МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«Национальный исследовательский университет ИТМО»

(ФГАОУ ВО «ИТМО»)

Факультет: «Факультет программной инженерии и компьютерной техники»

Дисциплина: «Программирование на Python».

Лабораторная работа № 8

Группа: P3121

Выполнила:

Шукалович Екатерина Андреевна

Проверил:

Жуков Николай Николаевич

Санкт-Петербург 2025

Цель:

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
    a. name — имя автора
    b. group — учебная группа
2. App
    a. name — название приложения
    b. version — версия приложения
    c. author — объект Author
3. User
    a. id — уникальный идентификатор
    b. name — имя пользователя
4. Currency
    a. id — уникальный идентификатор
    b. num_code — цифровой код
    c. char_code — символьный код
    d. name — название валюты
    e. value — курс
    f. 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>

5. UserCurrency
    a. id — уникальный идентификатор
    b. user_id — внешний ключ к User
    c. currency_id — внешний ключ к Currency
    d. Реализует связь «много ко многим» между пользователями и валютами.

Структура проекта:

```
lr8/
├── models/
│   ├── __init__.py        # импорт всех моделей
│   ├── author.py
│   ├── app.py
│   ├── user.py
│   ├── currency.py
│   └── user_currency.py
├── templates/
│   ├── index.html
│   ├── users.html          # список пользователей
│   └── currencies.html     # список валют с текущими курсами
│   ├── error.html          # файл для обработки ошибок, связанных с неверно
│   ├── author_project.html  # указанным ID
│   ├── user.html
├── myapp.py              # запуск сервера и маршрутизация
└── utils/
    └── currencies_api.py     # функция get_currencies и get_currency_history
```

Описание реализации:

В проекте были созданы модели Author, User, App, Currency, UserCurrency. Доступ к свойствам каждой модели осуществляется через геттеры и сеттеры. Сеттеры выполняют проверку типов и корректности значений (например, если в User введен некорректный тип данных или задано пустое имя, то выбрасывается ValueError).

Для обработки запросов используется модуль HTTPServer и класс BaseHTTPRequestHandler.

1. Поддерживаемые маршруты:
    a. / — главная страница с информацией о приложении и авторе
    b. /users — список пользователей
    c. /user?id=... — информация о конкретном пользователе и его подписках
    d. /currencies — список валют с текущими курсами
    e. /author — информация об авторе
2. Маршрутизация реализуется через self.path и urllib.parse.parse_qs.

Для шаблонов используется библиотека Jinja2. Инициализация Environment:

env = Environment(loader=FileSystemLoader("templates"))

Далее функция show(filename, **kwargs) загружает нужный шаблон и рендерит его.

Функция get_currencies() получает XML с официального сайта ЦБ РФ, разбирает его и возвращает словарь с актуальными курсами валют.

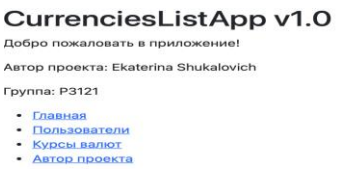
Основные шаги:

1. Выполняется GET-запрос.
2. XML парсится через xml.etree.ElementTree.
3. Из XML извлекаются только нужные валюты (на которые подписаны пользователи).
4. В случае ошибок (плохой JSON, отсутствие валют, сбой сети) выбрасываются исключения.

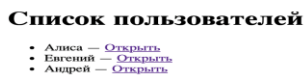
Эти данные затем выводятся на странице пользователя и по маршруту /currencies.

Скриншоты страниц:

/



/users



/user?id=... (Пример одного из пользователей)

Для отображения графиков изменения курса валют за последние 3 месяца использовались следующие библиотеки:

1. Requests (для получения данных)
2. Matplotlib (для построения графика. Ox - дата, Oy - значение курса)
3. Datetime (для вычисления последних 90 дней и преобразования в удобный для чтения графиков формат представления полученных данных)

Тестирование:

Пример теста для моделей (если имя не задано или задано неверно):

```
class TestModels(unittest.TestCase):

 def test_author_getters_setters(self):
    a = Author("Ekaterina", "P3121")
    self.assertEqual(a.name, "Ekaterina")
    self.assertEqual(a.group, "P3121")

    a.name = "Anna"
    a.group = "T1101"
    self.assertEqual(a.name, "Anna")
    self.assertEqual(a.group, "T1101")

 def test_author_invalid_name(self):
    with self.assertRaises(ValueError):
       Author("", "P3121")
```

Пример теста для контроллера (у пользователя отсутствует ID):

```
class TestController(unittest.TestCase):
 def test_user_without_id(self):
    r = requests.get("http://localhost:8001/user")
    self.assertEqual(r.status_code, 200)
    self.assertIn("Не указан id", r.text)
```

Пример теста для функции get_currencies с использованием unittest.mock (неверный XML):

```
class TestCurrenciesAPI(unittest.TestCase):
 @patch("requests.get")
 def test_invalid_xml(self, mock_get):
    mock_get.return_value.status_code = 200
    mock_get.return_value.content = b"<invalid>. </xml>"
    with self.assertRaises(ValueError):
       get_currencies()
```

Выводы:

Проблемы, возникшие в ходе работы:

Несовпадение кодов валют с ID ЦБ РФ. Из-за этого у пользователя отсутствовала подписка на валюты, хотя она была прописана.

Применение MVC:

Принципы MVC были применены при реализации моделей (Author, Currency и т.д.), при использовании шаблонизатора Jinja2 и при обработке запрос с помощью HTTPServer и BaseHTTPRequestHandler.

Полученные знания:

Принцип работы шаблонов Jinja2 и их рендеринг, работа с принципами MVC и их применение в реальном проекте, обработка ошибок при работе с API.