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 № 9

Группа: P3121

Выполнила:

Шукалович Екатерина Андреевна

Проверил:

Жуков Николай Николаевич

Санкт-Петербург 2025

Цель работы:

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

Описание моделей, их свойств и связей:

Каждый объект является отдельным классом в /models:

Author

1. **name - имя автора проекта**
2. **group - группа автора проекта**

App

1. **name - название приложения**
2. **version - версия**
3. **author - объект класса Author**

User

1. **ID - первичный ключ**
2. **name - имя пользователя**

Currency

1. **ID - первичный ключ**
2. **num_code - цифровой код**
3. **char_code - символьный код**
4. **name - название валюты**
5. **value - курс**
6. **nominal - номинал**

UserCurrency

1. **user_id - внешний ключ**
2. **currency_id - внешний ключ**

Структура проекта с назначением файлов:

```
lab9crud/
├── models/      # объекты приложения
│  ├── __init__.py
│  ├── author.py
│  ├── app.py
│  ├── user.py
│  ├── currency.py
│  └── user_currency.py
├── templates/   # HTML файлы с шаблонизатором Jinja2
│  ├── index.html
│  ├── users.html
| ├── user.html
| ├── author_project.html
| ├── currencies.html
| ├── currency_edit.html
| ├── currency_add.html
| ├── currency_delete.html
│  └── error.html
├── controllers/  # бизнес-логика для CRUD
│  ├── currency_controller.py
│  ├── subscription_controller.py
│  └── user_controller.py
├── app.py       # главный файл приложения с роутером и шаблонизатором Jinja2
├── db.py        # создает базу данных
└── tests/       # тесты
│  ├── test_user.py
│  └── test_user_currency.py
```

Реализация CRUD и SQL-запросы:

CRUD:

Create:

```
INSERT INTO currency(num_code, char_code, name, value, nominal)
VALUES (?, ?, ?, ?, ?);
```

Read:

```
SELECT id, num_code, char_code, name, value, nominal
FROM currency;
```

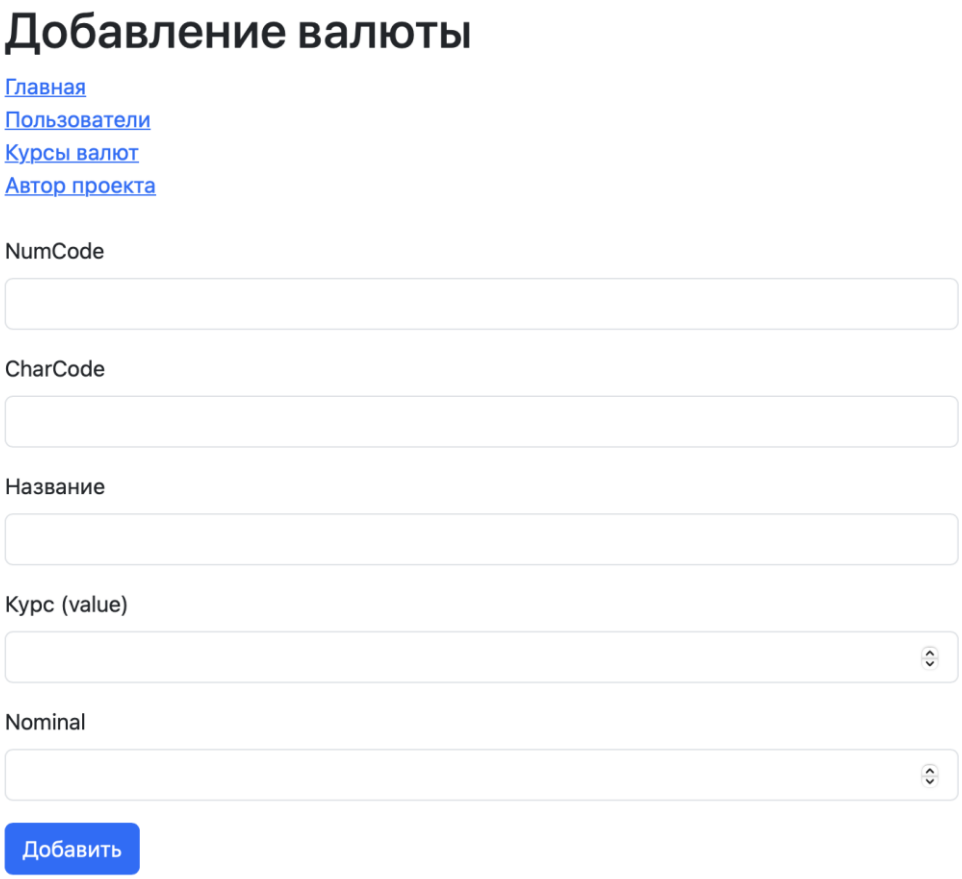
Update:

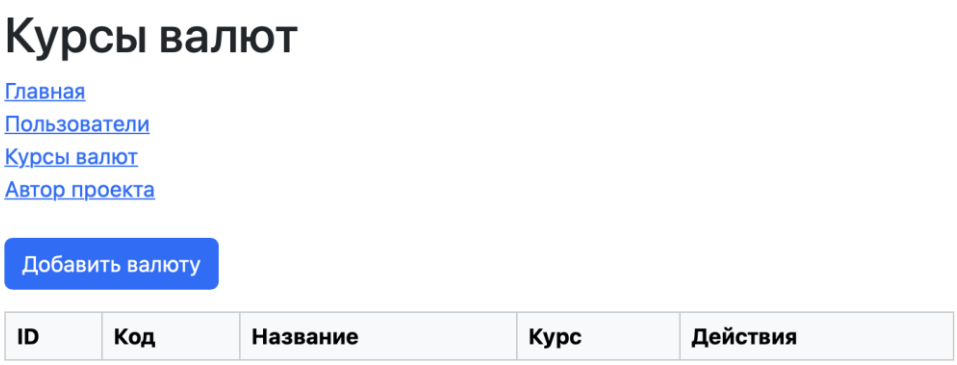
```
UPDATE currency
SET value = ?
WHERE id = ?;
```

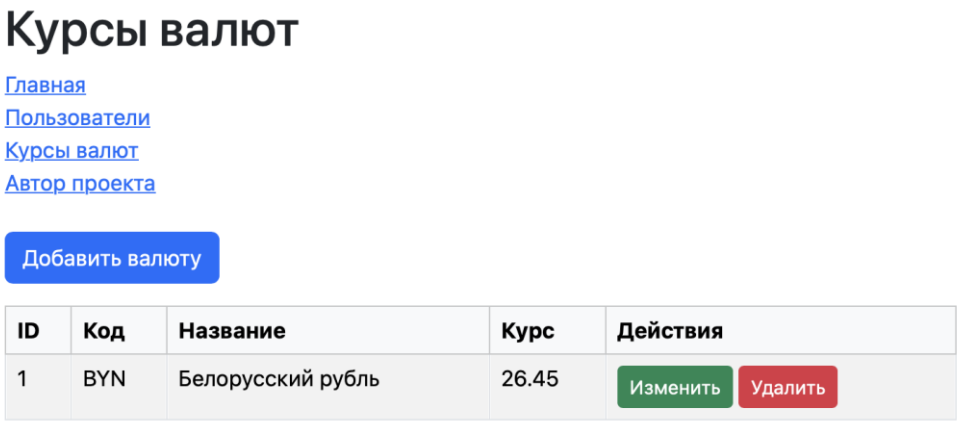
Delete:

```
DELETE FROM currency WHERE id = ?;
```

Скриншоты работы приложения:







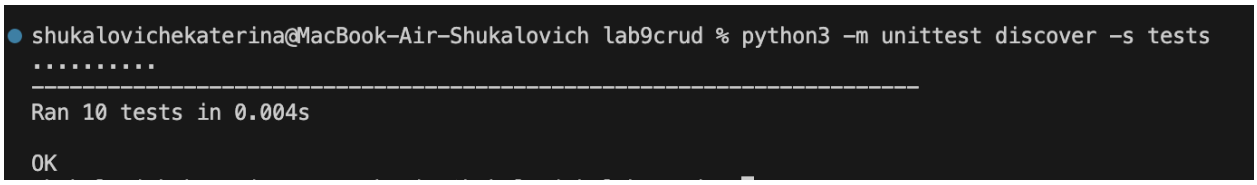
Примеры тестов с unittest.mock и результаты их выполнения:

```
import unittest
from unittest.mock import MagicMock

from controllers.currency_controller import (
add_currency,
get_all_currencies,
update_currency_value,
delete_currency
)
```

```
class TestCurrencyControllerSimple(unittest.TestCase):

def test_add_currency(self):
conn = MagicMock()
cursor = MagicMock()

conn.cursor.return_value = cursor
cursor.lastrowid = 1

cid = add_currency(conn, "840", "USD", "Dollar", 90.0, 1)

# проверяем, что курсор вызывался
cursor.execute.assert_called_once()
conn.commit.assert_called_once()

self.assertEqual(cid, 1)

def test_get_all_currencies(self):
conn = MagicMock()
cursor = MagicMock()

conn.cursor.return_value = cursor
cursor.fetchall.return_value = [
{"id": 1, "char_code": "USD", "value": 90.0}
]

currencies = get_all_currencies(conn)

cursor.execute.assert_called_once()
self.assertEqual(len(currencies), 1)
self.assertEqual(currencies[0]["char_code"], "USD")

def test_update_currency(self):
conn = MagicMock()
cursor = MagicMock()

conn.cursor.return_value = cursor
cursor.rowcount = 1
result = update_currency_value(conn, 1, 95.5)

cursor.execute.assert_called_once()
conn.commit.assert_called_once()
self.assertTrue(result)

def test_delete_currency(self):
```

```
conn = MagicMock()
cursor = MagicMock()

conn.cursor.return_value = cursor
cursor.rowcount = 1

result = delete_currency(conn, 1)

cursor.execute.assert_called_once()
conn.commit.assert_called_once()
self.assertTrue(result)


if __name__ == '__main__':
unittest.main()
```

Результат выполнения:



Выводы:

В ходе выполнения работы была построена мини веб-платформа, основанная на архитектурных принципах MVC, в которой реализованы обработка маршрутов, взаимодействие с базой данных SQLite и рендеринг HTML-страниц через шаблонизатор Jinja2.

Архитектура MVC:

Model: User, Currency, UserCurrency, Author

View: HTML-шаблоны Jinja2

Controller: обработка запросов с помощью функции get_currencies, CRUD

Работа с SQLite:

База данных, создаваемая в оперативной памяти, обеспечивает высокую скорость создания таблиц, выполнения запросов, удобство в тестировании. Все запросы являются параметризованными, что защищает систему от SQL-инъекций.

Обработка маршрутов:

Обработка маршрутов выполняется через self.path, что позволяет включить в программу работу роутера без использования сторонних фреймворков.

Рендеринг шаблонов:

Рендеринг шаблонов осуществляется шаблонизатором Jinja2 через функцию show(), которая загружает шаблон, подставляет параметры и отправляет полученную HTML-страницу.