

**Министерство науки и высшего образования Российской Федерации**  
**федеральное государственное автономное образовательное учреждение высшего образования**  
**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

<b>Факультет</b>	<b>Программной Инженерии и Компьютерной Техники</b>
<b>Направление подготовки (специальность)</b>	<b>Нейротехнологии и программирование</b>
<b>Дисциплина</b>	<b>Программирование на Python</b>

**Лабораторная работа 9**  
**ОТЧЕТ**

<b>Выполнил студент:</b>	<b>Иголкин Владислав Андреевич (504623)</b>
<b>Группа:</b>	<b>P3124</b>
<b>Преподаватель:</b>	<b>Жуков Николай Николаевич (261087)</b>

г. Санкт-Петербург  
2025 г.

## 1. Цель работы:

- Реализовать **CRUD** (Create, Read, Update, Delete) для сущностей бизнес-логики приложения.
- Освоить работу с **SQLite** в памяти (:memory:) через модуль `sqlite3`.
- Понять принципы **первичных и внешних ключей** и их роль в связях между таблицами.
- Выделить **контроллеры** для работы с БД и для рендеринга страниц в отдельные модули.
- Использовать архитектуру **MVC** и соблюдать разделение ответственности.
- Отображать пользователям таблицу с валютами, на которые они подписаны.
- Реализовать полноценный **роутер**, который обрабатывает GET-запросы и выполняет сохранение/обновление данных и рендеринг страниц.
- Научиться тестировать функционал на примере сущностей `currency` и `user` с использованием **`unittest.mock`**.

## 2. Описание моделей, их свойств и связей

Модель `Currency`:

Поле	Тип	Описание
<code>id</code>	<code>INTEGER</code>	Уникальный идентификатор
<code>name</code>	<code>TEXT</code>	Название валюты
<code>code</code>	<code>TEXT</code>	Символьный код валюты (USD, EUR, GBP...)
<code>rate</code>	<code>REAL</code>	Курс валюты относительно базовой
<code>last_update</code>	<code>TEXT</code>	Дата и время последнего обновления курса

Связей с другими таблицами нет, так как приложение содержит только одну сущность валют.

## 2. Описание моделей, их свойств и связей

Модели:

`Currency`

Поле	Тип	Описание
<code>id</code>	<code>INTEGER</code>	Уникальный идентификатор
<code>num_code</code>	<code>TEXT</code>	Цифровой код валюты
<code>char_code</code>	<code>TEXT</code>	Символьный код валюты (USD...)
<code>name</code>	<code>TEXT</code>	Название валюты
<code>value</code>	<code>FLOAT</code>	Курс валюты
<code>nominal</code>	<code>INTEGER</code>	Номинал валюты

`User`

Поле	Тип	Описание
<code>id</code>	<code>INTEGER</code>	Уникальный идентификатор
<code>name</code>	<code>TEXT</code>	Имя пользователя

Связь между моделями: `user_currency` — связь «многие ко многим» между пользователями и валютами.

## 3. Структура проекта с назначением файлов

`project/`

<code>myapp.py</code>	# Основной файл приложения, маршруты, запуск сервера
<code>models/</code>	

```

|   |— __init__.py
|   |— currency.py      # Модель Currency
|   |— user.py          # Модель User
|   |— author.py        # Модель Author
|   |— controllers/
|       |— __init__.py
|       |— databasecontroller.py  # CRUD для Currency и User
|       |— currencycontroller.py  # Контроллер для валют
|       |— usercontroller.py      # Контроллер для пользователей
|       |— cbr_api.py            # Подключение к API ЦБ
|   |— templates/
|       |— index.html
|       |— currencies.html
|       |— author.html
|       |— users.html
|       |— user.html
|   |— static/
|       |— style.css
|   |— tests/
|       |— test_myapp.py        # Тесты с unittest и unittest.mock

```

## 4. Реализация CRUD с примерами SQL-запросов

### 4.1 Создание валюты (Create)

```

data = {
    "num_code": "840",
    "char_code": "USD",
    "name": "Доллар США",
    "value": 90.0,
    "nominal": 1
}
db_controller._create([data])

```

SQL-запрос:

```

INSERT INTO currency(num_code, char_code, name, value, nominal)
VALUES ('840', 'USD', 'Доллар США', 90.0, 1);

```

### 4.2 Чтение валют (Read)

```

currencies = db_controller._read()

```

SQL-запрос:

```

SELECT * FROM currency;

```

### 4.3 Обновление курса валюты (Update)

```

db_controller._update({"USD": 91.5})

```

SQL-запрос:

```

UPDATE currency SET value = 91.5 WHERE char_code = 'USD';

```

#### 4.4 Удаление валюты (Delete)

db\_controller.\_delete(1) # по id

SQL-запрос:

DELETE FROM currency WHERE id = 1;

#### 4.5 CRUD для пользователей

user\_id = db\_controller.create\_user("Игорь")

users = db\_controller.read\_users()

db\_controller.assign\_currency\_to\_user(user\_id, 1)

user\_currencies = db\_controller.get\_user\_currencies(user\_id)

### 5. Примеры работы приложения (главная страница, таблица валют, обновление и удаление).

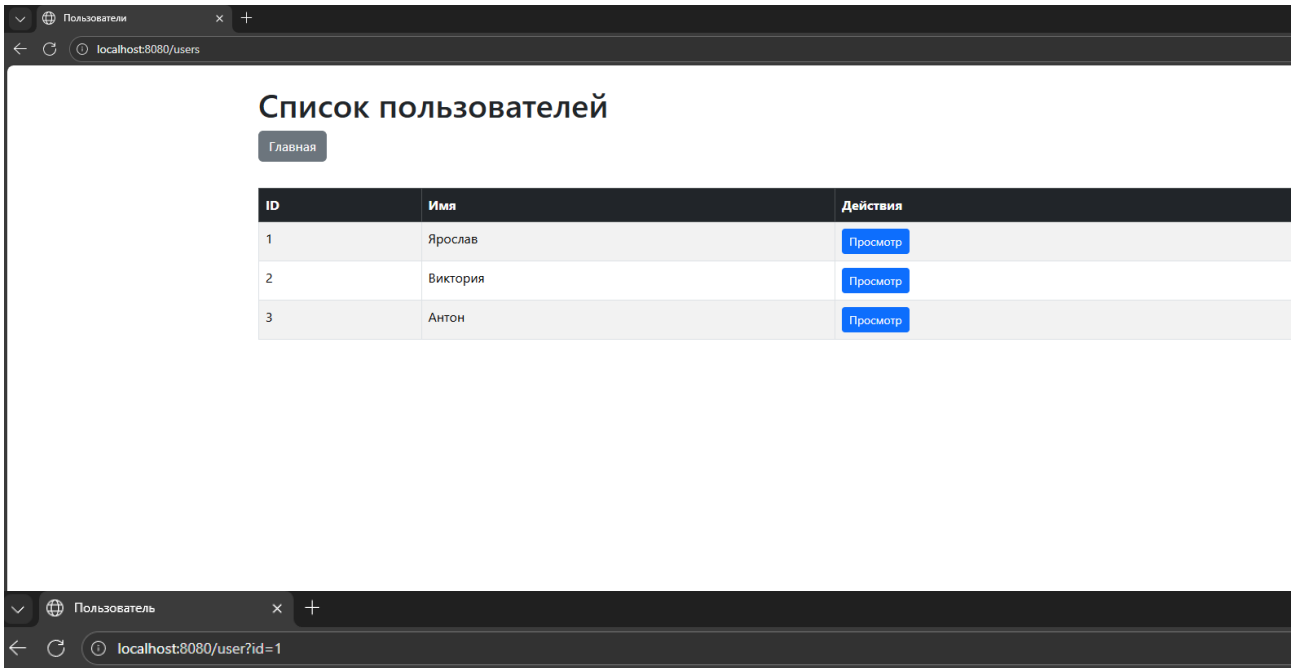
The screenshot displays two views of the 'CurrenciesListApp' web application. The top view is the main page, which includes a header with the app name, author information, and navigation links. Below the header is a table titled 'Валюты' (Currencies) with columns for ID, NumCode, CharCode, Name, Value, Nominal, and Действия (Actions). The table lists four currencies: USD, EUR, GBP, and AUD. Each row has a red 'Удалить' (Delete) button in the Actions column. The bottom view is the 'Информация об авторе' (About Author) page, which displays the author's name and group, along with a blue button labeled 'На главную' (Back to Home).

**CurrenciesListApp**  
Автор: Иголкин Владислав 504623  
Группа: P3124  
[Главная](#) [Об авторе](#) [Пользователи](#)

**Валюты**

ID	NumCode	CharCode	Name	Value	Nominal	Действия
1	840	USD	USD	76.0382	1	<a href="#">Удалить</a>
2	978	EUR	EUR	88.7898	1	<a href="#">Удалить</a>
3	826	GBP	GBP	102.1041	1	<a href="#">Удалить</a>
4	036	AUD	AUD	51.744	1	<a href="#">Удалить</a>

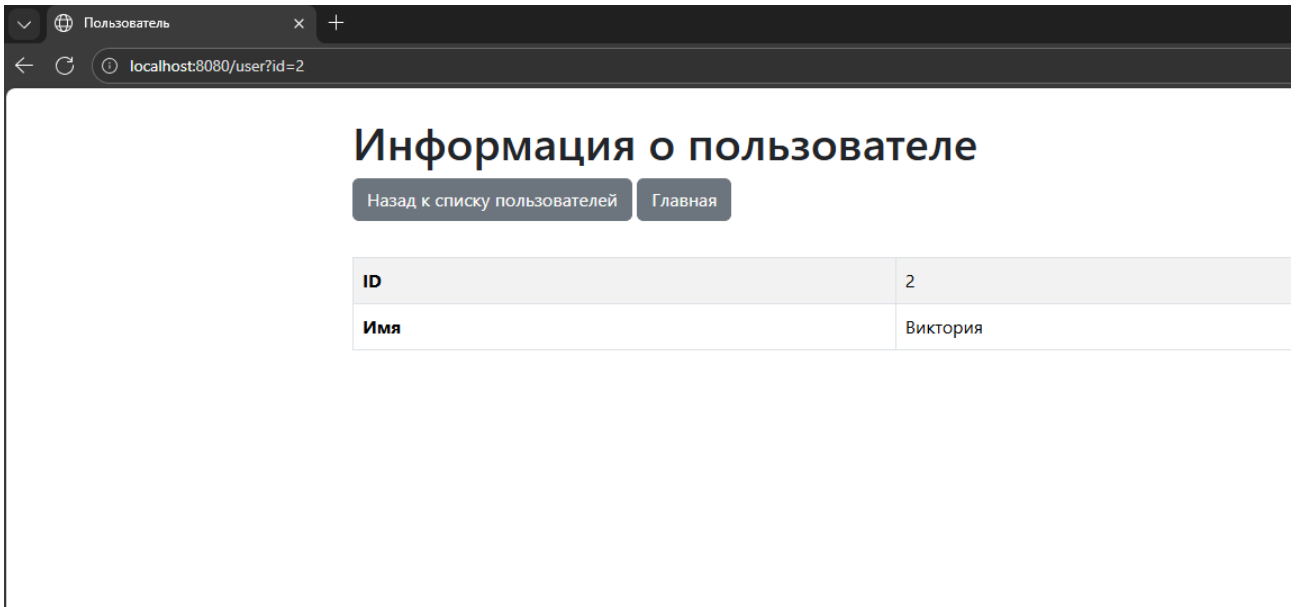
**Информация об авторе**  
Имя: Иголкин Владислав 504623  
Группа: P3124  
[На главную](#)



## Информация о пользователе

[Назад к списку пользователей](#) [Главная](#)

ID	1
Имя	Ярослав



# Информация о пользователе

Назад к списку пользователей

Главная

ID	3
Имя	АНТОН

## Все валюты

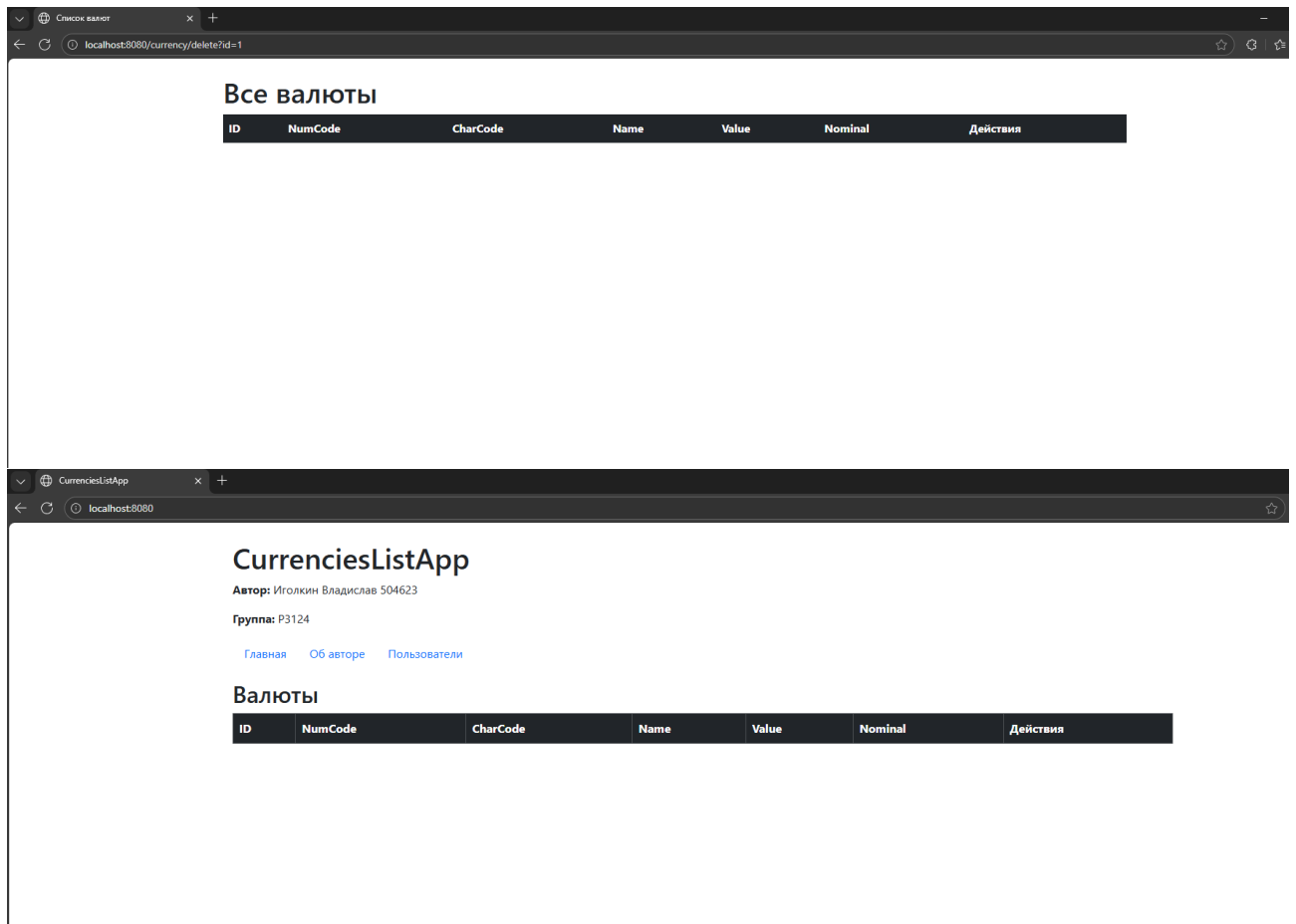
ID	NumCode	CharCode	Name	Value	Nominal	Действия
1	840	USD	USD	76.0382	1	<div>Удалить</div>
2	978	EUR	EUR	88.7898	1	<div>Удалить</div>
3	826	GBP	GBP	102.1041	1	<div>Удалить</div>

## Все валюты

ID	NumCode	CharCode	Name	Value	Nominal	Действия
1	840	USD	USD	76.0382	1	<div>Удалить</div>
2	978	EUR	EUR	88.7898	1	<div>Удалить</div>

## Все валюты

ID	NumCode	CharCode	Name	Value	Nominal	Действия
1	840	USD	USD	76.0382	1	<div>Удалить</div>



## 6. Примеры тестов с unittest.mock и результаты их выполнения.

```
import sys
import os
sys.path.append(os.path.dirname(os.path.abspath(__file__)))
import unittest
from unittest.mock import MagicMock
from .controllers.currencycontroller import CurrencyController
from .controllers.usercontroller import UserController

class TestCurrencyController(unittest.TestCase):
    def setUp(self):
        self.mock_db = MagicMock()
        self.controller = CurrencyController(self.mock_db)

    def test_list_currencies(self):
        self.mock_db._read.return_value = [{"id": 1, "char_code": "USD",
"value": 90}]
        result = self.controller.list_currencies()
        self.assertEqual(result[0]['char_code'], "USD")
        self.mock_db._read.assert_called_once()

    def test_update_currency(self):
        self.controller.update_currency("USD", 100.5)
        self.mock_db._update.assert_called_once_with({"USD": 100.5})

    def test_delete_currency(self):
        self.controller.delete_currency(1)
        self.mock_db._delete.assert_called_once_with(1)

class TestUserController(unittest.TestCase):
```

```

def setUp(self):
    self.controller = UserController()
    self.controller.add_user("Антон")
    self.controller.add_user("Мария")

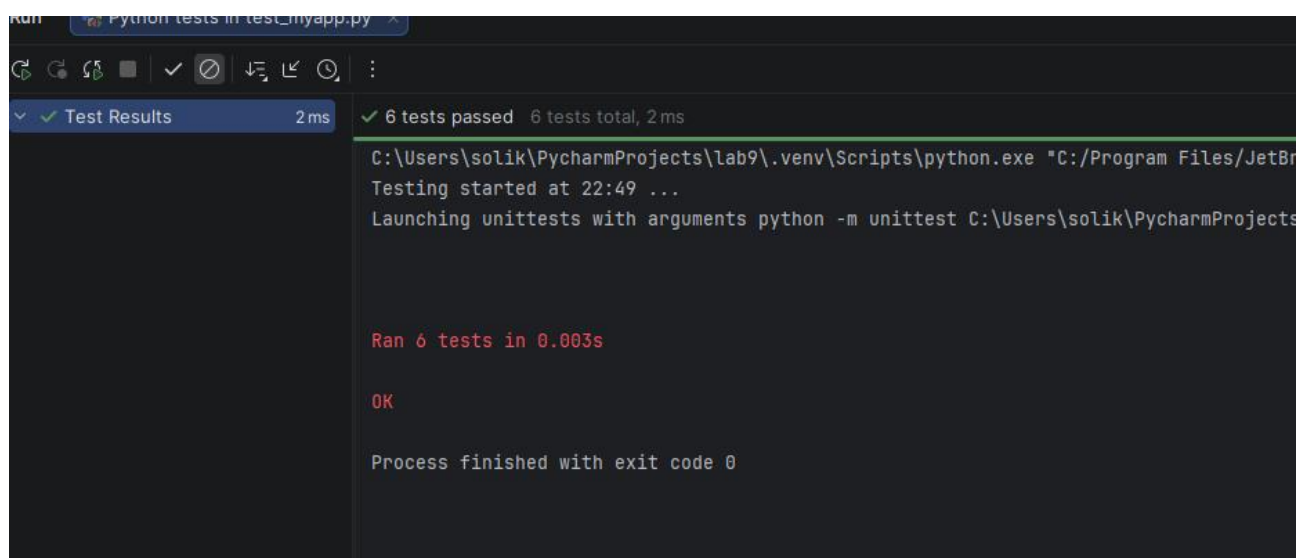
def test_list_users(self):
    users = self.controller.list_users()
    self.assertEqual(len(users), 2)
    self.assertEqual(users[0].name, "Антон")
    self.assertEqual(users[1].name, "Мария")

def test_get_user(self):
    user = self.controller.get_user(1)
    self.assertEqual(user.name, "Антон")
    self.assertIsNone(self.controller.get_user(99))

def test_add_user(self):
    self.controller.add_user("Игорь")
    users = self.controller.list_users()
    self.assertEqual(len(users), 3)
    self.assertEqual(users[2].name, "Игорь")

if __name__ == "__main__":
    unittest.main()

```



## 7. Выводы

- В работе реализован полный CRUD для валют и пользователей.
- Использование SQLite показало работу с базой через parameterized queries для защиты от SQL-инъекций.
- Архитектура MVC позволяет разделять логику (controllers), данные (models) и представление (templates).
- Тесты с unittest.mock позволяют проверять контроллеры без подключения к реальной базе.
- Jinja2 обеспечивает гибкий рендеринг шаблонов HTML с динамическими данными.