Сервис мониторинга и адаптивного распределения заявок на обслуживание от маломобильных пассажиров

Папуасы | МИСИС

1. Введение

Документация по работе с автоматизированной системой управления заявками на сопровождение маломобильных пассажиров, оптимизирующей процесс принятия, распределения и контроля выполнения заявок.

Основные компоненты включают в себя три сервиса:

- backend
- frontend
- algorithms

Сервис предоставляет функционал для работы с данными пассажиров, пользователей и заявок. Для всех сущностей реализована возможность создания, редактирования, поиска, фильтрации, удаления, удобного отображения.

Алгоритм позволяет распределять задачи и создавать расписание для сотрудников.

На бэкенде реализовано RESTful API с операциями CRUD, обработка ошибок и возвращение HTTP-статусов.

Для безопасности используется протокол HTTPS, для аутентификации используются токены, хешируются пароли. Для защиты от SQL-инъекций используется ORM, есть защита от XSS.

Сервис оптимизирован для большого количества запросов, скорость алгоритма позволяет обрабатывать тысячи заявок быстрее, чем за секунду.

Реализован адаптивный интерфейс, удобный для просмотра с мобильных устройств, асинхронный режим работы и предупреждения в виде всплывающих окон.

Реализованный функционал (часть из него):

- Аутентификация и авторизация пользователей
- Разделение уровня доступа по ролям пользователей
- Создание, редактирование, удаление заявок
- Подсказки для оформления заявок
- Сохранение истории изменений пользователем
- Мониторинг заявок за текущие сутки
- Отображение списка заявок, фильтрация и поиск по заявкам
- Создание, редактирование, удаление пассажиров
- Отображение списка пассажиров, фильтрация и поиск
- Создание, редактирование, удаление пользователей (рабочих)
- Отображение списка пользователей, фильтрация и поиск
- Автоматическое распределение заявок между сотрудниками
- Плановое распределение всех заявок на день и адаптивное подстраивание заявок под изменяющиеся условия
- Возможность руками проставить заявку
- Отображение расписания (результата распределения заявок)
- Соблюдение условий заявки
- Приоритет расчета на сотрудников
- Сотрудник прибывает на станцию за 15 минут до начала заявки
- Адаптивное время для каждой категории пассажиров
- Приоритет на меньшее количество пересадок
- Выделение времени обеда для сотрудников
- Возможность переноса времени заявки
- Просмотр геолокации сотрудников
- Просмотр сотрудником своих заявок на день
- Просмотр истории выполнения заявок

2. Требования к аппаратному и программному обеспечению

Необходимые для развертывания инструменты:

- Git
- Docker

Инструкция по установке:

Для пользователей ничего устанавливать не нужно. Достаточно перейти на сайт:

https://papuas.tech

Для развертывания на сервере:

Для запуска приложения сначала требуется скачать его исходный код с GitHub, используя Git. Убедитесь, что у вас установлен Git, следуя инструкциям для вашей операционной системы (Windows, Mac, Linux). После установки выполните следующую команду в терминале:

git clone https://github.com/nizhgo/lct-2024

Перед запуском приложения необходимо убедиться, что все значения переменных окружения в файле `.env` заполнены корректно. Для этого скопируйте файл `.env.example` и вставьте валидные значения. Вы можете выполнить следующую команду в терминале для копирования:

bashCopy code

cp .env.example .env

Затем откройте файл `.env` в текстовом редакторе и внесите необходимые значения для переменных окружения, такие как база данных, ключи API и другие конфигурационные параметры, которые могут быть необходимы для вашего приложения. После этого вы будете готовы к запуску вашего приложения.

Далее, для упрощения развертывания приложения, требуется установить Docker. Вы можете скачать и установить Docker и docker-compose с официального сайта Docker: https://www.docker.com/. После установки Docker выполните следующую команду в терминале :

docker compose up -d

3. Архитектура системы

Архитектура состоит из трех сервисов: бэкенд (сервис на FastApi, который также взаимодействует с базой данных PostgreSQL, хранилищем s3 и Speechkit), фронтенд и алгоритм.

Все они обернуты в docker-контейнер для удобного развертывания на сервере.

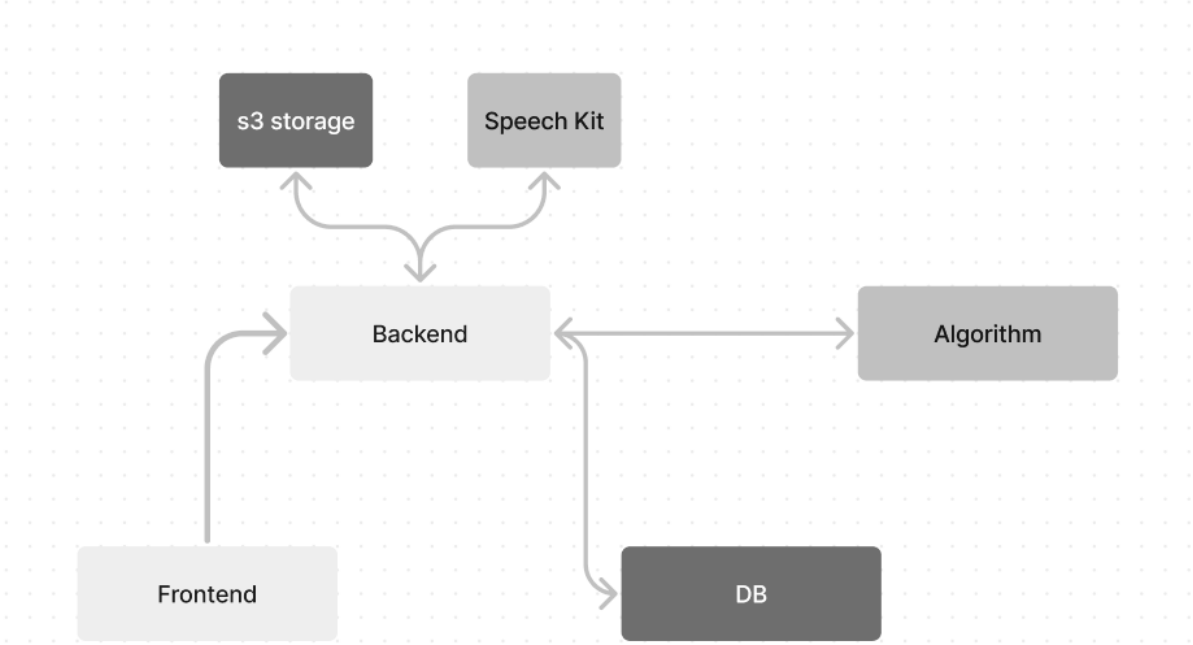

Рис. 1: Схема архитектуры системы

4. Инструкции по развертыванию

Для успешного развертывания на сервере см. п. 2: инструкция по установке.

Данные авторизации супер-пользователя будут подтянуты из данных файла окружения .env, переменных FIRST_SUPERUSER, FIRST_SUPERUSER_PASSWORD.

5. Инструкции по использованию

Описание доступных API методов, примеры запросов, endpoint’ов и ожидаемых ответов можно найти по ссылке на документацию (swagger):

https://api.papuas.tech/docs

6. Пояснения от участников

Выбор технологий обеспечен быстротой и эффективностью процесса разработки, а также предоставляют достаточную гибкость и производительность для реализации всех необходимых функций.

Алгоритм

Выбранный нами алгоритм показал наилучший результат среди всех экспериментов (также пробовали муравьиный алгоритм, метода отжига). Он обеспечивает гибкость настройки (пересчет всех событий или лишь последних, после конкретного времени и тд), работает безусловно быстро и качественно.

Подсчет расстояний между станциями и пересадками подсчитан с помощью алгоритма Дейкстры на графе метро, при этом мы осуществляем умный учет времени пересадки в зависимости от категории инвалидности пассажира, так как понимаем, что у разных категорий будет разное время дороги.

Алгоритм также состоит из сортировки событий с учетом времени, обработки их открывающих и закрывающих границ. Для этого используется метод сканирующей прямой. При попытке назначить работника на задачу, мы учитываем все его интервалы отдыха, и если по времени они блокируют задачу - назначаем другого.

Генерация времени обеда происходит относительно всех заявок, так, чтобы блокировать как можно меньше потенциальных задач.

Алгоритм пытается распределить как можно больше заявок без переноса времени. Если оказывается, что какую-то заявку выполнить нельзя, то ее мы переносим и распределяем на доступное время. Это происходит в крайнем случае, когда физически уже невозможно обеспечить своевременное выполнение каждой заявки.

FastAPI для бэкенда

1. Высокая производительность: FastAPI построен на основе Starlette и Pydantic, что позволяет ему обрабатывать большое количество запросов с низкой задержкой. Это важно для хакатона, где необходимо быстрое и отзывчивое приложение.
2. Простота и скорость разработки: FastAPI использует аннотации типов Python, что делает код читаемым и легко поддерживаемым. Это сокращает время на написание и отладку кода, что критично в условиях ограниченного времени хакатона.
3. Автоматическая генерация документации: FastAPI автоматически генерирует OpenAPI и JSON Schema документацию, что упрощает тестирование и интеграцию. Это также полезно для демонстрации проекта жюри и другим участникам команды.
4. Совместимость с asyncio: Поддержка асинхронного программирования позволяет обрабатывать множество конкурентных запросов эффективно, что может быть важно для масштабируемости и производительности приложения.

PostgreSQL для базы данных

1. Надежность и масштабируемость: PostgreSQL — это проверенная временем реляционная база данных, известная своей надежностью и стабильностью. Она хорошо масштабируется и поддерживает большое количество одновременных пользователей.
2. Богатый функционал: PostgreSQL поддерживает сложные запросы, транзакции, индексы и множество других функций, которые могут понадобиться для реализации сложной бизнес-логики.
3. Расширяемость: Благодаря поддержке расширений (например, PostGIS для географических данных), PostgreSQL может быть адаптирован для самых разных задач.
4. Сообщество и поддержка: Активное сообщество и множество доступных ресурсов и библиотек упрощают разработку и решение возникающих проблем.

React и TypeScript для фронтенда

1. Декларативный стиль: React позволяет создавать компонентный пользовательский интерфейс декларативным образом, что упрощает разработку и поддержку кода.
2. Большая экосистема: React имеет огромную экосистему библиотек и инструментов, что ускоряет разработку и позволяет легко интегрировать различные функциональности (например, маршрутизация с React Router, управление состоянием с Redux или Context API).
3. TypeScript для повышения надежности: TypeScript добавляет статическую типизацию к JavaScript, что позволяет избежать множества ошибок на этапе разработки. Это улучшает качество кода и ускоряет процесс отладки.
4. Реактивность и производительность: React оптимизирован для высокопроизводительных пользовательских интерфейсов. Его виртуальный DOM обеспечивает высокую скорость обновления интерфейса, что важно для создания отзывчивых приложений.