**🧱 Архитектурное предложение: гибридная AI-платформа (Cloud + Edge)**

**1. Frontend**

* **Технологии**: Next.js + Tailwind CSS + i18n для поддержки мультиязычности
* **Дополнительно**: PWA, офлайн-режим, edge-first рендеринг с использованием Astro или Qwik

**2. Backend**

* **API**: FastAPI (Python)
* **База данных**: PostgreSQL с расширением pgvector для поддержки векторного поиска
* **Кэширование**: Redis (например, Upstash)
* **Семантический поиск**: ChromaDB для гибридного поиска

**3. AI и ML**

* **Облачные сервисы**: Google Cloud AutoML и Vertex AI для обучения и развертывания моделей
* **Edge-инференс**: Развёртывание моделей на периферийных устройствах с использованием llama.cpp или Mixtral 8x22B
* **Интеграция**: Поддержка как облачных, так и локальных AI-моделей для гибкости и отказоустойчивости

**4. Edge AI**

* **Платформы**: Gcore Inference at the Edge, Google LLM Inference API
* **Контейнеризация**: Docker/Kubernetes для развёртывания моделей на edge-устройствах

**5. Безопасность и конфиденциальность**

* **Защита**: Cloudflare WAF для защиты от веб-угроз
* **Приватность**: Использование дифференциальной приватности и клиентского шифрования с Web Crypto API
* **Управление доступом**: Google Cloud IAM для детального контроля прав доступа

**6. DevOps и мониторинг**

* **CI/CD**: GitHub Actions, Tekton, ArgoCD для автоматизации процессов
* **Мониторинг**: Prometheus и Grafana для отслеживания производительности и безопасности
* **MLOps**: Kubeflow для управления жизненным циклом моделей машинного обучения[Wikipedia+2Wikipedia+2Wikipedia+2](https://ru.wikipedia.org/wiki/Kubeflow?utm_source=chatgpt.com)

**7. Интеграции и события**

* **Событийная архитектура**: Поддержка webhooks, интеграция с мессенджерами (Telegram, Slack)
* **Автоматизация**: Cloud Scheduler для планирования задач[Reddit](https://www.reddit.com/r/Cloud/comments/1c222m8?utm_source=chatgpt.com)

**8. Мультиязычность и мультиарендность**

* **Локализация**: i18n для поддержки различных языков
* **Мультиарендность**: Supabase Row Level Security для изоляции данных между арендаторами

**9. SEO и SSR**

* **Оптимизация**: Next.js с встроенной поддержкой SEO и серверного рендеринга

**✅ Преимущества предложенной архитектуры**

* **Гибридный подход**: Сочетание облачных и edge-решений обеспечивает высокую производительность и отказоустойчивость
* **Open-source**: Использование открытых решений снижает зависимость от поставщиков и позволяет гибко настраивать систему
* **Безопасность**: Интеграция современных методов защиты данных и управления доступом
* **Масштабируемость**: Архитектура легко масштабируется как вертикально, так и горизонтально
* **Гибкость разработки**: Модульный подход облегчает внедрение новых функций и технологий

**🔧 Рекомендации по внедрению**

1. **Развёртывание edge-инференса**: Настроить окружение для запуска моделей на периферийных устройствах с использованием Docker и Kubernetes
2. **Интеграция с Vertex AI**: Использовать возможности Vertex AI для обучения и управления моделями
3. **Настройка мониторинга**: Внедрить Prometheus и Grafana для постоянного отслеживания состояния системы
4. **Обеспечение безопасности**: Настроить Cloudflare WAF и IAM для защиты и управления доступом[GCP Masters](https://gcpmasters.in/google-cloud-ai/?utm_source=chatgpt.com)

Данная архитектура обеспечивает баланс между производительностью, безопасностью и гибкостью, что делает её подходящей для широкого спектра AI-приложений, включая чат-ботов, рекомендательные системы и аналитические платформы.

Вот итоговая таблица архитектурных компонентов гибридной AI-платформы с поддержкой edge-инференса и использованием open-source решений:

| Категория | Компонент/Технология | Назначение

Вот итоговая таблица архитектурных компонентов гибридной AI-платформы с учетом edge-инференса и open-source решений. Она отражает ключевые подсистемы, технологии и их роль в платформе:

| **Категория** | **Компоненты / Технологии** | **Назначение** |
| --- | --- | --- |
| 💠 Ядро AI-инференса | - llama.cpp - Ollama / vLLM - Mixtral 8x22B | Локальный запуск моделей (edge AI), inference на устройстве или в локальном окружении |
| ☁️ Облачный AI | - Hugging Face Inference API - Triton Inference Server - OpenAI API | Облачный inference для тяжёлых моделей и fallback-обработка |
| 🧠 Векторное хранилище | - ChromaDB - Qdrant - Weaviate | Семантический поиск, хранение эмбеддингов, поддержка RAG и поиска по знаниям |
| 🧩 RAG/Обработка знаний | - LangChain / LlamaIndex - Milvus + FastAPI | Генерация ответов с контекстом, цепочки вызовов моделей, документ-ориентированный AI |
| 🗂️ Backend/API | - FastAPI - .NET Core Web API - Supabase (как open-source BaaS) | REST/GraphQL API, маршрутизация, бизнес-логика |
| 🌐 Frontend | - Next.js / React - Firebase Hosting (опционально) - Vite + Tailwind | UI, клиентская логика, SSR/SSG |
| 🔐 Авторизация/Регистрация | - Supabase Auth - Firebase Auth - Keycloak / Auth0 | Поддержка email, телефона, OAuth, KYC и RBAC |
| 🧩 Edge-инфраструктура | - Cloudflare Workers / Pages - Vercel Edge Functions - WASM | Ведение логики ближе к пользователю, минимизация задержек |
| ⚙️ DevOps/CI-CD | - GitHub Actions - Tekton Pipelines - ArgoCD | Автоматизация сборки, тестирования, деплоя |
| 📈 Мониторинг и логгинг | - Prometheus + Grafana - OpenTelemetry - Loki | Отслеживание метрик, логов и поведения моделей |
| 🔍 Интерпретируемость AI | - Captum (PyTorch) - SHAP / LIME | Интерпретация моделей, отладка, соответствие требованиям доверия к AI |
| 🔄 Data Pipeline & Augmentation | - Apache Airflow - Label Studio - Albumentations / DVC | Организация обработки данных, синтетическая генерация и аннотация |
| 🔐 Безопасность | - WAF (Cloudflare) - JWT / OAuth2 - Верификация и KYC | Защита API, ролевая модель доступа, контроль поставщиков |
| 🌍 Мультиязычность | - i18next - LLM translation (Gemini, GPT) - локализация через crowdin | Поддержка нескольких языков UI и контента |
| 🔎 SEO/индексация | - Next.js SSR/SSG - Structured Data (JSON-LD) - sitemap.xml / robots.txt | Видимость в поиске, оптимизация загрузки, динамическая индексация |
| 📊 Аналитика и CRM | - PostHog / Plausible - Supabase Realtime - Метрики поставщиков и пользователей | Сбор действий, аналитика вовлеченности и эффективности |

🔧 Примечания:

* Компоненты можно адаптировать в зависимости от конкретного хостинга (GCP, AWS, локально).
* Сочетание edge-инференса (llama.cpp, Ollama) и облачных fallback-инференсов обеспечивает баланс гибкости, приватности и надежности.
* Основной акцент — на open-source, чтобы минимизировать зависимость от облаков и обеспечить расширяемость.

Если вы хотите, могу собрать эту таблицу также в формате Excel, PDF или Miro-совместимой схемы.

На основе вашего запроса и текущих технологий, предлагаю следующую архитектуру гибридной AI-платформы с упором на edge-инференс и open-source решения:

**🧠 Edge-инференс: локальные LLM без облака**

**1. llama.cpp**  
Лёгкий и высокопроизводительный движок для запуска моделей LLaMA и других в C/C++. Поддерживает квантование до 2 бит, работает на CPU и GPU, включая Apple Silicon и NVIDIA CUDA. [GitHub](https://github.com/ggerganov/llama.cpp?utm_source=chatgpt.com)

**2. LlamaEdge**  
Обёртка над llama.cpp с OpenAI-совместимым API, встроенным UI и поддержкой моделей в формате GGUF. Позволяет быстро развернуть локальные LLM-приложения с минимальной настройкой. [GitHub+1GitHub+1](https://github.com/ggerganov/llama.cpp?utm_source=chatgpt.com)

**3. Mixtral 8x22B**  
Модель Mixture-of-Experts (MoE) от Mistral AI, оптимизированная для масштабируемого инференса. Поддерживается NVIDIA NIM и может быть развернута с использованием MoE-Lightning для эффективной работы на ограниченных GPU. [NVIDIA Developer+1arXiv+1](https://developer.nvidia.com/blog/mistral-large-and-mixtral-8x22b-llms-now-powered-by-nvidia-nim-and-nvidia-api/?utm_source=chatgpt.com)

**🔍 Семантический поиск и RAG**

**1. ChromaDB**  
Open-source векторная база данных, ориентированная на LLM-приложения. Поддерживает встроенные эмбеддинги (OpenAI, Cohere, SentenceTransformers) и интеграции с LangChain и LlamaIndex. [33rd Square](https://www.33rdsquare.com/semantic-search-applications/?utm_source=chatgpt.com)

**2. txtai**  
Лёгкий фреймворк для семантического поиска и RAG, поддерживает локальные модели и эффективную работу с большими наборами данных. [Reddit](https://www.reddit.com/r/Python/comments/1ea6t89?utm_source=chatgpt.com)

**⚙️ Инфраструктура и DevOps**

**1. LocalAI**  
Платформа для локального запуска AI-моделей, включая текст, речь и изображения. Поддерживает распределённый инференс с использованием llama.cpp.

**2. WebLLM**  
Инференс LLM непосредственно в браузере с использованием WebGPU. Поддерживает модели LLaMA, Mistral и другие, обеспечивая быстрый отклик без серверной части. [Slashdot](https://slashdot.org/software/ai-inference/for-mixtral-8x22b/?utm_source=chatgpt.com)

**📦 Рекомендованная архитектура**

graph TD

A[Пользователь] --> B[Frontend (Next.js / React)]

B --> C[API Gateway (FastAPI / .NET Core)]

C --> D[Edge-инференс (llama.cpp / LlamaEdge)]

C --> E[Семантический поиск (ChromaDB / txtai)]

C --> F[Облачный инференс (OpenAI API / Triton)]

E --> G[Векторное хранилище]

F --> H[Облачные модели (Mixtral 8x22B)]

**✅ Преимущества предложенного подхода**

* **Приватность и контроль**: Локальный инференс обеспечивает защиту данных и независимость от облачных провайдеров.
* **Гибкость**: Возможность переключения между локальными и облачными моделями в зависимости от задач и ресурсов.
* **Масштабируемость**: Использование MoE-моделей и распределённого инференса позволяет эффективно обрабатывать большие объёмы данных.
* **Open-source**: Все компоненты доступны с открытым исходным кодом, что обеспечивает прозрачность и возможность кастомизации.[arXiv+1arXiv+1](https://arxiv.org/abs/2401.14361?utm_source=chatgpt.com)

Если вам нужна более подробная информация или помощь в реализации данной архитектуры, пожалуйста, дайте знать.

**🧠 Обзор архитектуры AI-агрегатора**

Архитектура построена по принципу многоуровневой модульности, обеспечивая гибкость, отказоустойчивость и расширяемость.

**1. Входной уровень (Frontend & API Gateway)**

* **Пользовательский интерфейс (UI):** Веб-приложение на базе Next.js или React, обеспечивающее интерактивный доступ к AI-сервисам.
* **API Gateway:** FastAPI или .NET Core Web API, выполняющий маршрутизацию запросов, аутентификацию и агрегацию данных.

**2. Уровень оркестрации и агрегации**

* **Сервис-агрегатор:** Микросервис, реализующий паттерн агрегатора, объединяющий ответы от различных AI-сервисов и обеспечивающий единый интерфейс для клиентов.
* **Сервисная шина (Service Bus):** RabbitMQ или Apache Kafka для асинхронного взаимодействия между микросервисами.

**3. Уровень AI-инференса**

* **Локальный инференс (Edge):** llama.cpp или LlamaEdge для выполнения моделей на устройствах пользователя или в локальной инфраструктуре.
* **Облачный инференс:** Интеграция с облачными AI-провайдерами (например, OpenAI, Hugging Face) для выполнения ресурсоёмких задач.

**4. Уровень данных и знаний**

* **Векторное хранилище:** ChromaDB или Qdrant для хранения эмбеддингов и реализации семантического поиска.
* **База знаний:** Интеграция с системами управления знаниями и документами для поддержки RAG.

**5. Уровень управления и мониторинга**

* **DevOps и CI/CD:** GitHub Actions или ArgoCD для автоматизации развертывания и обновления сервисов.
* **Мониторинг и логирование:** Prometheus и Grafana для сбора метрик и визуализации состояния системы.

**🔄 Поток обработки запроса**

1. Пользователь отправляет запрос через веб-интерфейс.
2. API Gateway аутентифицирует запрос и перенаправляет его в сервис-агрегатор.
3. Сервис-агрегатор определяет, какие AI-сервисы необходимы для обработки запроса, и направляет соответствующие подзапросы.
4. Результаты от AI-сервисов агрегируются и возвращаются пользователю через API Gateway.

**📌 Преимущества предложенной архитектуры**

* **Масштабируемость:** Микросервисная структура позволяет легко добавлять новые AI-сервисы и масштабировать существующие.
* **Гибкость:** Поддержка как локального, так и облачного инференса обеспечивает оптимальный баланс между производительностью и затратами.
* **Расширяемость:** Интеграция с различными источниками данных и AI-моделями позволяет быстро адаптироваться к новым требованиям.
* **Надёжность:** Использование асинхронного взаимодействия и мониторинга повышает устойчивость системы к сбоям.

Если вам нужна визуальная схема архитектуры или более подробная информация о конкретных компонентах, пожалуйста, сообщите, и я предоставлю дополнительные материалы.

**🧭 Уровень 1: Диаграмма контекста**

На этом уровне отображаются основные участники системы и их взаимодействие с AI-агрегатором.

**Участники:**

* **Пользователи:** конечные пользователи, взаимодействующие с системой через веб-интерфейс или мобильное приложение.
* **AI-агрегатор услуг:** центральная система, обрабатывающая запросы пользователей и взаимодействующая с различными AI-сервисами.
* **Внешние AI-сервисы:** облачные и локальные AI-сервисы, предоставляющие различные функциональности (например, обработка естественного языка, генерация изображений и т.д.).

**Взаимодействие:**

* Пользователи отправляют запросы в AI-агрегатор.
* AI-агрегатор обрабатывает запросы, при необходимости обращаясь к внешним AI-сервисам.
* Результаты обработки возвращаются пользователям.

**🧱 Уровень 2: Диаграмма контейнеров**

Этот уровень показывает основные контейнеры внутри AI-агрегатора и их взаимодействие.

**Контейнеры:**

* **Веб-интерфейс (Frontend):** предоставляет пользователям доступ к функциональности AI-агрегатора.
* **API Gateway:** обрабатывает входящие запросы, выполняет аутентификацию и маршрутизацию.
* **Сервис-агрегатор:** координирует выполнение задач, обращаясь к соответствующим AI-сервисам.
* **AI-сервисы:** выполняют конкретные задачи, такие как обработка текста, изображений и т.д.
* **База данных:** хранит информацию о пользователях, запросах и результатах.

**Взаимодействие:**

* Frontend взаимодействует с API Gateway.
* API Gateway перенаправляет запросы в Сервис-агрегатор.
* Сервис-агрегатор обращается к соответствующим AI-сервисам и базе данных.
* Результаты возвращаются через API Gateway в Frontend.

**⚙️ Уровень 3: Диаграмма компонентов**

На этом уровне детализируются компоненты внутри каждого контейнера.

**Примеры компонентов:**

* **Frontend:**
  + Компонент пользовательского интерфейса.
  + Компонент обработки пользовательских событий.[YesChat+3Microsoft Learn+3AIPURE+3](https://learn.microsoft.com/ru-ru/azure/architecture/browse/?azure_categories=ai-machine-learning&utm_source=chatgpt.com)
* **API Gateway:**
  + Компонент аутентификации и авторизации.
  + Компонент маршрутизации запросов.
* **Сервис-агрегатор:**
  + Компонент управления задачами.
  + Компонент взаимодействия с AI-сервисами.
* **AI-сервисы:**
  + Компонент обработки текста.
  + Компонент генерации изображений.[Freepik](https://ru.freepik.com/premium-ai-image/diagram-cloud-architecture-with-different-service-models-iaas-paas-saas_307591897.htm?utm_source=chatgpt.com)
* **База данных:**
  + Компонент хранения пользовательских данных.
  + Компонент хранения результатов обработки.

**🧩 Уровень 4: Диаграмма кода**

На этом уровне можно представить структуру кода для каждого компонента, включая классы, методы и их взаимодействие. Это может быть полезно для разработчиков при реализации системы.

**📌 Примечания**

* Модель C4 позволяет гибко адаптировать визуализацию архитектуры под различные аудитории: от бизнес-аналитиков до разработчиков.
* Для создания диаграмм можно использовать инструменты, такие как [Lucidscale](https://lucid.co/ru/lucidscale), [draw.io](https://app.diagrams.net/) или [PlantUML](https://plantuml.com/ru/).
* Автоматизация создания диаграмм возможна при наличии репозитория архитектурных данных, как описано в статье на Хабре: [Простая Enterprise Architecture](https://habr.com/ru/articles/730868/).[Wikipedia](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_C4?utm_source=chatgpt.com)[Lucid Software](https://lucid.co/ru/lucidscale?utm_source=chatgpt.com)[Habr](https://habr.com/ru/articles/730868/?utm_source=chatgpt.com)

Если вам требуется помощь в создании конкретных диаграмм или выборе инструментов для визуализации, пожалуйста, сообщите, и я предоставлю дополнительные рекомендации.

📘 Диаграмма классов (PlantUML)

@startuml

package "Frontend" {

class UserInterface {

+sendRequest()

+displayResponse()

}

}

package "API Gateway" {

class APIGateway {

+authenticateRequest()

+routeRequest()

}

}

package "Aggregator Service" {

class AggregatorService {

+processRequest()

+aggregateResponses()

}

}

package "AI Services" {

class TextProcessingService {

+processText()

}

class ImageGenerationService {

+generateImage()

}

}

package "Data Layer" {

class Database {

+storeData()

+retrieveData()

}

}

UserInterface --> APIGateway : sendRequest()

APIGateway --> AggregatorService : routeRequest()

AggregatorService --> TextProcessingService : call()

AggregatorService --> ImageGenerationService : call()

AggregatorService --> Database : store/retrieve

APIGateway --> UserInterface : sendResponse()

@enduml

**Если вам необходима диаграмма другого типа (например, последовательностей или компонентов) или более подробная детализация отдельных компонентов системы, пожалуйста, сообщите, и я предоставлю соответствующую информацию.**

**🛠️ Инструменты для генерации диаграмм**

Для создания и визуализации диаграмм на основе кода можно использовать следующие инструменты:

* **PlantUML**: позволяет создавать различные типы UML-диаграмм с использованием простого текстового описания.[getanalyst.ru](https://getanalyst.ru/database/create-uml-in-chatGPT?utm_source=chatgpt.com)
* **Code to Diagrams Generator**: инструмент на основе ИИ, который преобразует код в диаграммы, поддерживает различные типы диаграмм и форматы. [AI Graph Maker+2YesChat+2YesChat+2](https://www.yeschat.ai/ru/gpts-2OToA7yZUJ-GPTChart?utm_source=chatgpt.com)
* **AI Graph Maker**: позволяет создавать интерактивные графики и диаграммы с помощью ИИ, поддерживает экспорт в форматы PNG, SVG и Mermaid. [AI Graph Maker](https://aigraphmaker.net/ru?utm_source=chatgpt.com)

**🧩 Диаграмма компонентов (UML Component Diagram)**

Эта диаграмма иллюстрирует основные компоненты системы и их взаимодействие.

@startuml

package "Пользовательский интерфейс" {

[Web UI] as UI

}

package "API Gateway" {

[API Gateway] as APIGW

}

package "Сервис-агрегатор" {

[Aggregator Service] as AGG

}

package "AI-Сервисы" {

[Text Processing Service] as TextAI

[Image Generation Service] as ImageAI

}

package "Слой данных" {

[Database] as DB

}

UI --> APIGW : HTTP-запрос

APIGW --> AGG : REST API

AGG --> TextAI : gRPC

AGG --> ImageAI : gRPC

AGG --> DB : SQL

APIGW --> UI : HTTP-ответ

@enduml

🔄 Диаграмма последовательностей (UML Sequence Diagram)

@startuml

actor User

participant UI

participant APIGW as "API Gateway"

participant AGG as "Aggregator Service"

participant TextAI as "Text Processing Service"

participant ImageAI as "Image Generation Service"

database DB

User -> UI : Ввод запроса

UI -> APIGW : Отправка запроса

APIGW -> AGG : Перенаправление запроса

AGG -> TextAI : Обработка текста

TextAI --> AGG : Результат обработки

AGG -> ImageAI : Генерация изображения

ImageAI --> AGG : Сгенерированное изображение

AGG -> DB : Сохранение результатов

AGG --> APIGW : Ответ с результатами

APIGW --> UI : Отправка ответа

UI --> User : Отображение результатов

@enduml

**🧱 Детализация ключевых компонентов**

**1. Web UI (Пользовательский интерфейс)**

* **Функции:**
  + Предоставляет интерфейс для ввода запросов и отображения результатов.
  + Обеспечивает взаимодействие с API Gateway через HTTP.

**2. API Gateway**

* **Функции:**
  + Обрабатывает входящие HTTP-запросы от Web UI.
  + Маршрутизирует запросы к соответствующим сервисам.
  + Обеспечивает безопасность и аутентификацию.

**3. Aggregator Service (Сервис-агрегатор)**

* **Функции:**
  + Координирует взаимодействие между различными AI-сервисами.
  + Агрегирует результаты от TextAI и ImageAI.
  + Сохраняет результаты в базе данных.

**4. Text Processing Service (Сервис обработки текста)**

* **Функции:**
  + Обрабатывает текстовые запросы пользователя.
  + Может использовать модели NLP для анализа и генерации текста.

**5. Image Generation Service (Сервис генерации изображений)**

* **Функции:**
  + Генерирует изображения на основе текстовых описаний или других входных данных.
  + Может использовать модели генеративного ИИ, такие как GAN или Diffusion Models.

**6. Database (База данных)**

* **Функции:**
  + Хранит результаты обработки запросов.
  + Обеспечивает возможность последующего анализа и отчетности.

Эти диаграммы и описания предоставляют подробное представление об архитектуре и взаимодействии компонентов универсального AI-агрегатора услуг. Если вам нужны дополнительные детали или помощь в реализации конкретных частей системы, пожалуйста, сообщите.

**📁 Структура документа Word: «AI-агрегатор услуг — Архитектурная спецификация»**

Документ будет структурирован по модели C4 и включать следующие разделы:

1. **Введение**
   * Цель документа
   * Область применения
   * Определения и сокращения
2. **Обзор архитектуры**
   * Описание гибридной архитектуры (Cloud + Edge)
   * Принципы проектирования
   * Технологический стек
3. **Диаграммы архитектуры**
   * Диаграмма контекста (Context Diagram)
   * Диаграмма контейнеров (Container Diagram)
   * Диаграмма компонентов (Component Diagram)
   * Диаграмма последовательностей (Sequence Diagram)
4. **Описание компонентов**
   * Frontend (Web UI)
   * API Gateway
   * Aggregator Service
   * AI-сервисы (Text Processing, Image Generation и др.)
   * База данных
   * Службы мониторинга и логирования[vc.ru+4Medium+4Microsoft Learn+4](https://vnikitinsky.medium.com/5-%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%BE%D0%B2-%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B9-%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9-%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D1%8B-9af6869a280b?utm_source=chatgpt.com)
5. **Интеграции и взаимодействие**
   * Взаимодействие между компонентами
   * Интеграция с внешними сервисами
   * Обработка ошибок и отказоустойчивость
6. **Безопасность и соответствие требованиям**
   * Аутентификация и авторизация
   * Шифрование данных
   * Соответствие нормативным требованиям[Microsoft Learn+1arXiv+1](https://learn.microsoft.com/ru-ru/power-platform/architecture/reference-architectures/ai-document-processing?utm_source=chatgpt.com)
7. **Приложения**
   * Глоссарий
   * Список используемых технологий и инструментов
   * Ссылки на внешние ресурсы[Habr](https://habr.com/ru/companies/bcs_company/articles/651765/?utm_source=chatgpt.com)

**📊 Дополнительные материалы**

* **Excel-файл**: Таблицы с описанием компонентов, их характеристик, взаимодействий и зависимостей.
* **PDF-файл**: Версия документа Word в формате PDF для удобства просмотра и распространения.
* **Miro-совместимая схема**: Визуальная схема архитектуры, экспортированная в формат, совместимый с Miro, для совместной работы и обсуждения.