Test Project - Game Achievements API

Background

Achievements are a great way to increase users' engagement within mobile games. A developer can implement achievements in her games to encourage players to experiment with features they might not normally use, or to approach a game with entirely different play styles. Achievements can also be a fun way for players to compare their progress with each other and engage in light-hearted competition.

Task

Design and implement an HTTP/JSON web service (API) for managing achievements for a pre-existing set of games.

Requirements Specification

1. Data Model

Achievements

- **Id*** is a unique string that is generated by the system. You'll use this unique ID (for example *“fjh3hjw-sd4s-k87j”*) to refer to the achievement in your API
- **Display Name*** is a short name of the achievement (for example, "*Master Swordsman*"). The value can be up to 100 characters.
- **Description*** is a concise description of your achievement. Usually, this tells your player how to earn the achievement (for example, "*Win 10 games by only using your sword*"). The value can be up to 500 characters.
- **Icon** denotes the URL of the icon for this specific achievement (for example, *“www.example.com/example_icon.png”*
- **Display Order** is the order in which the achievements are returned (for example, “1”)
- **Created*** is a timestamp that denotes when the achievement was created.
- **Updated*** is a timestamp that denotes when the achievement was last updated.
- Each achievement is associated with a specific Game

Games

- **Id*** is a unique string that is generated by the system. You'll use this unique ID (for example "*gnwpdp-sd4s-m17lj*") to refer to the game in your API
- **Display Name*** is a short name of the game (for example "*Ninja Warrior Pro Deluxe Plus*")
- Each game has multiple Achievements associated with it

Attributes marked with * are mandatory.

2. API Specification

Game Achievements API is used to configure achievements for a specific game by exposing all achievement-related CRUD operations as API endpoints:

- **Create Achievement** should allow adding a new achievement. It should take all of the relevant parameters in the request and persist this information in the database.
- **Get All Game Achievements** should return all achievements for the supplied game Id. Order of returned achievements is determined by the **Display Order** attribute (smaller value means achievement is displayed first).
- **Get Achievement** returns a single achievement for the supplied achievement id
- **Update Achievement** should allow updating an existing achievement. It should take all of the relevant parameters in the request and persist this information in the database.
- **Delete Achievement** deletes a single achievement from the database by the supplied achievement id.

API should be exposed as a JSON-based REST web service. **All validation errors should be handled accordingly.** Authentication and authorization are **not** part of the project's scope.

3. Non-functional Requirements

The solution should also satisfy these non-functional requirements:

- Must use Java
- Must rely on a Relational Database for data persistence (For example, PostgreSQL, MySQL, Oracle RDBMS, HSQLDB, etc.)
- Must use an ORM framework (For example, Hibernate, EclipseLink, etc.)
- API (de)serialization format must be JSON
- Must employ some kind of build system (For example, Maven, Gradle, etc.)
- Must include SQL code for creating all necessary database tables

- Must include Postman project file, with prepared requests for all API operations specified above
- All written material should be in English. This includes names of constants, variables, classes, interfaces, etc., comments in code, git commit messages, any documentation, etc.
- Bonus points for using Spring Boot
- Bonus points for using a database migration tool

Submitting Your Work

The final deliverable for this project is a compilable, working codebase which satisfies the given requirements. The code should be put in a **private** Git repo (preferably on GitLab) and access should be granted to milan.stojanovic@ingsoftware.com & antonije.karadzic@ingsoftware.com Furthermore, any and all assumptions must be clearly stated in the accompanying email when submitting the code. Any extra features will be considered a plus, although the focus should be on the quality of the solution.

Important Note

Upon getting familiar with the requirements (i.e. this document), feel free to contact Toni (antonije.karadzic@ingsoftware.com) to discuss anything that potentially needs clearing up (can even jump on a call if necessary). Asking questions is not only OK - it is strongly encouraged! We'd love to see how you think and organize your work, or what kind of problems you encounter and how you communicate them.

Good luck! ;)