**"Прокси Чекер"**

**1. Техническое задание**

**1.1. Цель проекта**

Целью проекта является разработка приложения для проверки работоспособности прокси-серверов с использованием Python и PyQt5. Приложение должно предоставлять удобный графический интерфейс для взаимодействия пользователя, а также возможность сохранять результаты проверки в базе данных SQLite.

**1.2. Функциональные требования**

1. **Графический интерфейс**:
   * Окно для загрузки списка прокси из файла.
   * Кнопки для запуска проверки и сохранения результатов.
   * Таблица для отображения состояния прокси (IP, порт, статус, страна).
   * Поле для выбора протокола (HTTP, SOCKS4, SOCKS5).
2. **Логика проверки прокси**:
   * Проверка доступности прокси через заданный протокол.
   * Определение страны по IP с использованием внешнего API.
3. **Работа с базой данных**:
   * Сохранение результатов проверки в SQLite.
   * Автоматическое создание базы данных при первом запуске приложения.
4. **Экспорт данных**:
   * Сохранение рабочих прокси в текстовый файл.

**1.3. Нефункциональные требования**

* Приложение должно работать на Windows 10/11.
* Высокая производительность при работе с большими файлами (до 10 000 записей).
* Простая настройка без необходимости вручную создавать базу данных.

**2. Структура проекта**

**2.1. Основные модули**

1. **gui.py**:
   * Реализация графического интерфейса с использованием PyQt5.
   * Обработка событий (клик по кнопкам, загрузка файла, выбор протокола).
2. **proxy\_checker.py**:
   * Логика проверки доступности прокси.
   * Подключение через HTTP, SOCKS4, SOCKS5.
   * Определение страны прокси через API.
3. **database.py**:
   * Подключение к базе данных SQLite.
   * Автоматическое создание базы и таблиц при запуске.
   * Сохранение и выборка данных.
4. **exporter.py**:
   * Логика экспорта рабочих прокси в файл.
5. **tests/**:
   * Набор тестов для проверки функциональности проекта с использованием PyTest.
6. **requirements.txt**:
   * Список всех необходимых зависимостей для работы проекта.

**2.2. Логика взаимодействия**

1. Пользователь загружает файл с прокси через интерфейс.
2. Выбирает протокол и нажимает кнопку "Проверить".
3. Модуль proxy\_checker.py обрабатывает список, проверяет доступность каждого прокси и определяет страну.
4. Результаты передаются обратно в gui для отображения в таблице.
5. Пользователь может сохранить результаты в базу данных через database.py или экспортировать рабочие прокси через exporter.py.

**3. Документация**

**3.1. Установка**

**Системные требования**

* Python 3.10+
* SQLite (встроено в Python).
* ОС: Windows 10/11.

**Шаги установки**

1. Установите Python.
2. Клонируйте репозиторий:
3. git clone <ссылка на репозиторий>
4. Откройте папку с проктом
5. Перейдите в папку «dist».
6. Запустите приложение: proxy\_checker\_gui

**3.2. Использование**

**Интерфейс пользователя**

1. Загрузите файл с прокси (формат: IP:PORT).
2. Выберите протокол (HTTP, SOCKS4, SOCKS5).
3. Нажмите "Проверить прокси".
4. После проверки:
   * Рабочие прокси отображаются в таблице.
   * Статус нерабочих прокси помечается как "Failed".
5. Нажмите "Сохранить в базу" для записи данных в SQLite.
6. Нажмите "Экспортировать рабочие", чтобы сохранить рабочие прокси в файл.

**Доступ к данным из базы SQLite**

1. Убедитесь, что файл sqlite3.exe установлен и доступен на вашем компьютере.
2. Выполните следующие шаги для просмотра данных из базы:
3. C:\path\to\sqlite3.exe C:\PythonProject2\dist\proxies.db
4. Введите команды:
5. .tables
6. SELECT \* FROM proxies;
7. Для выхода из sqlite3 введите:
8. .exit

**3.3. Логика работы кода**

1. **Инициализация базы данных**:
   * При запуске приложения вызывается функция setup\_database, которая автоматически создаёт файл базы данных proxies.db и таблицу proxies (если они ещё не существуют).
2. def setup\_database():
3. conn = sqlite3.connect(DB\_NAME)
4. cursor = conn.cursor()
5. cursor.execute('''CREATE TABLE IF NOT EXISTS proxies (
6. id INTEGER PRIMARY KEY AUTOINCREMENT,
7. ip\_port TEXT NOT NULL,
8. country TEXT NOT NULL
9. )''')
10. conn.commit()
11. conn.close()

Эта функция проверяет наличие базы данных, а если её нет, создаёт таблицу proxies с полями для идентификатора, IP-адреса и страны.

1. **Графический интерфейс**:
   * Реализован в модуле gui.py.
   * Основной интерфейс включает возможность загрузки файла, выбора протокола и отображения результатов.
2. self.protocol\_combo = QComboBox()
3. self.protocol\_combo.addItems(["http", "socks4", "socks5"])

Поле выбора протокола предоставляет пользователю опции проверки через разные типы прокси.

1. **Проверка прокси**:
   * Логика проверки описана в функции check\_proxy.
   * Проверяются прокси через HTTP, SOCKS4 и SOCKS5. Для каждого типа протокола используется соответствующий метод подключения.
2. def check\_proxy(proxy, protocol):
3. try:
4. if protocol == "http":
5. proxies = {"http": f"http://{proxy}", "https": f"http://{proxy}"}
6. response = requests.get("https://httpbin.org/ip", proxies=proxies, timeout=5)
7. if response.status\_code == 200:
8. return "Working"
9. elif protocol in ["socks4", "socks5"]:
10. socks.set\_default\_proxy(socks.SOCKS4 if protocol == "socks4" else socks.SOCKS5, proxy)
11. socket.socket = socks.socksocket
12. return "Working"
13. except Exception:
14. return "Failed"

Эта функция проверяет каждый прокси, создавая запросы через указанный протокол. Рабочие прокси сохраняются в базе данных.

1. **Определение страны**:
   * Используется API http://ip-api.com/json/ для получения данных о стране по IP.
2. def get\_country\_by\_ip(ip):
3. response = requests.get(f"http://ip-api.com/json/{ip}")
4. data = response.json()
5. return data.get("country", "Unknown")
6. **Экспорт данных**:
   * Рабочие прокси сортируются и записываются в файл через функции sort\_proxies\_by\_country и save\_sorted\_proxies.
7. def save\_sorted\_proxies(file\_path, proxies):
8. with open(file\_path, "w") as file:
9. for proxy in proxies:
10. file.write(proxy + "\n")
11. **Просмотр базы данных**:
    * Пользователь может открыть базу данных в SQLite для анализа данных. Примеры запросов включают .tables для отображения всех таблиц и SELECT \* FROM proxies; для просмотра всех записей.

**3.4. Экспорт в .exe**

1. Установите PyInstaller:
2. pip install pyinstaller
3. Соберите проект:
4. pyinstaller --onefile --windowed gui.py
5. Исполняемый файл будет находиться в папке dist/.

**3.5. Тесты в PyTest.**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Все тесты хранятся в следующей папке: PythonProject2\tests

**4. Заключение**

Разработанный проект "Прокси Чекер" представляет собой универсальное и простое в использовании приложение, которое автоматизирует проверку прокси-серверов и предоставляет пользователю удобный инструмент для работы с большими списками прокси. Основные функциональные возможности включают выбор протокола, проверку доступности, определение страны и сохранение результатов в базу данных SQLite.

**Жизненный цикл программного обеспечения**

1. **Инициация проекта**: Определение целей и требований для приложения.
2. **Разработка**: Реализация интерфейса и логики на Python с использованием PyQt5, SQLite и других библиотек.
3. **Внедрение**: Сборка исполняемого файла (.exe) с помощью PyInstaller и его распространение.
4. **Эксплуатация**: Пользователи запускают приложение, загружают списки прокси и используют его функционал.
5. **Сопровождение**: Внесение изменений и исправление ошибок на основе обратной связи от пользователей.