Отчёт по проекту мультиагентной системы LikeProvodnik

1. Состав команды и ставки

Роль	Ставка (руб./час)
Проджект-менеджер	4000
Программист	1300
Промпт-инженер-лоукодер	900
Ресёрчер-аналитик	1000

---

2. Хронология и этапы проекта

Этап 1. Low-code прототип (n8n, no-code, быстрый MVP)

Планировалось:

- Быстро собрать MVP мультиагентной системы на базе n8n и внешних low-code инструментов.
- Каждый агент реализуется как отдельный воркфлоу в n8n.
- Интеграция с Telegram через стандартные ноды.
- RAG-система и парсер Get Course — отдельные прототипы.

Что было сделано:

- Проведён анализ предметной области и требований.
- Разработан базовый парсер для Get Course.
- Создан прототип RAG-системы (на Python).
- В n8n реализованы 6 воркфлоу: AI-Дайджест, AI-Генератор лайфхаков, AI-Коуч, AI-Путь, AI-Наставник, Еженедельный дайджест.
- Начата интеграция компонентов.

Что переиспользовано далее:

- Логика агентов (структура сценариев, промпты, основные цепочки обработки сообщений).
- Прототипы промптов и сценариев для каждого агента.
- RAG-система и парсер Get Course — как отдельные Python-скрипты.

Этап 2. Микросервисная архитектура (Python, FastAPI/Flask)

Планировалось:

- Перейти на архитектуру микросервисов (Python, FastAPI/Flask).
- Каждый агент — отдельный сервис, взаимодействие через API Gateway.
- Хранение состояния и истории — в Postgres и Redis.
- Telegram-бот — отдельный сервис, маршрутизация через gateway.
- Интеграция RAG-системы для агентов.

Что было сделано:

- Спроектирована и реализована архитектура микросервисов.
- Разработаны базовые версии всех 6 интеллектуальных агентов (AI-Path, Coach, Mentor, NeuroExpert, Lifehack, Digest).
- Реализован API Gateway с универсальной маршрутизацией.
- Интегрирована RAG-система для обогащения ответов агентов.
- Разработан Telegram-бот с поддержкой FSM и проксированием сообщений к агентам.
- Подготовлена документация и тестовые сценарии.

Что переиспользовано далее:

- Код агентов (основные обработчики, промпты, логика взаимодействия).
- RAG-система и парсер Get Course — интегрированы как отдельные сервисы.
- Логика FSM и сценариев Telegram-бота.
- Структура docker-compose, подход к логированию, .env.

Этап 3. Универсальная система (единый API Gateway, централизованное управление, доработка архитектуры)

Причины перехода:

- Неудовлетворительная скорость интеграции и тестирования в микросервисной схеме.
- Сложности с поддержкой и масштабированием большого количества сервисов.
- Необходимость унификации логики агентов, централизованного логирования и управления состоянием.

Планировалось и реализовано:

- Вся маршрутизация и проксирование запросов реализованы через единый API Gateway.
- Все агенты переведены на работу через универсальные REST API (Flask/FastAPI).
- Telegram-бот реализует FSM для выбора и смены агента, все сообщения проксируются через gateway.
- Хранение состояния пользователя и истории — через Postgres и Redis.
- Вся система запускается и управляется через docker-compose, логи всех сервисов централизованы.
- Внедрены универсальные эндпоинты для агентов, поддержка новых агентов реализуется декларативно.
- Ведётся работа по доработке и унификации логики агентов, улучшению обработки ошибок и логированию.

Что переиспользовано:

- Код агентов и их API из микросервисной итерации (адаптирован под универсальный gateway).
- Промпты, сценарии, логика FSM Telegram-бота.
- RAG-система и парсер Get Course — интегрированы в новую архитектуру.
- Подход к логированию, централизованное хранение логов.
- Документация и тестовые сценарии.

---

3. Выполненные задачи и затраченное время

Управление проектом (Проджект-менеджер)

- Первичное планирование, координация, пересмотр архитектуры: **11 часов**

Исследование и анализ (Ресёрчер-аналитик)

- Анализ предметной области, требований, сравнение подходов: **13 часов**

Разработка и интеграция (Программист)

- Прототипирование парсера Get Course: **10 часов**
- Прототип RAG-системы: **12 часов**
- Разработка и интеграция n8n-воркфлоу: **14 часов**
- Разработка микросервисов-агентов: **20 часов**
- Разработка и доработка API Gateway: **13 часов**
- Интеграция Telegram-бота: **8 часов**
- Миграция на новую архитектуру, рефакторинг, docker-compose, логирование: **16 часов**
- Интеграция и тестирование: **10 часов**
- Итого: **103 часа**

Промпт-инженер-лоукодер

- Разработка и оптимизация промптов, настройка воркфлоу: **22 часа**
- Доработка промптов и сценариев для новой архитектуры: **8 часов**
- Итого: **30 часов**

Документирование и тестирование (все роли)

- Подготовка документации, тестовых сценариев, ревизия: **5 часов**

---

4. Общее время и стоимость проекта (на текущий момент)

Роль	Часы	Ставка (руб./час)	Сумма (руб.)
Проджект-менеджер	11	4000	44,000
Ресёрчер-аналитик	13	1000	13,000
Программист	103	1300	133,900

Роль	Часы	Ставка (руб./час)	Сумма (руб.)
Промпт-инженер-лоукодер	30	900	27,000
Итого	157		**217,900**

---

5. Текущие результаты

- Проведён полный анализ предметной области и требований.
- Реализованы прототипы парсера Get Course и RAG-системы.
- Разработаны и протестированы все 6 интеллектуальных агентов (AI-Path, Coach, Mentor, NeuroExpert, Lifehack, Digest).
- Внедрена микросервисная архитектура, затем проведён рефакторинг в сторону универсального API Gateway.
- Реализован Telegram-бот с FSM, поддержкой выбора и смены агентов, проксированием сообщений.
- Вся система запускается через docker-compose, логи всех сервисов централизованы.
- Подготовлена и актуализирована документация, описаны пользовательские сценарии для каждого агента.
- Ведётся интеграционное тестирование и доработка логики агентов.

---

6. Причины смены архитектуры и подхода

- **Ограничения n8n**: неудобство поддержки сложных сценариев, невозможность масштабирования, проблемы с хранением состояния.
- **Микросервисы без единого gateway**: сложности с синхронизацией, дублирование логики, проблемы с интеграцией и тестированием.
- **Переход к универсальному gateway**: позволил централизовать маршрутизацию, упростить добавление новых агентов, унифицировать логику и повысить отказоустойчивость.
- **Требования к масштабируемости и поддержке**: новая архитектура позволяет легко добавлять новых агентов, централизованно управлять логами и состоянием, быстро реагировать на изменения требований.

---

7. Задачи в процессе выполнения

- Доработка и оптимизация логики агентов (обработка ошибок, хранение истории, улучшение промптов).
- Завершение интеграционного тестирования всех компонентов.
- Оптимизация производительности и отказоустойчивости системы.
- Финальная доработка документации и пользовательских сценариев.
- Подготовка к деплою на сервер и запуску в продуктиве.

---

8. План доработок (по состоянию на сейчас)

- ◯ Доработать универсальные эндпоинты в API Gateway для всех агентов.
- ◯ Перевести хранение состояния пользователя (активный агент, история) в Redis.
- ◯ Завершить интеграцию и тестирование всех агентов через gateway.
- ◯ Реализовать планировщик задач для агентов по расписанию.
- ◯ Провести ревизию логирования и централизовать все логи в папке logs.
- ◯ Провести финальное тестирование пользовательских сценариев в Telegram-боте.
- ◯ Подготовить и актуализировать документацию (README, OpenAPI).
- ◯ Провести деплой и запуск системы на сервере.

---

9. Выводы

- Проект прошёл через три архитектурные итерации: от low-code/n8n к микросервисам, затем к универсальному gateway.
- На каждом этапе переиспользовались наработки: промпты, сценарии, код агентов, RAG-система, парсер, логика FSM.
- Все ключевые компоненты реализованы, система готова к финальной интеграции и запуску.
- Принятые архитектурные решения позволили повысить гибкость, масштабируемость и удобство поддержки проекта.