Test assignment for back-end developer

Overview

You are tasked with building a RESTful API service using PHP with the Laravel framework to manage a "Company Device Tracker" application. This service will manage operations related to tracking devices like laptops, screens, and keyboards that employees keep at home. The project should utilize Docker for the environment setup, MariaDB for the database, Redis for caching, OpenAPI for API documentation, and PEST for testing. The API will use JWT for authenticating requests.

Requirements

1. API Development
    - Develop a Laravel application providing API endpoints for the following functionalities:
        - GET /devices: List all devices, showing which employee holds them, paginated. Requires authentication.
        - POST /devices: Register a new device to the system. Requires authentication.
        - PUT /devices/{id}: Update device details (e.g., change the holder, status). Requires authentication.
        - DELETE /devices/{id}: Remove a device from the system. Requires authentication.
        - GET /employees: List all employees with the devices they hold, paginated. Requires authentication.
        - POST /employees: Add a new employee. Requires authentication.
        - PUT /employees/{id}: Update an existing employee's details. Requires authentication.
        - DELETE /employees/{id}: Remove an employee from the system. Requires authentication.
    - Implement a POST /login endpoint for JWT authentication where users provide credentials and receive a token to use for authenticated requests.
2. Database
    - Create a schema with at least three tables: devices, employees, and users for managing credentials.

- The users table should include fields like id, username, password_hash, email.
- Proper foreign key constraints should link devices to employees.

3. Docker
   - Include a Dockerfile and docker-compose.yml for setting up Laravel, MariaDB, Redis, and any necessary services for JWT handling.
4. Redis
   - Implement Redis for caching data of GET requests to reduce load times and database calls.
5. Testing
   - Write tests using PEST to ensure functionality and security of the API, covering all endpoints, with specific attention to authentication and proper access control.
6. API Documentation
   - Use OpenAPI to document all endpoints, including authentication details, parameters, and expected responses.
7. Submission
   - Provide a GitHub repository containing all source code, Docker files, database schemas, and documentation.
   - A README.md should be included with detailed setup instructions, usage guidelines, and a discussion of the API design.