# Candidate Interview Project: Node + GraphQL

## Project Overview

Welcome to the interview project! This project is designed to assess your skills in building a backend API using Node and GraphQL.

## Goals

* Understand your proficiency with Node and GraphQL.
* Assess your ability to design and implement a scalable API.
* Evaluate your coding practices and problem-solving approach.
* Test your ability to work with different ORMs and databases.

## Project Details

The project includes two SQLite databases, one using Prisma ORM and the other using TypeORM. Both ORMs are already configured within the project. You are free to choose any of them for your test.

The database consists of a single table:

**Table "pokemons"**

* id: Int
* name: String
* type: String
* created\_at: Datetime

## Setup Instructions

1. Clone the Repository: <https://github.com/MatheusGNachtigall/Node-GraphQL-Test>
2. Follow installation guide on README.MD.
3. Run the Project.
4. Access GraphQL Playground: Navigate to http://localhost:4000/graphql.

### Tasks

1. Implement CRUD Operations (as resolvers) for the pokemons Table:
   * createOnePokemon
   * updateOnePokemon
   * deleteOnePokemon
   * findOnePokemon
   * findManyPokemon
2. (Optional) Enhance the API with Extra Features
   * Query filters (e.g., filter by type, partial name).
   * Pagination.
   * Rate limiting.
   * Caching for performance optimization.
3. (Optional) Data Validation and Error Handling:
   * Ensure input validation using class-validator.
   * Handle errors and return appropriate responses.
4. (Optional) Testing:
   * Write unit tests for resolvers.
5. (Optional) Convert the “type” field into a many-to-many table

**Table "types"**

id: Int

name: String **(unique)**

created\_at: Datetime

1. (Optional) **Go Wild! Do you feel like there’s anything else you would like to include? Do so! Add as many extra features as you want. Write down in the README what you added and why.**

## Evaluation Criteria

* Code Quality and Structure.
* Proper use of GraphQL schema and resolvers.
* Error handling and input validation.
* Test coverage.
* Ability to implement extra features.
* Clear and maintainable code.

## Submission

* Create a public repository on your personal GitHub and send the link to the reviewers
* Include a brief README with any assumptions or additional instructions.

Good luck!