Промежуточный отчет по программному проекту

1. Основные планы и задачи разработчика Зобова А.А.

1.1 Краткое описание проекта:

«Веб-приложение для комплексной автоматизации процессов поискового продвижения веб-сайтов» («SEO Мастер») — это высокоавтоматизированная, микросервисная платформа, предназначенная для комплексного управления и оптимизации SEO-продвижения веб-сайтов. Платформа обеспечивает независимость от дорогих внешних SEO API, внедряет механизм Human-in-the-Loop (HITL) для контроля критических изменений и фокусируется на бизнес-ориентированной аналитике (FF-Score, E-E-A-T Score).

Название проекта:

Веб-приложение для комплексной автоматизации процессов поискового продвижения веб-сайтов

Цель работы (Зобов А.А.):

Разработка ключевых управляющих компонентов платформы: Management Service (оркестратор бизнес-логики), API Gateway (фронтенд и шлюз), Client API Gateway (безопасный шлюз для внедрения изменений) и обеспечение общей инфраструктуры проекта (контейнеризация, конфигурация).

Краткое описание задач:

1. Разработка и реализация Management Service (Оркестратор, логика HITL, переприоритизация задач).
2. Разработка и реализация API Gateway / Frontend Service (UI, личный кабинет, интерфейс HITL-одобрений).
3. Разработка и реализация Client API Gateway и адаптеров (WordPress, PHP, Python) для безопасного внедрения изменений.
4. Инициализация и настройка общей инфраструктуры проекта (Docker, Celery/ RabbitMQ, миграции БД).

1.2 Планы и этапы выполнения проекта (Специфично для Разработчика 1)

Компонент	Ключевые модули	Технологии	Статус/Сроки
Management Service	Оркестратор (`orchestrator.py`), Периодические задачи (`periodic_tasks.py`), Логика HITL	Python (FastAPI/ Django), Celery	11.01.26 – 25.02.26
API Gateway / Frontend	Backend (роуты-прокси, аутентификация), Frontend (Dashboard, HITL UI)	FastAPI, React/ Vue.js, JWT	01.02.26 – 20.03.26
Client API Gateway	Ограниченные PATCH эндпоинты, Аутентификация HMAC-SHA256, Адаптеры (WordPress, PHP, Python)	FastAPI, PHP, Python	01.03.26 – 25.03.26

Компонент	Ключевые модули	Технологии	Статус/Сроки
Инфраструктура	Конфигурация Docker Compose, Инициализация БД, Настройка Celery/RabbitMQ	Docker, PostgreSQL, Celery	17.12.25 – 25.02.26

2. Используемый технологический стек и его обоснование

2.1 Перечень используемых технологий

Технология/Инструмент	Назначение	Обоснование
Python (FastAPI/Django)	Основной язык для Backend-логики Management Service и API Gateway	Высокая производительность, асинхронность (FastAPI), удобство для оркестрации
React / Vue.js	Разработка Frontend (SPA) для личного кабинета и интерфейса HITL	Современный стек для создания интерактивных и сложных пользовательских интерфейсов
Celery / RabbitMQ	Оркестрация и асинхронное выполнение задач	Обеспечение надежной работы «SEO Robot» и механизма переприоритизации задач
Docker / Docker Swarm	Контейнеризация и Production-развертывание	Обеспечение стандартизированного окружения для всех микросервисов и упрощение деплоя
PostgreSQL (с JSONB)	Основное хранилище данных для задач, одобрений HITL и Changelog	Надежность, поддержка транзакций, необходимая для критических операций HITL
HMAC-SHA256	Механизм аутентификации для Client API Gateway	Обеспечение высокого уровня безопасности при внедрении изменений на сайт клиента
Visual Studio Code	Основная интегрированная среда разработки (IDE) для реализации backend- и frontend-компонентов проекта	Бесплатная, кроссплатформенная, высоконастраиваемая среда с широкой экосистемой расширений для Python, JavaScript/TypeScript, Docker, Git и других используемых технологий

Технология/Инструмент	Назначение	Обоснование
Git	Управление версиями	Удобство разработки в команде, сохранение версий разработки

2.2 Обоснование выбранного технологического стека

Выбор **Python** и **FastAPI/Django** для **Management Service** и **API Gateway** обусловлен необходимостью быстрой разработки и высокой производительности для обработки большого количества запросов и управления асинхронными задачами.

React/Vue.js выбран для **Frontend** для создания современного, отзывчивого и функционального интерфейса, особенно для сложного компонента **HITL Approval Detail** с просмотром различий (**DiffViewer**).

Celery и **RabbitMQ** являются основой для реализации **Management Service** как Оркестратора. Они позволяют реализовать логику **SEO Robot** — запуск циклов оптимизации, переприоритизацию задач по **Impact x Effort** и обработку событий от других микросервисов.

Использование **Docker** и **Docker Swarm** критически важно для обеспечения воспроизводимости окружения и упрощения развертывания сложной микросервисной архитектуры, за что отвечает Разработчик 1.

Client API Gateway с аутентификацией **HMAC-SHA256** обеспечивает безопасность и контроль при внедрении изменений на сайт клиента, что является ключевым требованием методологии **Human-in-the-Loop (HITL)**.

3. Критерии оценивания работы (Зобов А.А.)

Критерий	Описание	Измеримый показатель
Функциональность Management Service	Корректная работа оркестратора, правильная логика переприоритизации задач (Impact x Effort), надежная обработка событий HITLApproved.	90% покрытия функциональных требований, 100% обработки доменных событий
Функциональность Frontend / API Gateway	Работоспособность аутентификации, корректное отображение Dashboard, полная реализация интерфейса HITL-одобрений с DiffViewer.	Время отклика UI < 2.5с, 95% покрытия кейсов использования
Функциональность Client API Gateway	Безопасность (HMAC-SHA256), корректная работа ограниченных PATCH эндпоинтов, успешное внедрение изменений через адаптеры.	0 критических уязвимостей, 90% успешных внедрений изменений

Критерий	Описание	Измеримый показатель
Инфраструктура и Деплой	Полная и корректная конфигурация Docker-окружения для всех микросервисов, работоспособность Celery/RabbitMQ, наличие скриптов миграции БД.	Время развертывания < 10 мин, 99.9% доступности сервисов
Соблюдение архитектуры	Строгое следование принципам DDD и EDA, корректная генерация и потребление доменных событий (`TaskCreated`, `HITLApproved`).	100% соответствия архитектурным паттернам, покрытие кода тестами > 80%
Качество кода	Соответствие стандартам кодирования, наличие документации, эффективность реализации	Статический анализ: 0 критических замечаний, покрытие unit-тестами > 75%

4. Особые пометки

Настоящий документ описывает только индивидуальную часть проекта, разрабатываемую Зобовым А.А. В рамках командного проекта распределение ответственности следующее:

Зобов А.А. (Разработчик 1) отвечает за:

- Разработку и реализацию Management Service
- Разработку и реализацию API Gateway / Frontend Service
- Разработку и реализацию Client API Gateway и адаптеров для клиентов
- Инициализацию и настройку общей инфраструктуры проекта (Docker, Celery/RabbitMQ, миграции БД)

Пухова А.И. (Разработчик 2) отвечает за:

- Разработку Audit Service (краулер, индексатор, анализ Core Web Vitals)
- Разработку Semantic Service (LLM-анализ, расчет E-E-A-T Score и FF-Score)
- Реализацию Reporting Service (формирование отчетов, интеграция с внешними API)
- Участие в разработке Client API Gateway (адаптеры)

Интеграция компонентов и совместное тестирование системы будет выполняться обоими разработчиками в период с 16.04.26 по 20.04.26 согласно графику проекта.