Команда Aiker

Оньекаба Екенедиличукву

Frontend / Backend / DevOps

Хафизов Георгий

DevOps

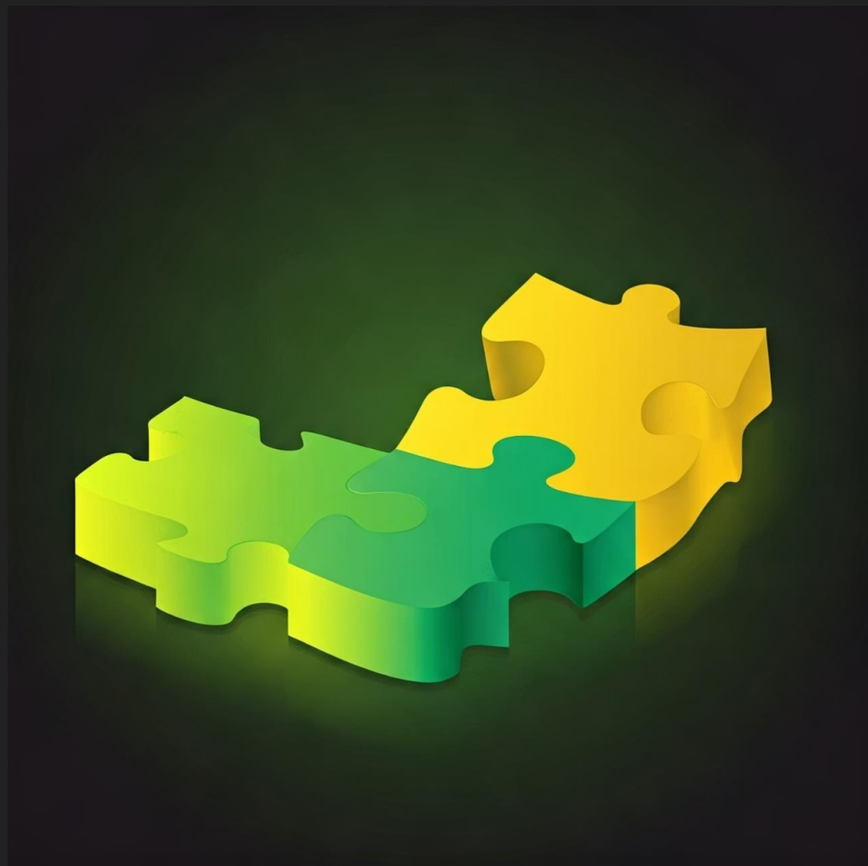
Проблемы и бизнес-контекст

Для кого:
Наш пользователь — QA-инженер (Manual/Automation), участвующий в регулярных релизах и регрессионном тестировании.

Проблема:
Огромные затраты времени на рутину: написание тест-кейсов, генерацию автотестов, анализ отчетов о дефектах.

- Менеджеры тратят часы на распределение задач.
- QA-инженеры тратят дни на написание кода для тестов.
- Высокие риски ошибок из-за ручной обработки данных.
- Риск пропустить критичный баг из-за "замыленного глаза« или нехватки времени.

Представьте: пятница, вечер, срочный хотфикс. QA-инженеру нужно срочно написать тесты на новую логику скидок. Вручную это займет 3 часа. Релиз откладывается на понедельник



Наше решение: Интеллектуальная платформа

Kenzy QA Copilot — AI-агентная система на базе MCP для автоматизации задач тестирования.

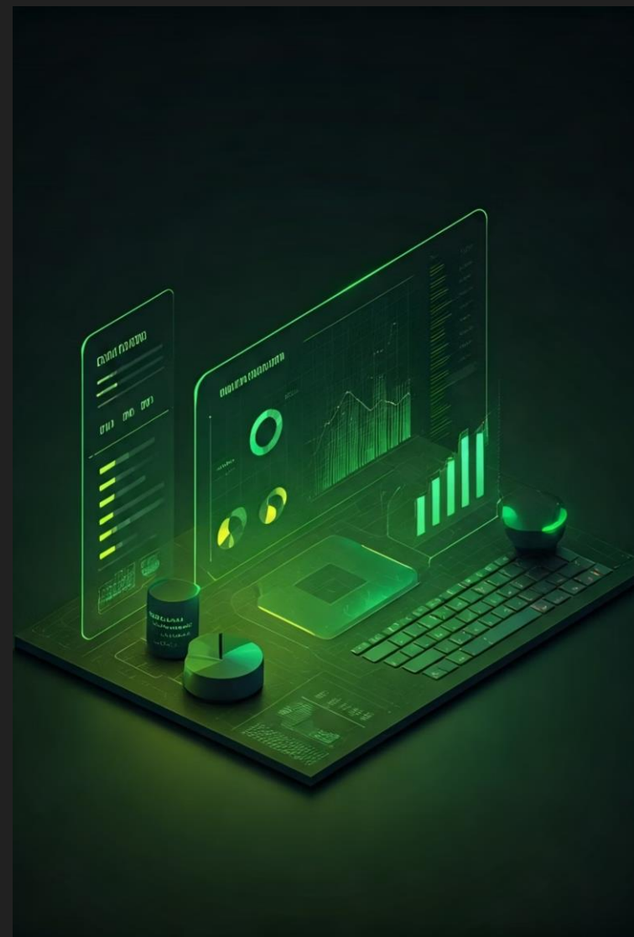
Что делает агент: «Понимает запрос на естественном языке, анализирует код/требования и генерирует готовые артефакты (тесты, отчеты)».

Интеграции:

- **GitLab API**: для чтения кода и управления merge request'ами.
- **Cloud.ru Evolution (LLM)**: "мозг" системы для генерации контента.
- **Публичные API**: интеграция с баг-трекерами и CI системами.

Ключевые особенности:

- Использует **MCP (Model Context Protocol)** для стандартизированного доступа к инструментам.
- Валидация данных через Pydantic.
- Встроенная обработка ошибок и повторные попытки (Retry logic).
- Полный цикл: от анализа до готового кода.



Архитектура решения

Решение построено по клиент-серверной архитектуре с использованием агентного подхода.

AI-Агент (Backend): Реализован на **Python (FastAPI)**. Выступает как оркестратор.

1. Использует **Cloud.ru Foundation Models** (OpenAI-compatible API) для интерпретации намерений пользователя.
2. Реализует логику выбора инструментов (Router).

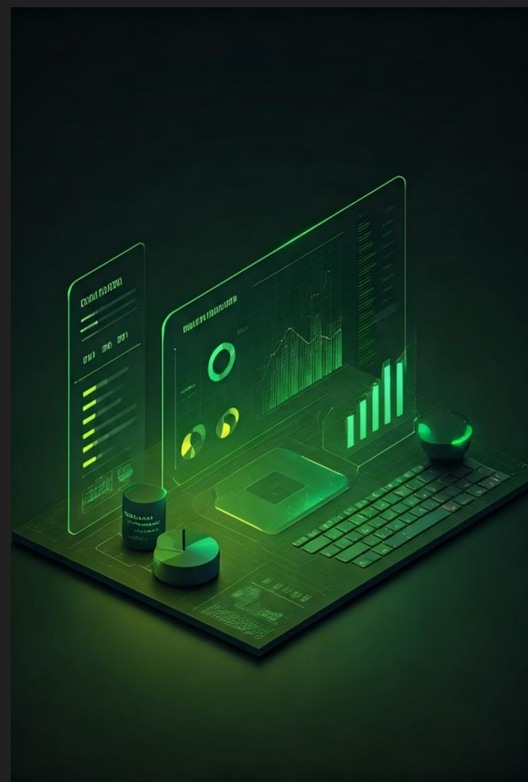
MCP Инструменты (Services):

1. gitlab_api: работа с репозиториями.
2. auto_test_generator: генерация кода автотестов (Pytest/Playwright).
3. defect_analyzer: анализ логов и ошибок.
4. test_validator: проверка корректности сгенерированных тестов.

Frontend: React-приложение для удобного чата с агентом и просмотра результатов.

Технологии:

- **LLM:** Cloud.ru Evolution.
- **Backend:** FastAPI, Pydantic, Tenacity (retries).
- **Deployment:** Docker, Docker Compose (готово к on-premise деплою).



Демонстрация: Решение в действии

Эффект и измеримый результат

Что изменилось после внедрения

Показатель	До	После
Время на задачу	4 ч/нед	5–10 мин
Покрытие	~60%	>85%
Скорость онбординга	1–2 дня	30 секунд

Теперь QA-инженер экономит до 15 часов в неделю

Ошибки в тестах <1%

Мгновенный ответ вместо дневного ожидания

Масштабирование решения и его будущее

Рынок: Любые компании с внутренней разработкой (FinTech, Retail, E-com).

Интеграция: Легкое подключение Jira, Yandex Tracker для создания баг-репортов.

Развитие:

- Multi-agent режим: один агент пишет тесты, второй делает Code Review.
- RAG по внутренней документации компании.
- Запуск тестов прямо из чата (интеграция с Jenkins/GitLab CI).

Чем гордимся:
Мы сделали не просто "обертку над ChatGPT", а рабочий инструмент, интегрированный в реальные процессы разработки через GitLab и специализированные сервисы. Решение надежно (retries, валидация) и готово к использованию.