Task: Employee Management System

Overview:
You are tasked with creating a basic Employee Management System (EMS) that allows for managing departments and employees within those departments. The system will feature a user authentication mechanism, CRUD operations for managing departments and employees, relationships between departments and employees, soft deletion of records, and an API to interact with the system.

Requirements:

1. **Authentication**:
   - Implement a simple authentication system where users can register, login, and logout.
   - Ensure that only authenticated users can access the CRUD functionalities.

2. **CRUD Operations**:
   - Create interfaces for adding, updating, viewing, and deleting departments and employees.
   - Utilize form requests to validate data before storing it in the database.

3. **Models and Relationships**:
   - **Department Model**: Should have a `name` and `description`.
   - **Employee Model**: Should have fields like `first_name`, `last_name`, `email`, `department_id`, and `position`.
   - Implement a **Many-to-Many** relationship between Employees and Projects (assume a project model exists and employees can work on multiple projects).
   - Use **Polymorphic Relations** (Morph Relation) to allow both departments and employees to have notes (assume a notes model that can be associated with multiple models).

4. **Soft Deletes**:
   - Implement soft deleting for departments and employees, allowing them to be restored or permanently deleted.

5. **API Development**:
   - Develop an API that allows third-party applications to get the list of departments, view employees within a department, and manage employees (add, update, delete).
   - Secure the API with API tokens.

6. **Extra Features**:
   - Implement setters and getters in the Employee model to ensure that the `first_name` and `last_name` are properly capitalized upon retrieval and storage.

Practical Implementation Steps:

1. Setup a New Laravel Project:
   - Use Laravel's 10 version and set up the environment and database connections.

2. Authentication:
   - Use Laravel's built-in authentication scaffolding (JWT) to set up the auth system.

3. Database Migrations and Models:
   - Create migrations for departments, employees, projects, and notes.
   - Define models with the necessary relationships (hasMany, belongsTo, belongsToMany, morphMany).

5. API Routes and Controllers:
   - Define RESTful routes in Laravel for the API.

- Implement controllers with methods for handling the API requests, utilizing Resource Controllers for clean code.

Submission:
Students are expected to submit a GitHub repository link containing their complete Laravel project. The README file should include instructions on setting up the project, running migrations, seeding the database, and any other necessary configuration details.