

Faculty of Automation and Computer Science

**Sofware Design**

StackOverflow

**Student:**

Stan Andreea-Alexandra

Group 30413

Year III

**Table of contents:**

1. Introduction.................................................................................................................pg 2

2. Tech Stack................................................................................................................pg3

3.Use case diagrams ..............................................................pg4

4. Software architecture ................................................................................................pg5

5. Package diagram......................................................................................pg17

6. Class Diagram........................................................................................................pg18

7. Database diagram.....................................................................................................pg19

8. Endpoints requests.........................................................................................pg22

9. Architecture of the frontend..................................................................................pg22

8. Routing ......................................................................................................pg22

1. Introduction

The purpose of this application is to replicate the main features of StackOverflow, in a simplified manner. Stack Overflow is a question and answer website for programmers. It features questions and answers on certain computer programming topics

1.1 Project Overview

My StackOverflow-like system will allow users to ask questions, provide answers, and engage in a voting system to express their opinion on certain questions and answers. The system will have two types of users: a normal user and a moderator

1.2 Key Features

The following features will be implemented in our StackOverflow system:

* Feature 1: Asking Questions

Users will be able to ask questions, the question including information such as the author, title, text, creation date and time, as well as an optional picture. Each question can be associated with one or more tags, and users should be able to create new tags if necessary (if the name of the tag they want to add doesn’t exist yet).

The list of questions will be displayed, sorted by the creation date, with the most recent question appearing first. Users will be able to filter questions by tags, conduct text-based searches, filter by specific users, or view only their own questions. The text search functionality will primarily focus on the question title.

Question authors will have the ability to edit or delete their own questions.

* Feature 2: Answering Questions

Each question can receive one or more answers from any user, including the original author of the question. Answers will consist of an author, text, an optional picture, and a creation date and time. Answer authors will have the ability to edit or delete their own answers.

When viewing a question individually, the list of answers associated with that question will be displayed.

* Feature 3: Voting System

Users will have the ability to vote on both questions and answers. Voting options include upvoting and downvoting for questions and answers. Each user can only vote once on each question or answer. Users cannot vote on their own questions or answers.

The vote count for each question and answer will be displayed.

2. Tech Stack

The stackoverflow website project is being developed using a combination of Java Spring and Angular technologies. Java Spring is a popular framework for developing web applications, providing a robust and scalable platform for building complex applications. Angular, on the other hand, is a popular front-end framework for building dynamic and responsive web applications.

The backend of the stackoverflow website is being developed using the Java Spring framework. The Spring framework provides a comprehensive set of tools and libraries for building web applications, including database access, and security. The Spring framework also provides support for RESTful web services, which are used to facilitate communication between the frontend and backend.

For the frontend, the Angular framework is being used. Angular provides a powerful set of tools and libraries for building dynamic and responsive web applications, including support for modular architecture, data binding, and dependency injection. The Angular framework also provides support for reactive programming, which enables the website to respond to user actions and updates in real-time.

Throughout the development process, various libraries will also be used, frameworks, and tools specific to Angular and Java Spring, such as Angular CLI, TypeScript, Spring Boot, Hibernate, and Spring Security.

The project also includes a number of other technologies and tools, including HTML, CSS, TypeScript, and Bootstrap. HTML is used for creating the structure of the web pages, while CSS is used for styling and layout. TypeScript is a superset of JavaScript that provides enhanced type checking and tooling for developing web applications. Bootstrap is a popular CSS framework that provides a set of pre-built components and styles for building responsive web applications.

The IDE I’m using is IntelliJ IDEA, which is a popular Integrated Development Environment for Java developers. IntelliJ IDEA that supports a wide range of technologies and frameworks, including Java, Spring, Angular, and many others. It provides a wide range of tools and features, such as code completion, refactoring, debugging, and testing.

IntelliJ also includes a range of debugging and testing tools, which can help. It provides integration with popular testing frameworks such as JUnit and TestNG, as well as support for debugging code in real-time.

Another useful feature of IntelliJ IDEA is its support for version control systems such as Git. It provides integration with popular Git hosting platforms such as GitHub, GitLab, and Bitbucket, making it easy to share code and collaborate on projects.

3. Use-case diagrams

A use case diagram is a type of behavioral diagram in UML that depicts the interactions between users (actors) and the system to accomplish specific goals or tasks. It provides a high-level overview of the system's functionalities and the actors involved.

O imagine care conține text, captură de ecran, cerc, lună

Descriere generată automatThis is the use case diagram for a regular user. The regular user should be able to: register, login, ask a questions, answer a question, filter questions, edit their own questions or answer, delete their own questions or answers and vote on questions and answers.

This is the use case diagram for a moderator. The moderator can do additionally to what a regular user can do the following: ban/unban users, delete or edit any question or answer

O imagine care conține text, captură de ecran, cerc, primăvară

Descriere generată automat

4. Software architecture

Software architecture refers to the basic structure of any software system and incorporates any aspect that makes a system function and behaves as it should.

The Layered architecture is considered as the most popular and extensively adopted architectural pattern in software development, also referred to as n-tier architecture. It comprises multiple horizontal layers that work together as a cohesive software unit. A layer in this architecture represents a logical division of code or components. Components that are related or that are similar are usually placed on the same layers. However, each layer is different and contributes to a different part of the overall system.

O imagine care conține diagramă

Descriere generată automat

There are four layers in this architecture where each layer has a connection between modularity and component within them. From top to bottom, they are: the presentation layer, business layer, persistence layer and database layer.

* Presentation layer: This layer is responsible for handling user interactions and presenting the data to the end-users. It includes components such as controllers, views, and templates. The presentation layer is built using web technologies such as HTML, CSS, and JavaScript. It communicates with the business layer to retrieve data and perform necessary operations.
* Business layer: This layer is responsible for implementing the core functionality of the application. It includes business rules, algorithms, and workflows that govern how the application works. The business layer is built using programming languages such as Java or C# and may include components such as services, managers, or facades. It communicates with the presentation layer to retrieve data and send results.
* Persistence layer: This layer is responsible for interacting with the database or other data sources used by the application. It includes components such as repositories, data access objects (DAOs), or data models. The persistence layer is built using technologies such as SQL, JDBC, or ORM frameworks such as Hibernate or Entity Framework. It communicates with the business layer to perform database operations.
* Database layer: This layer is responsible for storing and managing the data used by the application. It includes components such as database servers, tables, and schemas. The database layer is built using database technologies such as MySQL, Oracle, or MongoDB.

Each layer communicates with the layer above and below it through well-defined interfaces, which allows for separation of concerns and makes the system more modular and maintainable. The Layered architecture provides a clear separation of concerns, making it easier to develop, test, and maintain software applications.

5. Package Diagram

A package diagram is a type of UML diagram that depicts the structure and organization of a software system by illustrating the dependencies between various packages or modules within the system. It helps to visualize the high-level architecture of an application and the relationships between different components.

* Entity package:

The entity package contains the classes that represent the fundamental entities or objects within the application's domain. In the context of StackOverflow, entities include User, Content, Tag, QuestionTag, Vote and Email. These classes define the attributes and behaviors of the objects, representing the real-world entities or concepts that the application manages.

* Repository package:

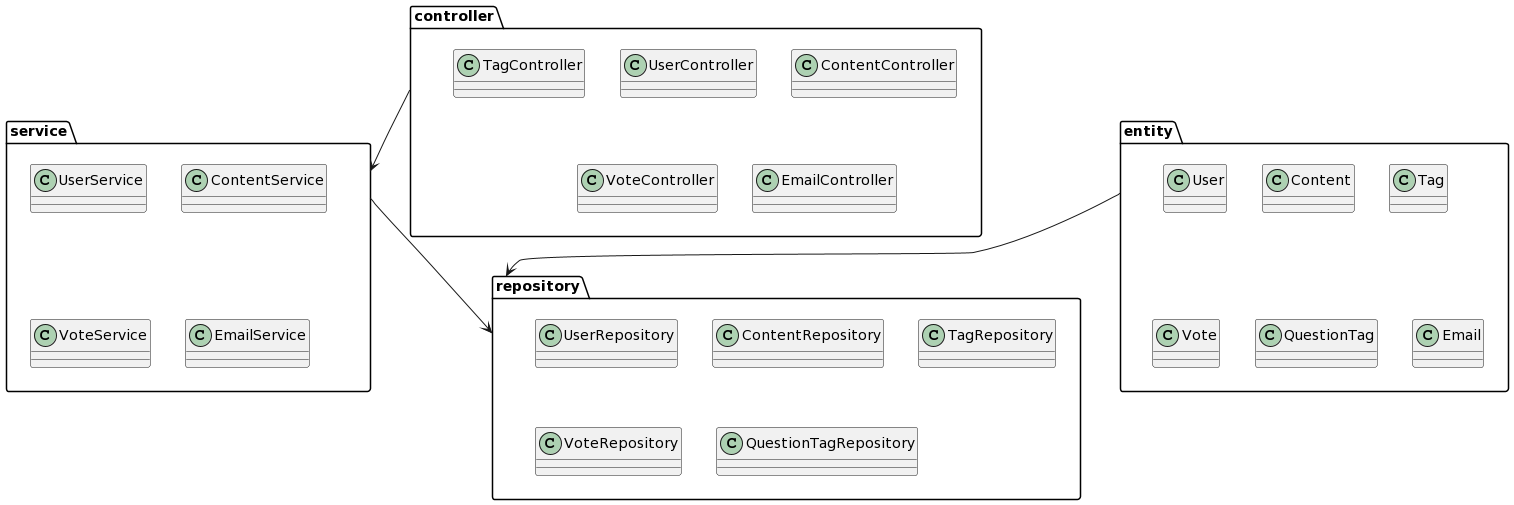
The repository package contains classes responsible for data persistence and retrieval. These classes interact with the underlying database or data storage mechanism and provide methods for querying, creating, updating, and deleting data entities. The repositories act as an abstraction layer between the services and the data storage, allowing the services to interact with the entities without directly dealing with the database operations.

* Service package:

The service package contains the classes that implement the core business logic of the application. These classes provide various services and operations required by the application, such as user authentication, question and answer management, user profile handling, and other functionality specific to the StackOverflow. Services encapsulate the business rules and orchestrate interactions between entities and repositories.

* Controller package:

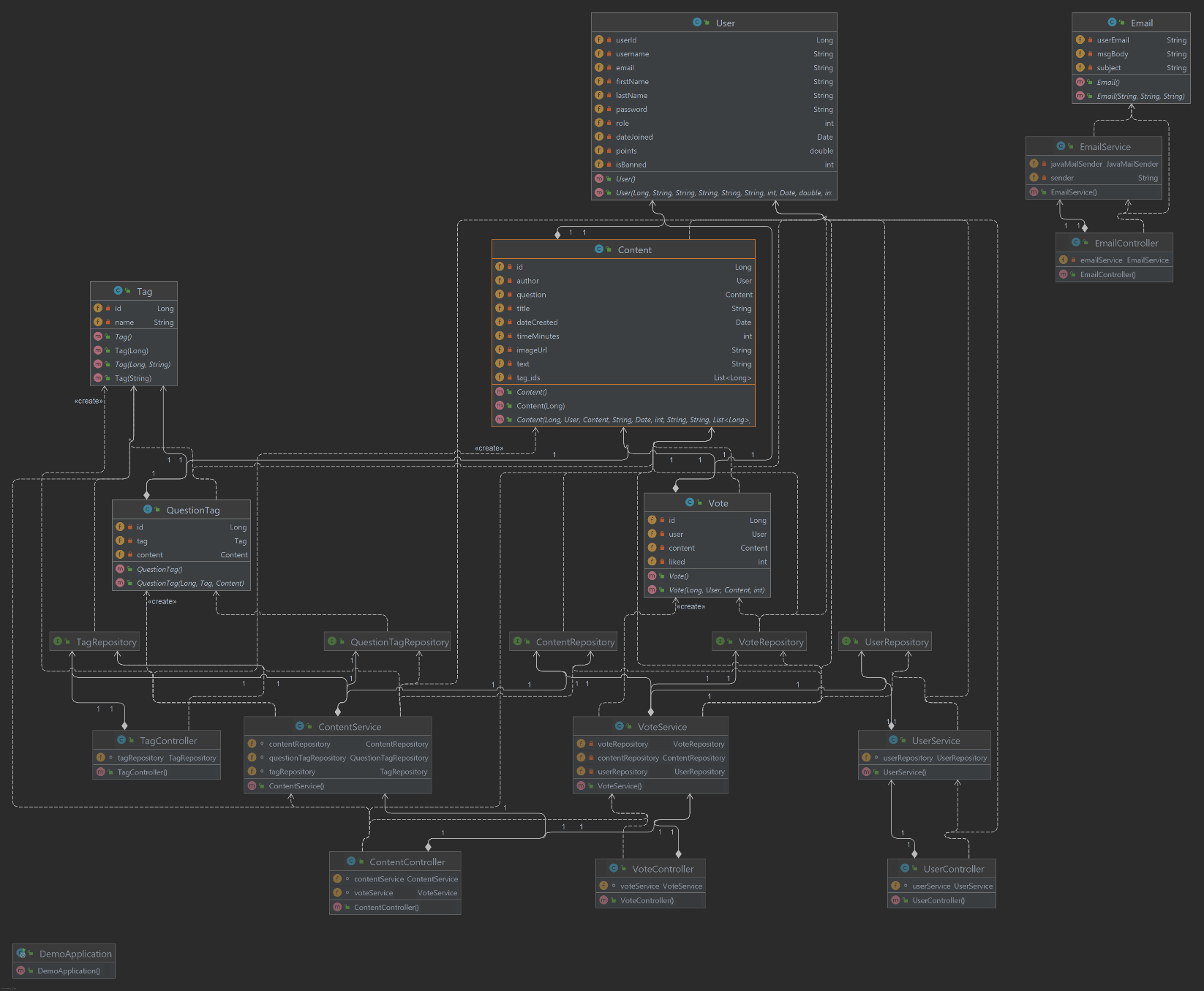
The controller package contains the classes responsible for handling incoming requests, managing the flow of data, and coordinating the overall behavior of the application. Controllers receive requests from the client, invoke the appropriate services, and return the responses. They act as an interface between the user or external system and the underlying logic of the application.

This is the package diagram for my implementation:

6. Class Diagram

A class diagram is another type of UML diagram that represents the static structure of a system by illustrating the classes, their attributes, methods, and the relationships between them. It provides a detailed view of the classes and their interactions within an application

This is the diagram for my implementation of the StackOverflow app.



* User: The User class represents a user entity in the StackOverflow app. It holds information about users, such as their name, email, password, and other relevant details.
* Content: The Content class represents various types of content entities, such as questions, answers, and comments, in the StackOverflow app. It encapsulates the data and attributes associated with content items.
* Tag: The Tag class represents tags associated with questions and other content in the StackOverflow app. It contains information about tags and their relationships to content items.
* Vote: The Vote class represents the voting action performed on content in the StackOverflow app. It stores information about votes, including the voter, the content being voted on, and the voting value (e.g., upvote or downvote).
* QuestionTag: The QuestionTag class represents the relationship between questions and tags in the StackOverflow app. It associates tags with their corresponding questions and helps in organizing and categorizing content.
* Email: The Email class is used for sending an email. It contains details such as the sender, recipient, subject, and content of an email notification.
* UserRepository: The UserRepository class provides the data access layer for user-related operations in the StackOverflow app. It handles database interactions for user entities, such as retrieving user information, storing user details, and performing user-related queries.
* ContentRepository: The ContentRepository class acts as the data access layer for content-related operations in the StackOverflow app. It handles database interactions for content entities, such as creating, retrieving, updating, and deleting content items.
* TagRepository: The TagRepository class provides the data access layer for tag-related
* VoteRepository: The VoteRepository class acts as the data access layer for vote-related operations in the StackOverflow app.
* QuestionTagRepository: The QuestionTagRepository class provides the data access layer for managing the relationship between questions and tags in the StackOverflow app. It handles database interactions for associating tags with questions, retrieving questions by tags, and managing question-tag relationships.
* UserService: The UserService class encapsulates the business logic for user-related operations in the StackOverflow app. It interacts with the UserRepository to perform tasks such as user authentication, profile management, and user-specific functionality.
* ContentService: The ContentService class provides the core functionality for managing various types of content in the StackOverflow app. It communicates with the ContentRepository to handle tasks such as content creation, retrieval, updates, and deletions.
* VoteService: The VoteService class manages the voting functionality for content in the StackOverflow app. It interacts with the VoteRepository to handle voting operations on questions, answers, and other content items.

EmailService:

* The EmailService class handles email-related functionality in the StackOverflow app.

It communicates with external services or libraries to send email notifications to users.

* UserController: The UserController class is responsible for handling requests related to user management in the StackOverflow app. It interacts with the UserService to perform operations such as user registration, login, profile management, and other user-related actions.
* ContentController: The ContentController class handles requests related to content management. It works with the ContentService to handle content operations.
* VoteController: The VoteController class manages requests related to voting on content in the StackOverflow app. It interacts with the VoteService to handle voting operations on questions and answers.
* EmailController: The EmailController class handles requests related to email notifications and communication. It works with the EmailService to send and manage email notifications.

7. Database diagram

A database diagram visually represents the structure of a database, including tables, columns, relationships, and constraints. It provides an overview of how the data is organized and the relationships between different entities.

O imagine care conține text, captură de ecran, Font, linie

Descriere generată automatThis is my database diagram

* User table:

The User table includes columns such as id, username, email, first\_name, last\_name, password, role, date\_joined, points, and is\_banned. There are associations between the User table and other tables through foreign keys.

* Content table:

The Content table contains columns such as id, author\_id, question\_id, title, date\_created, time\_minutes, image\_url, and text. This table has a foreign key (author\_id) that references the User table, indicating that a user can be the author of multiple content entries.

* Tag table:

The Tag table includes columns such as id and the name of the tag.

* Question\_tag table:

The Question\_tag table contains columns such as id, tag\_id, and question\_id. The Question\_tag table has a foreign key (question\_id) that references the Content table, and a another foreign key (tag\_id) that references the Tag table, indicating a relationship between a question and its associated tags.

* Vote table:

The Vote table includes columns such as id, user\_id, content\_id, and like. The Vote table has a foreign key (content\_id) that references the Content table and also another one (user\_id) that references the User table, representing the relationship between a user and their votes. The like column is used to represent the like or dislike of a user regarding the content. If likes equals 1, it means that the user liked the content and if it is 0 it means that the user disliked it.

8. Endpoints requests

Endpoint requests refer to the different types of HTTP requests that can be made to the various endpoints of an application's API. These requests allow clients (such as web browsers or mobile apps) to interact with the server and perform specific actions or retrieve information. Here are some used endpoint requests :

* GET: The GET request is used to retrieve data from the server. It is a read-only operation that does not modify any data on the server. This request is typically used to retrieve information, such as fetching a list of resources or retrieving details of a specific resource.
* POST: The POST request is used to send data to the server to create a new resource. It is commonly used for submitting forms, creating new records, or performing other actions that result in the creation of data on the server. The data to be created is usually included in the body of the request.
* PUT: The PUT request is used to update an existing resource on the server. It replaces the entire resource with the new data provided in the request. This request is commonly used for updating existing records or performing full replacements of resource data.
* DELETE: The DELETE request is used to remove a specified resource from the server. It is used to delete data or records from the server. The resource to be deleted is typically identified in the URL or request parameters.

These endpoint requests, along with their corresponding HTTP methods, provide a way to perform different operations on the server's resources. By using the appropriate request method and providing the necessary input data, clients can interact with the server and perform actions such as retrieving data, creating new records, updating existing records, or deleting resources.

Here are some examples of endpoint requests I am using:

1. Endpoint: /user/getAll

HTTP Method: GET

Description: This endpoint is used to retrieve a list of users.

Input: No input parameters required.

O imagine care conține text, software, Pictogramă computer, număr

Descriere generată automatOutput: The response will include a list of user objects with attributes such as id, username, email, first\_name, last\_name, role, and points.

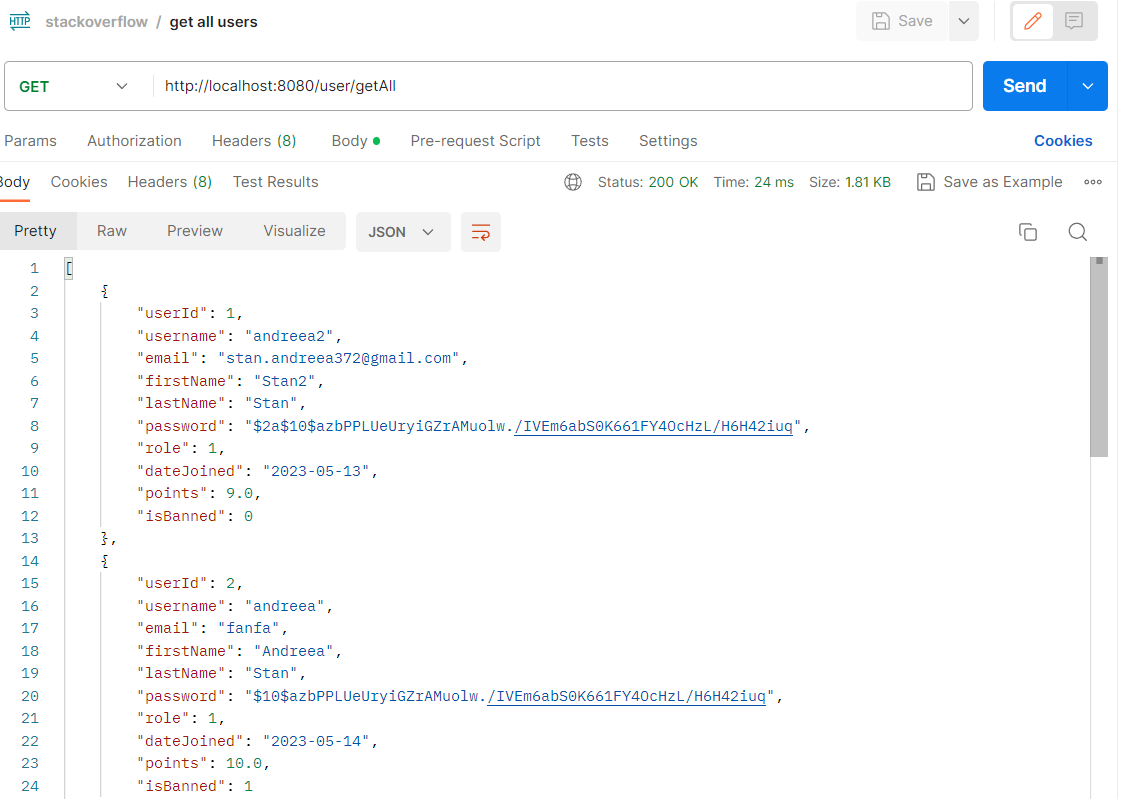
2. Endpoint: /users/getById/{id}

HTTP Method: GET

Description: This endpoint is used to retrieve details of a specific user by their ID.

Input: The id path parameter specifies the ID of the user to retrieve.

Output: The response will include the user object with attributes such as id, username, email, first\_name, last\_name, role, and points.



3. Endpoint Request: /user/deleteById/{id}

HTTP Method: DELETE

Description: This request is used to delete a user with a certain id from the server.

Input: No additional input parameters required apart from the user ID specified in the URL.

Output: Will show an error message if the delete couldn’t be done

4. Endpoint Request: http://localhost:8080/user/insert

HTTP Method: POST

Description: This request is used to insert a new user into the server.

Input: The input data required for creating the user is sent in the request body, in a json format.

Output: The response will be the newly created user object with its generated ID. If there are any issues with the insertion, an appropriate error message may be returned.

O imagine care conține text, captură de ecran, software, Pagină web

Descriere generată automat

5. Endpoint Request: http://localhost:8080/content/update/{id}

HTTP Method: PUT

Description: This request is used to update the content with a given id.

Input: The input data required for updating the content is typically sent in the request body, following the appropriate data format specified by the server's API. It may include the fields that need to be modified.

Output: The response will be the updated content. If there are any issues with the update, an appropriate error message may be returned.

6. Endpoint: /questions/getAll

HTTP Method: GET

Description: This endpoint is used to retrieve the list of questions.

Input: No input parameters required.

Output: The response will include a list of question objects with attributes such as id, author\_id, title, date\_created, and time\_minutes.

9. Architecture of the Frontend

In order to implement this part of the assignment, I have used the Angular framework. The Angular framework follows a component-based architecture, which encourages the modular design of applications. This approach makes it easier to maintain and scale the application over time.

The project consists of several components, including the login component, the signup component, the logout component, the home component, the questions-page component, and the question-detail component. Each component has its own responsibilities and encapsulates a specific part of the application's functionality.

* The login and signup components handle user authentication and registration, respectively.
* The logout component is used to handle users logging out of their accounts, removing them from the current sessionStorage.
* The home component displays a welcome message to the user and provides links to other parts of the application, links which depend on wheter the user is logged in or not.
* The question list component displays a list of all the questions available in the application, and a form for the user currently logged in to be able to ask a new question.
* The question detail component displays the details of a particular question, including answer. Here there is also the functionality for the user logged in to be able to answer a question
* The profile component is used for displaying the information of the user currently logged in.

O imagine care conține text, captură de ecran, software

Descriere generată automat

The application also makes use of several services, including the Authentication service, AuthenticationGuard service, HTTP interceptor service. and the Question service. The use of services ensures that the business logic of the application is separated from the UI logic, making it easier to test and maintain.

* The Authentication service handles user authentication, including methods for the login and registering of an user, for checking if a user is logged in or for retrieving some data about the user currently logged in.
* The Question service is responsible for retrieving and managing questions and their associated data.
* The AuthenticationGuard service manages the access authorization to certain links.It uses an event called canActivate. The canActivate is like a constructor. It will be called before accessing the routes. The canActivate has to return true to access the page. If it returns false, the user cannot access the page.
* The HttpInterceptor service is used to intercept all HTTP requests sent by the Angular application. The service checks if the user is logged in and adds authentication headers to the request if the user is authenticated. If the user is not authenticated, the service passes the request without modifying it.

O imagine care conține text, captură de ecran, Font

Descriere generată automatIn the Angular project, there are several models/interfaces defined to represent the data objects used throughout the application. These models/interfaces provide a structure for the data objects and help maintain consistency in the way data is passed between components and services.

O imagine care conține text, captură de ecran, Font, software

Descriere generată automatThere are three models/interfaces in this project: User, Question, and Answer.

* The User interface defines the properties of a user object, including id, username, password, and email. This interface is used to create user objects and ensure that each user object has the necessary properties.
* The Question interface defines the properties of a question object, including id, authorId, title, description, and answers. The answers property is an array of Answer objects. This interface is used to create question objects and ensure that each question object has the necessary properties.
* The Answer interface defines the properties of an answer object, including id, questionId, authorId, text. The author property is a string representing the username of the user who posted the answer. This interface is used to create answer objects and ensure that each answer object has the necessary properties.

In this project, HTML, CSS, and TypeScript were to create a dynamic and responsive web application.

HTML is the standard markup language used to create the structure and content of web pages. In this project, HTML was used to define the layout and structure of the various pages, as well as to create forms and tables to display data.

SCSS is a style sheet language used to describe the presentation of a document written in HTML. CSS was used extensively in this project to style the various elements of the web pages, including fonts, colors, backgrounds, borders, and positioning.

TypeScript is a superset of JavaScript that adds optional static typing and other features to the language. In this project, TypeScript was used to write the business logic and data handling code for the application, including services, components, and models.

O imagine care conține text, captură de ecran, software, Pagină web

Descriere generată automat

**2. Routing**

The routing module in an Angular application maps specific URLs to specific components. In this project, the AppRoutingModule is used to define the application routes.

The RouterModule and Routes are imported from the @angular/router package. The Routes array defines the different routes in the application, with each route having a path and a corresponding component.

The path property in the route specifies the URL path that should match with the corresponding component. If the URL matches with the path, the corresponding component is rendered.

In this project, the AuthGuard is also used to restrict access to certain routes based on the user's login status. For example, the QuestionDetailComponent is guarded by the AuthGuard, which means that only authenticated users can access the component.

Once the routes are defined, they are exported as an AppRoutingModule class which is then imported in the app.module.ts file to be used by the application. This allows the application to navigate to different pages by clicking on links or typing URLs in the browser.

O imagine care conține text, captură de ecran, Font

Descriere generată automatO imagine care conține text, captură de ecran, software

Descriere generată automat