|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДАЮ |
|  |  | Руководитель образовательной программы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сергеева Е. Г.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  М.П. |

**РУКОВОДСТВО ПРОГРАММИСТА**

«Система автоматизированного тестирования безопасности веб-приложений»

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО |  | СОГЛАСОВАНО |
| Колледж ВятГУ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вершинина Н.Д.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  М.П. |  | Руководитель УП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженкова М. Л.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2025

### 1. Установка и запуск программы

### 1.1. Подготовка окружения

1. Скачайте файлы программы из репозитория: https://github.com/gickso-qno/Web-security-self-testing.
2. Убедитесь, что на вашем компьютере установлен Python 3.11.
3. Убедитесь, что у вас есть следующие библиотеки и инструменты:

* Fastapi;
* uvicorn;
* pydantic;
* python-jose;
* passlib;
* bcrypt;
* celery;
* redis;
* sqlalchemy;
* alembic;
* asyncio;
* aiohttp;
* httpx;
* aiofiles;
* jinja2;
* orjson;
* loguru;
* tenacity;
* requests;
* beautifulsoup4;
* lxml;
* tldextract;
* pyyaml;
* ruamel.yaml;
* playwright;
* pyppeteer;
* Pillow;
* opencv-python;
* semgrep;
* gitpython;
* pathspec;
* minio;
* cryptography;
* pyotp;
* hashlib;
* auditlog;
* prometheus-client;
* structlog;
* pytest;
* pytest-asyncio;
* pytest-cov;
* faker;
* respx;
* python-dotenv;
* typer;
* click;
* black;
* isort;
* flake8;
* mypy;
* sentry-sdk.

1. Для установки зависимостей выполните команду:

pip install -r requirements.txt

### 1.2. Настройка конфигурации

Конфигурация системы осуществляется с использованием централизованных файлов настроек, размещённых в каталоге configs/.

Ключевые параметры, необходимые для работы микросервисов, хранятся в конфигурационном файле .env, который загружается каждым сервисом при запуске контейнера.

Файл .env содержит основные параметры окружения, включая сведения о подключении к базе данных, брокеру сообщений, файловому хранилищу, а также ключи безопасности и настройки почтового сервера.

Помимо файла .env, система использует отдельные конфигурации для настройки очередей задач, логирования, мониторинга и области допустимного сканирования.

### 1.3. Подключение базы данных

В качестве основной системы управления базами данных используется PostgreSQL. Связь с базой данных осуществляется с помощью библиотеки SQLAlchemy в асинхронном режиме через драйвер asyncpg.

Подключение выполняется динамически на основе параметров, указанных в файле .env.

Для удобства миграции схемы данных используется инструмент Alembic, обеспечивающий автоматическое создание и обновление таблиц.

### 1.4. Подключение брокера задач и хранилища артефактов

Для организации обмена сообщениями между микросервисами применяется брокер Redis, а для хранения файловых артефактов – объектное хранилище MinIO.

Связь с Redis осуществляется посредством библиотеки Celery, которая обеспечивает управление задачи и выполнения асинхронных операций.

MinIO используется для хранения отчетов, скриншотов и других артефактов Proof-of-Concept, доступ к которым осуществляется по безопасным ключам доступа.

### 1.5. Средства контейнеризации и запуск системы

Система построена на микросервисной архитектуре и запускается в конвейеризированной среде с помощью Docker Compose. Каждый компонент (API-шлюз, оркестратор задач, воркеры анализа, хранилище, бркоер и база данных) представлен отдельным контейнером, взаимодействующим по сети Docker.

### 1.6. Запуск системы

1. создание конфигурации;

На основе шаблона env.example создается файл .env с параметрами окружения.

1. сборка контейнеров;
2. запуск всех сервисов;

После успешного запуска становятся доступны веб-интерфейс, панель управления MinIO, база данных и брокер сообщений Redis.

1. инициализация базы данных;
2. проверка работоспособности.

### 2. Проверка работоспособности

### 2.1. Проверка запуска

1. убедитесь, что приложение на базе app/main.py запускается корректно и отвечает на запросы по адресу http://localhost:8000/docs;
2. проверьте, что оркестратор задач успешно подключается к брокеру Redis принимает задания;
3. убедитесь, что веб-интерфейс корректно загружается и отображает форму авторизации;
4. проверьте, что все контейнеры запущены и находятся в состоянии healthy;
5. проверьте соединение с базой данных.

### 2.2. Тестирование функциональности

1. Проверьте работу всех основных функций:

* запуск сканирования веб-приложение по URL;
* получение статуса задачи и процентов выполнения;
* формирование и просмотр отчета с уязвимостями;
* воспроизведение РоС через headless-браузер;
* экспорт отчетов в формате PDF и JSON;
* верификация найденных уязвимостей и пометка ложных срабатываний;
* взаимодействие c CI/CD.

1. Для проверки взаимодействия с базой данных и API выполните тесты:

Pytest –cov=api tests/

1. Проверьте логи работы системы.
2. Убедитесь, что Prometheus и Grafana отображают активность воркеров и статусы задач.

### 3. Поддержка и обновление программы

### 3.1. Обновления

1. Регулярно обновляйте зависимости:

pip install --upgrade -r requirements.txt

1. Следите за изменениями в библиотеках fastapi, celery, redis, sqlalchemy, playwright, jinja2.
2. Обновляйте шаблоны уязвимостей:

Nuclei -update-templates

Semgrep --update

### 3.2. Мониторинг ошибок

1. Используйте встроенные логеры для остлеживания ошибок:

* логи сервисов сохраняются в директории logs/.
* уровень логирования настраивается в config.py.

1. Настройте оповещения через Ptometheus + Grafana или email-уведомления о сбоях задач.

### 3.3. Документация и уведомления

1. Обновляйте README.md и документацию о новых функциях в репозитории.
2. Уведомляйте пользователей через GitHub Issues.

### 3.4. Резервное копирование

1. Регулярно сохраняйте резервные копии базы данных PostgreSQL:

pg\_dump -U <user> -d <database> > backup.sql

### 4. Модернизация программы

### 4.1. Разработка новых функций

1. Создайте новую ветку для разработки:

git checkout -b feature/<название\_функции>

1. Реализуйте функции в соответствующих модулях:

* API и логика: app/services/;
* воркеры и задачи: works/tasks/;
* интерфейс пользователя: webui;
* конфигурация: .env.

1. При добавлении новых подсистем внесите их описание в техническую документацию и обновите схему архитектуру.

### 4.2. Тестирование

1. Напишите тесты для новой функции в директории tests/:

pytest tests/

1. Убедитесь, что покрытие кода тестами выше 80%:

pytest --cov=api --cov=workers

1. Убедитесь, что новая функция корректно взаимодействует с БД, очередью и другими модулями системы.
2. Выполните ручное тестирование через веб-интерфейс и API, убедившись в корректности ответов и стабильности работы.

### 4.3. Внесение изменений

1. После успешного тестирования и код-ревью выполните слияние изменений в основную ветку
2. При необходимости выполните обновление контейнеров.
3. Обновите документацию и при необходимости добавьте новую схему API.

### 5. Основная структура проекта

|  |  |
| --- | --- |
| Директория | Описание |
| ci/ | CI/CD конфиги |
| charts/ | Helm charts / k8s manifests |
| infra/ | Инфраструктура (terraform, ansible, scripts) |
| services/ | Микроскрвисы (API, workers, orchestration и т.д.) |
| webui/ | Frotend |
| docs/ | Документация |
| tools/ | Вспомогательные инструменты (osint collectors, oob server) |
| configs/ | Глобальные конфиги и шаблоны (env.example, yaml) |
| migrations/ | SQL / alembic миграции |
| tests/ | Интеграционные / e2e тесты |
| monitoring/ | Prometheus/grafana/loki dashboards |
| artifacts/ | Локальное хранилище для PoC/скриншотов (используется minio в prod) |

### 6. Часто задаваемые вопросы (FAQ)

1. Система не запускается. Что делать?

Проверьте, чтобы все сервисы были запущены: docker-compose ps

Если какой-то контейнер имеет статус Exited, перезапустите его: docker-compose up -d –build.

1. Что делать, если не открывается веб-панель?

Проверьте, запущен ли фронтенд: npm run dev. Если ошибка не исчезает – очистите кэш и перезапустите сервер.

1. Почему сканирование не завершается?

Проверьте логи Celery-воркеров: docker logs worker. Если есть ошибки подключения – убедитесь, что Redis доступен и параметры в .env корректны.

1. Можно ли запустить систему без HTTPS?

Не рекомендуется. Для локальных тестов можно включить режим INSECURE\_MODE=True в .env, но в боевой среде должен использоваться только HTTPS.

1. Как ограничить область сканирования?

Настройте параметры scope\_whitelist и scope\_blacklist в интерфейсе или через API.