JUCR
Backend Challenge

Hi, Welcome to JUCR Technical Challenge!🚀

Challenge Proposal: Building a Comprehensive Project Documentation and Scalable Data Import

Objective: This challenge aims to assess your ability to create detailed project documentation, provide a clear solution overview, and implement a scalable data import algorithm using the OpenChargeMap API.

Problem Statement: In this challenge, you are tasked with enhancing an existing platform by integrating Points of Interest (POIs) from OpenChargeMap through a microservice. This integration is crucial as these POIs will serve as core data for the platform. The implementation must not only be robust and resilient but also highly monitored to ensure the accuracy and reliability of the data. Additionally, it must seamlessly interact with the remaining services of the platform.

Deliverables:

1. *Documentation Part - Project Documentation:*

- Project Overview: Provide a **high-level overview of the project**, its goals, and its significance within the organization.
- Architecture Diagram: Create an **architectural diagram** illustrating the components of the system, their interactions, and data flow.
- API Documentation: Document the **API endpoints**, request/response formats, and authentication mechanisms.
- Database Documentation: **Describe the database schema**, including table structures, relationships, and indexing strategies.
- Deployment Instructions: Provide step-by-step instructions for deploying the application (e.g. to a Kubernetes cluster), **there is no need to create the deployment itself**, only documenting the required resources for that.
- Reliability and Scalability: Explain how the project **ensures high uptime and scalability**, addressing timeouts and fault tolerance.
- GraphQL Integration: Describe how the GraphQL endpoint is implemented and **how it serves the imported charging station data**.
- Monitoring and Logging: Describe **how the project can be monitored** and how errors and exceptions are logged.
- **The documentation must contain sufficient details in order to support the development process of the whole project**

2. *Implementation Part - Scalable Data Import:*

- **Concurrent Import**: Implement a scalable algorithm to **concurrently import all Points of Interest (POIs) from the OpenChargeMap API**. You can choose a concurrency strategy of your preference.
- **Efficient Data Retrieval**: **Optimize the data retrieval process** to minimize API requests and **efficiently handle computational resources**.

- **Data Storage Strategy:** Define how imported data is stored in the MongoDB database, including the use of UUIDs (v4) and any relevant data transformation or normalization.
- **Scalability:** The process should be able to be scaled horizontally, in order to handle a large volume of POIs effectively.
- **Error Handling:** Implement error handling and retry mechanisms to ensure data integrity and reliability during import.
- **Testing:** Provide unit tests and at least one end-to-end (E2E) test for the data import functionality.
- **Docker:** Provide a docker-compose with all the used resources to be used as a local development environment.

General Guidelines:

- Use Typescript for the implementation.
- Maintain clear and well-structured documentation throughout the project.
- Commit your code frequently to show your git practices.
- If you encounter any ambiguities or uncertainties in the requirements, make reasonable assumptions and document **them.**
- Follow best practices for **clean and maintainable code**.
- **UUIDv4** should be used as the _id key in MongoDB.
- We are big fans of the **functional programming** paradigm. While it's **not required**, feel free to use it if you want.
- The implementation part **requires only the code related to the data import** until it's saved on the database. **Endpoints, queries, and mutations are not required**.
- API Documentation: https://openchargemap.org/site/develop/api
- API Key: ff82541f-c8d1-4507-be67-bd07e3259c4e

Evaluation Criteria:

- Quality and comprehensiveness of project documentation.
- Clarity and correctness of the solution overview.
- Efficiency and scalability of the data import algorithm.
- Adherence to best practices for error handling, testing, and logging.
- The ability to create a clean and functional solution.

We look forward to reviewing your documentation and the scalability of your data import algorithm. Good luck!