**План выполнения проекта**

**Цель проекта:**  
Разработка инструмента для массовой проверки и фильтрации прокси-серверов с акцентом на скорость, масштабируемость и расширяемость.

**1. Анализ требований и постановка задач**

* Определить целевые сценарии использования (массовая проверка, фильтрация по типу, скорости, анонимности).
* Изучить лучшие решения на рынке и их ключевые функции: массовая проверка, фильтрация по "живости", определение типа, скорости, геолокации, анонимности, работа с разными форматами прокси[3](https://googiehost.com/blog/best-proxy-checker-tools/)[4](https://www.pyproxy.com/information/what-are-the-best-ip-proxy-checker-tools-to-use.html)[5](https://hackread.com/tools-for-testing-your-proxy-servers/)[6](https://www.thordata.com/blog/proxies/best-proxy-checker-tools).
* Согласовать с заказчиком желаемые форматы ввода/вывода и интеграции.

**2. Декомпозиция задач**

**1. Архитектура и проектирование**

* Выбрать основной стек: Python (с возможной оптимизацией Cython/C для ускорения).
* Спроектировать модульную структуру (разделение логики парсинга, проверки, фильтрации, конфигурирования).

**2. Реализация основных модулей**

* Модуль парсинга (parse2ip.py, parse\_table.py):
  + Поддержка различных форматов списков прокси.
  + Валидация и нормализация входных данных.
* Модуль проверки (main.py, import requests.py):
  + Проверка доступности прокси (подключение, таймауты).
  + Определение типа (HTTP/SOCKS4/SOCKS5).
  + Измерение скорости и анонимности.
  + Проверка на "черные списки" (по возможности).
* Модуль фильтрации и вывода (utils.py, cfg.py):
  + Фильтрация по заданным критериям (скорость, тип, анонимность).
  + Формирование итогового списка валидных прокси.
  + Логирование и отчеты.

**3. Тестирование и оптимизация**

* Юнит-тесты для каждого модуля.
* Оптимизация скорости (асинхронность, многопоточность, Cython/C).
* Тестирование на больших объемах данных.

**4. Документация и упаковка**

* Документирование кода и инструкции по запуску.
* Подготовка README и примеров использования.
* Подготовка отчета для заказчика.

**3. Сравнение вариантов реализации**

| **Критерий** | **Python (requests, asyncio)** | **Python + Cython/C** | **Использование готовых сервисов/API** |
| --- | --- | --- | --- |
| Скорость | Средняя | Высокая | Очень высокая (зависит от сервиса) |
| Гибкость | Высокая | Высокая | Низкая/Средняя |
| Масштабируемость | Хорошая | Отличная | Зависит от тарифов |
| Зависимости | Минимальные | Требуется компиляция | Нет (если через API) |
| Стоимость | Бесплатно | Бесплатно | Платно (при больших объемах) |
| Контроль над данными | Полный | Полный | Частичный |

**Вывод:**  
Для фриланс-проекта оптимально использовать Python с возможностью доработки критичных участков на Cython/C. Это даст максимальную гибкость, контроль и масштабируемость без дополнительных затрат на сторонние сервисы[4](https://www.pyproxy.com/information/what-are-the-best-ip-proxy-checker-tools-to-use.html" \t "_blank)[5](https://hackread.com/tools-for-testing-your-proxy-servers/" \t "_blank)[6](https://www.thordata.com/blog/proxies/best-proxy-checker-tools" \t "_blank).

**4. Этапы реализации и отчетность**

**Этап 1. Проектирование (1-2 дня)**

* Согласование требований.
* Разработка архитектуры.
* Отчет: схема модулей, описание логики.

**Этап 2. Разработка (5-7 дней)**

* Реализация парсинга и базовой проверки.
* Добавление фильтрации и конфигурирования.
* Отчет: демонстрация работы на тестовых данных.

**Этап 3. Оптимизация и расширение (3-5 дней)**

* Внедрение асинхронности/многопоточности.
* При необходимости - оптимизация через Cython/C.
* Отчет: сравнение производительности до/после.

**Этап 4. Тестирование и документация (2 дня)**

* Проведение тестов на реальных списках.
* Подготовка документации.
* Итоговый отчет: инструкция, примеры, рекомендации.

**5. Итог**

В результате заказчик получает масштабируемый и быстрый инструмент, полностью соответствующий современным требованиям к массовой проверке прокси и легко расширяемый под новые задачи.