требований проекта.

2.4. Архитектура проекта до рефакторинга

Все Flask-приложение находится в главной папке src. Оно создается в одном файле flask_se.py. В том же файле происходит инициализация всех маршрутов. Файл se_models.py, где хранятся пользовательские данные и классы, представляющие собой модели для базы данных. Некоторые конфигурационные файлы были написаны в том же файле, где создается экземпляр приложения, а другие были выделены в отдельный файл. При написании сервиса по учебным практикам, дополнительные настройки были созданы в отдельном файле. Таким образом получается 3 файла содержат настройки для одного приложения.

3. Реализация

3.1. Создание экземпляра приложения

Для инициализации приложения был реализован паттерн фабрика приложений. Создан специальный файл с функцией создания Flask-приложения, включающий в себя добавление необходимых конфигураций и регистрацию чертежей. В ней происходит инициализация экземпляра приложения, добавление конфигурационных настроек через объект, представляющий собой класс с переменными. После это регистрируются чертежи и различные расширения.

Листинг 1: Функция создания Flask приложения

```
def create_app(config_name):
    # creating application
    app = Flask(...)
    # add config to app
    app.config.from_object(config[config_name])
    # register different blueprints
    app.register_blueprint(...)
    return app
```

3.2. Конфигурации

Для конфигураций было решено создать отдельный файл config.py, в котором должны находиться классы, определяющие конфигурационные данные. Для улучшения сопровождения кода в дальнейшем сделан словарь с названием конфигурации и класса, который ему соответствует. Так при создании экземпляра приложения необходим импорт словаря, далее можно устанавливать различные конфигурационные настройки, указывая необходимое название для них.

Листинг 2: Класс конфигураций

```
# Creating a configuration class
class Config():
```

```
    ...
    # fields
    SECRET_KEY = ...
    SQLITE_DATABASE_NAME = ...
    ....

# A dictionary in which we find a class by name.
# Can be extended with other classes if necessary.
config = {"default": Config}
```

3.3. Хранение данных

В проекте для инициализации базы данных хранятся данные представляющие собой информацию о преподавателях, предметах обучения, должностях, информация небходимая для практик и ВКР. В связи с тем, что ранее информация хранилась в файле с моделями для Flask, принято решение переместить все данные в отдельный файл.

3.4. Чертежи

Приложение было разбито на различные модули, изображенные на рисунке 2 синими цветом. Каждая компонента включает в себя создание самого экземпляра чертежа в файле __init__.py и маршрутов в файле routes.py. Так как были написаны чертежи, необходимо было поменять все ссылки в HTML файлах на новые.

3.5. Иерархия проекта

Учитывая выше описанные изменения, теперь проект состоит из файла app.py для создания приложения, файла config.py для конфигурационных переменных. В папке src лежат все модули и остальные файлы. Модели находятся в файле models.py, а данные в data.py. На рисунке 2 показана схема проекта с указанными файлами и чертежами.

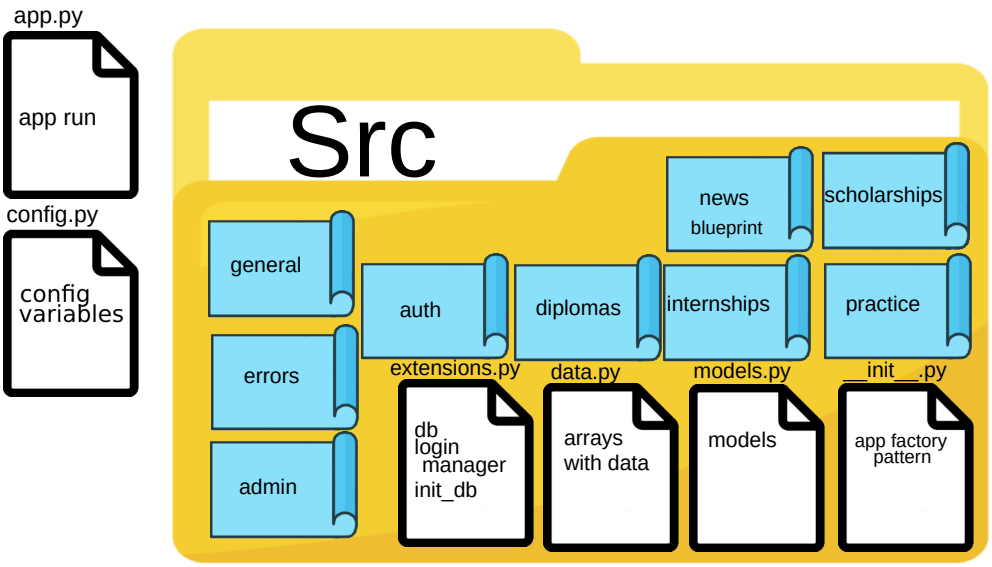

Рис. 1: Архитектура проекта после изменения.

Заключение

В итоге достигнуты следующие результаты.

- спроектирована архитектура проекта.
- разделены конфигурационные файлы от приложения.
- разделены данные и приложение.

Ссылка на репозиторий GitHub:
https://github.com/Michael-Kolesnikov/spbu_se_site

Список литературы

[1] Application factory. — URL: https://flask.palletsprojects.com/en/2.3.x/patterns/appfactories/ (дата обращения: 11 декабря 2023 г.).

[2] Blueprints. — URL: https://flask.palletsprojects.com/en/3.0.x/blueprints/ (дата обращения: 11 декабря 2023 г.).

[3] Flask. — URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 15 декабря 2023 г.).

[4] Python. — URL: https://ru.wikipedia.org/wiki/Python (дата обращения: 11 декабря 2023 г.).

[5] Spbu-se site. — URL: https://github.com/spbu-se/spbu_se_site (дата обращения: 11 декабря 2023 г.).

[6] TOML. — URL: https://en.wikipedia.org/wiki/TOML (дата обращения: 7 января 2024 г.).

[7] Wikipedia. Backend. — URL: https://en.wikipedia.org/wiki/Frontend_and_backend (дата обращения: 11 декабря 2023 г.).