НИЯУ МИФИ

Прогноз нагрузки на дороги и метро

nuclear intelligence

02.10.2024

Задачи проекта

1. Исследование потребностей: провести анализ требуемого функционала для решения задачи и обеспечения возможности масштабирования проекта на сеть общественного транспорта.

2. Разработка **REST API**: создать серверную часть приложения для обработки запросов и взаимодействия с базой данных.

3. Разработка клиентской части: реализовать пользовательский интерфейс для предоставления доступа к функционалу платформы.

4. Тестирование: провести комплексное тестирование для обеспечения надежности, производительности и безопасности системы.

5. Развертывание и масштабирование: организовать развертывание приложения и подготовить его к масштабированию с использованием **Docker Compose**.

Технологический стек

Backend:

Языки программирования: Java и Kotlin

Фреймворки:

Spring Boot: Фреймворк для создания веб-приложений и микросервисов. Он обеспечивает быстрое развертывание, управление зависимостями и интеграцию с базами данных и внешними сервисами.

Безопасность: Spring Security: Комплексное решение для управления аутентификацией и авторизацией, которое защищает доступ к API и пользовательским данным, поддерживает OAuth, JWT и другие протоколы.

Микросервисная архитектура:

Eureka: Используется для регистрации и обнаружения микросервисов, что обеспечивает динамическую маршрутизацию и масштабирование приложений. Eureka облегчает управление отказоустойчивостью системы.

Load Balancer: Компонент для балансировки нагрузки между микросервисами. Он распределяет запросы между доступными узлами, снижая нагрузку на серверы и обеспечивая высокую доступность системы.

Интеграция с мессенджером:

Телеграм-бот с Spring Client: Реализован бот, который взаимодействует с пользователями через Telegram. Spring Client обеспечивает легкую интеграцию с API для отправки запросов на сервер.

База данных:

PostgreSQL: Реляционная СУБД, которая используется для хранения данных. PostgreSQL известна своей поддержкой транзакций, расширяемыми типами данных и возможностями для работы с большими объемами данных.

Frontend:

Язык программирования:

JavaScript: Используется для разработки интерактивной клиентской части. JavaScript обеспечивает динамическое взаимодействие с интерфейсом и API.

Алгоритм:

Predict Service использует алгоритм, основанный на поиске в ширину (BFS), для предсказания транспортной нагрузки. Этот алгоритм разработан для моделирования пассажиропотока в метро с учётом закономерностей движения пассажиров в час пик.

Шаги алгоритма:

1. Инициализация:

   Алгоритм начинается с начального распределения пассажирской нагрузки, основанного на исторических данных и площадях зданий.

   Каждой станции в метро присваивается начальное количество пассажиров.

2. Обход в ширину (BFS):

   Обход в ширину запускается от начальной станции, имитируя движение пассажиров.

   Обход следует по соединениям в метро.

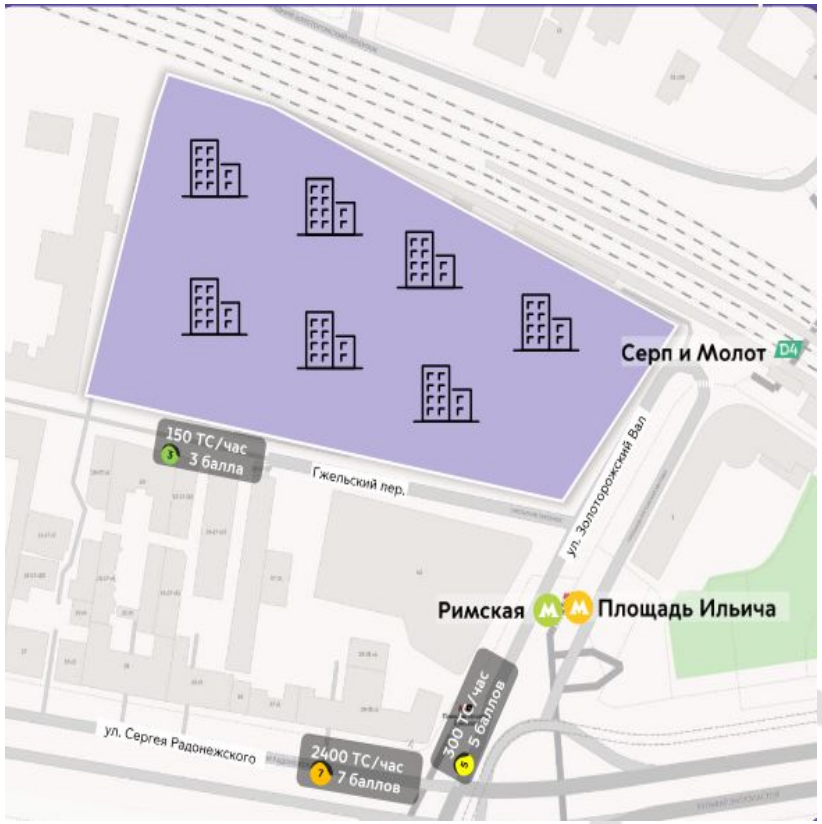
3. Распределение пассажиропотока:

   По мере того, как BFS проходит по сети, к каждой станции добавляется дополнительная пассажирская нагрузка: количество пассажиров, вышедших на этой станции.

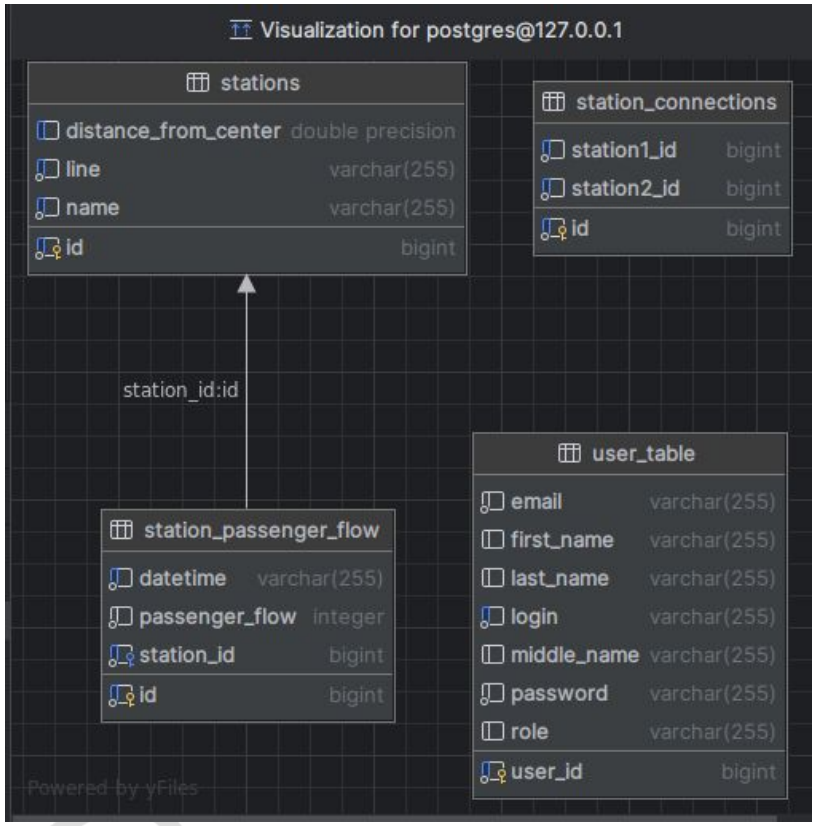
   Эта дополнительная пассажирская нагрузка распределяется по следующему правилу:

   Если текущая станция находится за пределами кольцевой линии (коричневая), то к центру движется примерно 80% дополнительной пассажирской нагрузки. Остальные 20% распределяются пропорционально другим станциям, соединённым с текущей станцией.

   Иначе пассажиропоток распределяется по всем смежным станциям равномерно.



Реализация БД



База данных состоит из нескольких основных таблиц. Таблица "stations" хранит данные о станциях транспортной сети, включая их название, линию, расстояние до центра и связи с пассажиропотоками (связь один ко многим с таблицей "station_passenger_flow"), где записаны данные о пассажиропотоках на станции в разное время (дата и количество пассажиров). Таблица "station_connections" описывает связи между станциями, указывая идентификаторы связанных станций для построения маршрутов. Таблица "user_table" хранит информацию о пользователях системы: логин, пароль, email, а также их персональные данные и роль в системе.

Spring mircosevrices

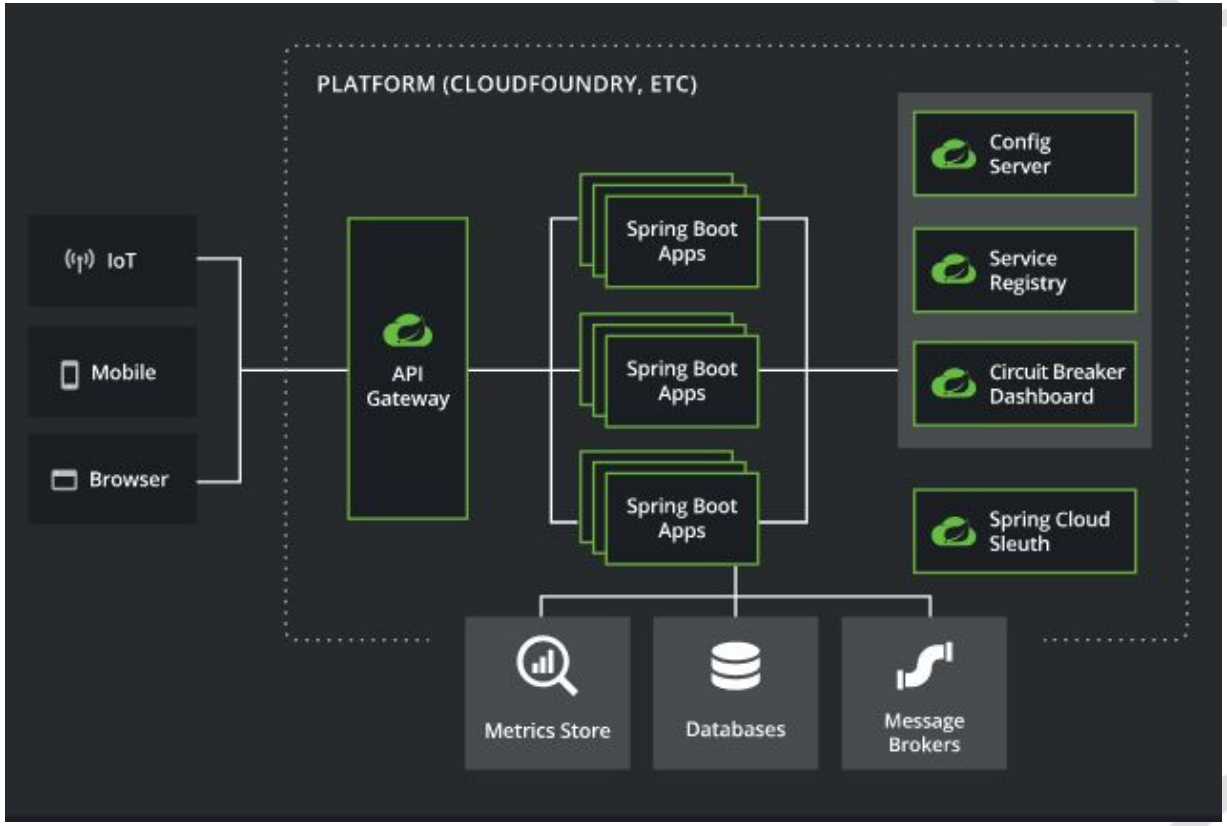
Гибкость развертывания: Микросервисы можно развертывать независимо, что упрощает обновления и масштабирование отдельных компонентов.

Устойчивость к отказам: Отказ одного сервиса не обязательно приводит к сбою всей системы.

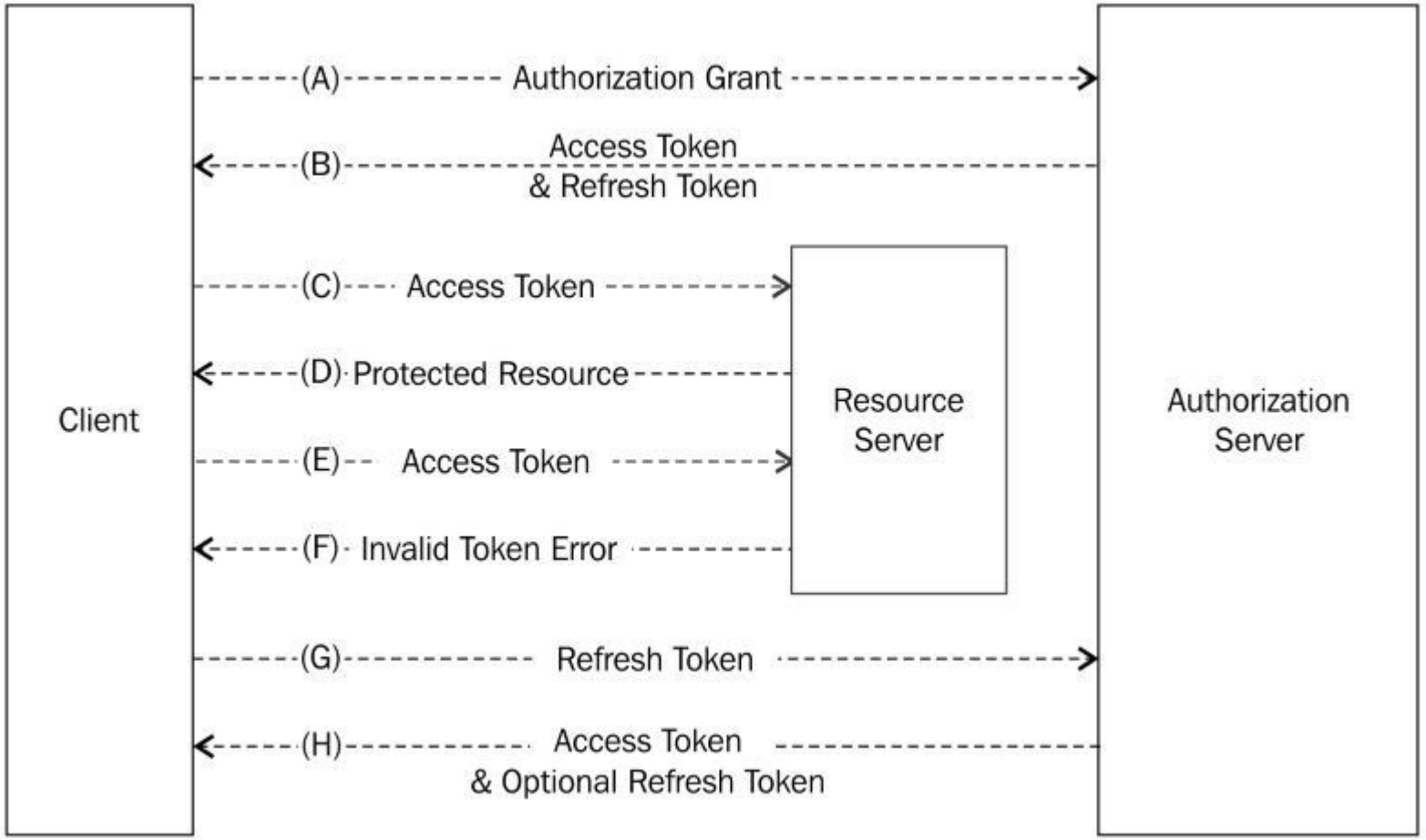
Масштабируемость: Микросервисы легче масштабировать по сравнению с монолитными приложениями.

Гибкость в выборе технологий: Разные микросервисы могут использовать разные технологии и языки программирования, что позволяет использовать наиболее подходящие инструменты для каждой задачи.

Упрощение разработки: Меньший объем кода для каждого сервиса упрощает понимание и поддержку системы в целом.



Spring Security



Access Token: Краткосрочный токен, который предоставляет доступ к API и ресурсам. Он обычно имеет короткий срок действия и должен быть обновлен с помощью Refresh Token.

Refresh Token: Долгосрочный токен, который используется для получения нового Access Token после его истечения. Refresh Token имеет более длительный срок действия и обеспечивает возможность пользователям оставаться в системе без необходимости повторного ввода учетных данных.

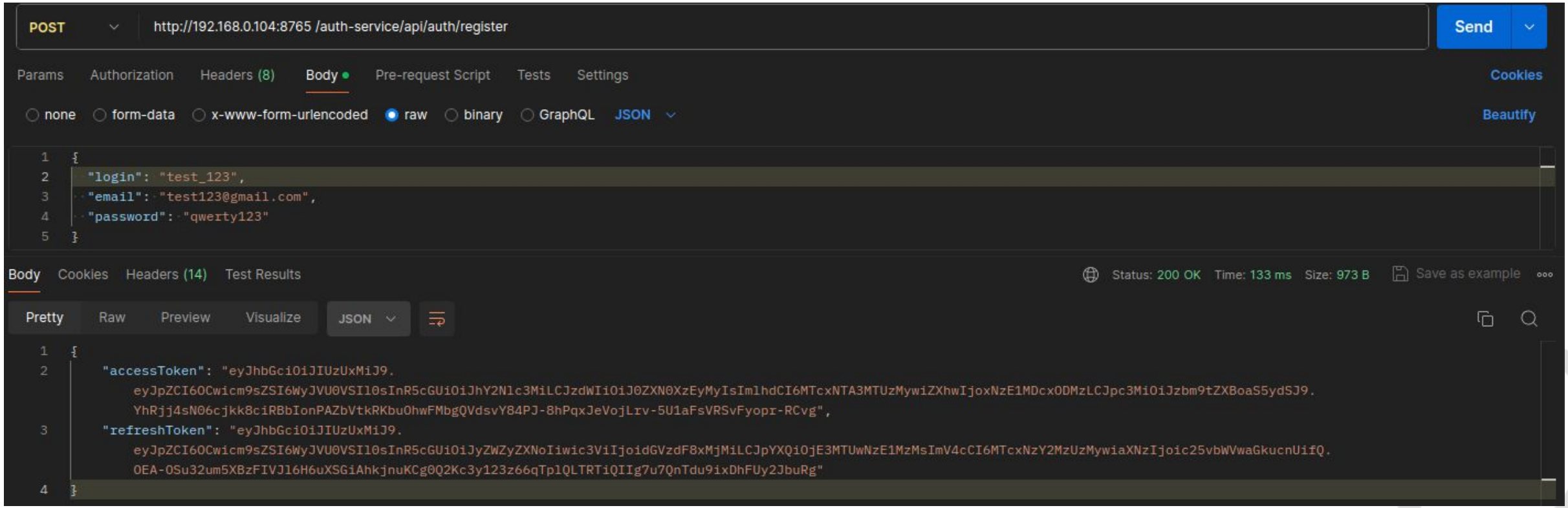
Spring Security

Использование JWT в Spring Security позволяет:

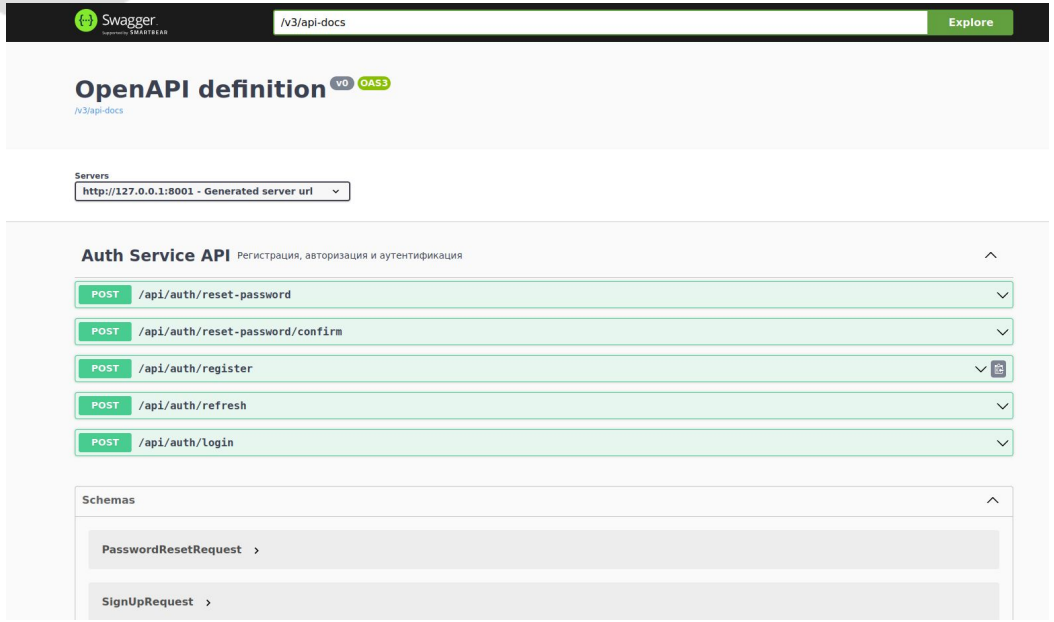
Уменьшить количество запросов к серверу для проверки аутентификации, так как информация о пользователе зашифрована в токене.

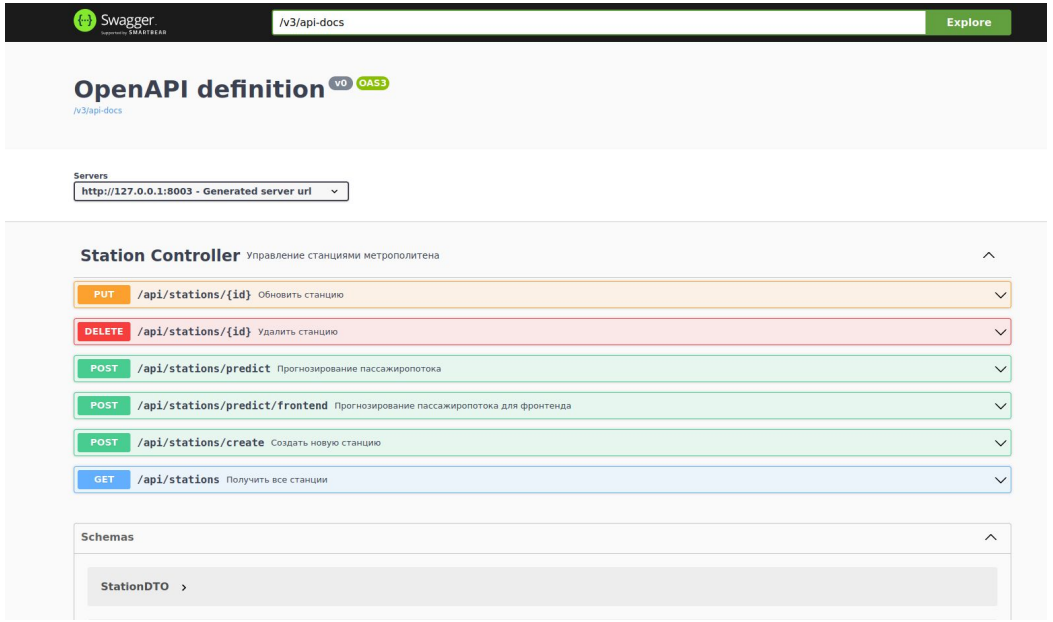
Обеспечить безопасность передачи данных, благодаря цифровой подписи.

Упростить масштабируемость приложения, поскольку токены не требуют хранения состояния сессии на сервере.



Документирование кода





Swagger и **OpenAPI** обеспечивают:

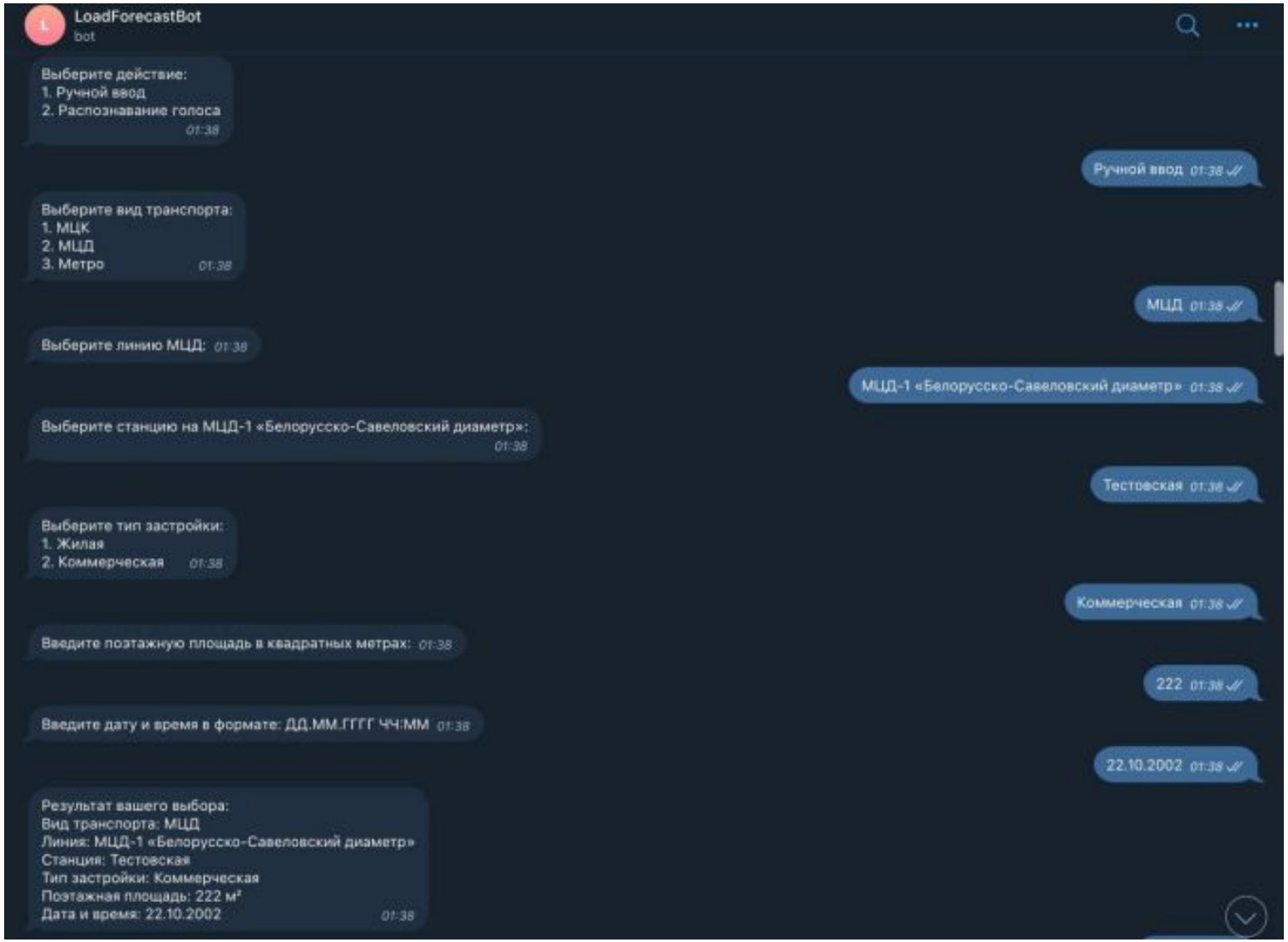
- **Стандартизацию:** Они помогают стандартизировать документацию API для улучшения совместимости и взаимодействия.
- **Удобство использования:** Интерактивная документация позволяет разработчикам тестировать API прямо в браузере.
- **Ускорение разработки:** Автоматическая генерация кода сокращает время, необходимое для создания клиентских и серверных приложений.

http://127.0.0.1:8765/auth-service/swagger-ui/index.html#/

Frontend


Дата:
02.10.2024
Время:
17:00
Линия метро:
Замоскворецкая линия (зел
Название станции:
Театральная
Площадь (кв. м):
10000
Тип здания:
Офисное
Предсказать
Театральная
Нагруженность 43%

Telegram-бот



https://github.com/ITech7750/mostrans_api

Спасибо за внимание!