ChatGPT

Техническое задание на разработку универсального агрегатора услуг с AI и ChatBot-ассистентом

Описание проекта

Проект представляет собой **универсальный агрегатор услуг** (юридических, бытовых, образовательных, ИТ и смешанных), объединивший в себе функциональность маркетплейса и интеллектуального помощника на базе ИИ. Система обеспечивает поиск, выбор и заказ услуг разных категорий, а также консультации через многоязычный **ChatBot-ассистент**. Приложение будет модульным и масштабируемым, с современным веб-интерфейсом (SSR/SEO-оптимизированный) и мобильным приложением (PWA или кроссплатформенный Flutter). Серверная часть размещается на VPS (Linux Ubuntu) с гибридной архитектурой «облако+edge».

Ключевые особенности:
- **Категории услуг:** правовые, бытовые, образовательные, ИТ и т.д.
- **Модули AI:** интеллектуальный поиск (семантический), чат-бот, персональные рекомендации, анализ отзывов, генерация контента.
- **Мультиязычность:** интерфейс и контент на русском, английском, казахском и узбекском (с перспективой расширения).
- **Безопасность:** WAF на уровне Cloudflare, клиентское шифрование (Web Crypto API), конфиденциальная обработка данных (TensorFlow Privacy).
- **Интеграции:** платёжные шлюзы (Stripe, ЮKassa, PayPal и др.), мессенджеры (Telegram, WhatsApp via Bot API), email-уведомления, событийная архитектура (webhooks).
- **DevOps:** CI/CD с GitHub Actions, контейнеризация (Docker), оркестрация (ArgoCD/Tekton), мониторинг (Prometheus + Grafana).

Цели и задачи проекта

- **Удобство пользователей:** собрать в одном месте различные услуги, упростить поиск и заказ, предоставить персональные рекомендации.
- **Интеллектуальность:** задействовать ИИ для автоматизации консультаций, ответов, анализа данных (отзывы, поведение) и генерации контента.
- **Масштабируемость и гибкость:** обеспечить модульную архитектуру, способную легко расширяться новыми категориями услуг и функциями.
- **Мультиязычность:** поддержка нескольких языков позволит выйти на рынки СНГ и СНГ.
- **SEO-оптимизация:** Web-версия должна быть SSR/SSG-рендерингом, чтобы поисковики индексировали контент эффективно. Использование Next.js обеспечит **готовый HTML-код на сервере** для каждого запроса [1], что значительно улучшает индексируемость и Core Web Vitals.

Задачи: организовать разработку фронтенда и мобильного приложения, настроить бэкенд и БД, интегрировать AI-модули, реализовать все пользовательские сценарии (поиск, заказ, чат, профиль и т.д.), обеспечить безопасность и надежность, настроить CI/CD и мониторинг.

Архитектура системы

Система строится по гибридной архитектуре «Edge+Cloud». **Edge-устройства** (региональные серверы, CDN, кэширующие прокси) обрабатывают критически важные запросы ближе к пользователю, снижая задержку и уменьшая трафик к центральному узлу. **Облачный** компонент предоставляет централизованное хранилище, мощность для AI-инференса и глобальную координацию [2] . Такая гибридная архитектура позволяет достигать низкой задержки и высокой доступности одновременно [2] . В системе используется **микросервисный подход**: каждый сервис (например, авторизация, обработка платежей, поиск, AI-инференс) разворачивается и масштабируется независимо. Это повышает гибкость и отказоустойчивость [3] .

Схематично система включает следующие уровни:
- **Клиентская часть (UI):** Web-приложение на Next.js (SSR) для SEO и отзывчивого интерфейса, PWA или кроссплатформенный Flutter для мобильных устройств с поддержкой офлайн-режима.
- **API-сервер (Backend):** FastAPI (Python) с REST/GraphQL интерфейсом для бизнес-логики.
- **База данных:** PostgreSQL (SQL хранилище) и Redis (кэш и очередь сообщений).
- **Поисковый движок:** векторная (семантическая) поисковая система на базе **ChromaDB + pgvector**, позволяющая искать по смыслу и генерировать рекомендации.
- **AI-инфраструктура:** на пограничных серверах (edge) – LLM на `llama.cpp`/Mixtral для локального инференса, в облаке – API Hugging Face/OpenRouter для сложных задач. ChatBot-ассистент обрабатывает естественный язык и формирует ответы.
- **Безопасность:** Cloudflare WAF защищает от атак OWASP Top-10, а шифрование через Web Crypto API обеспечивает безопасность данных на клиенте. Конфиденциальность при машинном обучении усиливается с помощью TensorFlow Privacy [4] .
- **DevOps/CI-CD:** репозиторий на GitHub с Actions для сборки, Docker-контейнеры для каждого сервиса, оркестрация через ArgoCD/Tekton. Система мониторинга с Prometheus + Grafana собирает метрики и алерты.

Для обработки **асинхронных событий** (Webhooks, очереди на Redis/Kafka) используется событийно-ориентированная архитектура. События от платежных шлюзов, мессенджеров или внутренних сервисов обрабатываются приёмниками, запускающими соответствующие workflow. Такой подход хорошо подходит для гибридных систем с требованием моментальной реакции [5] .

Компоненты системы

Компонент	Функции/Назначение
Web-фронтенд (Next.js)	SSR/SSG-рендеринг страниц, SEO-оптимизация, UI/UX, локализация с i18n.
Мобильное приложение	PWA/Flutter клиент для смартфонов и планшетов, офлайн-доступ, push-уведомления.
Backend API (FastAPI)	Обработка запросов клиентов, бизнес-логика, интеграции с БД и внешними сервисами.
База данных (PostgreSQL)	Сохранение пользователи, услуг, заказов, отзывов. Поддержка pgvector для семантических векторов.
Кэш и очередь (Redis)	Кэширование частых запросов, сессии, временные данные; брокер сообщений для асинхронных задач.

Компонент	Функции/Назначение
Поисковый модуль	Семантический поиск и генерация рекомендаций (ChromaDB + pgvector).
AI-модуль (Edge)	Локальный инференс LLM (`llama.cpp/Mixtral`) для чат-бота, предварительной обработки запросов и офлайн-работы.
AI-модуль (Cloud)	Облачные ML-инференсы через Hugging Face API/OpenRouter для сложных анализов, генерации контента.
Платежный модуль	Интеграция с платёжными шлюзами (Stripe, YooKassa, PayPal и др.) для приёма платежей.
Мессенджер-боты	Telegram/WhatsApp-боты (Bot API/Webhook) для взаимодействия пользователей и уведомлений.
Email-сервис	Отправка email-уведомлений (SMTP/сервисы рассылки) для подтверждений, рассылок.
Система безопасности	Файрвол (Cloudflare WAF), серверные SSL/TLS, клиентское шифрование (Web Crypto), защита данных (TensorFlow Privacy).
DevOps/CI-CD	CI/CD конвейер (GitHub Actions), контейнеры Docker, оркестрация (ArgoCD/Tekton), мониторинг (Prometheus+Grafana).

Функциональные модули и AI-функции

- **Поиск и каталог услуг:** пользователи могут искать услуги по ключевым словам и фильтрам, а также по смыслу благодаря семантическому поиску (ChromaDB + pgvector). Система учитывает предпочтения и поведение пользователя, чтобы выдавать релевантные варианты. Семантический поиск позволяет анализировать контекст запросов и персонализировать рекомендации – по исследованиям, **персональные рекомендации** повышают вероятность покупки и лояльность клиентов [6].
- **ChatBot-ассистент:** многоязычный чат-бот, способный вести диалог, отвечать на вопросы об услугах, помогать с оформлением заказа и давать общие консультации. Бот использует LLM (локально или в облаке) и семантический поиск для понимания запросов. Семантические технологии позволяют боту точно интерпретировать сложные вопросы и выдавать полезные ответы [7]. Важно, что подобный интент-ориентированный подход улучшает качество ответов по сравнению с традиционными системами.
- **Автогенерация ответов:** система автоматически формирует ответы на типовые вопросы (например, часто задаваемые) с помощью языковых моделей. Также предусмотрена генерация описаний новых услуг и SEO-контента: исследования показали, что при подходе семантического SEO контент создается «за рамками простых ключевых слов» и лучше соответствует запросам пользователей, улучшая видимость сайта [8].
- **Анализ отзывов и фидбека:** сбор и автоматизированный анализ отзывов клиентов с применением методов обработки естественного языка. Это помогает выявлять проблемы, аномалии и тренды качества услуг.
- **Персональные рекомендации:** на основе истории взаимодействий и семантического профилирования пользователя система предлагает подходящие услуги. Например, при наличии паттернов запроса мы можем выводить «Вам также могут понравиться...», повышая конверсию. Как показано в практике, использование семантического анализа позволяет делать подбор, который на 76% повышает вероятность повторных покупок [6].

- **Антифрод и мониторинг:** модули машинного обучения следят за подозрительной активностью (мошеннические покупки, фальшивые отзывы и т.д.). Система раннего обнаружения аномалий может блокировать транзакции или активировать дополнительную модерацию.
- **Генерация SEO-контента и карточек услуг:** на основе ключевых слов, статистики запросов и популярных вопросов AI-модуль автоматически генерирует тексты для карточек услуг, блог-постов, описаний для лучшего продвижения. Это ускоряет заполнение каталога и улучшает видимость проекта в поисковых системах.

Пользовательские роли

- **Гость:** может просматривать публичные страницы каталога услуг, читать описания, искать и фильтровать предложения. Без регистрации доступно ограниченное взаимодействие (например, просмотр услуг и FAQ).
- **Зарегистрированный пользователь:** имеет персональный профиль, может оставлять заказы, отзывы, получать рекомендации, сохранять избранное и участвовать в программах лояльности. Управляет своими данными и настройками уведомлений.
- **Поставщик услуг (Партнер):** регистрируется как организация или индивидуальный предприниматель. Заполняет каталог своих услуг, получает заказы через платформу, управляет ценами, расписанием и информацией о своем профиле. Получает аналитику и отчеты по своим услугам.
- **Модератор:** проверяет поступающий контент (новые услуги, отзывы, сообщения в чат), следит за соблюдением правил. Может редактировать или удалять неподходящие записи, одобрять профиль новых партнеров.
- **Администратор:** полный доступ ко всем функциям системы. Управляет пользователями, правами доступа, может настраивать глобальные параметры (категории услуг, тарифы, интеграции). Отвечает за решение инцидентов.
- **SEO-менеджер:** занимается продвижением и контентом: редактирует SEO-параметры страниц, управляет генерацией текстов и метаданных, анализирует трафик. Имеет доступ к инструментам аналитики и создания семантических карточек.

Каждая роль имеет свои права в системе авторизации. Например, только поставщик услуг может создавать и редактировать карточки услуг, модератор – активировать/деактивировать их, а SEO-менеджер – управлять ключевыми словами и мета-тегами.

Интеграции и внешние системы

- **Платежные шлюзы:** интеграция с несколькими провайдерами (Stripe, YooKassa, PayPal и др.) для приема оплат картами и электронными деньгами. Система позволяет подключать дополнительные региональные провайдеры, если необходимо.
- **Мессенджеры:** Telegram и WhatsApp (через Bot API и вебхуки) для общения с пользователями и отправки уведомлений о статусе заказа. Бот может оповещать о новых сообщениях в чате, изменениях в заказе, акциях и т.д.
- **Email-уведомления:** отправка писем через SMTP или сервисы рассылок (например, SendGrid, Mailgun) для подтверждения регистрации, оплаты, рассылки новостей и т.д.
- **Webhooks / Event-driven:** система поддерживает **расширяемую событийно-ориентированную архитектуру**. Любые внешние системы (CRM, партнерские площадки) могут подписываться на события через вебхуки. Например, появление нового заказа генерирует событие, которое может быть передано в ERP или BI-систему. Такая интеграция строится на механизме подписки и асинхронной обработки, что упрощает масштабирование и расширение системы.

Мультиязычность

Система изначально разрабатывается с поддержкой **четырех языков** интерфейса: русский, английский, казахский и узбекский. Внедряется механизм интернационализации (i18n), позволяющий легко добавлять новые языки. Все текстовые ресурсы (UI, уведомления, тексты услуг) хранятся в виде переводимых строк. AI-модули (например, чат-бот) тоже обучены на многоязычных наборах, чтобы обрабатывать запросы на указанных языках. Мультиязычность важна для охвата большой аудитории и улучшения SEO – страницы на родном языке пользователей лучше индексируются в локальном сегменте поисковых систем.

Безопасность и конфиденциальность

- **Cloudflare WAF:** на уровне сети используется Web Application Firewall от Cloudflare, который блокирует распространенные угрозы (SQLi, XSS, CSRF, брутфорс и пр.). Это обеспечивает первичную защиту публичного веб-приложения.
- **Шифрование:** все коммуникации по HTTPS/TLS. На клиенте (в браузере) используется **Web Crypto API** для шифрования чувствительных данных (например, частей пароля или платежной информации) перед отправкой на сервер.
- **Аутентификация:** используется безопасная модель аутентификации (например, JWT с коротким TTL или OAuth 2.0). Реализована защита от атаки повторного воспроизведения и CSRF.
- **Confidential ML:** при обработке пользовательских данных (например, истории заказов, отзывов) применяется дифференциальная приватность. Библиотека TensorFlow Privacy позволяет гарантировать, что при обучении моделей не будет утечек личной информации [4]. Это особенно важно для анализа отзывов и поведения, чтобы нельзя было восстановить данные конкретного пользователя из модели.
- **Антифрод:** модули мониторинга транзакций используют машинное обучение для выявления мошенничества. Например, резкое изменение объема заказов или подозрительная активность приводит к проверке.
- **Разграничение доступа:** данные партнёров и пользователей изолированы (см. White-label). Реализовано многоуровневое разграничение прав, чтобы, например, партнер не видел чужие заказы, а модератор мог управлять только публичным контентом.

Масштабируемость и отказоустойчивость

Архитектура обеспечивает горизонтальное масштабирование. Каждый микросервис может масштабироваться независимо: при росте нагрузки добавляются новые инстансы контейнеров. **Redis** используется для кэширования часто запрашиваемых данных (сессии, списки популярных услуг), что снижает нагрузку на базу. БД PostgreSQL может работать в режиме репликации (read replicas) для распределения нагрузки на чтение.

Используется **load balancer** (балансировщик нагрузки) при росте трафика между edge-серверами и облаком. Контейнеры разворачиваются с оркестратором (Kubernetes или аналог) через ArgoCD/Tekton, что позволяет легко обновлять и балансировать версии. Мониторинг (Prometheus + Grafana) отслеживает метрики производительности: когда загрузка процессоров или задержки запросов растут, система автомасштабирования разворачивает дополнительные ресурсы.

Кэширование и CDN: статические ресурсы и сервисы кэшируются на CDN, что снижает задержку для пользователей из разных регионов и разгружает бэкенд.

Безотказность: за счет репликации баз данных, активной гарды (high-availability) и резервных инстансов, система переживает выход из строя отдельных узлов. Регулярные бэкапы БД и контейнерные образы обеспечивают восстановление после критических сбоев.

DevOps и CI/CD

- **Репозиторий GitHub:** все компоненты хранятся в Git. После пулл-реквестов система GitHub Actions автоматически запускает сборку, тесты (unit, integration, lint) и деплой контейнеров. Это обеспечивает быструю итерацию разработки.
- **Docker:** каждый сервис упакован в собственный контейнер. Docker Compose или Helm-чарты описывают развёртывание.
- **Orchestration:** для продакшена рекомендуется использование кластера (Kubernetes, Rancher или альтернативы). ArgoCD или Tekton отвечают за автоматическое применение изменений в кластере при обновлении образов.
- **Мониторинг и логирование:** Prometheus собирает метрики (загрузка CPU, задержки API, число запросов), Grafana строит дашборды. Логи централизуются (например, Elastic Stack или Loki) для анализа инцидентов. Алерты (Slack/email/SMS) настраиваются по критическим метрикам (падение сервиса, ошибки 5xx).
- **Тестирование:** помимо модульных тестов, развернуто staging-окружение для ручного и автоматизированного тестирования UI (E2E-тесты).

Хостинг и инфраструктура

- **Сервер:** VPS на базе Linux Ubuntu (минимум 4 ядра CPU, 8 GB RAM для начальной нагрузки) или кластер из таких узлов. Можно использовать облачные провайдеры (AWS, Azure, GCP, Hetzner и т.д.) или собственные дата-центры. В продакшене желательно иметь балансировщик нагрузки и хотя бы 2+ серверных узла для отказоустойчивости.
- **Edge-узлы/CDN:** региональные точки присутствия (например, через Cloudflare/CDN-провайдеров) для раздачи статики и ускорения взаимодействия. Также на таких узлах могут размещаться LLM-модели (через контейнеры с `llama.cpp`), чтобы в популярных регионах облегчить нагрузку на основной сервер.
- **Хранилище:** PostgreSQL с SSD-хранилищем (сервер или managed service). Redis на тех же или отдельном сервере. Для больших объёмов данных (архивы, бэкапы) можно подключить объектное хранилище (S3-совместимое).
- **SSL/TLS:** сертификаты (Let’s Encrypt или коммерческие) для HTTPS.
- **Резервное копирование:** автоматические бэкапы баз данных и критичных данных (ежедневный снимок БД, ежечасные дампы наиболее активных таблиц, копии файлов) на удалённое хранилище.

White-label решение и расширяемость

Система должна поддерживать **multi-tenant** («white-label») режим. Т.е. можно развернуть независимые порталы (или брендировать под разных клиентов) на одном бекенде. С помощью **Row Level Security** в PostgreSQL (например, через Supabase) достигается изоляция данных между партнёрами: каждый видит только свои записи [9]. RLS позволяет писать сложные правила доступа (каждое SQL-запрос автоматически «маскируется» фильтрами по tenant_id), обеспечивая defense-in-depth [9]. Это даст гибкость для будущего развития проекта как платформы-провайдера.

Дополнительные требования

- **PWA/offline:** мобильное веб-приложение (PWA) должно работать офлайн для чтения ранее сохраненных данных (например, просмотра каталога услуг, истории заказов) с помощью Service Worker. Это повысит отказоустойчивость при нестабильном соединении.
- **Генерация документации:** техническая документация (API, архитектура) генерируется частично с помощью AI-инструментов (например, на основе Annota AI или GPT) и поддерживается актуальной по мере развития системы.
- **Edge-инференс:** там, где критична скорость ответа (например, простой чат-бот или подсказки в офлайн-приложении), LLM разворачиваются на пограничных узлах или даже на устройствах пользователей (через WebAssembly-копию `llama.cpp`). В качестве альтернативы можно использовать сервисы вроде Gcore.ai для edge-инференса, чтобы обеспечить быстрый отклик без отправки данных в основной дата-центр.

Ограничения и допущения

- Проект разрабатывается для раскладки на **Linux/Ubuntu VPS**. Система должна эффективно работать на распространенной серверной инфраструктуре.
- Из-за санкций возможность интеграции с некоторыми внешними сервисами (например, Stripe) следует проверять по текущим ограничениям.
- Наличие высокопроизводительных GPU на сервере не гарантируется, поэтому локальный инференс LLM ориентирован на CPU (с quant-запусками). Основной тяжёлый инференс может идти через облачные API.
- Сервис должен быть SEO-френдли, поэтому критичная часть контента (описания услуг) доступна через server-side рендеринг.

Вывод: данное техническое задание описывает модульный, масштабируемый агрегатор услуг с продвинутой поддержкой AI. Архитектура «Cloud+Edge» и современные технологии обеспечат высокую производительность и гибкость. Наличие ChatBot-ассистента и аналитических AI-модулей повысит ценность сервиса для пользователей. При четкой реализации описанных требований проект сможет эффективно объединить различные категории услуг в единой мультифункциональной платформе.

Источники: современные рекомендации по Next.js (SSR/SEO) [1], концепции гибридной облачной архитектуры [2] [5], практики использования семантического поиска и рекомендаций [6] [7], сведения о возможностях LLM на пограничных устройствах [10], а также материалы по безопасности данных (RLS, дифференциальная приватность) [9] [4].

---

1 Next.js SEO Benefits and Optimization in 2025
https://focusreactive.com/how-nextjs-can-improve-seo/

2 3 5 Edge-Cloud Architecture in Distributed System | GeeksforGeeks
https://www.geeksforgeeks.org/edge-cloud-architecture-in-distributed-system/

4 Implement Differential Privacy with TensorFlow Privacy | Responsible AI Toolkit
https://www.tensorflow.org/responsible_ai/privacy/tutorials/classification_privacy

6 7 8 The definitive guide to semantic search engines - Algolia Blog | Algolia
https://www.algolia.com/blog/ai/the-definitive-guide-to-semantic-search-engines

9 Row Level Security | Supabase Docs
https://supabase.com/docs/guides/database/postgres/row-level-security

10 GitHub - ggml-org/llama.cpp: LLM inference in C/C++
https://github.com/ggml-org/llama.cpp