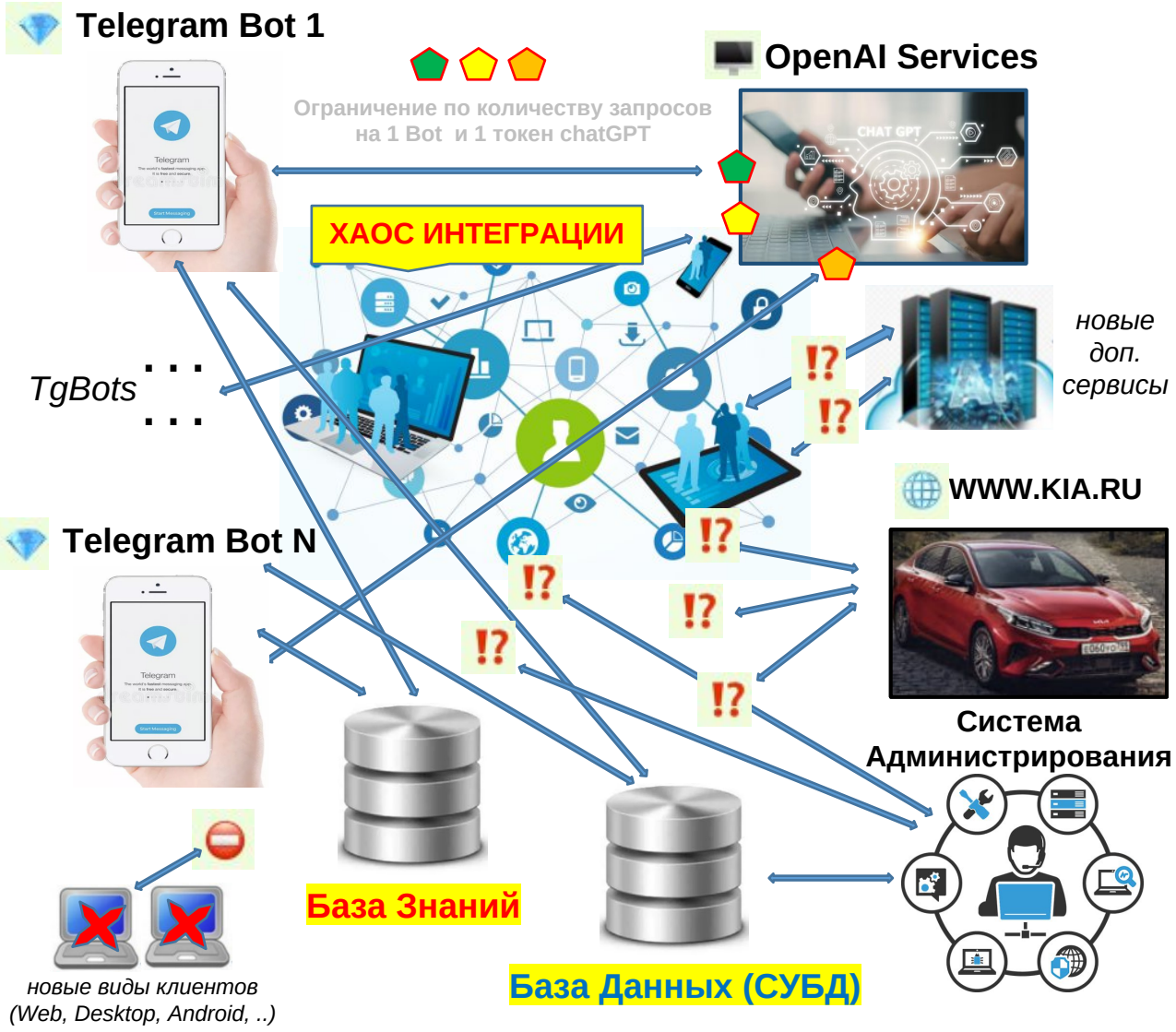
Ожидаемые результаты по ТЗ (Киа):
Нейро-консультант, отвечающий на вопросы клиентов компании по продуктам и услугам.

Путь 1 (текущий): Быстрое внедрение TgBot (noSQL) ⟺ OpenAI (+...?)



Основные возможности, плюсы и потенциальные проблемы:

Сценарий 1.А) Быстрая Демонстрация «верхнего уровня» решения (как пример обращения Бота к chatGPT из Телеграм), ограниченного по количеству пользователей и одновременных запросов, без комплексной системы управления и администрирования (и/или без СУБД). Все этапы проекта, подготовки Базы Знаний, сервисные функции и результаты будут хаотично создаваться и управляться из Google Colab, либо ручным способом в MS Word + MS Excel, что неправильно предлагать Заказчику! Нельзя портировать решение. Т.О. развитие проекта в данном направлении затруднительно!

Сценарий 1.Б) Попытка успеть за 1-2 мес. создать минимально-необходимый функционал без API => приведет к необходимости создания нескольких клиентов (ботов), подключаемых к разным токенам OpenAI и кучи дополнительных микросервисов. *"Хаос интеграции" или связи "многие ко многим" в контексте системной интеграции обычно описывают сценарий, в котором множество компонентов или систем напрямую связаны друг с другом без единого интерфейса или прослойки управления. Это может привести к ряду проблем:*

- **Сложность управления**: При изменении одного элемента нужно удостовериться, что это не сломает другие связанные элементы. Нет согласованности систем.
- **Жёсткая связанность**: Внесение изменений в одну систему может потребовать изменений во всех связанных системах. При переходе на новое ПО - все заново.
- **Сложность масштабирования**: Необходимо учитывать взаимодействие всех пар систем, что становится непрактичным при увеличении их числа.
- **Проблемы с безопасностью**: Без единого уровня управления или защиты, уязвимость одной системы может подвергнуть риску все остальные. К Боту или к OpenAI начнут подключаться случайные пользователи, для регистрации нужна БД.
- **Невозможность разделения тестирования на многие подзадачи:** Любые нововведения или принципиальные изменения в коде будут ожидать все участники проекта. Тестирование альтернативных клиентов и эффективной работы с СУБД будет невозможно. При блокировке VPN все делать заново для локальных LLM.
- **Трудности в отладке и мониторинге**: Отследить, какие системы взаимодействуют и как, может быть непростой задачей, как и совместное тестирование ряда LLM.

Ожидаемые результаты по ТЗ (Киа):

Полноценный и масштабируемый **Нейро-консультант, отвечающий на вопросы клиентов компании по продуктам и услугам**

Путь 2 (FastAPI): Масштабируемая Клиент-Серверная архитектура с API *(упомянута в лекции Д.Романова)*

Основные этапы внедрения:

1. Настройка min. API-эндпоинтов
2. Расширение точек тестирования
3. Предложения по развитию
4. Серьезное масштабирование

Основные преимущества архитектуры:

- ✓ Возможность поэтапного развития
- ✓ Быстрое подключение через Swagger UI/CLI
- ✓ Единый интерфейс для всех клиентов
- ✓ Быстрое и простое портирование решения
- ✓ Возможность работы на локальном сервере
- ✓ Простое администрирование и логгирование
- ✓ Быстрая работа с историей запросов и СУБД
- ✓ Высокая стабильность и защищенность
- ✓ Возможность добавления любых сервисов
- ✓ Простое масштабирование при нагрузке

Основные недостатки:

- ✓ Необходимость выделения Сервера (лучше с GPU)
- ✓ Опыт работы с RESTfull, Linux и Docker
- ✓ Выше затраты на Ресурсы

WWW.KIA.RU

Парсер или CRM API

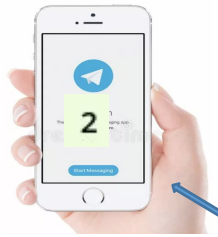
OpenAI Services

Автоматическое распределение нагрузки (чередование токенов)

новые доп. сервисы

Telegram Bot

- Linux
- Windows
- MacOS
- Android
- Web-Client
- Python IDE



Web-клиент

- Linux
- Windows
- MacOS
- Android
- Web-Client
- Python IDE

HTML JS CSS IMAGES

Desktop App (ex. PyQt6)

- Linux
- Windows
- MacOS
- Android
- Web-Client
- Python IDE

/CRM /MODEL /PROMPT /LOGS /ADMIN /AUTH /TOKEN

Create → POST
Read → GET
Update → PUT
Delete → DELETE
HTTP Methods

REST

Swagger UI

Система Администрирования

FastAPI

Web App и/или Telegram

docker HOST

Контейнеры на внутренней сети Docker

Выделенная среда Linux (Ubuntu, Debian, ..) + GPU

Server 1

Android-клиент

Server 2

Доступ CLI (curl) или Swagger UI

colab + BASH

/SberAPI /Whisper /GPT4All /Saiga2

Автоматическое масштабирование
Docker Swarm
или
Kubernetes

и/или

Базы Знаний **База Данных (СУБД)**

1. Предварительная Обработка (Pre-Processing):
2. Основная Обработка (Main Processing):
3. Пост-Обработка (Post-Processing):

Архив отобранных лучших комбинаций prompt (json)

Yandex Cloud