Отчет
По лабораторной №8

по дисциплине «Программирование на Python»

Тема: «Клиент-серверное приложение на Python с использованием Jinja2»

Выполнила:
Родина Виктория Юрьевна
P3121
504638
Преподаватель:
Жуков Николай Николаевич

Цель работы:

1. Создать простое клиент-серверное приложение на Python без серверных фреймворков.
2. Освоить работу с HTTPServer и маршрутизацию запросов.
3. Применять шаблонизатор Jinja2 для отображения данных.
4. Реализовать модели предметной области (User, Currency, UserCurrency, App, Author) с геттерами и сеттерами.
5. Структурировать код в соответствии с архитектурой MVC.
6. Получать данные о курсах валют через функцию get_currencies и отображать их пользователям.
7. Реализовать функциональность подписки пользователей на валюты и отображение динамики их изменения.
8. Научиться создавать тесты для моделей и серверной логики.

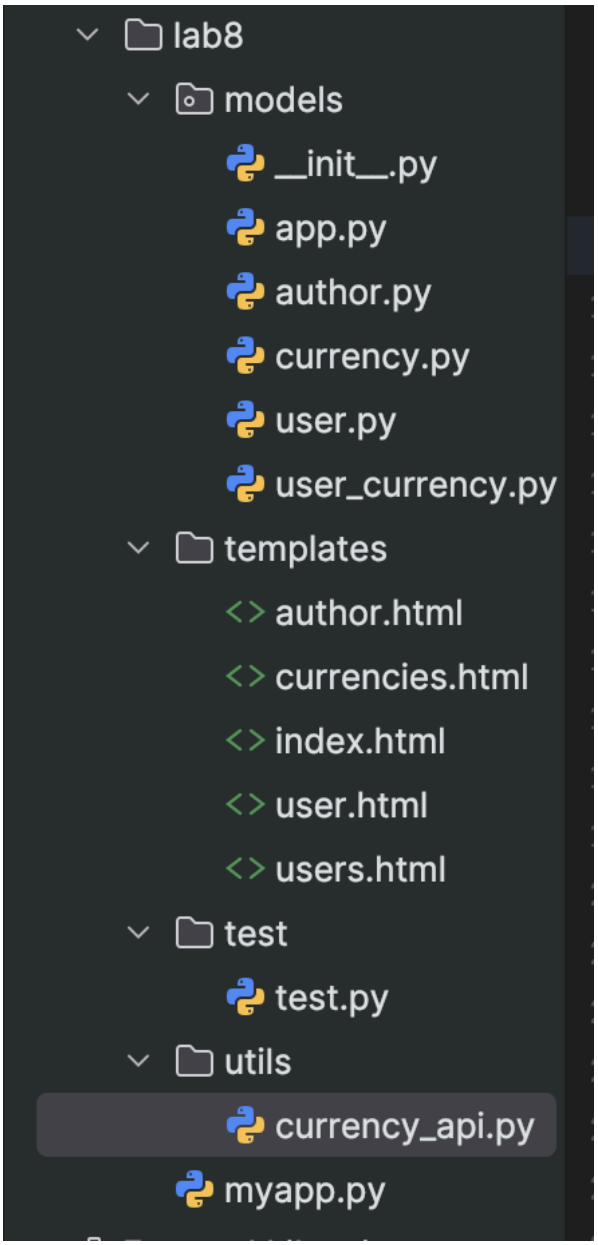
2. Описание предметной области

Приложение предназначено для управления информацией о курсах валют и пользовательских подписках. Система включает следующие сущности:

Основные модели:

1) Author - разработчик приложения
   - name (строка, минимальная длина 2 символа) - имя автора
   - group (строка, длина > 5 символов) - учебная группа
2) App - описание программного продукта
   - name (строка, ≥2 символа) - название приложения
   - version (непустая строка) - версия приложения
   - author (экземпляр Author) - автор разработки
3) Currency - финансовая валюта
   - id (непустая строка) - уникальный идентификатор
   - num_code (строка из цифр) - цифровой код валюты
   - char_code (ровно 3 символа) - буквенный код (USD, EUR и т.д.)
   - name (строка, ≥2 символа) - полное название
   - value (положительное число) - курс к рублю
   - nominal (положительное целое число) - номинал
4) User - пользователь системы
   - id (целое число > 0) - уникальный идентификатор
   - name (строка, ≥2 символа) - имя пользователя
   - subscriptions (список) - подписки на отслеживание валют
   - Методы: add_subscription(), remove_subscription(), has_subscription()
5) UserCurrency - связь пользователя с отслеживаемой валютой
   - id (целое число > 0) - идентификатор связи
   - user_id (целое число > 0) - ID пользователя
   - currency_id (строка) - ID валюты

3. Структура проекта:




```
# Пакет моделей данных (Model в MVC)
```

```
# HTML шаблоны (View в MVC)
```

```
# Модуль тестирования
```

```
# Вспомогательные утилиты
```

```
# Основной файл приложения, точка входа
```

4. Описание реализации:

4.1 Реализация моделей и их свойств

Все модели реализованы с использованием инкапсуляции. Приватные атрибуты обозначены двойным подчеркиванием. Для доступа к данным используются свойства с геттерами и сеттерами. В сеттерах реализована валидация: проверка типов данных, длин строк, диапазонов числовых значений. Например, для валюты должен быть ровно 3 символа, value — положительным числом. При невалидных данных выбрасывается исключение ValueError с описанием ошибки.

4.2 Реализация маршрутов и обработки запросов

Обработка HTTP-запросов реализована в классе CurrencyHandler, наследующем BaseHTTPRequestHandler. Метод do_GET анализирует URL и перенаправляет запрос соответствующим методам: show_index, show_users, show_user, show_currencies, show_author. Для маршрута /user извлекается query-параметр id. Все ошибки (404, 500) обрабатываются с отправкой соответствующих HTTP-статусов.

4.3 Использование шаблонизатора Jinja2

Jinja2 инициализируется с указанием папки templates в качестве источника шаблонов. Включено автоматическое экранирование для безопасности. В контроллере реализован метод render_template, который объединяет базовый контекст (название приложения, автор, навигация) с переданными данными, рендерит шаблон и отправляет HTML-ответ.

4.4 Интеграция функции get_currencies

Функция get_currencies в модуле currency_api отправляет GET-запрос к API Центробанка России. Полученный JSON-ответ парсится, извлекаются курсы для запрошенных валют. Реализована обработка ошибок сети и некорректных данных. В контроллере при отображении страницы /currencies эта функция вызывается для обновления курсов в реальном времени.

5. Скриншот:

Главная страница:

CurrencyTracker

Версия: 1.0

Автор: Виктория Родина

Группа: P3121

Главная | Пользователи | Валюты | Об авторе |

Описание

Простое приложение для отслеживания курсов валют.

Статистика

- Пользователей: 3
- Валют: 3
- Подписок: 4

Пользователи:

Пользователи

Автор: Виктория Родина

Группа: P3121

Главная | Пользователи | Валюты | Об авторе

Список пользователей

ID	Имя	Email	Подписок	Действие
1	Алексей Петров	алексей.петров@example.com	2	Просмотреть
2	Мария Сидорова	мария.сидорова@example.com	1	Просмотреть
3	Иван Иванов	иван.иванов@example.com	1	Просмотреть

← На главную

Пользователь: Алексей Петров

Автор: Виктория Родина

Группа: P3121

Основная информация

- **ID:** 1
- **Имя:** Алексей Петров
- **Email:** алексей.петров@example.com
- **Подписок:** 2

Подписки на валюты

Код	Название	Курс	Номинал
USD	Доллар США	92.5 руб.	1
EUR	Евро	101.3 руб.	1

← Назад к списку пользователей

Валюты:

Курсы валют

Автор: Виктория Родина

Группа: P3121

Текущие курсы

Обновить курсы

Код	Название	Курс	Номинал	ID
USD	Доллар США	77.83 руб.	1	R01235
EUR	Евро	90.54 руб.	1	R01239
GBP	Фунт стерлингов	104.2 руб.	1	R01035

← На главную

Об авторе

Имя: Виктория Родина

Группа: P3121

Приложение: CurrencyTracker

Версия: 1.0

Информация о разработчике

Студент, изучающий Python, веб-разработку и фреймворки.

Данное приложение создано в рамках учебного проекта.

Используемые технологии:

- Python
- Jinja2
- HTTPServer
- MVC архитектура

← На главную

Тестирования:

Пример:

```python

class TestAuthorModel(unittest.TestCase):
    """Тестирование модели Author."""

    def test_01_author_creation(self):
        """1. Проверка создания автора."""
        author = Author("Вика", "P31211")
        self.assertEqual(author.name, second: "Вика")
        self.assertEqual(author.group, second: "P31211")

    def test_02_author_name_validation(self):
        """2. Проверка валидации имени автора."""
        author = Author("Вика", "P31211")
        with self.assertRaises(ValueError):
            author.name = "B"  # Слишком короткое

    def test_03_author_group_validation(self):
        """3. Проверка валидации группы автора."""
        author = Author("Вика", "P31211")
        with self.assertRaises(ValueError):
            author.group = "P123"  # Длина должна быть > 5

```

Результаты тестирования:

```
Ran 15 tests in 0.002s

OK

Process finished with exit code 0
```

Вывод:

Применение принципов MVC

Архитектура MVC была успешно применена:

- Model (модели) — бизнес-логика и валидация данных в папке models/
- View (представления) — HTML-шаблоны в папке templates/ с использованием Jinja2
- Controller (контроллер) — обработка HTTP-запросов в my_app.py

Такое разделение позволило четко распределить ответственность: модели отвечают за данные и их целостность, шаблоны — за отображение, контроллер — за координацию между ними и обработку запросов.

Новые знания и навыки:

В ходе работы освоила создание HTTP-сервера с обработкой GET-запросов и query-параметров, настройку шаблонизатора Jinja2 для динамической генерации HTML-страниц, интеграцию с REST API Центробанка для получения актуальных курсов валют с обработкой JSON-ответов и ошибок соединения. Углубила понимание архитектурных принципов MVC, инкапсуляции данных и важности валидации на уровне моделей.