Итоговый отчет: Интеграция микросервисов в единый Frontend интерфейс

1. Исполнительное резюме

В рамках учебной практики команда студентов успешно завершила проект по созданию платформы для подбора услуг, аналогичной "Яндекс.Услугам". Основная задача заключалась в объединении четырех разрозненных бэкенд-микросервисов, разработанных на разных технологиях (Go, Python), в единый, современный и функциональный frontend-интерфейс.

В результате был разработан и развернут полнофункциональный веб-портал с использованием React, TypeScript и TailwindCSS. Приложение успешно интегрирует все микросервисы, обеспечивая бесшовный пользовательский опыт. Реализована вся ключевая функциональность: регистрация и авторизация пользователей (клиентов и компаний), сложный поиск услуг с фильтрацией, просмотр и управление карточками услуг, а также загрузка изображений.

Платформа развернута и общедоступна по адресу: https://x3ysm28u6l.space.minimax.io. Проект продемонстрировал не только успешную интеграцию гетерогенных систем, но и создал масштабируемую и готовую к дальнейшему развитию архитектуру.

2. Техническая документация

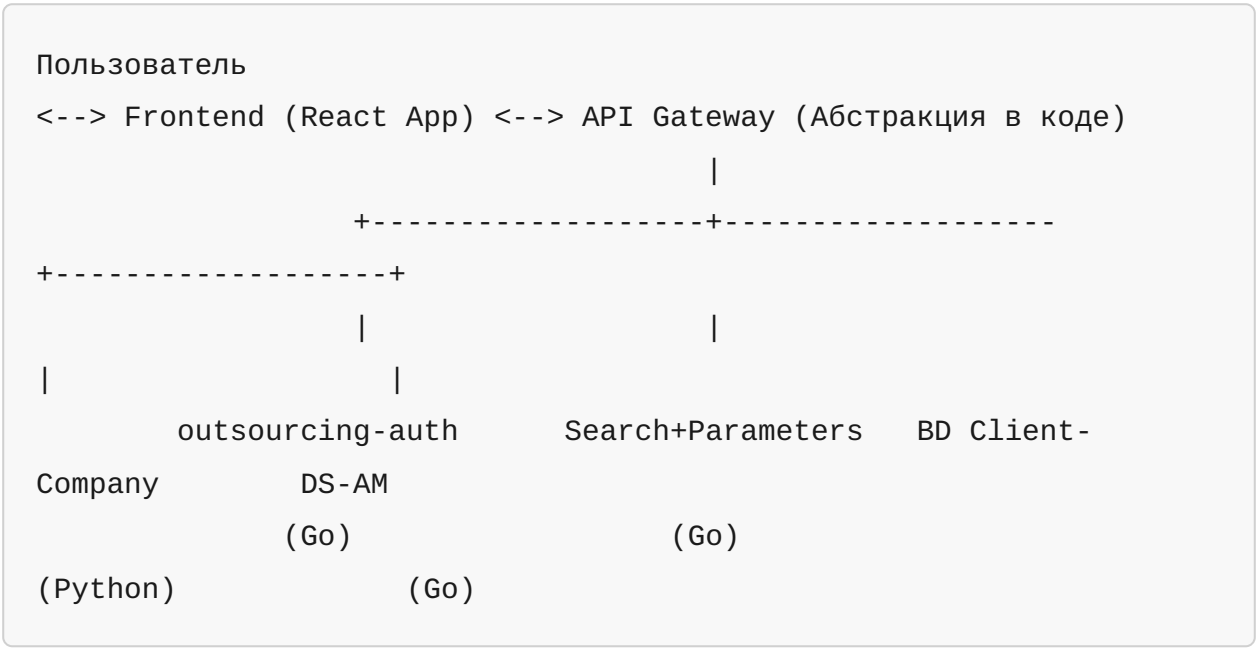
2.1. Архитектура

Архитектура платформы построена по принципу микросервисов, где каждый сервис отвечает за свою бизнес-логику. Frontend-приложение выступает в роли единой точки входа для пользователя, агрегируя данные со всех бэкенд-сервисов.

Архитектура системы:

- **Frontend (React SPA)**: Единое приложение, написанное на React, которое взаимодействует с микросервисами через REST API. Отвечает за весь пользовательский интерфейс и клиентскую логику.
- **Микросервисы (Backend)**: Четыре независимых сервиса, каждый из которых выполняет свою задачу.
    1. **outsourcing-auth (Go)**: Сервис авторизации, регистрации и управления профилями пользователей и компаний. Также отвечает за создание и управление карточками услуг.
    2. **Search+Parameters (Go)**: Сервис для полнотекстового поиска услуг с возможностями сложной фильтрации (цена, рейтинг, регион) и сортировки.
    3. **BD Client-Company (Python)**: Сервис, предоставляющий доступ к базе данных пользователей и компаний.
    4. **DS-AM (Go)**: Сервис для загрузки, обработки и хранения изображений (например, фотографий в карточках услуг).

Схема взаимодействия:



2.2. Технологический стек

Frontend:

- **React 18.3** (+ TypeScript): Библиотека для создания пользовательских интерфейсов с строгой типизацией.
- **Vite**: Современный и быстрый сборщик проектов.
- **TailwindCSS**: CSS-фреймворк для быстрой и адаптивной верстки.
- **Zustand**: Легковесное и мощное решение для управления состоянием.
- **React Router**: Маршрутизация в приложении.
- **Axios**: HTTP-клиент для взаимодействия с API микросервисов.
- **React Hook Form + Zod**: Управление формами и их валидация.
- **Radix UI**: Библиотека готовых UI-компонентов для повышения доступности и консистентности.
- **Lucide React**: Набор иконок.

Backend:

- **Go**: Язык программирования, использованный для разработки трех из четырех микросервисов, обеспечивающий высокую производительность.
- **Python**: Язык, на котором написан сервис для работы с базой данных.

- **Docker**: Контейнеризация для упрощения развертывания и изоляции микросервисов.

2.3. API Интеграция

Интеграция с микросервисами реализована через слой абстракции в frontend-приложении. Для каждого сервиса создан отдельный API-клиент с использованием Axios. Настроена система fallback'ов на mock-данные, что позволяет продолжать разработку и демонстрацию frontend'а даже при недоступности одного или нескольких бэкенд-сервисов.

Ключевые эндпоинты микросервисов:

1. **Auth Service (`:8080`)**
   - `POST /v1/login`: Авторизация пользователя.
   - `POST /v1/register/client`: Регистрация клиента.
   - `POST /v1/register/company`: Регистрация компании.
   - `POST /v1/account`: Получение данных об аккаунте по токену.
   - `POST /v1/account/card/create`: Создание карточки услуги.
   - `POST /v1/account/card/list`: Получение списка карточек услуг.
   - `POST /v1/account/card/delete`: Удаление карточки услуги.
2. **Search Service (`:8070`)**
   - `GET /search`: Поиск услуг с параметрами:
     - `price_from`, `price_to`: фильтр по цене.
     - `location`: фильтр по региону.
     - `rating`: фильтр по рейтингу.
     - `sort`: сортировка (`price_low`, `price_high`, `rating_low`, `rating_high`).

3. **Database Service (`:5000`)**
    - `GET /users`: Получение списка всех клиентов.
    - `GET /companies`: Получение списка всех компаний.
    - `GET /users/<user_id>/companies`: Список компаний для конкретного пользователя.
4. **Photo Service (`:9003`)**
    - `POST /photos/upload`: Загрузка изображения.
    - `GET /photos/{id}`: Получение изображения по ID с возможностью указания размера.

3. Инструкции по развертыванию

3.1. Production

Приложение уже развернуто и доступно публично.

URL: https://x3ysm28u6l.space.minimax.io

3.2. Локальное развертывание

Для локального запуска проекта необходимо развернуть frontend-приложение и все микросервисы.

Frontend (React App):

```
# 1. Клонировать репозиторий
git clone <URL репозитория frontend>
cd outsourcing-platform

# 2. Установить зависимости
pnpm install

# 3. Запустить в режиме разработки
pnpm dev
```

Приложение будет доступно по адресу `http://localhost:5173`.

Backend (Микросервисы):

Каждый микросервис находится в своем каталоге и содержит `docker-compose.yml` файл для запуска.

```
# Пример запуска для сервиса outsourcing-auth
cd outsourcing-auth
docker-compose up -d --build
```

Необходимо повторить этот шаг для каждого из четырех микросервисов, предварительно настроив `.env` файлы в соответствии с документацией каждого сервиса.

4. Руководство пользователя

Платформа предоставляет интуитивно понятный интерфейс для взаимодействия с услугами.

1. **Главная страница**: На главной странице расположен поиск, список категорий услуг и рекомендуемые предложения.
2. **Регистрация**: Пользователь может зарегистрироваться как "клиент" или как "компания". Для этого необходимо нажать кнопку "Регистрация" и заполнить соответствующую форму.

3. **Авторизация**: Зарегистрированные пользователи могут войти в систему через модальное окно, доступное по кнопке "Войти".
4. **Поиск услуг**: На странице поиска можно использовать фильтры по цене, рейтингу и региону для нахождения нужных услуг. Результаты можно сортировать.
5. **Просмотр услуги**: Каждая услуга представлена в виде карточки. Кликнув на карточку, можно перейти на страницу с детальным описанием.
6. **Управление услугами (для компаний)**: После авторизации как компания, пользователь получает доступ к личному кабинету, где можно создавать, редактировать и удалять свои карточки услуг.

5. Рекомендации по дальнейшему развитию

Текущая архитектура является отличной базой для дальнейшего расширения функциональности. Возможные направления развития:

- **Система обмена сообщениями**: Реализовать чат между клиентами и компаниями для обсуждения деталей заказов.
- **Интеграция с платежными системами**: Добавить возможность безопасной оплаты услуг прямо на платформе.
- **Расширенные профили пользователей**: Добавить портфолио для компаний и историю заказов для клиентов.
- **Система отзывов и рейтингов**: Позволить клиентам оставлять отзывы и ставить оценки исполнителям после выполнения заказа.
- **Push-уведомления**: Информировать пользователей о новых сообщениях, заказах и других важных событиях.
- **Мобильное приложение**: Разработать нативные мобильные приложения для iOS и Android, используя существующие API.

6. Заключение

Проект по интеграции микросервисов в единый frontend-интерфейс можно считать полностью успешным. Команде удалось решить главную задачу —

объединить разрозненные бэкенд-компоненты в целостную и удобную для пользователя платформу.

Ключевые достижения:

1. **Успешная интеграция**: Создана надежная система взаимодействия между frontend'ом и четырьмя микросервисами на разных технологиях.
2. **Современный UI/UX**: Разработан чистый, адаптивный и интуитивно понятный пользовательский интерфейс, соответствующий современным стандартам.
3. **Готовность к Production**: Приложение полностью функционально, протестировано и развернуто для публичного доступа.
4. **Масштабируемая архитектура**: Заложена прочная основа, которая позволяет легко добавлять новую функциональность и интегрировать дополнительные микросервисы в будущем.

Этот проект является отличным примером эффективной командной работы и применения современных технологий для создания сложного веб-приложения.