PROJECT #8.1

Objective:

to develop a Course Management System using C# and .NET that allows users to create, manage, and track booking and resources. The system must implement CRUD operations, user authentication, and RESTful APIs. The APIs must be consumed by a web interface, that allows user to interact with them.

Description

Create a full application that covers the managements of computer labs inside a company that allows to book both physical resources, such as computer, blackboards, projector, and software resource, such as specific applications, remote access to library, etc.
The student must develop the backend, the front end and some test units covering the backend functionalities.

Requirements

User registration and authentication

1. Users should be able to register an account with their email and password.
2. Implement user authentication using tokens and persistence.
3. Ensure that only authenticated users can access certain functionalities.
4. There exists different categories of users that have different access rules(*)

User admin

1. User admin can create, modify, update any user profile (except password)
2. User admin can access all users' information.
3. User admin can block or re-enable any user.
4. User admin can performs some report on users (such as number of login, time of use, etc.)(*)

Lab management

1. Lab Administrator can add new Computer specifying (at least) a name, a description, some technical specification, and the status. Once inserted the date of creation must be stored and a unique id (string of 10 character) must be generated and saved.
2. Lab administrator can modify an existing computer but cannot change the unique id and creation date.
3. Lab administrator can change the status of each computer resources (state can be available, maintenance, out of order, removed, reserved)
4. Lab administrator manage the list of available software and assign them to specific Computer or group of computers(*)
5. Lab administrator manage the list of available resources
6. Lab administrator assign o move a computer to a classroom

User access

1. Once logged, the user can search for a resource and book it if available, if the resource is a computer you can reserve up to 2 slots each day.
2. The user can delete a booking.
3. The user can signal that they are using the computer.
4. The user can receive a Report of their use of resources. (*)
5. Each reservation is managed in one-hour time slots, from Monday to Friday, from 9:00 AM to 6:00 PM.

Commentato [MG1]: Non ho ben compreso questa specifica

RESTful API:

1. Implement a RESTful API that exposes endpoints for User management and Lab management (CRUD operations).
2. Note that the ai endpoint should be distinguished according to their functionalities and should be implemented as decoupled as possible. (*)
3. API should follow RESTful principles, with clear and meaningful URLs.
4. Include proper HTTP methods (GET, POST, PUT, DELETE) for each operation.

Persistence

1. The persistence can be implemented using the preferred mechanism and format.
2. The location of data must be independent from specific locations. (*)

User Interface:

1. Create the REST API consumer and a simple web interface for users to interact with the services.
2. Feel free to use bootstrap (*)
3. The web interface must require user authentication before showing any other content.
4. Once the user is logged-in, different content must be shown to match user's role (admin or simple user) and their capabilities.
5. Personalize a bit the web UI to show the role and username/email.
6. Perform form validation every time user input is required. Display any validation errors.
7. Provide feedback to the user when actions are successful (not necessarily by showing a message) or not (display the errors).

Deliverable:

1. A working-class management system.
2. A brief report of the architecture using UML class diagram.
3. A presentation including use cases (*).

Grading Criteria:

1. Functionality: Completeness and correctness of user registration, authentication, task management, and CRUD operations.
2. API Design: Proper use of HTTP methods, meaningful URLs, and adherence to RESTful principles.
3. Data Storage: Effective implementation of data storage.
4. User Interface: Usability and user-friendliness of the web interface.
5. Authentication and Authorization: Secure user authentication and appropriate access controls.

6. Code Quality: Well-structured, readable, and maintainable code.
7. Presentation: Ability to explain and present the project.
8. Test: completeness, correctness of the approach and covering of the test proposed

Note that requirements marked with (*) are optional, however they contribute to the valuation.