Лабораторная работа №4:

Цель задания

Разработать приложение для получения курсов валют с сайта Центробанка России, используя объектно-ориентированный подход и архитектурный паттерн MVC (Model-View-Controller).

Требования к функционалу

1. Компонент Model:
    - Представляет собой класс CurrencyRates, который отправляет запросы к сайту ЦБ РФ.
    - На вход принимает список отслеживаемых валют (строки длины три, например, ["USD", "EUR", "GBP"]).
    - Реализует паттерн "Одиночка" (Singleton), чтобы в приложении существовал только один объект данного класса.
    - Использует геттер для получения текущих курсов валют.
    - Использует сеттер для обновления списка отслеживаемых валют.
2. Компонент View:
    - Использует шаблонизатор Jinja2 или какую-либо другую библиотеку (пример) для отображения данных.
    - Позволяет пользователю просматривать курсы валют в удобном формате.
3. Компонент Controller:
    - Обрабатывает запросы и управляет логикой приложения.
    - Сохраняет полученные данные в базу данных SQLite3 (реализован именованный стиль для параметризации запросов в БД).
    - Передаёт данные в представление для отображения.

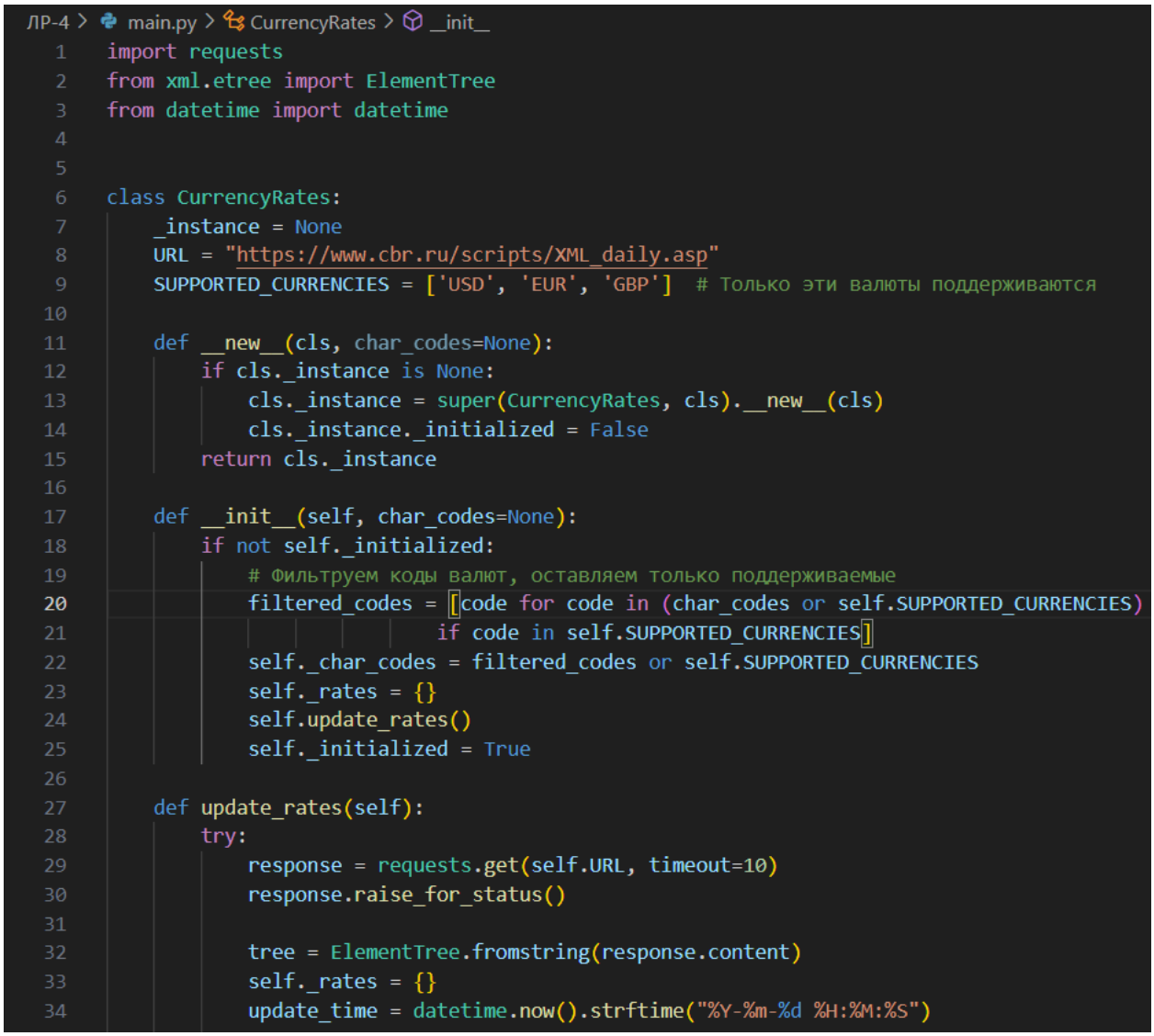
Дополнительные требования

- Данные должны загружаться с официального API или XML-файла Центробанка России.
- Для взаимодействия с API можно использовать библиотеку requests.
- Использование библиотек sqlite3 для работы с базой данных и jinja2 для шаблонов.

Этапы выполнения

1. Реализовать класс CurrencyRates (паттерн "Одиночка").
2. Реализовать контроллер для обработки запросов и сохранения данных в SQLite3.
3. Разработать шаблоны на Jinja2 для отображения информации.
4. Написать код для интеграции всех компонентов в MVC-структуре.
5. Протестировать приложение, проверив работу с разными валютами.

main.py:



```
            for valute in tree.findall('.//Valute'):
                char_code = valute.find('CharCode').text
                if char_code in self._char_codes:
                    value = float(valute.find('Value').text.replace(',', '.'))
                    nominal = int(valute.find('Nominal').text)
                    self._rates[char_code] = {
                        'value': round(value / nominal, 4),
                        'name': valute.find('Name').text,
                        'date': update_time
                    }
            return True
        except Exception as e:
            print(f"Ошибка при загрузке курсов: {e}")
            return False

    def get_all_rates(self):
        return [
            (code, info['name'], info['value'], info['date'])
            for code, info in self._rates.items()
        ]

    @property
    def char_codes(self):
        return self._char_codes

    @char_codes.setter
    def char_codes(self, new_codes):
        # Фильтруем новые коды валют
        filtered_codes = [code for code in new_codes if code in self.SUPPORTED_CURRENCIES]
        if filtered_codes:
            self._char_codes = filtered_codes
            self.update_rates()
```

__init__.py:

ЛР-4 > controllers > __init__.py > CurrencyRatesCRUD

```python
import sqlite3
from datetime import datetime


class CurrencyRatesCRUD:
    def __init__(self, currency_rates_obj):
        self.__connection = sqlite3.connect('currency_rates.db')
        self.cursor = self.__connection.cursor()
        self.currency_rates = currency_rates_obj
        self._create_table()

    def _create_table(self):
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS currency_rates (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                char_code TEXT NOT NULL,
                name TEXT NOT NULL,
                value REAL NOT NULL,
                date TEXT NOT NULL
            )
        """)
        self.__connection.commit()

    def create(self, data=None):
        try:
            if data is None:
                raw_data = self.currency_rates.get_all_rates()
                data = [
                    {
                        'char_code': code,
                        'name': name,
                        'value': value,
                        'date': date
                    }
                    for code, name, value, date in raw_data
                ]

            if not data:
                print("Нет данных для записи")
                return False
```

```
            sql = """
                INSERT INTO currency_rates
                (char_code, name, value, date)
                VALUES (:char_code, :name, :value, :date)
            """

            self.cursor.executemany(sql, data)
            self.__connection.commit()
            print(f"Успешно записано {len(data)} записей")
            return True

        except Exception as e:
            print(f"Ошибка при записи в БД: {e}")
            self.__connection.rollback()
            return False

    def read(self, char_code=None):
        try:
            if char_code:
                self.cursor.execute("""
                    SELECT char_code, name, value, date
                    FROM currency_rates
                    WHERE char_code = ?
                    ORDER BY date DESC
                """, (char_code,))
            else:
                self.cursor.execute("""
                    SELECT char_code, name, value, date
                    FROM currency_rates
                    ORDER BY date DESC
                """)
            return self.cursor.fetchall()
        except sqlite3.Error as e:
            print(f"Ошибка при чтении из БД: {e}")
            return []
```

```
78      def update_rates(self):
79          try:
80              if not self.currency_rates.update_rates():
81                  return False
82              return self.create()
83          except Exception as e:
84              print(f"Ошибка при обновлении данных: {e}")
85              return False
86
87      def close(self):
88          try:
89              self.__connection.close()
90          except sqlite3.Error as e:
91              print(f"Ошибка при закрытии соединения: {e}")
92
93
94  class ViewController:
95      def __init__(self, currency_rates):
96          self.currency_rates = currency_rates
97
98      def __call__(self):
99          rates = self.currency_rates.get_all_rates()
100         if not rates:
101             return "Нет данных о курсах валют"
102
103         result = ["Текущие курсы валют:"]
104         for code, name, value, date in rates:
105             result.append(f"{code} ({name}): {value} руб. (на {date})")
106
107         return "\n".join(result)
```

app.py:

ЛР-4 > app.py > ...

```python
from main import CurrencyRates
from controllers import CurrencyRatesCRUD, ViewController


def main():
    # Инициализация с поддержкой только USD, EUR, GBP
    c_r = CurrencyRates(['USD', 'EUR', 'GBP'])

    # Обновляем курсы валют
    if not c_r.update_rates():
        print("Не удалось обновить курсы валют")
        return

    # Инициализация контроллера БД
    crud = CurrencyRatesCRUD(c_r)

    # Запись данных в БД
    if not crud.create():
        print("Не удалось записать данные в БД")
        return

    # Вывод информации
    view = ViewController(c_r)
    print(view())

    # Закрытие соединения с БД
    crud.close()

if __name__ == "__main__":
    main()
```

Результат:

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс
313/python.exe "c:/Users/sambu/OneDrive/Рабочий стол/Учеба в Ге
Успешно записано 3 записей
Текущие курсы валют:
GBP (Фунт стерлингов): 106.7723 руб. (на 2025-06-12 22:14:00)
USD (Доллар США): 79.0028 руб. (на 2025-06-12 22:14:00)
EUR (Евро): 90.0068 руб. (на 2025-06-12 22:14:00)
```