DEINE AUFGABE

We were approached by one of our recently gained customer called “The Galactic Empire” who is in dire need for an app to manage their fleet in an upcoming “business expansion project” (their words).

Note: the most important part is the backend functionality. Fancy UI is not as important.

- Set up Django REST Framework for building APIs.
- Configure Django models for user management with different roles (e.g., Admin, Manager, Employee).
- Implement authentication and authorization using Django’s built-in authentication system, Django REST Framework’s token-based authentication or using JSON Web Tokens (JWT).
- Ensure that users can have different roles assigned during registration or by an admin user.
- Define permissions and roles for accessing different parts of the application (e.g., Admin can create/update/delete users, Managers can manage specific resources, Employees have read-only access, etc.).
- Implement middleware or decorators to enforce role-based access control on API endpoints.
- Ensure that unauthorized users receive appropriate error responses when trying to access restricted resources.
- Implement some models with appropriate relations and (at least the following) values/properties:
    - Person
        - `name`
        - `rank` (Options (in increasing order): Recruit, Lieutenant, Captain, Commander, Admiral)
        - `stationed_currently` (the ship the person is currently stationed at)
        - `stationed_past` (ships the person was stationed at)
    - Ship
        - `name`
        - `crew` (list of Crew-members)
        - `size` (number of crew)

        - `fleet`
        - `ranks` (number of different ranks on ship)
    - Fleet
        - `name`
        - `admiral`
        - `ships` (list of ships belonging to fleet)
        - `size` (number of ships in fleet)
        - `ranks` (number of different ranks in fleet)
- Create instances for all models in your database
    - Make sure there are not more low-level ranks than higher-level ranks (e.g.: it is impossible to have 10 Commanders and 8 Captains)
    - Only 1 Admiral can supervise a fleet
    - At least 3 fleets
    - At least 8 ships (per fleet)
    - At least a crew of 100 per ship
- Create (at least) the following endpoints:
    - **GET** `/fleets`
        - List all fleets
        - Show name, admiral and size
    - **POST** `/fleets` (only admins)
        - Create a new fleet
    - **GET** `/fleets/<fleet-id>`
        - List only the specified fleet
        - Show name, admiral, ships, size, ranks
    - **GET** `/ships`
        - List of all ships
        - Show name, fleet, size
    - **POST** `/ships` (only admins)
        - Create a new ship
    - **GET** `/ships/<ship-id>`
        - List only the specified fleet

        - Show name, fleet, ranks, crew
    - **GET** `/people`
        - List of all people
        - Show name, rank, stationed_currently
    - **POST** `/people` (Admin + Manager)
        - Create a new person
    - **GET** `/people/<person-id>`
        - List of specified person
        - Show name, rank, stationed_currently, stationed_past
- Optimize database queries
    - Identify potential N+1 Queries
    - Implement queryset optimizations (e.g., select_related, prefetch_related) to minimize database queries and improve performance.
    - **Optional**: Use Django Debug Toolbar or similar tools to profile and analyze database queries before and after optimization.
- Bonus: We encourage creativity and initiative in suggesting improvements and optimizations to the project, for example regarding scalability and performance.
- Present your project in a convincing way.