

A
CAPSTONE PROJECT REPORT on

DoConnect



Submitted By:

C1 – Group 6

Ugandhar Venkata Sai Krishna

Pinniboina Yuva Rohith

Md. Faizan Khan

Devashish Singh

Sharanya K K

Abstract

Developers use Question and Answer (Q&A) websites to exchange knowledge and expertise. DoConnect is a popular Q&A website where developers discuss coding problems and share code examples. It only accepts questions about programming that are tightly focused on a specific problem. Questions of a broader nature—or those inviting answers that are inherently a matter of opinion—are usually rejected by the site's users, and marked as closed. DoConnect is intended to be a venue for broader queries, e.g. general questions about software development.

Closing questions is a main differentiation from other Q&A sites like Yahoo! Answers and a way to prevent low quality questions. It was created to be a more open alternative to earlier question and answer websites such as Experts-Exchange. The website serves as a platform for users to ask and answer questions, and, through membership by simply Signup and active participation answers up or down similar to Reddit and edit questions and answers in a fashion similar to a wiki.

Introduction

DoConnect is a question and answer site for professional and enthusiast programmers.. It features questions and answers on a wide range of topics in computer programming. It was created to be a more open alternative to earlier question and answer sites such as Experts-Exchange knowledge on the unknown things. Stackoverflow is sort of like the anti-experts-exchange (minus the nausea-inducing sleaze and quasi-legal search engine gaming) meets wikipedia meets programming reddit. It is by programmers, for programmers, with the ultimate intent of collectively increasing the sum total of good programming knowledge in the world. No matter what programming language you use, or what operating system you call home. Better programming is our goal.

It is created for asking questions and getting answers, sharin useful in the future coming days for the young generation which will be more as like Wiki for the present.

PROBLEM STATEMENT

DoConnect is a popular Q and A form in which techniques questions was asked and answered.

There are 2 users on the application: -

1.User

2.Admin

User Requisites :

1.As a user I should be able to login, Logout and Register into the application.

2.As a user I should be able to ask any question under any topic

3.As a user I should be able to search the question on any string written in search box

4.As a user I should be able to Answer any question asked

5.As a user I should be able to answer more than one question and more than one time

6.As a user I should be able to chat with other users.

7.As a user I should be able to upload images to refer

Admin Stories –

- 1.As an Admin I should be able to login, Logout and Register into the application.
- 2.As an Admin I should be able to get mail as soon as any new Question is asked or any Answers given.
- 3.As an Admin I should be able to approve the question and Answer. Any Question or Answer will be visible on the platform only if it is approved.
- 4.As an Admin I should be able to delete inappropriate Questions or Answers.

SYSTEM SPECIFICATIONS

SOFTWARE REQUIREMENTS:

Technologies:

- ✓ Angular
- ✓ Java
- ✓ Springoot

Languages:

- ✓ Type Script
- ✓ Java
- ✓ SQL Queries

IDE:

- ✓ Eclipse
- ✓ Visual Studio code

HARDWARE REQUIREMENTS:

Operating System:

- ✓ Windows 7/8/10/11 Linux distros MacOS X or later.

Processor:

- ✓ Intel or AMD dual core x86 processor.

RAM:

- ✓ 2 GB or above.

Hard disk:

- ✓ 500 MB of free disk space or more.

ANGULAR ARCHITECTURE

Angular is a platform or framework to build client-based applications in HTML and TypeScript. It is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that are imported into applications.

There are main eight blocks of Angular:

- ✓ Module
- ✓ Component
- ✓ Metadata
- ✓ Template
- ✓ *Data Binding*
- ✓ Service
- ✓ Directive

Dependency Injection Module:

Angular apps are modular and Angular has its own modularity system called Angular modules or NgModules. Every Angular app has at least one Angular module class, the root module, conventionally named AppModule. While the root module may be the only module in a small application, most apps have many more feature modules, each a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities.

NgModule is a decorator function that takes a single metadata object whose properties describe the module. The most important properties are:

- ✓ declarations - the view classes that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- ✓ exports - the subset of declarations that should be visible and usable in the component templates of other modules.
- ✓ imports - other modules whose exported classes are needed by component templates declared in this module.
- ✓ providers - creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- ✓ bootstrap - the main application view, called the root component, that hosts all other app views. Only the root module should set this bootstrap property.

Component:

Components are the most basic UI building block of an Angular app. An Angular app contains a tree of Angular components. Angular components are a subset of directives, always associated with a template. Unlike other directives, only one component can be instantiated for a given element in a template.

Metadata:

Metadata is used to decorate a class so that it can configure the expected behavior of the class. Following are the different parts for metadata. Annotations – These are decorators at the class level. This is an array and an example having both the @Component and @Routes decorator.

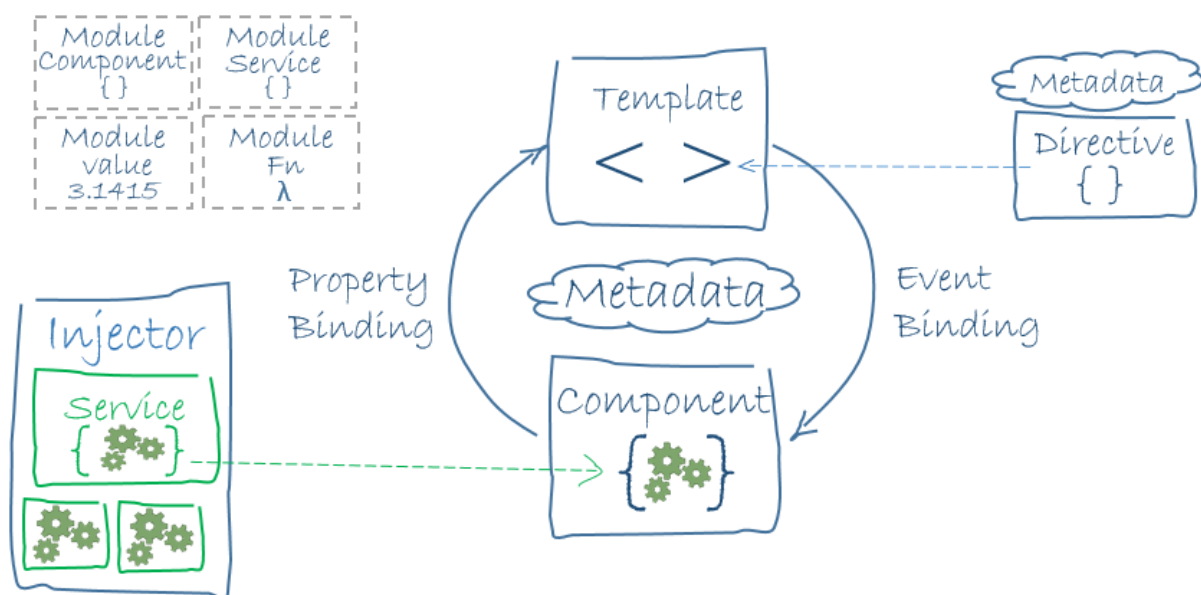


Fig: Angular Architecture

Template:

A template is a form of HTML that tells Angular how to render the component. Views are typically arranged hierarchically, allowing you to modify or show and hide entire UI sections or pages as a unit. The template immediately associated with a component defines that component's host view.

Data Binding:

Data binding in AngularJS is the synchronization between the model and the view. When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well.

Service:

Service is a broad category encompassing any value, function, or feature that your application needs. Almost anything can be a service. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

Examples include:

- ✓ logging service
- ✓ data service
- ✓ application configuration

There is nothing specifically Angular about services. Angular has no definition of a service. There is no service base class, and no place to register a service. Yet services are fundamental to any Angular application. Components are big consumers of services.

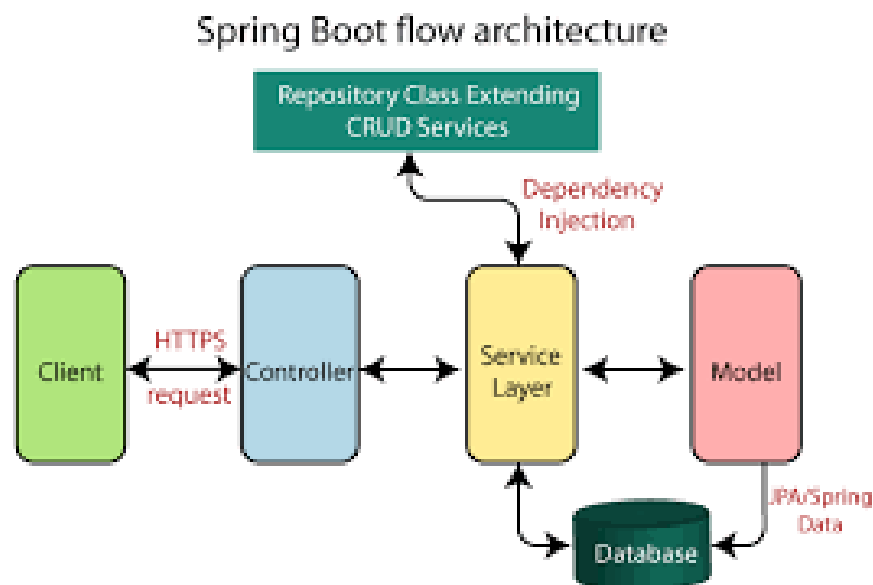
Directive:

Directives are classes that add new behavior or modify the existing behavior to the elements in the template. Basically directives are used to manipulate the DOM, for example adding/removing the element from DOM or changing the appearance of the DOM elements.

Dependency Injection:

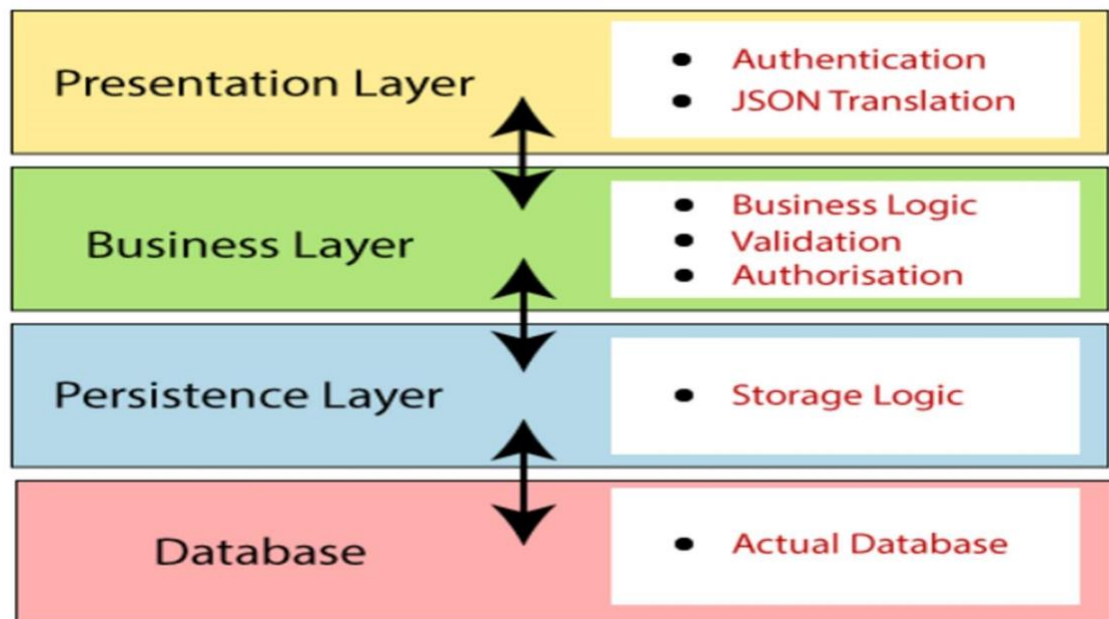
Dependency injection, or DI, is a design pattern in which a class requests dependencies from external sources rather than creating them. Angular's DI framework provides dependencies to a class upon instantiation. Use Angular DI to increase flexibility and modularity in your applications.

SPRINGBOOT Architecture



The spring boot consists of the following four layers:

- ✓ Presentation Layer – Authentication & Json Translation.
- ✓ Business Layer – Business Logic, Validation & Authorization.
- ✓ Persistence Layer – Storage Logic.
- ✓ Database Layer – Actual Database.



Presentation Layer:

The presentation layer is the top layer of the spring boot architecture. It consists of Views. i.e., the front-end part of the application. It handles the HTTP requests and performs authentication.

Business Layer:

The business layer contains all the business logic. It consists of services classes. It is responsible for validation and authorization. The persistence layer contains all the database storage logic.

Persistence Layer:

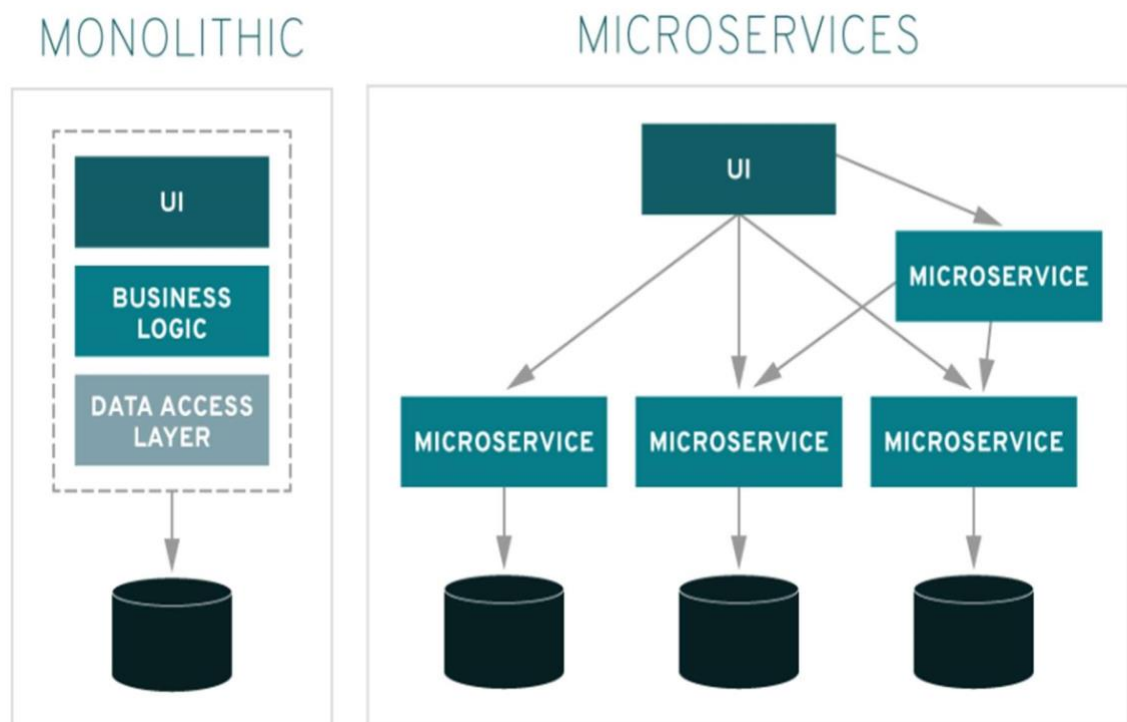
Persistence Layer: The persistence layer contains all the storage logic and translates business objects from and to database rows.

Database Layer: In the database layer, CRUD (create, retrieve, update, delete) operations are performed.

Database:

The database layer contains all the databases such as MySQL, MongoDB, etc. This layer can contain multiple databases. It is responsible for performing the CRUD operation.

MICROSERVICE ARCHITECTURE

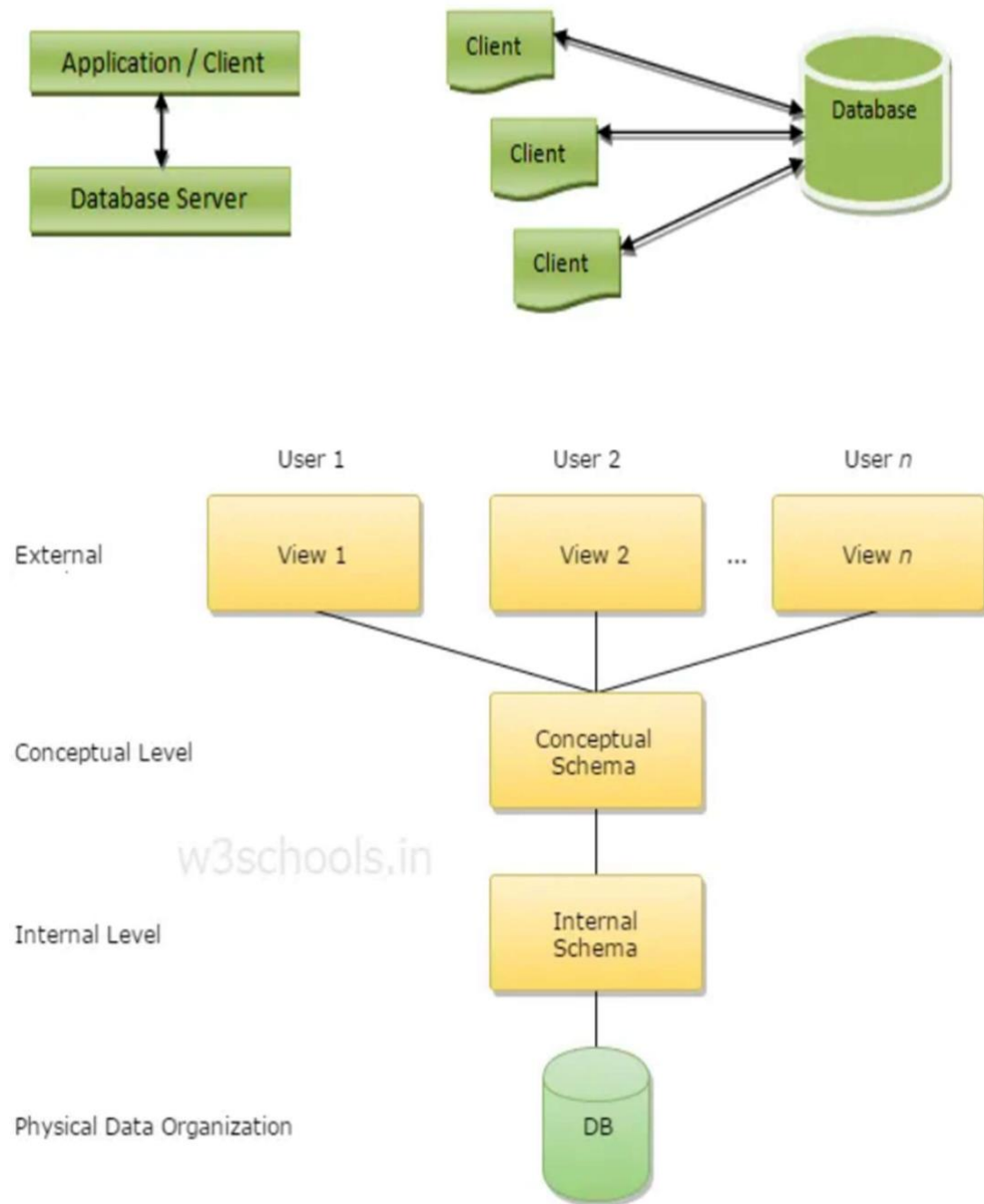


- Typically, micro services are used to speed up application development.
- Micro services architectures built using Java are common, especially Spring Boot ones.
- Microservices architecture (often shortened to microservices) refers to an architectural style for developing applications. Microservices allow a large application to be separated into smaller

independent parts, with each part having its own realm of responsibility.

- When you are ready to start adopting a microservices architecture and the associated development and deployment best practices, you'll want to follow the three C's of microservices: componentize, collaborate, and connect.

DATABASE ARCHITECTURE



- A Database Architecture is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the

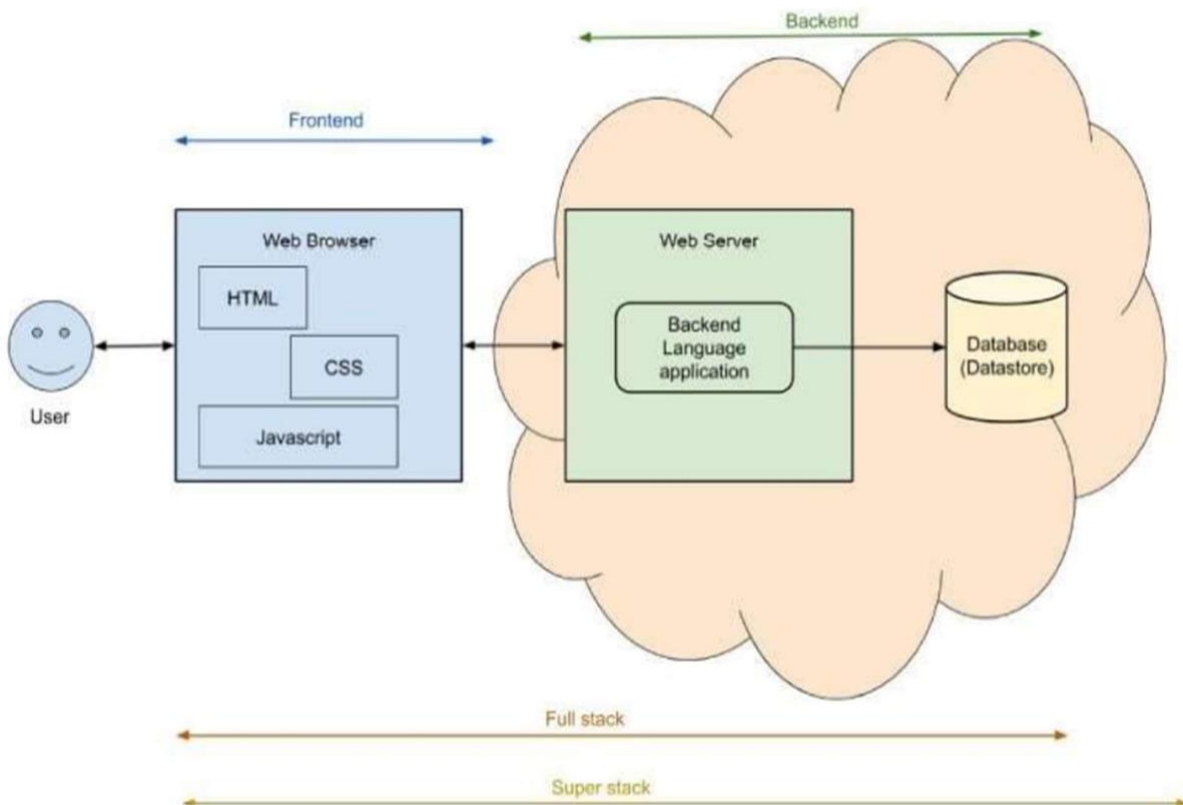
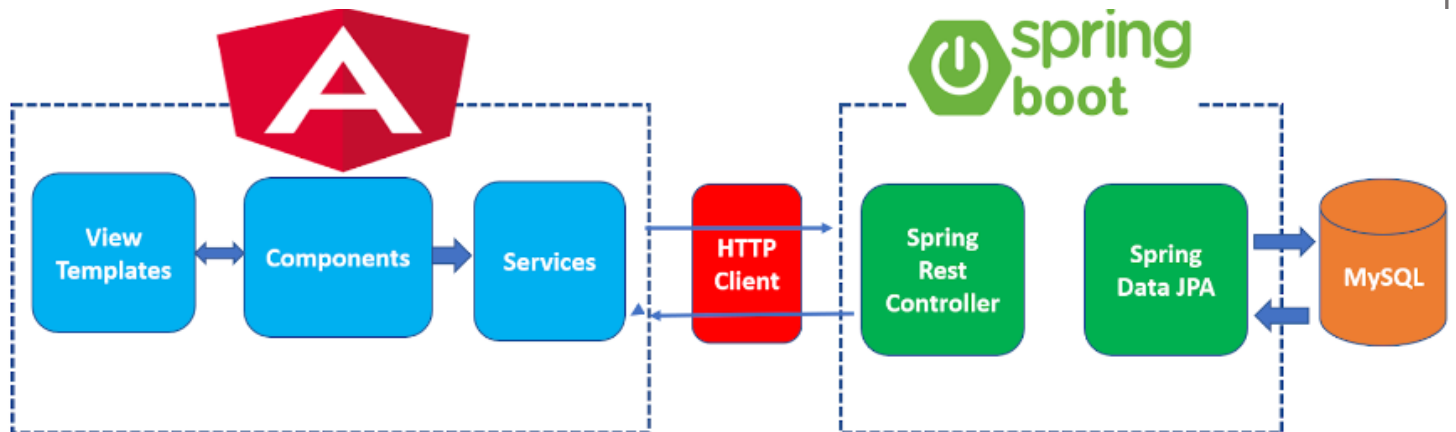
database system into individual components that can be independently modified, changed, replaced, and altered.

- The Database Management System (DBMS) architecture shows how data in the database is viewed by the users. It is not concerned about how the data are handled and processed by the DBMS. It helps in implementation, design, and maintenance of a database to store and organize information for companies.

Entities in the Database include:

- users
- admin
- questions
- answers
- topics

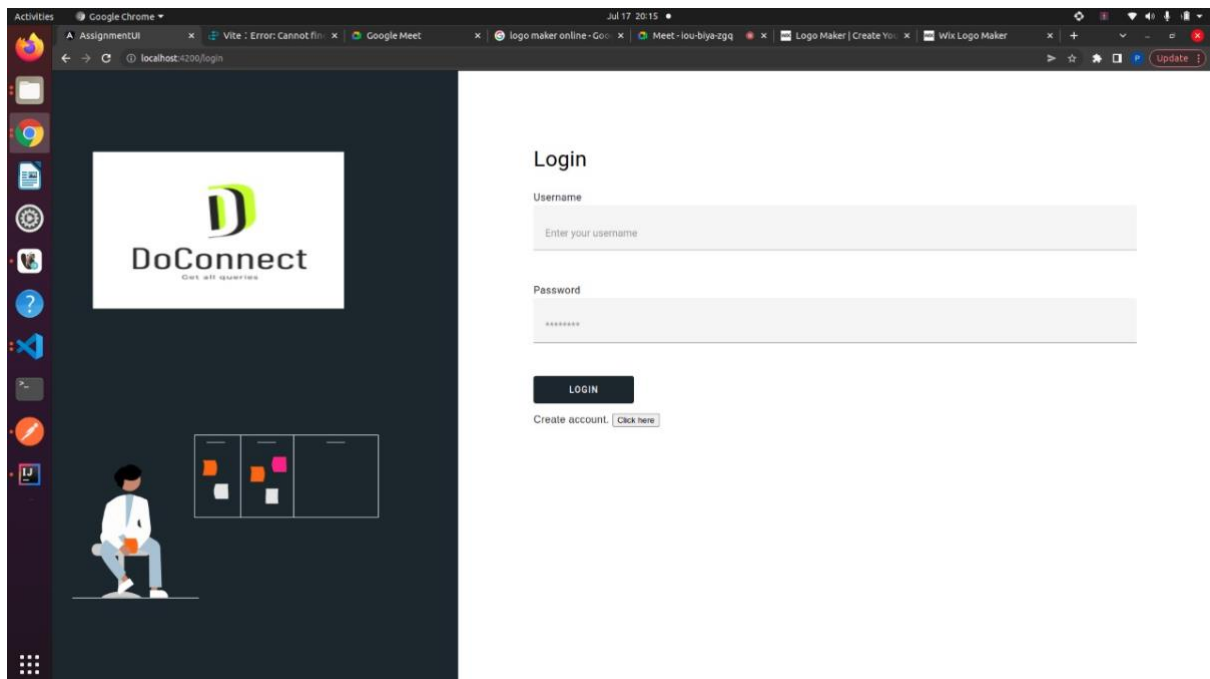
TOTAL PROJECT OVERVIEW

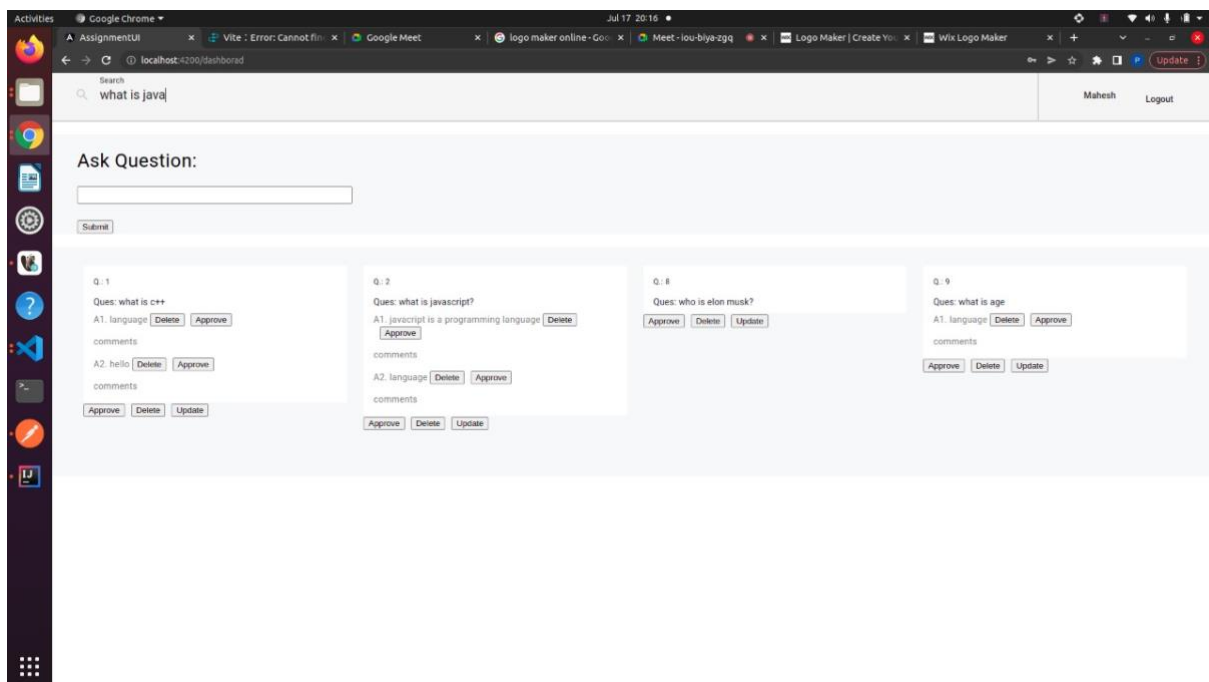
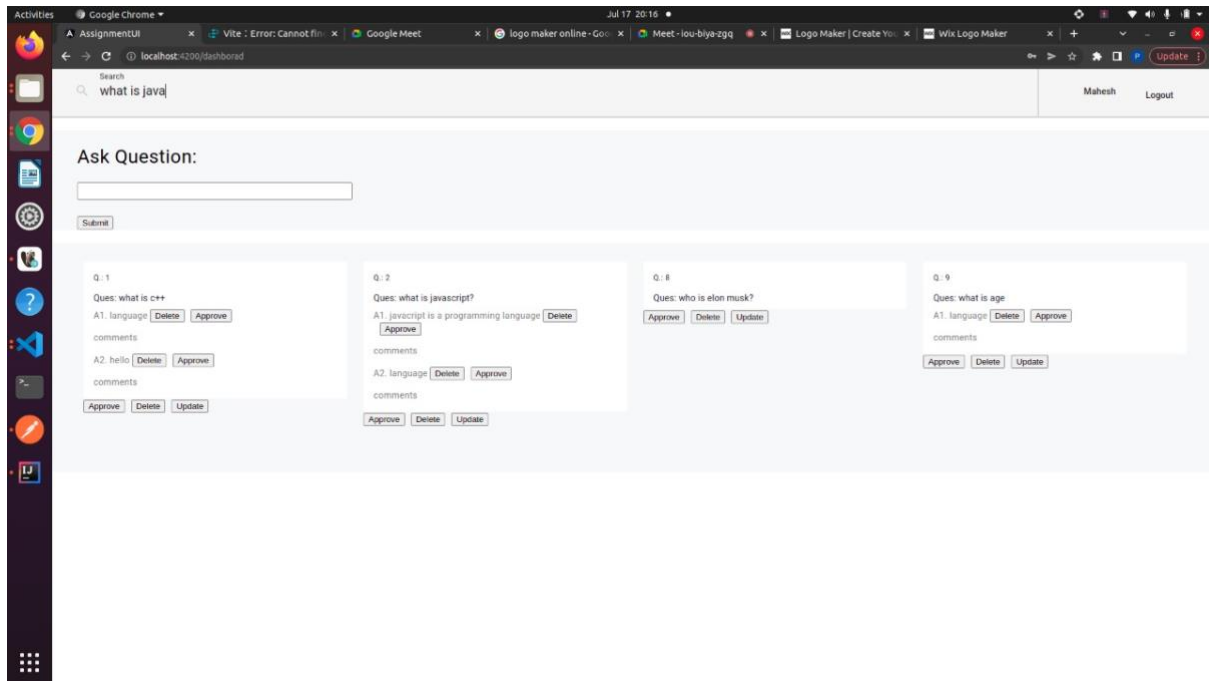


DoConnect...

Front End Design:

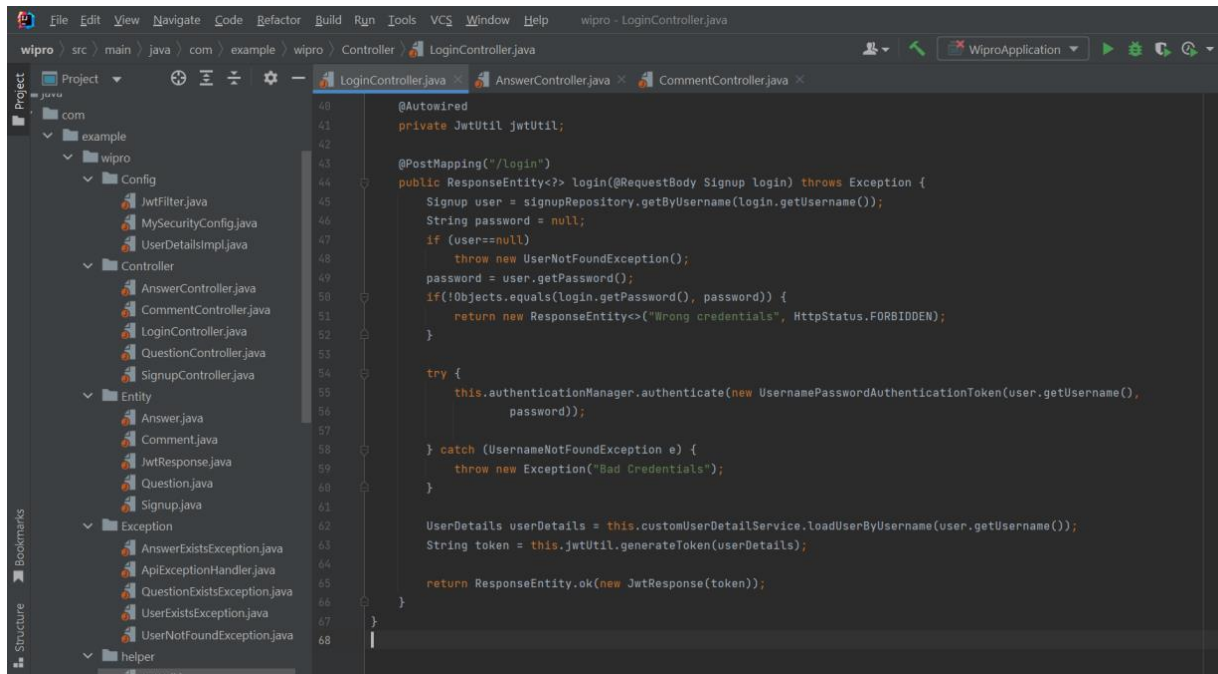
The whole frontend part for the project is developed by using Angular.





Backend Code

1. As a user I should be able to login, Logout and Register into the application.

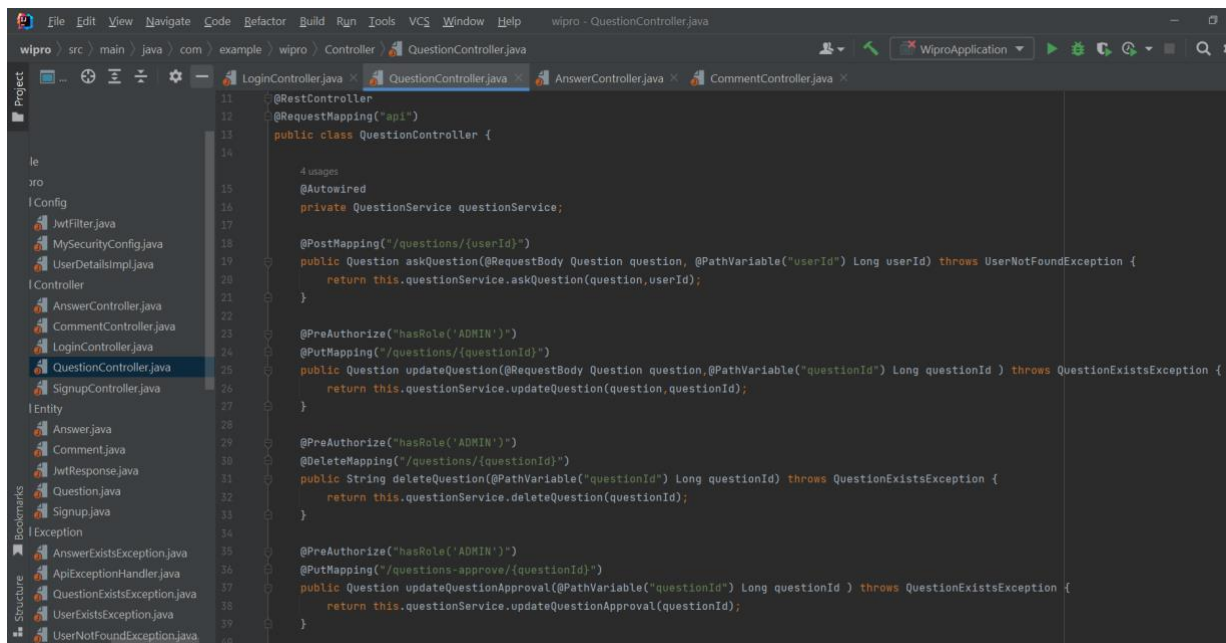


```

40 @Autowired
41 private JwtUtil jwtUtil;
42
43 @PostMapping("/login")
44 public ResponseEntity<?> login(@RequestBody Signup login) throws Exception {
45     Signup user = signupRepository.getByUsername(login.getUsername());
46     String password = null;
47     if (user == null)
48         throw new UserNotFoundException();
49     password = user.getPassword();
50     if (!Objects.equals(login.getPassword(), password)) {
51         return new ResponseEntity<>("Wrong credentials", HttpStatus.FORBIDDEN);
52     }
53
54     try {
55         this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(user.getUsername(),
56             password));
57     } catch (UsernameNotFoundException e) {
58         throw new Exception("Bad Credentials");
59     }
60
61     UserDetails userDetails = this.customUserDetailsService.loadUserByUsername(user.getUsername());
62     String token = this.jwtUtil.generateToken(userDetails);
63
64     return ResponseEntity.ok(new JwtResponse(token));
65 }
66
67 }
68

```

2. As a user I should be able to ask any question under any topic.



```

11 @RestController
12 @RequestMapping("api")
13 public class QuestionController {
14
15     4 usages
16     @Autowired
17     private QuestionService questionService;
18
19     @PostMapping("/questions/{userId}")
20     public Question askQuestion(@RequestBody Question question, @PathVariable("userId") Long userId) throws UserNotFoundException {
21         return this.questionService.askQuestion(question, userId);
22     }
23
24     @PreAuthorize("hasRole('ADMIN')")
25     @PutMapping("/questions/{questionId}")
26     public Question updateQuestion(@RequestBody Question question, @PathVariable("questionId") Long questionId) throws QuestionExistsException {
27         return this.questionService.updateQuestion(question, questionId);
28     }
29
30     @PreAuthorize("hasRole('ADMIN')")
31     @DeleteMapping("/questions/{questionId}")
32     public String deleteQuestion(@PathVariable("questionId") Long questionId) throws QuestionExistsException {
33         return this.questionService.deleteQuestion(questionId);
34     }
35
36     @PreAuthorize("hasRole('ADMIN')")
37     @PutMapping("/questions-approve/{questionId}")
38     public Question updateQuestionApproval(@PathVariable("questionId") Long questionId) throws QuestionExistsException {
39         return this.questionService.updateQuestionApproval(questionId);
40     }
41 }

```

3. As a user I should be able to Answer any question asked.

```

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("api")
public class AnswerController {

    @Autowired
    private AnswerService answerService;

    @PostMapping("answer/{questionId}/{username}")
    public Answer postAnswer(@RequestBody Answer answer, @PathVariable("questionId") Long questionId,
        @PathVariable("username") String username) throws QuestionExistsException, UserNotFoundException {
        return this.answerService.postAnswer(answer, questionId, username);
    }

    @PreAuthorize("hasRole('ADMIN')")
    @PutMapping("answer/{answerId}")
    public Answer updateAnswer(@RequestBody Answer answer, @PathVariable("answerId") Long answerId) throws AnswerExistsException {
        return this.answerService.updateAnswer(answer, answerId);
    }

    @PreAuthorize("hasRole('ADMIN')")
    @DeleteMapping("answer/{answerId}")
    public String deleteAnswer(@PathVariable("answerId") Long answerId) throws AnswerExistsException {
        return this.answerService.deleteAnswer(answerId);
    }
}

```

4. As an Admin I should be able to login, Logout and Register into the application

```

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/")
public class LoginController {

    @Autowired
    private JwtUtil jwtUtil;

    @PostMapping("/login")
    public ResponseEntity<> login(@RequestBody Signup login) throws Exception {
        Signup user = signupRepository.getByUsername(login.getUsername());
        String password = null;
        if (user == null) {
            throw new UserNotFoundException();
        }
        password = user.getPassword();
        if (!Objects.equals(login.getPassword(), password)) {
            return new ResponseEntity<>("Wrong credentials", HttpStatus.FORBIDDEN);
        }

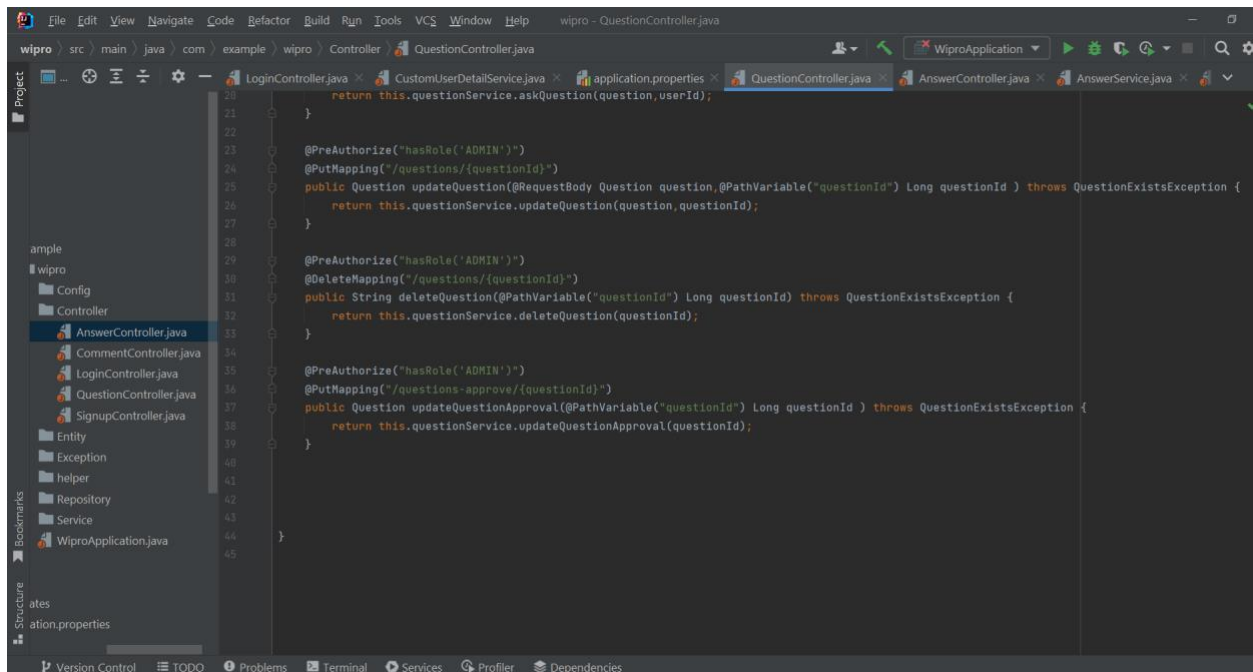
        try {
            this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(user.getUsername(),
                password));
        } catch (UsernameNotFoundException e) {
            throw new Exception("Bad Credentials");
        }

        UserDetails userDetails = this.customUserDetailsService.loadUserByUsername(user.getUsername());
        String token = this.jwtUtil.generateToken(userDetails);

        return ResponseEntity.ok(new JwtResponse(token));
    }
}

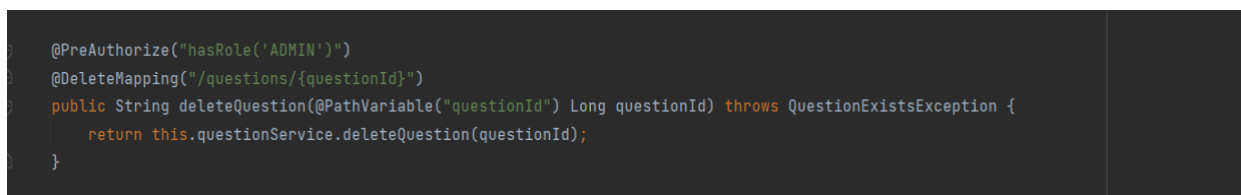
```

5. As an Admin I should be able to approve the question and Answer. Any Question or Answer will be visible on the platform only if it is approved.



```
21 }
22
23 @PreAuthorize("hasRole('ADMIN')")
24 @PutMapping("/questions/{questionId}")
25 public Question updateQuestion(@RequestBody Question question, @PathVariable("questionId") Long questionId) throws QuestionExistsException {
26     return this.questionService.updateQuestion(question, questionId);
27 }
28
29 @PreAuthorize("hasRole('ADMIN')")
30 @DeleteMapping("/questions/{questionId}")
31 public String deleteQuestion(@PathVariable("questionId") Long questionId) throws QuestionExistsException {
32     return this.questionService.deleteQuestion(questionId);
33 }
34
35 @PreAuthorize("hasRole('ADMIN')")
36 @PutMapping("/questions-approve/{questionId}")
37 public Question updateQuestionApproval(@PathVariable("questionId") Long questionId) throws QuestionExistsException {
38     return this.questionService.updateQuestionApproval(questionId);
39 }
40
41
42
43
44 }
45
```

6. As an Admin I should be able to delete inappropriate Questions or Answers.



```
@PreAuthorize("hasRole('ADMIN')")
@DeleteMapping("/questions/{questionId}")
public String deleteQuestion(@PathVariable("questionId") Long questionId) throws QuestionExistsException {
    return this.questionService.deleteQuestion(questionId);
}
```



THE END