**1. Архитектурная спецификация**

**Архитектурная спецификация AI чат-бота для универсального агрегатора услуг**

**Введение**

Данный документ описывает архитектуру AI чат-бота, предназначенного для универсального агрегатора услуг. Основные цели — минимизация затрат, масштабируемость, безопасность, простота разработки, AI-first подход, мультиязычность и SEO.

**Ключевые требования**

* **Минимизация затрат:** Использование бесплатных и low-cost технологий, open-source моделей.
* **Масштабируемость:** Поддержка роста пользователей, регионов и нагрузки.
* **Безопасность:** Защита данных, API, антифрод, соблюдение GDPR.
* **Простота разработки:** Lego-стиль, модульность, расширяемость.
* **AI-first:** Автоматизация генерации кода, аналитики, модерации.
* **Мультиязычность и мультирынки:** Поддержка i18n, мультиарендность.
* **SEO и SSR:** Встроенная поддержка для поисковой оптимизации.
* **Интеграции:** Webhooks, Telegram/Slack боты, event-driven архитектура.

**Архитектурный обзор**

Система построена по микросервисному принципу с возможностью старта с монорепозитория. Основные компоненты:

* **Frontend:** Next.js + Tailwind + i18n — обеспечивает SSR, SEO, мультиязычность.
* **Backend API:** Python + FastAPI — бизнес-логика, API, AI-интеграция.
* **Базы данных:** PostgreSQL (JSONB, полнотекстовый поиск), Redis (кэширование).
* **Semantic Search:** ChromaDB / FAISS — векторный поиск.
* **AI-модели:** Hugging Face API, llama.cpp, Mixtral 8x22B — генерация ответов, модерация, антифрод.
* **Edge-инференс:** Локальный инференс на edge-устройствах для снижения задержек.
* **Интеграции:** Webhooks, Telegram/Slack боты.
* **DevOps:** Docker, GitHub Actions, Tekton, ArgoCD — CI/CD и мониторинг.
* **Безопасность:** Cloudflare WAF, TensorFlow Privacy, Web Crypto API.

**Выбор технологий и обоснование**

* **FastAPI и Python:** Высокая производительность, простота интеграции AI.
* **Next.js:** SEO, SSR, PWA, поддержка i18n.
* **PostgreSQL:** Универсальность, масштабируемость, JSONB.
* **Redis:** Быстрый доступ, кэширование.
* **ChromaDB/FAISS:** Эффективный semantic search.
* **Hugging Face и llama.cpp:** Гибкость, open-source, поддержка локального инференса.
* **Cloudflare:** Безопасность и WAF.
* **GitHub Actions и Tekton:** Автоматизация и масштабируемость DevOps.

**Безопасность и приватность**

* Дифференциальная приватность для защиты данных.
* Клиентское шифрование через Web Crypto API.
* Защита API через Cloudflare WAF.
* Соответствие GDPR и другим регуляциям.

**DevOps и CI/CD**

* Контейнеризация через Docker.
* Автоматическое тестирование и деплой через GitHub Actions или Tekton + ArgoCD.
* Мониторинг с Prometheus и Grafana.

**Расширяемость и интеграции**

* Lego-стиль архитектуры позволяет легко заменять и добавлять модули.
* Поддержка webhooks и ботов для интеграции с внешними сервисами.
* Мультиарендность через Supabase RLS.

**2. Таблица компонентов и их назначение**

| **Компонент** | **Назначение** | **Технологии / Инструменты** |
| --- | --- | --- |
| Frontend | UI/UX, SSR, SEO, мультиязычность | Next.js, Tailwind, i18n |
| Backend API | Обработка запросов, бизнес-логика, AI-интеграция | Python, FastAPI |
| База данных | Хранение данных, JSONB, полнотекстовый поиск | PostgreSQL |
| Кэширование | Быстрый доступ к часто используемым данным | Redis |
| Semantic Search | Поиск по смыслу, векторный поиск | ChromaDB / FAISS / Weaviate |
| AI-модели | Генерация ответов, модерация, антифрод | Hugging Face, llama.cpp, Mixtral |
| Edge-инференс | Локальный AI-инференс для снижения задержек и повышения отказоустойчивости | llama.cpp, WebAssembly |
| Интеграции | Внешние сервисы, уведомления, мессенджеры | Webhooks, Telegram Bot, Slack Bot |
| DevOps | CI/CD, контейнеризация, мониторинг | Docker, GitHub Actions, Tekton, Prometheus |
| Безопасность | Защита API, шифрование, антифрод, приватность | Cloudflare WAF, TensorFlow Privacy, Web Crypto API |
| Мультитенантность | Поддержка нескольких рынков, кастомизация | Supabase RLS |

**3. UML-диаграмма компонентов (текстовое описание для визуализации)**

* **Пользовательский интерфейс (Next.js)**  
  ↔ взаимодействует с →
* **Backend API (FastAPI)**  
  ↔ взаимодействует с →
* **PostgreSQL** (хранение данных)  
  ↔ взаимодействует с →
* **Redis** (кэширование)
* **Backend API** ↔ вызывает → **AI-модели** (Hugging Face API / локальный llama.cpp)
* **Backend API** ↔ обращается к → **Semantic Search** (ChromaDB/FAISS)
* **Backend API** ↔ взаимодействует с → **Интеграциями** (webhooks, Telegram/Slack боты)
* **DevOps** (Docker, GitHub Actions, Tekton) поддерживает весь процесс разработки и деплоя
* **Безопасность** (Cloudflare WAF, TensorFlow Privacy) обеспечивает защиту API и данных

**4. Диаграмма последовательностей (текстовое описание)**

**Сценарий: Пользовательский запрос в чат-бот**

1. Пользователь отправляет запрос через UI (Next.js).
2. UI передаёт запрос в Backend API (FastAPI).
3. Backend API проверяет безопасность (Cloudflare WAF).
4. Backend API обращается к Semantic Search (ChromaDB/FAISS) для поиска релевантных данных.
5. Backend API вызывает AI-модель (локальный edge-инференс llama.cpp или Hugging Face API) для генерации ответа.
6. Ответ кэшируется в Redis для ускорения последующих запросов.
7. Backend API отправляет ответ обратно в UI.
8. UI отображает ответ пользователю.
9. При необходимости Backend API отправляет уведомления через webhooks (Telegram/Slack).
10. Все действия логируются и мониторятся (Prometheus, Grafana).