Программное средство по сканированию файлов на уязвимости

1. Цели разработки продукта

* **Обеспечение безопасности кода и файлов**: Автоматизация процесса проверки исходного кода, контейнеров и зависимостей на наличие уязвимостей.
* **Удобный пользовательский интерфейс**: Обеспечение простоты использования через веб-интерфейс, позволяющий пользователям загружать файлы и получать отчеты.
* **Масштабируемость**: Возможность обработки большого количества запросов и файлов одновременно без снижения производительности.
* **Надежность и устойчивость к сбоям**: Система должна быть стабильной, с возможностью восстановления после сбоев.

1. Функциональные требования

* **Загрузка и сканирование файлов**:

Пользователь должен иметь возможность загружать файлы для сканирования на наличие уязвимостей.

Использование сокетов для быстрого получения результата.

* **Использование Trivy**:

Сканирование файлов с помощью Trivy для выявления уязвимостей в контейнерах и файловых системах.

* **Визуализация отчетов**

Отчеты должны предоставляться в понятном пользователю виде

1. Основные бизнес-кейсы:  
     
   **Сканирование файлов и зависимостей:**

* **Описание**: Пользователь загружает файл через фронтенд. Микросервис вызывает Trivy для сканирования и анализирует результаты.
* **Основные шаги**: Загрузка файла → Обработка файла → Запуск Trivy → Анализ результата → Отображение результата пользователю.

1. Требования к производительности, масштабируемости и надежности системы  
     
   **Производительность:**

* **Низкая задержка обработки**: Время от загрузки файла до получения результата должно быть минимальным, желательно в пределах секунд или нескольких минут, в зависимости от размера файла.
* **Низкая нагрузка на сеть**: Использование минимального количества запросов, для получения результатов.

**Масштабируемость:**

* **Горизонтальное масштабирование**: Возможность масштабирования микросервисов, таких как Trivy, за счет добавления новых инстансов.
* **Облачная инфраструктура**: Поддержка работы в облачных средах с использованием контейнеров.
* **Распределенная обработка**: Разделение нагрузки между различными сервисами и серверами для обеспечения стабильной работы при высокой нагрузке.

**Надежность:**

* **Обработка ошибок**: Все сервисы должны иметь встроенные механизмы обработки ошибок и ведения логов для анализа и быстрого реагирования на сбои.
* **Защита данных**: Обеспечение безопасного хранения и передачи данных, включая результаты сканирования, путем использования шифрования.

1. Архитектура:

В данном приложение будет реализовано веб-приложение, с которым будет взаимодействовать пользователь. ASGI Сервер – на который будут идти запросы от пользователя. И WSGI сервер на который будут приходить отчеты от trivy, и который будет отсылать результаты проверки клиенту. Запросы на обработку будут попадать в RabbitMQ, а дальше переходить к микросервисам.

1. Выбор технологий:  
     
   JavaScript, React;  
   Python, Django;  
   Python, FastApi;  
   MongoDB;
2. Конфигурация RabbitMQ

* **Очередь: durable** — очередь должна быть устойчивой, чтобы сохранить сообщения при перезапуске RabbitMQ.
* **Exclusive** и **Auto-delete** можно отключить, чтобы очередь была доступна для других потребителей и не удалялась при отключении потребителя.
* Применение **подтверждения сообщений от брокера** (publisher confirms) — чтобы убедиться, что сообщение успешно доставлено в очередь.
* **prefetch** — установить количество сообщений, которое потребитель может одновременно обработать (например, 1 или 10), чтобы предотвратить перегрузку.