

Отчет по лабораторной работе 9

Тема: Реализация MVC-архитектуры и CRUD-операций в веб-приложении на Python.

Выполнила: Чикомазова Алиса 504497

1. Цель работы

Реализовать полноценное веб-приложение, использующее архитектуру MVC (Model-View-Controller). Основные задачи включали:

1. Организацию взаимодействия с базой данных SQLite (в режиме memory) через модуль sqlite3.
2. Реализацию полного набора CRUD-операций (Create, Read, Update, Delete) для сущностей предметной области.
3. Разделение логики приложения на слои: модели данных, контроллеры БД и бизнес-логики, представления (HTML-шаблоны).
4. Настройку маршрутизации (routing) для обработки HTTP-запросов и рендеринга динамических страниц с помощью Jinja2.
5. Написание модульных тестов с использованием библиотеки unittest.mock

2. Описание моделей, их свойств и связей

В приложении разработаны три основные сущности, соответствующие таблицам в базе данных.

Модель User

Представляет пользователя системы.

- **id:** Уникальный идентификатор
- **name:** Имя пользователя

Модель Currency

Хранит информацию о валютах.

- **Id:** Уникальный идентификатор
- **num_code:** Цифровой код валюты
- **char_code:** Буквенный код
- **name:** Название валюты
- **value:** Текущий курс
- **nominal:** Номинал

Модель UserCurrency

Таблица подписок пользователей на валюты. Реализует связь между таблицами user и currency.

- **Id:** Идентификатор записи
- **user_id:** Внешний ключ, ссылается на user(id)
- **currency_id:** Внешний ключ, ссылается на currency(id).

3. Структура проекта

Проект организован в соответствии с паттерном MVC:

- **controllers/** — Логика управления
 - databasecontroller.py: Отвечает за прямое выполнение SQL-запросов (слой доступа к данным)
 - currencycontroller.py: Преобразует данные из БД в объекты Python-моделей (бизнес-логика)
- **models/** — Классы данных (User, Currency, Author и др.)
- **templates/** — HTML-шаблоны Jinja2
- **static/** — CSS стили
- **utils/** — Вспомогательные модули (парсинг API ЦБ РФ)
- **myapp.py** — Точка входа. Здесь инициализируется сервер, контроллеры и происходит маршрутизация запросов
- **tests.py** — Модульные тесты

4. Реализация CRUD и примеры SQL-запросов

Create (Создание):

Добавление новой валюты

```
# CRUD для Валют
def create_currency(self, num_code, char_code, name, value, nominal): 1 usage
    sql = """
    INSERT INTO currency(num_code, char_code, name, value, nominal)
    VALUES(:num_code, :char_code, :name, :value, :nominal)
    """
    self.cursor.execute(sql, parameters: {
        'num_code': num_code,
        'char_code': char_code,
        'name': name,
        'value': value,
        'nominal': nominal
    })
    self.conn.commit()
    return self.cursor.lastrowid
```

Read (Чтение):

Получение списка валют или поиск по ID

Чтение всех валют

```
def get_all_currencies(self): 1 usage
    self.cursor.execute("SELECT * FROM currency")
    return self.cursor.fetchall()
```

Чтение подписок пользователя (JOIN запрос)

```
def get_user_subscriptions(self, user_id): 1 usage
    """JOIN запрос для получения валют пользователя"""
    sql = """
        SELECT c.* FROM currency c
        JOIN user_currency uc ON c.id = uc.currency_id
        WHERE uc.user_id = ?
    """
    self.cursor.execute(sql, parameters: (user_id,))
    return self.cursor.fetchall()
```

Update (Обновление): Изменение курса валюты.

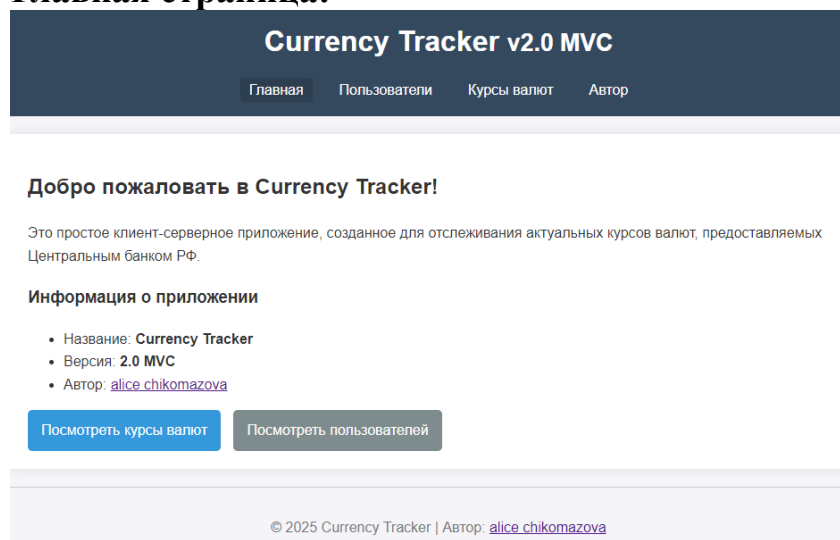
```
def update_currency_value(self, char_code, new_value): 1 usage
    sql = "UPDATE currency SET value = ? WHERE char_code = ?"
    self.cursor.execute(sql, parameters: (new_value, char_code))
    self.conn.commit()
```

Delete (Удаление): Удаление валюты по ID.

```
def delete_currency(self, currency_id): 1 usage
    sql = "DELETE FROM currency WHERE id = ?"
    self.cursor.execute(sql, parameters: (currency_id,))
    self.conn.commit()
```

5. Скриншоты работы приложения

1. Главная страница:



2. Таблица валют:

Currency Tracker v2.0 MVC

[Главная](#)[Пользователи](#)[Курсы валют](#)[Автор](#)

Актуальные курсы валют ЦБ РФ

Курсы обновляются при каждой загрузке этой страницы.

Код	Название	Номинал	Курс к RUB
AED	Дирхам ОАЭ	1	20.7199 RUB
AMD	Армянских драмов	100	19.9459 RUB
AUD	Австралийский доллар	1	50.374 RUB
AZN	Азербайджанский манат	1	44.761 RUB
BDT	Так	100	62.2032 RUB
BGN	Болгарский лев	1	45.3879 RUB
BHD	Бахрейнский динар	1	202.3331 RUB
BOB	Боливiano	1	11.0121 RUB
BRL	Бразильский реал	1	14.3711 RUB
BYN	Белорусский рубль	1	26.4517 RUB
CAD	Канадский доллар	1	54.5396 RUB
CHF	Швейцарский франк	1	94.7736 RUB
CNY	Юань	1	10.7328 RUB
CUP	Кубинских песо	10	31.7057 RUB
NOK	Норвежских крон	10	75.3448 RUB
NZD	Новозеландский доллар	1	43.8642 RUB
OMR	Оманский риал	1	197.903 RUB
PLN	Злотый	1	20.9549 RUB
QAR	Катарский риал	1	20.9049 RUB
RON	Румынский лей	1	17.41 RUB
RSD	Сербских динаров	100	75.5778 RUB
RUB	Российский рубль	1	1.0 RUB
SAR	Саудовский риял	1	20.2917 RUB
SEK	Шведских крон	10	80.9437 RUB
SGD	Сингапурский доллар	1	58.7097 RUB
THB	Батов	10	23.7815 RUB
TJS	Сомони	10	82.2937 RUB
TMT	Новый туркменский манат	1	21.7411 RUB
TRY	Турецких лир	10	17.9402 RUB
UAH	Гривен	10	18.0386 RUB
USD	Доллар США	1	76.0937 RUB
UZS	Узбекских сумов	10000	63.6063 RUB
VND	Донгов	10000	30.2547 RUB
XDR	СДР (специальные права заимствования)	1	103.8945 RUB
ZAR	Рэндов	10	44.9413 RUB

3. Пример обновления:

/currency/update?USD=100

SGD	Сингапурский доллар	1	58.7097 RUB
THB	Батов	10	23.7815 RUB
TJS	Сомони	10	82.2937 RUB
TMT	Новый туркменский манат	1	21.7411 RUB
TRY	Турецких лир	10	17.9402 RUB
UAH	Гривен	10	18.0386 RUB
USD	Доллар США	1	100.0 RUB
UZS	Узбекских сумов	10000	63.6063 RUB
VND	Донгов	10000	30.2547 RUB
XDR	СДР (специальные права заимствования)	1	103.8945 RUB
ZAR	Рэнд	10	44.9413 RUB

Результат удаления:

46	UAH	Гривен	10	18.0386 RUB	Удалить
17	USD	Доллар США	1	76.0937 RUB	Удалить
45	UZS	Узбекских сумов	10000	63.6063 RUB	Удалить
46	UAH	Гривен	10	18.0386 RUB	Удалить
45	UZS	Узбекских сумов	10000	63.6063 RUB	Удалить

6. Тестирование с unittest.mock

Для проверки бизнес-логики использовался модуль unittest и MagicMock. Это позволило протестировать работу контроллера в изоляции от реальной базы данных.

```
class TestCurrencyController(unittest.TestCase):
    def test_list_currencies(self):
        mock_db = MagicMock()
        mock_db.get_all_currencies.return_value = [
            {
                "id": 1,
                "num_code": "840",
                "char_code": "USD",
                "name": "Dollar",
                "value": 90.5,
                "nominal": 1
            }
        ]
        controller = CurrencyController(mock_db)
        result = controller.get_currencies_list()
        self.assertEqual(len(result), 1)
        self.assertEqual(result[0].char_code, "USD")
        self.assertEqual(result[0].value, 90.5)
        # Проверяем, что метод БД действительно вызывался
        mock_db.get_all_currencies.assert_called_once()
```

```
def test_add_currency_check(self):
    """Тест, что валюта не добавляется, если она уже есть"""
    mock_db = MagicMock()
    mock_db.get_currency_by_char_code.return_value = {"id": 1}
    controller = CurrencyController(mock_db)
    currency_mock = MagicMock()
    currency_mock.char_code = "USD"
    controller.add_currency_from_api(currency_mock)
    # create_currency НЕ должен вызываться
    mock_db.create_currency.assert_not_called()
```

```
✓ Test Results 2 ms    ✓ Tests passed: 2 of 2 tests – 2 ms
C:\Users\user\PythonProject8\.venv\Sc
Testing started at 15:15 ...
Launching unittests with arguments py

Ran 2 tests in 0.003s

OK

Process finished with exit code 0
```

7. Выводы

В ходе выполнения работы были освоены ключевые принципы веб-разработки на Python:

1. **Архитектура MVC:** Разделение кода на Модели, Контроллеры и Шаблоны значительно упростили структуру проекта. Логика работы с базой данных отделена от логики отображения HTML.
2. **Работа с SQLite:** Использование базы данных в памяти (memory) позволило быстро прототипировать приложение без создания лишних файлов на диске. Параметризованные запросы обеспечили безопасность.
3. **Маршрутизация:** Ручная обработка путей в методе `do_GET` дала понимание того, как работают веб-фреймворки (разбор URL, параметров запроса).
4. **Шаблонизация:** Использование Jinja2 позволило создавать динамические HTML-страницы, подставляя данные из контроллеров, что делает интерфейс гибким.