Лабораторная работа 8. Клиент-серверное приложение на Python с использованием Jinja2

Отчёт

Выполнил Жохов Даниил (502506)

Группа P3123

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.

Описание предметной области

Модели

1. **Author**
   - `name` — имя автора
   - `group` — учебная группа
2. **App**
   - `name` — название приложения
   - `version` — версия приложения
   - `author` — объект Author
3. **User**
   - `id` — уникальный идентификатор
   - `name` — имя пользователя
4. **Currency**
   - `id` — уникальный идентификатор
   - `num_code` — цифровой код
   - `char_code` — символьный код
   - `name` — название валюты

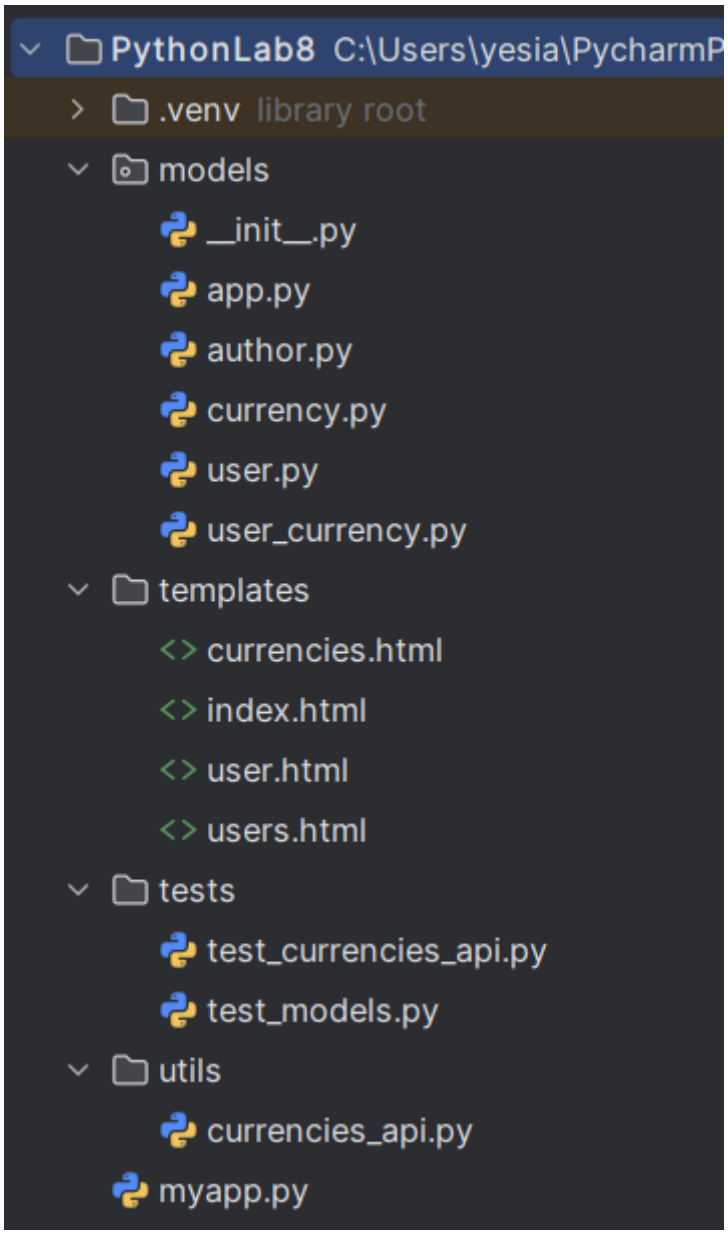
- `value` — курс
- `nominal` — номинал (за сколько единиц валюты указан курс)

Пример XML:

```
<Valute ID="R01280">
 <NumCode>360</NumCode>
 <CharCode>IDR</CharCode>
 <Nominal>10000</Nominal>
 <Name>Рупий</Name>
 <Value>48,6178</Value>
</Valute>
```

7. **UserCurrency**
    - `id` — уникальный идентификатор
    - `user_id` — внешний ключ к User
    - `currency_id` — внешний ключ к Currency
    - Реализует связь «много ко многим» между пользователями и валютами.

Структура проекта



Описание реализации

1. Реализация моделей и их свойств (геттеры/сеттеры)
Модели предметной области (Author, App, User, Currency, UserCurrency) реализованы как отдельные классы в пакете models. Каждое свойство снабжено геттером и сеттером, реализованными с помощью декоратора @property.

Сеттеры выполняют строгую валидацию входных данных:

Проверка типов (например, строка, целое число, объект другого класса).
Проверка содержательной корректности (например, char_code валюты — ровно 3 символа, num_code — 3 цифры, имя пользователя — непустая строка).

При нарушении условий выбрасывается исключение (ValueError или TypeError).
Такой подход обеспечивает целостность данных на уровне модели и предотвращает попадание некорректных значений в бизнес-логику приложения.

Для автоматической генерации уникальных идентификаторов (id) у User и UserCurrency используются статические счётчики (_next_id), что моделирует простейшую логику автоинкремента без базы данных.

2. Реализация маршрутов и обработка запросов
Маршрутизация реализована с использованием стандартных средств Python:

http.server.HTTPServer и http.server.BaseHTTPRequestHandler.
В методе do_GET класса-обработчика (MyHandler) происходит разбор входящего URL с помощью urllib.parse.urlparse и urllib.parse.parse_qs.
В зависимости от значения path (/, /users, /user, /currencies, /author) сервер:

Формирует соответствующий контекст данных (например, список пользователей или курсы валют).
Вызывает нужный HTML-шаблон через Jinja2.
Отправляет HTTP-ответ с заголовком Content-type: text/html; charset=utf-8 и телом, закодированным в UTF-8.
Обработка параметров запроса (например, /user?id=1) выполняется через извлечение значения из query = parse_qs(...). При отсутствии пользователя возвращается ошибка 404 (в упрощённой форме).

Такой подход демонстрирует ручную маршрутизацию, без использования фреймворков, и чётко соответствует роли контроллера в архитектуре MVC.

3. Использование шаблонизатора Jinja2
Шаблонизатор Jinja2 используется для отделения логики от представления. Инициализация выполняется один раз при старте приложения:

```
from jinja2 import Environment, PackageLoader, select_autoescape

env = Environment(
    loader=PackageLoader("myapp"),
    autoescape=select_autoescape()
)
```

PackageLoader("myapp") указывает, что шаблоны находятся в папке templates внутри пакета myapp (корневой директории проекта).
select_autoescape() включает автоматическую экранировку HTML-содержимого, что повышает безопасность (защита от XSS).
Объект env кэширует загруженные шаблоны, что повышает производительность при множественных запросах.
В обработчике запросов шаблоны загружаются один раз:

python
12
template = env.get_template("currencies.html")
html = template.render(currencies=currencies_list)
Передача данных в шаблоны осуществляется через именованные аргументы, что позволяет гибко формировать страницы без жёсткой привязки к HTML-коду в контроллере.

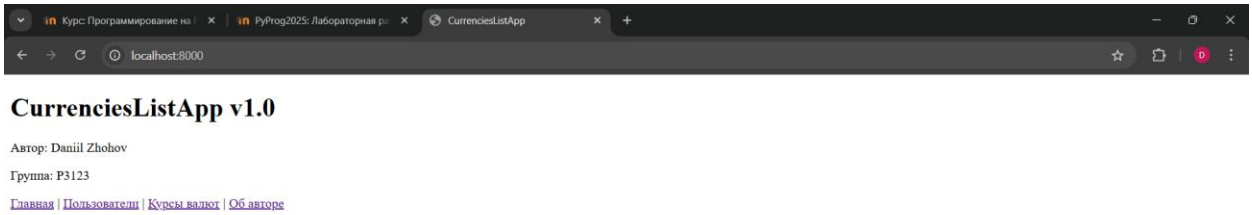
4. Интеграция функции get_currencies для получения актуальных курсов
Функция get_currencies() реализована в модуле utils.currencies_api. Она:

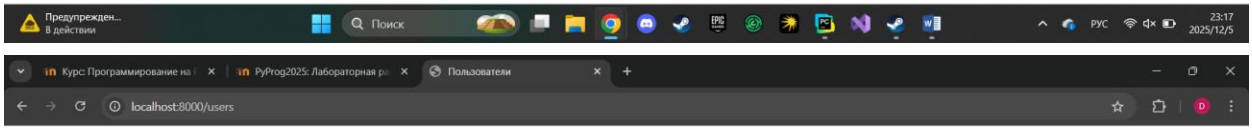
Отправляет HTTP-запрос к официальному API ЦБ РФ (https://www.cbr.ru/scripts/XML_daily.asp).
Парсит полученный XML-документ с помощью xml.etree.ElementTree.
Извлекает данные о каждой валюте (CharCode, Name, Value, Nominal и др.).
Преобразует строковое значение курса (с запятой в качестве разделителя) в float.
Создаёт и возвращает список объектов класса Currency.
Эта функция вызывается при каждом запросе к маршруту /currencies, а также при отображении подписок пользователя (/user?id=...), что гарантирует актуальность курсов при каждом просмотре.

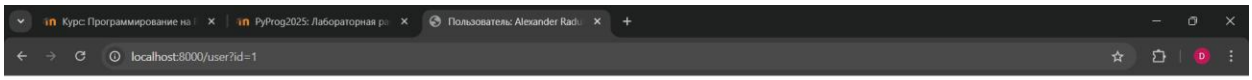
Интеграция с моделями происходит напрямую: объекты Currency, созданные в get_currencies, передаются в шаблоны и отображаются пользователю. Это обеспечивает согласованность данных между слоем модели и представления.

Таким образом, внешний API инкапсулирован в отдельной утилите, а бизнес-логика оперирует уже валидированными объектами предметной области.

Примеры работы приложения





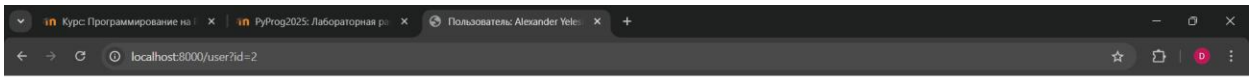


Пользователь: Alexander Radulov

Подписки на валюты:

- USD (Доллар США): 76.0937 за 1
- EUR (Евро): 88.7028 за 1

← Назад к списку

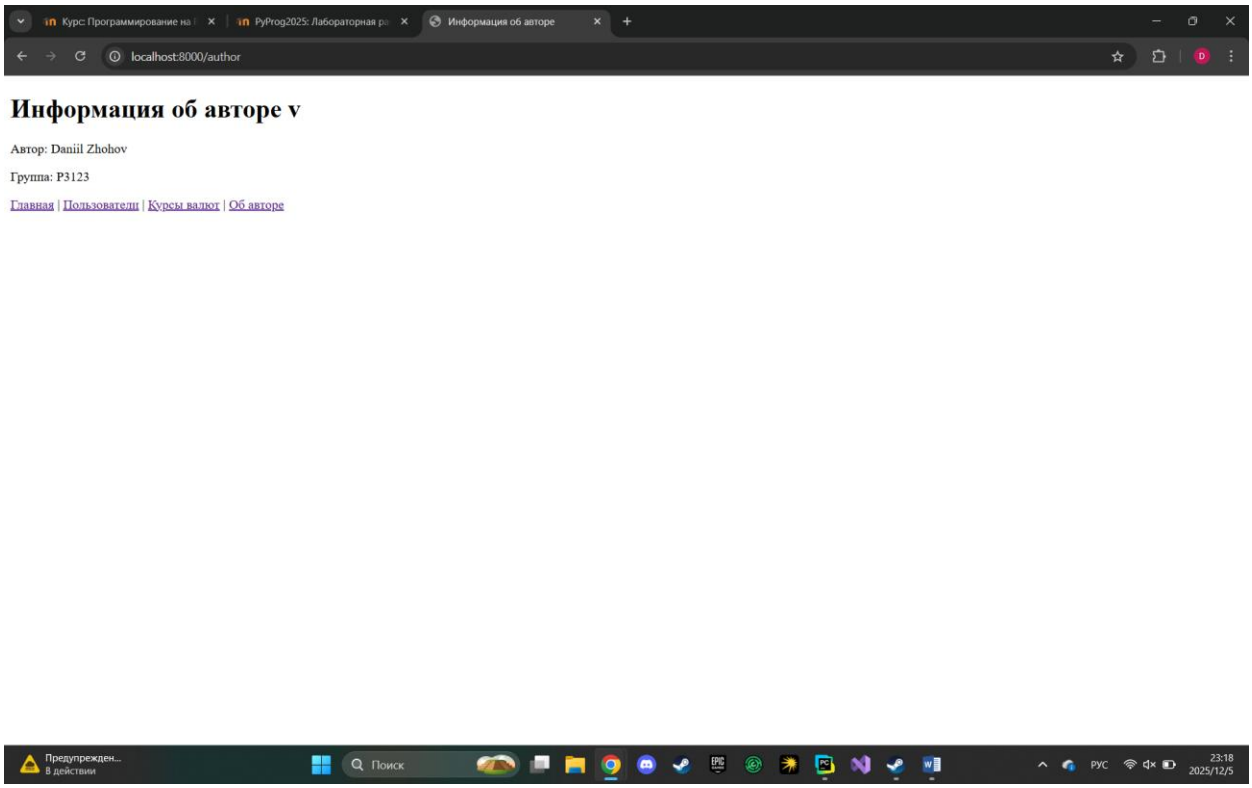


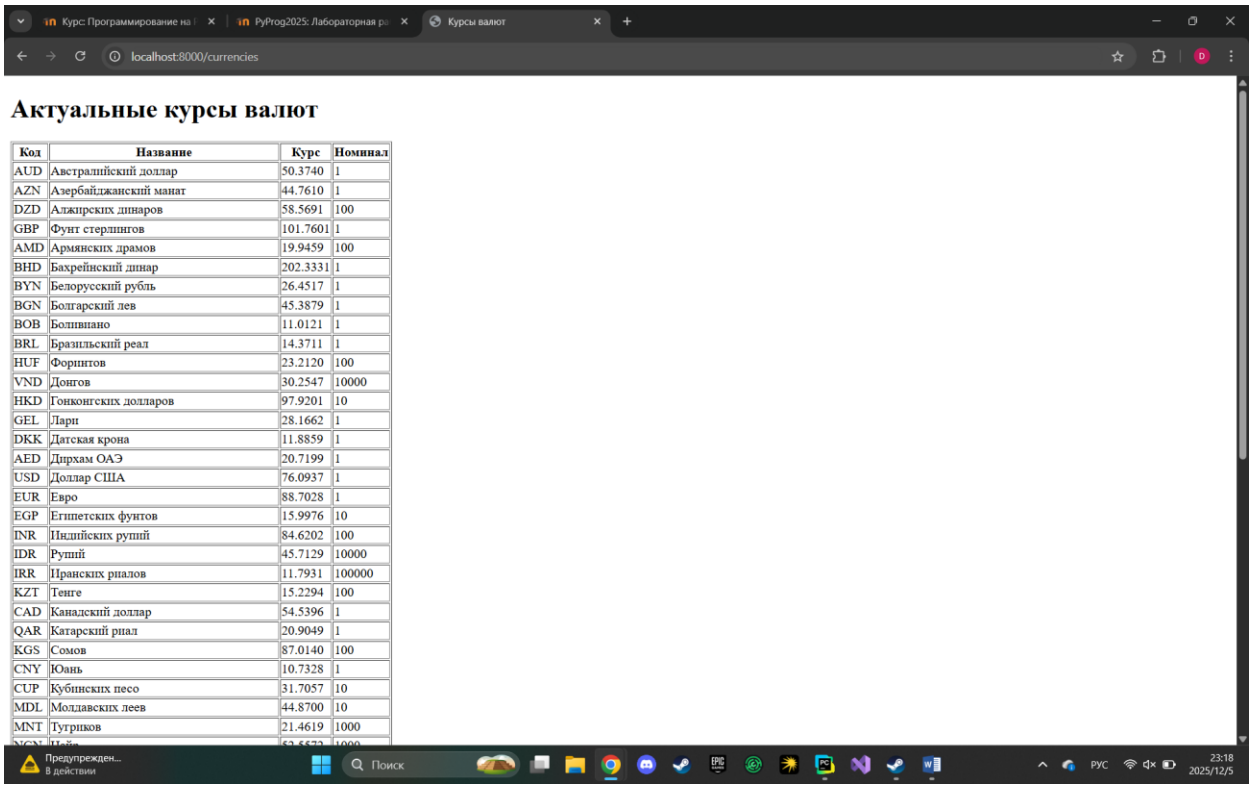
Пользователь: Alexander Yelesin

Подписки на валюты:

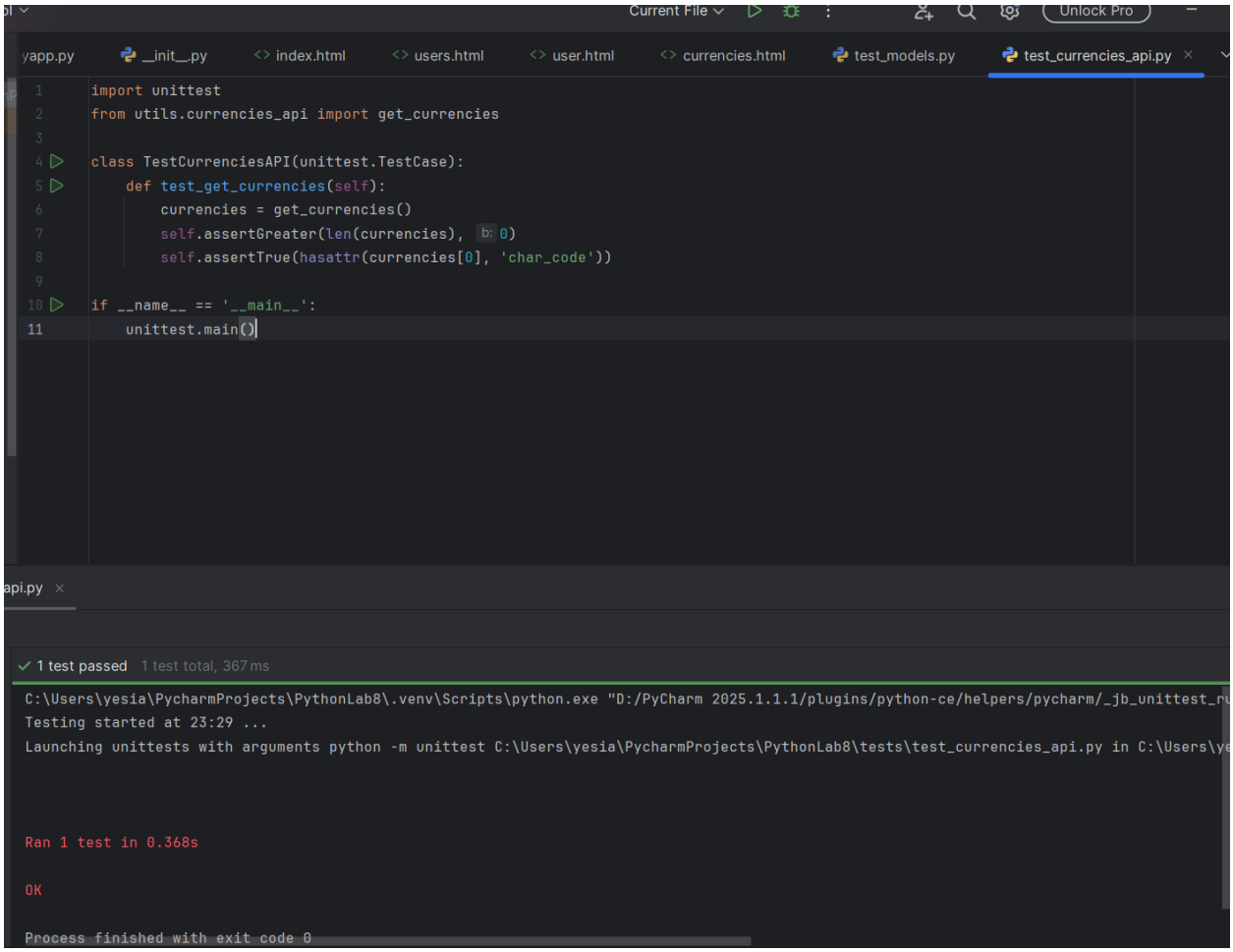
- CNY (Юань): 10.7328 за 1

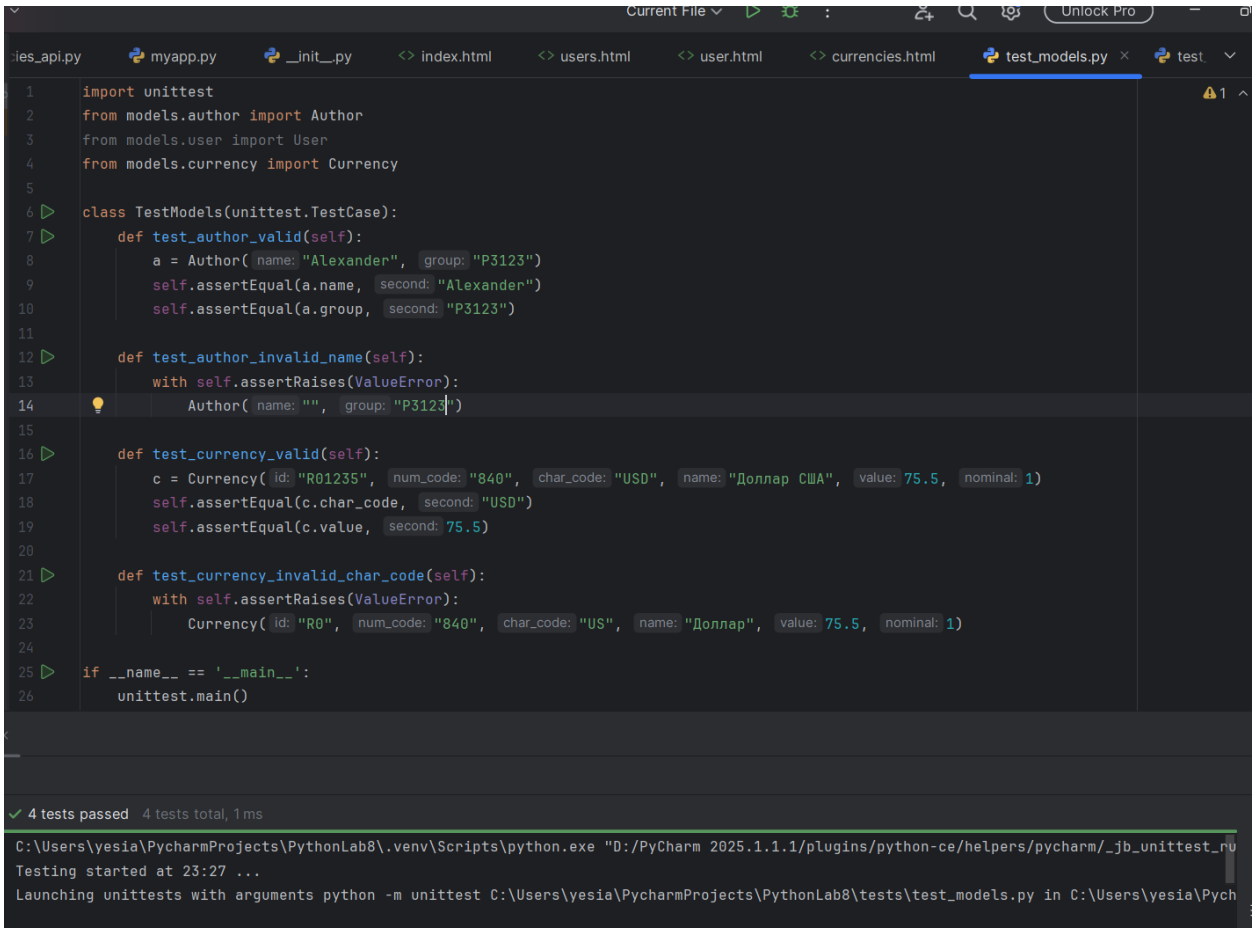
← Назад к списку


Курс: Программирование на
PyProg2025: Лабораторная ра
Информация об авторе
localhost:8000/author
Информация об авторе v
Автор: Daniil Zhohov
Группа: P3123
Главная | Пользователи | Курсы валют | Об авторе
Предупрежден...
В действии
Поиск
РУС
23:18
2025/12/5


Курс: Программирование на
PyProg2025: Лабораторная ра
Курсы валют
localhost:8000/currencies
Актуальные курсы валют
Код | Название | Курс | Номинал
AUD | Австралийский доллар | 50.3740 | 1
AZN | Азербайджанский манат | 44.7610 | 1
DZD | Алжирских динаров | 58.5691 | 100
GBP | Фунт стерлингов | 101.7601 | 1
AMD | Армянских драмов | 19.9459 | 100
BHD | Бахрейнский динар | 202.3331 | 1
BYN | Белорусский рубль | 26.4517 | 1
BGN | Болгарский лев | 45.3879 | 1
BOB | Боливиано | 11.0121 | 1
BRL | Бразильский реал | 14.3711 | 1
HUF | Форинтов | 23.2120 | 100
VND | Донгов | 30.2547 | 10000
HKD | Гонконгских долларов | 97.9201 | 10
GEL | Лари | 28.1662 | 1
DKK | Датская крона | 11.8859 | 1
AED | Дирхам ОАЭ | 20.7199 | 1
USD | Доллар США | 76.0937 | 1
EUR | Евро | 88.7028 | 1
EGP | Египетских фунтов | 15.9976 | 10
INR | Индийских рупий | 84.6202 | 100
IDR | Рупий | 45.7129 | 10000
IRR | Иранских риалов | 11.7931 | 100000
KZT | Тенге | 15.2294 | 100
CAD | Канадский доллар | 54.5396 | 1
QAR | Катарский риал | 20.9049 | 1
KGS | Сомов | 87.0140 | 100
CNY | Юань | 10.7328 | 1
CUP | Кубинских песо | 31.7057 | 10
MDL | Молдавских леев | 44.8700 | 10
MNT | Тугриков | 21.4619 | 1000
Предупрежден...
В действии
Поиск
РУС
23:18
2025/12/5

Тестирование





Выводы

1. Проблемы, возникшие при реализации
В ходе реализации проекта возникло несколько технических и архитектурных сложностей:

Относительные импорты в Python: при попытке использовать from ..models import ... в модуле utils/currencies_api.py возникла ошибка ImportError: attempted relative import beyond top-level package. Это произошло из-за запуска myapp.py как standalone-скрипта. Проблема была решена переходом на абсолютные импорты (from models.currency import Currency), что упростило структуру и сделало запуск более предсказуемым.
Парсинг XML от ЦБ РФ: данные приходят с запятой в качестве десятичного разделителя (например, "75,2345"), что вызывает ошибку при конвертации в float. Требовалась предварительная замена запятой на точку.
Отсутствие постоянного хранилища: без базы данных пришлось моделировать подписки и пользователей статическими списками, а историю курсов — глобальной переменной. Это ограничивает масштабируемость, но допустимо в учебных целях.
Ручная маршрутизация: отсутствие фреймворка потребовало реализации разбора URL и параметров вручную через urlparse и parse_qs, что увеличивает объём кода и риск ошибок (например, при отсутствии параметра id).
2. Применение принципов MVC
Архитектура MVC была успешно применена:

Model (модель): все классы предметной области (User, Currency, Author и др.) реализованы

в пакете models. Они содержат только данные и логику валидации (через геттеры/сеттеры), без зависимости от представления или сети.
View (представление): шаблоны в папке templates/ отвечают исключительно за отображение данных. Они не содержат бизнес-логики — только HTML и Jinja2-конструкции ({{ }}, {% for %}).
Controller (контроллер): класс MyHandler, унаследованный от BaseHTTPRequestHandler, обрабатывает HTTP-запросы, извлекает данные из моделей (включая вызов get_currencies), формирует контекст и рендерит шаблоны.
Чёткое разделение позволило легко вносить изменения: например, добавление графика затронуло только контроллер и шаблон, не затрагивая модели.

3. Новые знания и навыки
В ходе работы были получены ценные практические навыки:

Работа с http.server: научился создавать простой HTTP-сервер без фреймворков, обрабатывать GET-запросы, парсить URL и формировать корректные HTTP-ответы с заголовками и кодировкой.
Использование Jinja2: освоил инициализацию Environment с PackageLoader, понял важность однократной инициализации для производительности, научился безопасно передавать данные в шаблоны и использовать фильтры (например, tojson для передачи данных в JavaScript).
Интеграция с внешним API: получил опыт работы с XML-API (ЦБ РФ), включая обработку сетевых ошибок, парсинг XML и преобразование данных в объекты предметной области.