Coding Challenge Backend Development: "API Development"

1. Project Description

Develop a CRUD service for managing Points of Interest (POIs), specifically focusing on petrol stations. Each POI encapsulates detailed information about the petrol station's operational status, location, available fuel pumps and products, and varied opening hours schedules.

2. Description of the POI Object

- Status: Reflects the operational state of the petrol station (e.g., ONLINE, OFFLINE).
- Address:
  - Country: The country where the station is located
  - Zip Code: The postal code where the station is located (e.g., 80939).
  - City: The city of the station (e.g., München).
  - Street: The street name (e.g., Ingolstaedter Str.).
  - House Number: The number on the street (e.g., 59).
- Opening Hours: Configurable for different operational patterns:
  - Case One: Open Monday to Friday from 8 AM to 8 PM, Saturdays from 8 AM to 6 PM, closed on Sundays and public holidays.
  - Case Two: Open daily, including public holidays.
  - Case Three: Open Monday to Thursday 6 AM to 8PM, Fridays from 6AM to 4 PM, closed on Saturdays, Sundays and public holidays.
- Pumps: An array of fuel pumps, each defined by:
  - Pump ID: A unique identifier (e.g., "91144c75-4e01-49bf-911b-4a6330962b17").
  - Pump Name: The name of the pump (e.g., 1,2,3).
  - Fuel Products: An array of available fuel types at each pump, including their names (e.g., "SUPER E10", "Diesel") and prices in multiple currencies.

3. Duration

2.5 hours

4. Tech Stack

- Backend Language: TypeScript
- Server Environment: Node.js
- Database: PostgreSQL (using ORM like TypeORM is recommended)
- API: RESTful services
- Testing: Jest

5. Challenge Scope

5.1 API Development

- Implement RESTful APIs to manage POI data with CRUD operations.
- Include pagination for list endpoints to manage large sets of POIs efficiently.

5.2 Database Design

- Create a PostgreSQL schema for storing POI information effectively, considering relationships and data normalization.

5.3 Data Handling

- Handle various configurations for opening hours and support multiple currencies for fuel pricing.

5.4 Architectural Considerations

- From an architectural point of view, detail any improvements, changes, or additional features you would introduce to your implementation.

6. Deliverables

- A GitHub repository link containing all source code.
- README.md with setup and operational instructions.
- Unit and integration tests demonstrating application functionality and resilience.
- A document outlining suggested improvements and architectural considerations.
- (Optionally) Database scripts for schema creation and data seeding.
- (Optionally) Postman collection or equivalent for API testing.
- (Optionally) SQL dump of the populated database, compressed as `*.sql.gz`.

7. Evaluation Criteria

- **Functionality**: Accuracy and completeness of the implemented features.
- **Code Quality**: Best practices and design patterns, appropriate use of TypeScript.
- **API Design**: REST principles, efficient pagination, and error handling.
- **Database Design**: Effective schema with consideration for future scalability and maintenance.
- **Scalability and Performance**: Application should handle varying loads efficiently.
- **Documentation**: Clarity and detail in documentation, including inline comments , API Documentation and API usage examples.
- **Testing and Testability**: Coverage of key functionalities with tests, ease of maintenance and extension of the test suite.
- **Architectural Improvements and Considerations**: Comprehensive evaluation of proposed architectural changes, emphasizing adherence to best practices, innovation, and detailed consideration of implications, risks, and solutions.

8. Notes

8.1 Note on AI Tools

Please **do not use AI coding tools** to support the completion of this project. It is important that you understand and explain the logic and structure of your code. Following the offline evaluation of your submission, we will organize a Q&A session where you will need to discuss your design and decision-making processes. This interaction is crucial as it not only reflects your ability to articulate and defend your technical choices but also demonstrates your readiness to manage the complexity of day-to-day work effectively.

8.2 Note on Time Management and Challenge Scope

Please note that it is not expected for candidates to complete all tasks within the 2.5-hour timeframe. This challenge is designed to assess your prioritization and time management skills as well as your technical capabilities. Plan your time accordingly, focusing on core functionalities first and then expanding to additional features as time allows.

Thank you in advance for investing your time in solving our coding challenge,
We really appreciate your efforts.

ryd