

Промежуточный отчет по программному проекту

1. Основные планы и задачи разработчика Пуховой А.И.

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).

Цель проекта: разработка веб-приложения для автоматизированного SEO-аудита и долгосрочного управления продвижением сайтов с использованием событийной архитектуры и аналитических метрик.

Название проекта:

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

Цель работы (Пухова А.И.):

Разработка ключевых микросервисов, отвечающих за сбор и анализ данных (Audit Service), семантический анализ и генерацию контента (Semantic Service), а также сбор внешних данных и отчетность (Reporting Service). Обеспечение надежной работы аналитического ядра платформы.

Краткое описание задач:

1. Разработка и реализация Audit Service (Краулер, Анализ данных GSC API для анализа ссылочной массы, Анализаторы CWV/Schema).

1.1. Режим полного аудита (для зарегистрированных пользователей):

- Полноценный краулер с JS-рендерингом
- Анализ данных GSC API для анализа ссылочной массы
- Анализаторы CWV/Schema

1.2. Режим публичного (первичного) аудита:

- Облегченный краулер (до 10 страниц, без JS-рендеринга)
- Базовые технические проверки (meta-теги, битые ссылки)
- Проверка CWV через публичный PageSpeed Insights API
- Проверка базовой Schema.org разметки

2. Разработка и реализация Semantic Service (LLM Адаптер, Расчет FF-Score, E-E-A-T Анализ, Контент-Анализ).

3. Разработка и реализация Reporting Service (Сборщики GSC/GA4/Yandex, Расчет Метрик, Генерация Отчетов CSV).

4. Обеспечение асинхронного взаимодействия между микросервисами через событийную модель (Celery/RabbitMQ).

1.2 Планы и этапы выполнения проекта (Специфично для Разработчика 2)

Микросервис	Ключевые модули	Технологии	Статус/Сроки
Общие задачи	Формализация логики FF-Score и HITL-механизма, включая реализацию событийной модели с использованием событий CrawlCompleted и FFScoreRecalculated	Celery / RabbitMQ	17.12.2025 – 10.01.2026
Audit Service	Полный режим: Краулер (technical_audit.py), Анализ GSC API (gsc_link_analyzer.py), Анализаторы CWV/Schema. Публичный режим: Облегченный краулер (public_crawler.py), базовые проверки, PSI API	Python, Scrappy, Playwright, PostgreSQL	11.01.2026 – 25.02.2026
Semantic Service	LLM Адаптер (llm_client.py), FF-Score (ff_score_calculator.py), E-E-A-T Анализ (eeat_analyzer.py), Контент-Анализ	Python, LLM API (OpenAI/Gemini), NLTK/spaCy	26.02.2026 – 20.03.2026
Reporting Service	Сборщики GSC/GA4/Yandex, Расчет Метрик (calculator.py), Генерация Отчетов CSV	Python, pandas, GSC/GA4 API	01.03.26 – 25.03.26

2. Используемый технологический стек и его обоснование

2.1 Перечень используемых технологий

Технология/Инструмент	Назначение	Обоснование
Python (FastAPI)	Основной язык для Backend-логики микросервисов	Высокая скорость разработки, богатая экосистема для NLP/SEO (Scrappy, pandas)
PostgreSQL (с JSONB)	Основное хранилище данных	Надежность, поддержка сложных запросов, JSONB для гибкого хранения сырых данных
Celery / RabbitMQ	Очередь задач для асинхронного выполнения долгих процессов	Необходимость для краулинга и LLM-генерации, которые являются долгими операциями
Scrappy / Playwright	Краулинг и рендеринг JS/SPA контента	Scrappy для эффективности, Playwright для обработки современного JS-рендеринга

Технология/Инструмент	Назначение	Обоснование
LLM API (OpenAI/Gemini)	Генерация контента и метаданных, семантический анализ	Обеспечение функционала LLM-генерации черновиков и E-E-A-T анализа
pandas	Сбор, обработка данных и генерация отчетов (Reporting Service)	pandas для эффективной работы с данными
Git. GitHub	Контроль версий исходного кода проекта и хранение репозитория	Позволяют отслеживать изменения в коде, управлять версиями проекта и обеспечивать сохранность результатов разработки
Visual Studio Code	Интегрированная среда разработки для реализации backend-компонентов проекта, включая разработку краулеров, анализаторов, модулей семантического и контентного анализа, а также средств формирования отчётности	Бесплатная кроссплатформенная среда разработки с широкой экосистемой расширений для Python, работы с краулерами и NLP-модулями, контейнеризации (Docker), системы контроля версий Git, а также сопутствующих инструментов аналитики и отчётности

2.2 Обоснование выбранного технологического стека

Выбор **Python** как основного языка обусловлен его доминированием в области **Data Science**, **NLP** и **Web Scraping**. Библиотеки **Scrapy**, **pandas**, **NLTK/spaCy** являются критически важными для реализации **Audit** и **Semantic** сервисов.

Использование **PostgreSQL** с типом данных **JSONB** позволяет эффективно хранить как структурированные данные (например, метаданные страниц), так и неструктурированные сырые данные краулинга.

Celery и **RabbitMQ** выбраны для реализации **Event-Driven Architecture (EDA)**. Это необходимо для асинхронного выполнения ресурсоемких задач, таких как полный краулинг с рендерингом JS и сложные расчеты **FF-Score**, предотвращая блокировку основного API.

Playwright обеспечивает возможность полного рендеринга страниц с **JavaScript**, что является ключевым требованием для **Audit Service** при анализе современных **SPA** (Single Page Applications) и корректной оценке **Mobile-First Indexing**.

Playwright выбран для JS-рендеринга, но имеет высокое потребление ресурсов:

- ~ 500 МБ RAM на инстанс браузера
- Рекомендуется ограничение 5 параллельных инстансов на 1 воркер (4 CPU, 8 GB RAM)
- Для масштабирования > 100К страниц необходима распределенная архитектура (например, browserless.io или собственный кластер Playwright)

Playwright используется только для полного режима аудита (зарегистрированные пользователи). Для публичного первичного аудита применяется облегченный краулинг без JS-рендеринга (статический HTML через requests/httpx), что обеспечивает:

- Быстрое выполнение (до 60 секунд на аудит)
- Минимальное потребление ресурсов (~50 МБ RAM на запрос)
- Возможность обработки до 10 параллельных публичных запросов

Использование **Git** и **GitHub** обусловлено необходимостью централизованного хранения исходного кода и контроля версий проекта. Это позволяет фиксировать этапы разработки, отслеживать изменения в кодовой базе и обеспечивать согласованную работу с архитектурными и программными артефактами проекта.

Visual Studio Code используется в качестве основной среды разработки благодаря встроенной поддержке языка **Python**, удобным инструментам отладки и интеграции с системой контроля версий. Применение данных **IDE** повышает эффективность разработки и упрощает сопровождение кода на всех этапах реализации проекта.

3. Критерии оценивания работы (Пухова А.И.)

Критерий	Описание
Функциональность Audit Service	Корректная работа JS-краулера, анализ ссылок через внешние API, валидация CWV и Schema.org. Работоспособность режима публичного аудита (ограниченный краулинг, базовые проверки), соблюдение лимитов производительности (60 сек), корректная публикация события PublicAuditCompleted.
Функциональность Semantic Service	Точность расчета FF-Score и E-E-A-T Score, корректная работа LLM-адаптера и генерации черновиков.
Функциональность Reporting Service	Успешное подключение к GSC/GA4/Yandex API, корректный расчет Cost-Efficiency, генерация отчетов CSV.
Соблюдение архитектуры	Строгое следование принципам DDD и EDA, корректная публикация и потребление доменных событий (CrawlCompleted, FFScoreRecalculated).
Качество кода	Соответствие Clean Architecture, покрытие тестами, читаемость и поддерживаемость кода.

4. Особые пометки

Настоящий документ описывает только индивидуальную часть проекта, разрабатываемую Пуховой А.И.

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 (WordPress, Tilda)
- Инициализацию и настройку общей инфраструктуры проекта (Docker, Celery/ RabbitMQ, миграции БД)

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

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

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