



Mergen

27/07/2019

—

Proje Sahipleri:

Onur ALBAYRAK

Enes KARANFİL

Ayça GÜRLEYİK

İçindekiler

1- Proje Tanımı

1.1 Gereksinim Analizleri

1.2 Tablolarımız

2-Sisteme Genel Bakış

3-Sistemin Kullandığı Teknolojiler

4-HIPO

5-EER/ER

6-Fonksiyonlarımız

7-Admin Modülü

8-Kullanıcı Arayüzü ile Uygulama Tanıtımı

9- Veritabanı Yapısı

10-Sistem Değerlendirmesi

11-Ekler

1 - Proje Tanımı

Günümüzde teknolojinin gelişmesi ve yaygınlaşması ile haberleşme imkanları da son derece ulaşılabilir olmuştur. Hayatın her alanında birçok farklı şekilde haberleşme türü kullanılmaktadır. Eskiden haberleşme için kullanılan fiziksel materyaller günümüzde uygun şartlar sağlandığı zaman yerini elektronik sistemlere bırakabiliyor. Bu sayede fiziksel materyalin ulaşımı esnasında harcanan zaman ortadan kalkıyor, haberleşme süresi kısalıyor ve kolaylık sağlanmış oluyor. Örnek olarak üniversitelerde kullanılan öğrenci-üniversite arasındaki ilişkiyi kuran bir elektronik sistem verilebilir.

Ders kapsamında yapmamız gereken proje için bu tür bir yazılımı geliştirmek istedik. Projemizin amacı, dönem içinde ders veren akademisyenler ve asistanlar ile dersi alan öğrenciler arasında bir köprü olarak görev yapacak, üniversite içi verileri güvenli bir şekilde muhafaza edip yüksek oranda erişilebilir bir sistem inşa etmektir.

1.1 - Gereksinim Analizleri

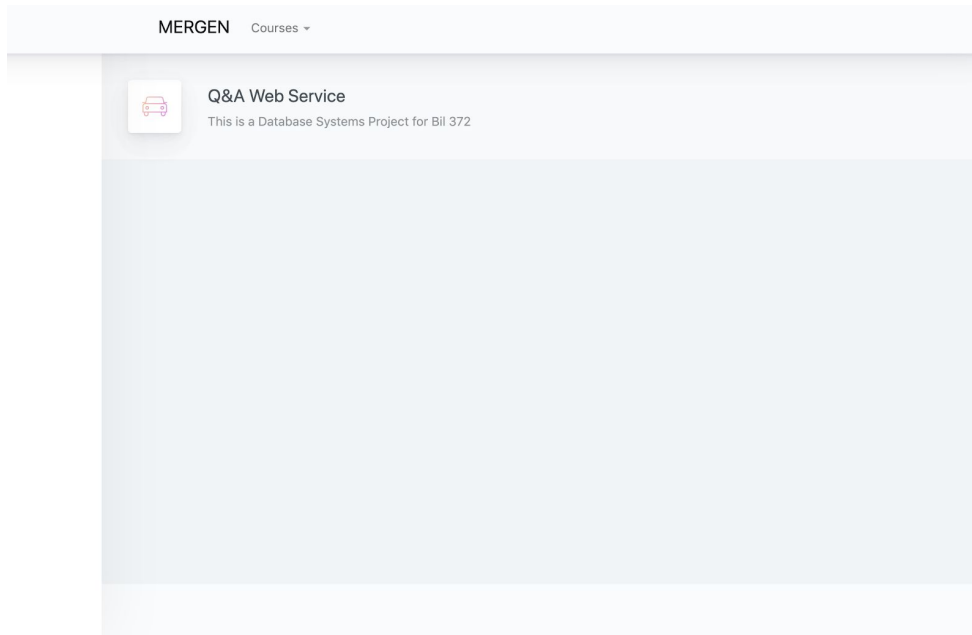
1. Kurulan sistemin amacı üniversite ortamındaki öğrenciler ile akademisyenler ve asistanların birlikte interaktif bir şekilde etkileşime geçebileceği ortam sunmaktır.
2. Bu sistemi oluştururken buna benzer modellerden fikir alarak kullanıcı deneyimini maksimize etmekteyiz.
3. Bu sistemde akademisyenler yapması gereken duyuruları bu sistem üzerinde paylaşabilmektedir ve öğrenciler ise bu duyurulara kesintisiz bir şekilde çevrimiçi olarak ulaşabilmektedir.
4. Akademisyen ve ders asistanları bu platformda Resources başlığı altında öğrenciler ile ders notlarını ve kaynak paylaşımı yapabilmektedir.
5. Bu oluşturulan sistemde aynı zamanda kullanıcının web sayfamızdan mutlak olan maksimum verimi alabilmesi için web sitemizin tasarımına gerekli önemi vermekteyiz.
6. Veritabanımızda oluşturduğumuz kısıtlar ile sistemin tutarlılığını bozacak veri girişlerini engellemiş bulunmaktayız. Örnek olarak aynı ders adı ile aynı dönem içinde iki ders olmamaktadır.

1.2 - Tablolarımız

- Uygulamada USER'lar iki alt sınıftan meydana gelmektedir. Bunlar STUDENT ve INSTRUCTOR'dır.
- COURSES tablosu ile USER'lar arasında ilişkiler bulunmaktadır. STUDENT ve COURSES arasında TAKES (N,M) ilişkisi bulunmaktadır. INSTRUCTOR VE COURSES arasında GIVES (N,M) ilişkisi bulunmaktadır.
- ENROLLMENT tablosu ile dersler ve bu derslere kayıtlı olan kişiler tutulacaktır. COURSES ve ENROLLMENT arasında CONSISTS (N,1) ilişkisi bulunmaktadır.. STUDENT ile ENROLLMENT arasında ASSIGNS (N,M) ilişkisi bulunmaktadır.
- Uygulama içinde USER'lar için Q&A bölümü yer alacaktır. USER'lar post ve comment atabileceklerdir. Q&A'nın de iki alt sınıfı bulunmaktadır. Bunlar POSTS ve COMMENTS'dir.USER ile Q&A arasında CAN USE (N,M) ilişkisi bulunmaktadır.
- Bunlarla birlikte ENROLLMENT ile Q&A arasında da bir ilişki bulunmaktadır.Bu HAS (N,N) ilişkisidir.

- Yukarıda sayılan user story'de toplamda 8 adet tablo ve 6 adet relationship bulunmaktadır.
- Her USER'ın kendine özel bir id'si bulunmaktadır. Ayrıca name, surname, email, password, isOnline attribute'larına da sahiptir.
- Her POST'un kendine özel bir post_id'si bulunmaktadır. Ayrıca post_message, post_author, post_date attribute'larına da sahiptir.
- Her COMMENT'in kendine özel bir comment_id'si bulunmaktadır. Ayrıca comment_message, comment_author, comment_date, post_id attribute'larına da sahiptir.
- Her COURSE'un kendine özel bir course_id'si bulunmaktadır. Ayrıca course_name'e de sahiptir.
- Her ENROLLMENT'ın kendine özgü bir enrollment_id'si bulunmaktadır. Ayrıca course_id, course_name, student_list ve qalD attribute'larına da sahiptir.
- Her Q&A kendine özgü bir qalD'ye sahiptir.

2- SİSTEME GENEL BAKIŞ



Figür 0 - Mergen Anasayfası

Oluşturmuş olduğumuz sistem web destekli bir sistem olmakla birlikte üniversite ortamında öğrenciler, akademisyenler ve asistanlar arasında interaktif bir şekilde etkileşimi sağlayacak olan bir platformdur. Figür 0'da gösterilen sayfa bu uygulamanın ana sayfasıdır. Arayüz tasarımı sistemde kullanılabilen özelliklere göre ilgili kısımlara bölünmektedir. Sayfanın üst kısmındaki menü barda öğrencilerin kayıtlı oldukları dersleri seçebilecekleri bir drop down kısmı bulunmaktadır. Burada kayıtlı oldukları dersleri seçebilmelerinin yanı sıra

başka bir ders ekleyebilecekleri Add Another Courses seçeneği bulunmaktadır. Ancak bu sistemde öğrenciler kendilerini sisteme eklememekte bu işlemi Admin görevinde olan Instructor yapmaktadır. Bu kısmın yanında ise atılan postların ve cevapların olduğu bölüme yönlendirildikleri Q&A seçeneği bulunmaktadır. Ayrıca öğrenciler için Resources kısmında 3 farklı alt başlık bulunmaktadır. Ders içinde paylaşılan kaynakları ve dosyaları görüntüleyebilecekleri Resources'tır. Akademisyenler için bu menü bar yine geçerli olmakla birlikte vermiş oldukları derslerin listesinin tutulduğu drop down kısmı bulunmaktadır ve yine bu kısımdan Add Courses seçeneği ile yeni ders oluşturabilmektedirler. Ayrıca post atabilecekleri ve atılan postlara cevap verebilecekleri kısma ise Q&A bölümünden yönlendirilmektedirler. Resources kısmında . Sayfanın sol kısmında hem öğrenciler hem de akademisyenler için de aynı şekilde bulunan atılan postların ve bu postlara verilen cevapların sıralandığı bir scroll down bölümü bulunmaktadır. Bu kısmın üstünde New Post butonu yer almaktadır. Bu buton ile yeni post oluşturabilecekleri formun bulunduğu sayfaya yönlendirilmektedirler. Bu butonun yanında postları aratabilecekleri Search kısmı bulunmaktadır. Sayfanın sağ üst kısmında kullanıcıların profil bilgilerini görüntüleyebileceği bir bölüm ve yanında ise Ayarlar bölümü bulunmaktadır.

SİSTEMİN KULLANDIĞI TEKNOLOJİLER

Kullanılan Teknolojiler:

- Java
- Hibernate
- Spring Boot
- Html/Css, Javascript, Ajax, JQuery
- REST API
- Bootstrap

Spring Framework Java ve .NET için geliştirilmiş, açık kaynak olan bir uygulama geliştirme framework'üdür. Spring Framework'un temel özellikleri herhangi bir Java uygulaması tarafından kullanılabilir. İmplementasyonları gerçekleştirirken getirdiği kolaylıklar dolayısıyla ve grup üyelerinin ortak olarak hakim olduğu programlama dili Java olduğu için bu teknolojiyi seçmiş bulunmaktayız.

Html/Css uygulamamızın arayüz tasarımı aşamasında kullandık. Arayüz tasarımı için Bootstrap isimli kütüphaneden faydalandık. Twitter tarafında geliştirilen Bootstrap açık kaynak kodlu, web sayfaları veya uygulamaları geliştirmek için kullanılabilecek araçlar bütünüdür. Bootstrap, web sayfaları veya uygulamalarında kullanılabilecek, Html/Css tabanlı tasarım şablonlarını içerir. Herhangi bir sayfaya girildiğinde sayfanın yenilenmeden verilerin ekranda dinamik olarak gösterilmesi için Ajax teknolojisini kullandık.

Sayfalardaki formlara girilen bilgileri oradan almak veya sağlanan endpointlere REST API teknolojisi ile istek atmak için Javascript'i kullandık. REST bir veri transferi yöntemidir.

HTTP protokolü üzerinde çalışan REST, basit sorgular ile verilere kolay erişim sağlamaktadır. REST, bu verileri XML, JSON veya farklı bir veri formatında taşır.

Biz projenin büyük bir kısmında JSON formatını tercih ettik. Bunun nedeni JSON'un, XML'e göre daha okunaklı olması ve verileri daha küçük boyutlarda tutabilmesidir. Ancak kullanıcı kayıt ve giriş kısmında FormData kullandık bunun nedeni ise projenin backend kısmında implementasyonu kolaylaştırmasıdır. FormData içinde tıpkı Java'daki HashMap gibi key/value ikilileri içermektedir.

Hibernate, yalnızca Java sınıflarından veritabanı tablolarına eşleştirme ile ilgilenmez, aynı zamanda veri sorgulama ve alma olanakları sağlar. Böylece ek olarak bir SQL sorgusuna gerek kalmadan Java kodunun içinde bu işlemler halledilmiş olur. Bu nedenle sistemimizde SQL cümlecikleri bulunmamaktadır.

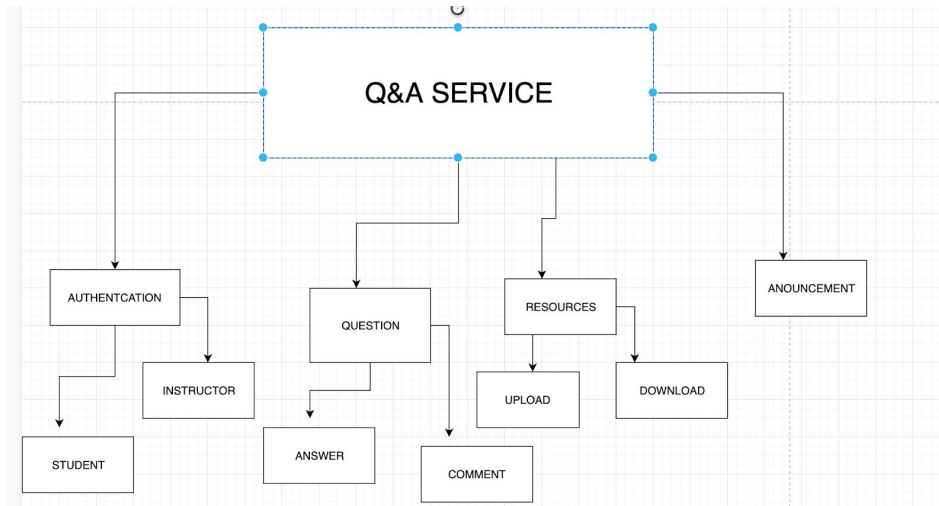
```
package com.example.springboot;

import ...

@Repository
public interface InstructorRepository extends JpaRepository<Instructor, Long> {
    Instructor findByInstructorMail(String mail);
    Instructor findByInstructorMailAndInstructorPassword(String mail, String psswd);
}
```

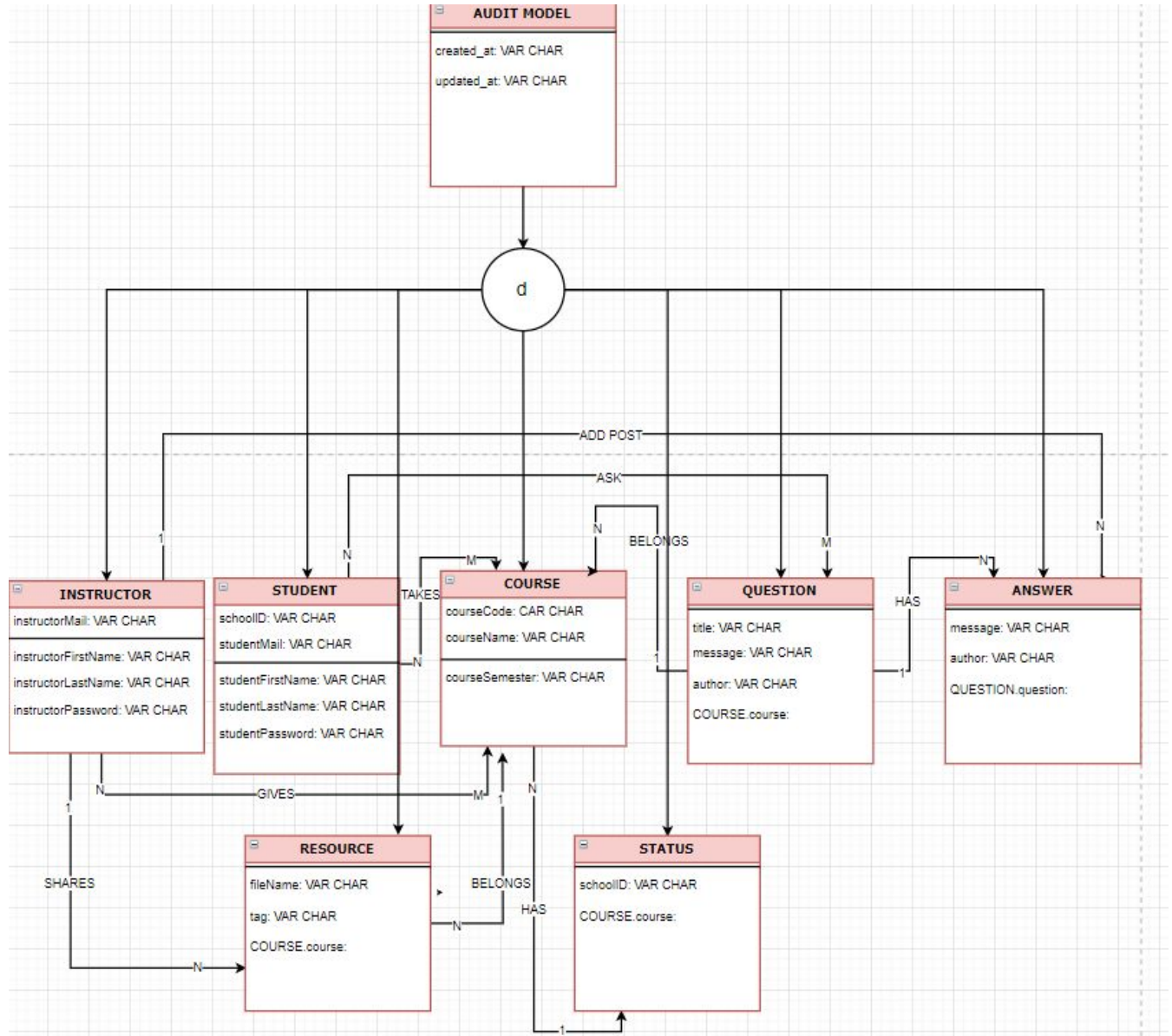
Figür 1 - Instructor Tablosunda Mail ve Şifre Sorgu Örneği

3- Hierarchical Input Process Output (HIPO)



HIPO Modelimiz

Enhanced Entity Relationship (EER)



Son halini almış EER Modelimiz

6- Fonksiyonlar

CourseController.java (RestController)


```

cow [-IdeaProjects/372 Q-A-web-service-Backend] - .../src/main/java/com/etu/cow/controller/CourseController.java [cow.main] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
372 Q-A-web-service-Backend src main java com etu cow controller CourseController
CourseController.java
14 @RestController
15 public class CourseController {
16     @Autowired
17     private CourseRepository courseRepository;
18
19     @GetMapping("/courses")
20     public ResponseEntity<List> getCourses() {
21         List<Course> courseList = courseRepository.findAll();
22         if (courseList == null) {
23             return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
24         }
25         return new ResponseEntity<>(courseList, HttpStatus.OK);
26     }
27
28     @GetMapping("/courses/{courseCode}")
29     public ResponseEntity<String> getCourse(@RequestParam String courseCode) {
30         Course tmp = courseRepository.findById(courseCode);
31         if (tmp == null) {
32             return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
33         }
34         String result = tmp.getCourseCode() + " " + tmp.getCourseName();
35         return new ResponseEntity<>(result, HttpStatus.OK);
36     }
37
38     @PostMapping("/courses")
39     public Course createCourse(@Valid @RequestBody Course course) {
40         return courseRepository.save(course);
41     }
42
43     @PutMapping("/courses/{courseId}")
44     public Course updateCourse(@PathVariable Long courseId,
45                               @Valid @RequestBody Course courseRequest) {
46         return courseRepository.findById(courseId)
47             .map(course -> {
48                 course.setCourseCode(courseRequest.getCourseCode());
49                 course.setCourseName(courseRequest.getCourseName());
50                 course.setCourseSemester(courseRequest.getCourseSemester());
51                 return courseRepository.save(course);
52             })
53             .orElseThrow(() -> new ResourceNotFoundException("Course not found with id " + courseId));
54     }
55
56     @DeleteMapping("/courses/{courseId}")
57     public ResponseEntity<?> deleteQuestion(@PathVariable Long courseId) {
58         return courseRepository.findById(courseId)
59             .map(course -> {
60                 courseRepository.delete(course);
61                 return ResponseEntity.ok().build();
62             })
63             .orElseThrow(() -> new ResourceNotFoundException("Question not found with id " + courseId));
64     }
65 }
66
67 CourseController : getCourse()
G: TODO Spring Terminal Java Enterprise Version Control
35:73 LF: UTF-8 4 spaces Git: master Friday July 26, 13:31

```

QuestionController.java (RestController)

```

cow [-IdeaProjects/372 Q-A-web-service-Backend] - .../src/main/java/com/etu/cow/controller/QuestionController.java [cow.main] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
372 Q-A-web-service-Backend src main java com etu cow controller QuestionController
QuestionController.java
21 @GetMapping("/courses/{courseId}/questions")
22 public ResponseEntity<List> getQuestions(@PathVariable Long courseId) {
23     List<Question> questionList = questionRepository.findByCourseId(courseId);
24     if (questionList == null) {
25         return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
26     }
27     return new ResponseEntity<>(questionList, HttpStatus.OK);
28 }
29
30 @GetMapping("/courses/{courseId}/questions/{questionId}")
31 public ResponseEntity<Question> getQuestion(@PathVariable Long courseId,
32                                             @PathVariable Long questionId) {
33     List<Question> questionList = questionRepository.findByCourseId(courseId);
34     Question question = null;
35     for (Question questionTemp : questionList) {
36         if (questionTemp.getId() == questionId) {
37             question = questionTemp;
38         }
39     }
40     if (question == null) {
41         return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
42     }
43     return new ResponseEntity<>(question, HttpStatus.OK);
44 }
45
46 @PostMapping("/courses/{courseId}/questions")
47 public Question createQuestion(@PathVariable Long courseId,
48                               @Valid @RequestBody Question question) {
49     return questionRepository.findById(courseId)
50         .map(course -> {
51             question.setCourse(course);
52             return questionRepository.save(question);
53         })
54         .orElseThrow(() -> new ResourceNotFoundException("Course not found with id " + courseId));
55 }
56
57 @PutMapping("/questions/{questionId}")
58 public Question updateQuestion(@PathVariable Long questionId,
59                               @Valid @RequestBody Question questionRequest) {
60     return questionRepository.findById(questionId)
61         .map(question -> {
62             question.setTitle(questionRequest.getTitle());
63             question.setMessage(questionRequest.getMessage());
64             question.setAuthor(questionRequest.getAuthor());
65             return questionRepository.save(question);
66         })
67         .orElseThrow(() -> new ResourceNotFoundException("Question not found with id " + questionId));
68 }
69
70 @DeleteMapping("/questions/{questionId}")
71 public ResponseEntity<?> deleteQuestion(@PathVariable Long questionId) {
72     return questionRepository.findById(questionId)
73         .map(question -> {
74             questionRepository.delete(question);
75             return ResponseEntity.ok().build();
76         })
77         .orElseThrow(() -> new ResourceNotFoundException("Question not found with id " + questionId));
78 }
79
80 QuestionController
G: TODO Spring Terminal Java Enterprise Version Control
18:1 LF: UTF-8 4 spaces Git: master Friday July 26, 13:31

```

AnswerController.java (RestController)

```

cow\IdeaProjects\372 Q-A-web-service-Backend\src\main\java\com\etu\cow\controller\AnswerController.java [cow.main] - IntelliJ IDEA

@WebController
public class AnswerController {

    @Autowired
    private AnswerRepository answerRepository;

    @Autowired
    private QuestionRepository questionRepository;

    @GetMapping("/questions/{questionId}/answers")
    public List<Answer> getAnswersByQuestionId(@PathVariable Long questionId) {
        return answerRepository.findByQuestionId(questionId);
    }

    @PostMapping("/questions/{questionId}/answers")
    public Answer addAnswer(@PathVariable Long questionId,
                           @Valid @RequestBody Answer answer) {
        return questionRepository.findById(questionId)
            .map(question -> {
                answer.setQuestion(question);
                return answerRepository.save(answer);
            })
            .orElseThrow(() -> new ResourceNotFoundException("Question not found with id " + questionId));
    }

    @PutMapping("/questions/{questionId}/answers/{answerId}")
    public Answer updateAnswer(@PathVariable Long questionId,
                              @PathVariable Long answerId,
                              @Valid @RequestBody Answer answerRequest) {
        if(!questionRepository.existsById(questionId)) {
            throw new ResourceNotFoundException("Question not found with id " + questionId);
        }
        return answerRepository.findById(answerId)
            .map(answer -> {
                answer.setMessage(answerRequest.getMessage());
                answer.setAuthor(answerRequest.getAuthor());
                return answerRepository.save(answer);
            })
            .orElseThrow(() -> new ResourceNotFoundException("Answer not found with id " + answerId));
    }

    @DeleteMapping("/questions/{questionId}/answers/{answerId}")
    public ResponseEntity<> deleteAnswer(@PathVariable Long questionId,
                                         @PathVariable Long answerId) {
        if(!questionRepository.existsById(questionId)) {
            throw new ResourceNotFoundException("Question not found with id " + questionId);
        }
        return answerRepository.findById(answerId)
            .map(answer -> {
                answerRepository.delete(answer);
                return ResponseEntity.ok().build();
            })
            .orElseThrow(() -> new ResourceNotFoundException("Answer not found with id " + answerId));
    }
}

```

S3Service.java (Service)

```

cow\IdeaProjects\372 Q-A-web-service-Backend\src\main\java\com\etu\cow\service\impl\S3ServicesImpl.java [cow.main] - IntelliJ IDEA

@Service
public class S3ServicesImpl implements S3Services {

    private Logger logger = LoggerFactory.getLogger(S3ServicesImpl.class);

    @Autowired
    private AmazonS3 s3Client;

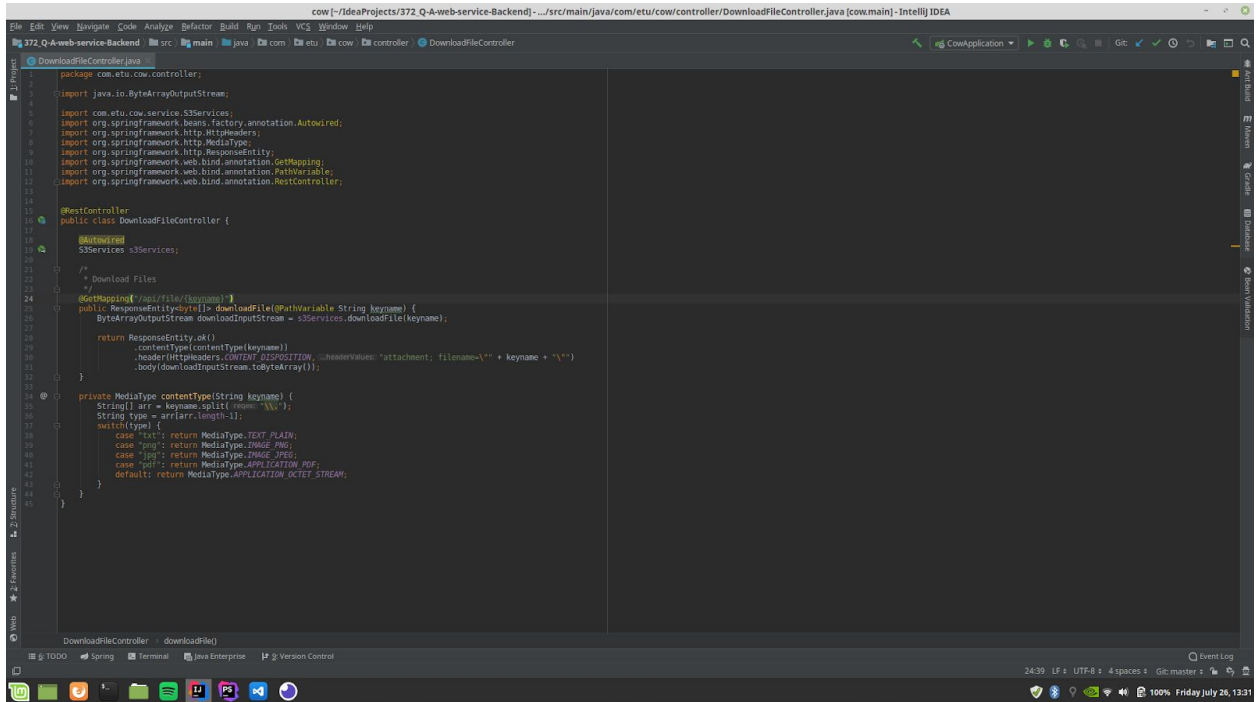
    @Value("${s3-bucket}")
    private String bucketName;

    @Override
    public ByteArrayOutputStream downloadFile(String keyName) {
        try {
            S3Object s3Object = s3Client.getObject(new GetObjectRequest(bucketName, keyName));
            InputStream is = s3Object.getObjectContent();
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            int len;
            byte[] buffer = new byte[1024];
            while ((len = is.read(buffer, 0, buffer.length)) != -1) {
                baos.write(buffer, 0, len);
            }
            return baos;
        } catch (IOException ioe) {
            logger.error("IOException: " + ioe.getMessage());
        } catch (AmazonServiceException ase) {
            logger.info("Caught an AmazonServiceException from GET requests, rejected reasons:");
            logger.info("Error Message: " + ase.getMessage());
            logger.info("HTTP Status Code: " + ase.getStatusCode());
            logger.info("AWS Error Code: " + ase.getErrorCode());
            logger.info("Error Type: " + ase.getErrorType());
            logger.info("Request ID: " + ase.getRequestId());
            throw ase;
        } catch (AmazonClientException ace) {
            logger.info("Caught an AmazonClientException: ");
            logger.info("Error Message: " + ace.getMessage());
            throw ace;
        }
        return null;
    }

    @Override
    public void uploadFile(String keyName, MultipartFile file) {
        try {
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentLength(file.getSize());
            s3Client.putObject(bucketName, keyName, file.getInputStream(), metadata);
        } catch (IOException ioe) {
            logger.error("IOException: " + ioe.getMessage());
        } catch (AmazonServiceException ase) {
            logger.info("Caught an AmazonServiceException from PUT requests, rejected reasons:");
            logger.info("Error Message: " + ase.getMessage());
            logger.info("HTTP Status Code: " + ase.getStatusCode());
            logger.info("AWS Error Code: " + ase.getErrorCode());
            throw ase;
        }
    }
}

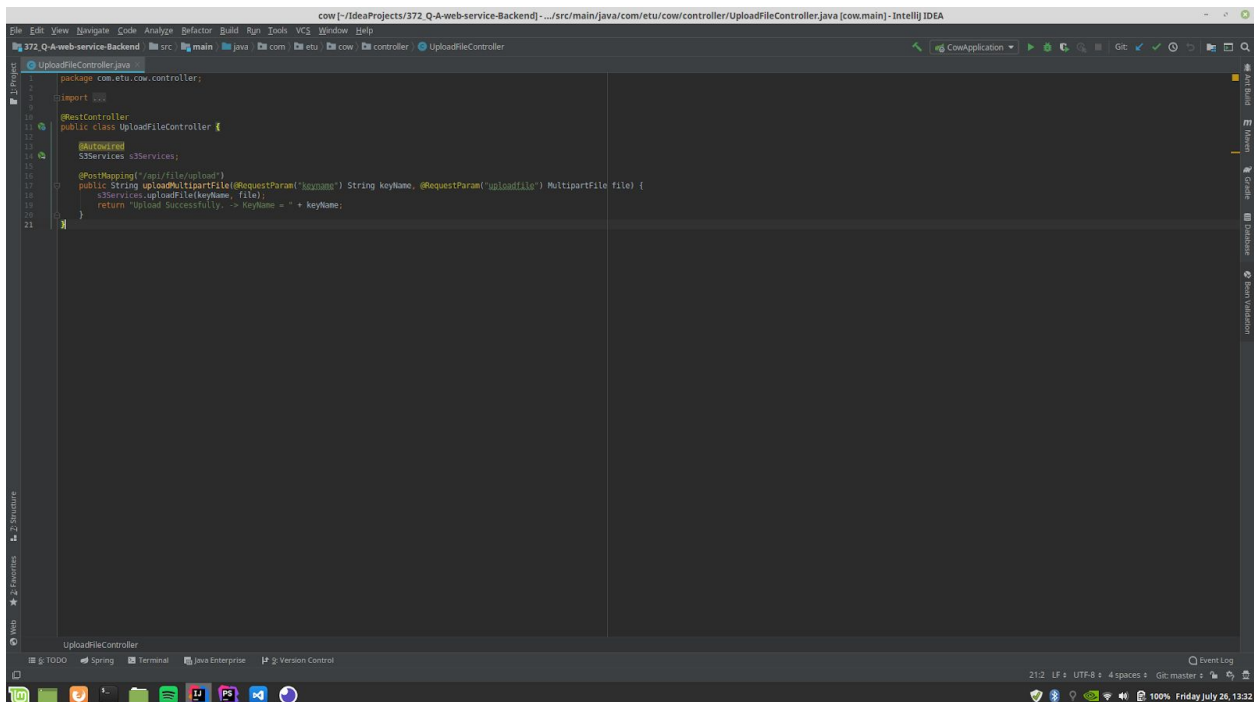
```

DownloadFileController.java (RestController)



```
1 package com.etu.cow.controller;
2 import java.io.ByteArrayOutputStream;
3
4 import com.etu.cow.service.S3Services;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.HttpHeaders;
7 import org.springframework.http.MediaType;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.RestController;
12
13 @RestController
14 public class DownloadFileController {
15     @Autowired
16     S3Services s3Services;
17
18     /*
19      * Download Files
20      */
21     @GetMapping("/api/file/{keyname}")
22     public ResponseEntity<byte[]> downloadFile(@PathVariable String keyname) {
23         ByteArrayOutputStream downloadInputStream = s3Services.downloadFile(keyname);
24         return ResponseEntity.ok()
25             .contentType(MediaType(keyname))
26             .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" + keyname + ".\"")
27             .body(downloadInputStream.toByteArray());
28     }
29
30     private MediaType contentType(String keyname) {
31         String[] arr = keyname.split(":");
32         String type = arr[arr.length-1];
33         switch(type) {
34             case "txt": return MediaType.TEXT_PLAIN;
35             case "png": return MediaType.IMAGE_PNG;
36             case "jpg": return MediaType.IMAGE_JPEG;
37             case "pdf": return MediaType.APPLICATION_PDF;
38             default: return MediaType.APPLICATION_OCTET_STREAM;
39         }
40     }
41 }
```

UploadFileController.java (RestController)



```
1 package com.etu.cow.controller;
2 import java.io;
3
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PostMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8 import org.springframework.web.multipart.MultipartFile;
9
10 @RestController
11 public class UploadFileController {
12     @Autowired
13     S3Services s3Services;
14
15     @PostMapping("/api/file/upload")
16     public String uploadMultipartFile(@RequestParam("keyname") String keyName, @RequestParam("uploadfile") MultipartFile file) {
17         s3Services.uploadFile(keyName, file);
18         return "Upload Successfully. -> KeyName = " + keyName;
19     }
20 }
```

The image shows a screenshot of an IDE (IntelliJ IDEA) with the following details:

- Top Bar:** Displays the file path: `File Edit View Navigate Code Analysis Refactor Build Run Tools VCS Window Help`. Below it, the project name is `372_Q-A-web-service-Backend` and the current file is `StudentController.java`.
- Left Panel:** Shows the project structure with folders like `src`, `main`, `java`, `com`, `edu`, `cow`, and `controller`. The `StudentController.java` file is selected under the `controller` folder.
- Code Editor:** Contains the following Java code:

```
import java.util.*;
import org.springframework.http.*;
import org.springframework.web.servlet.mvc.annotation.annotation.*;

@Controller
public class StudentController {

    @Autowired
    StudentRepository studentRepository;

    @PostMapping("/students")
    public Student createStudent(@Valid @RequestBody Student student) {
        Student tempStudent = new Student();
        tempStudent.setSchoolId(student.getSchoolId());
        tempStudent.setStudentFirstName(student.getStudentFirstName());
        tempStudent.setStudentLastName(student.getStudentLastName());
        tempStudent.setStudentMail(student.getStudentMail());
        tempStudent.setStudentPassword(student.getStudentPassword());
        return studentRepository.save(tempStudent);
    }

    @GetMapping("/students")
    public ResponseEntity<List<Student>> getStudents() {
        List<Student> studentList = studentRepository.findAll();
        if (studentList == null) {
            return new ResponseEntity<List<Student>>(<null>, HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<List<Student>>(studentList, HttpStatus.OK);
    }

    @GetMapping("/students/{schoolId}")
    public ResponseEntity<Student> getStudent(@PathVariable (value = "schoolId") String schoolId) {
        Student result = studentRepository.findById(schoolId);
        if(result == null) {
            return new ResponseEntity<Student>(<null>, HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<Student>(result, HttpStatus.OK);
    }

    @PostMapping("/students/login")
    public ResponseEntity<Student> login(@RequestParam (value = "studentMail") String studentMail,
                                      @RequestParam (value = "studentPassword") String studentPassword) {
        Student result = studentRepository.findById(studentMailAndStudentPassword(studentMail, studentPassword));
        if(result == null) {
            return new ResponseEntity<Student>(<null>, HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<Student>(result, HttpStatus.OK);
    }

    @DeleteMapping("/students/{id}")
    public ResponseEntity<Student> deleteStudent(@Valid @PathVariable Long id) {
        return studentRepository.findById(id).map(post -> {
            studentRepository.delete(post);
            return ResponseEntity.ok().build();
        }).orElseThrow(() -> new ResourceNotFoundException("Student id " + id + " not found"));
    }
}
```
- Bottom Panel:** Shows the `StudentController : login` method being executed. The status bar at the very bottom indicates `51/27` lines of code, `UTF-8` encoding, `4` spaces, and a clock showing `Friday, July 26, 2019`.

```
File Edit View Navigate Code Analysis Refactor Build Run Tools VCS Window Help
372_Q-A-web-service-Backend src main java com etu cow/controller java [CowApplication]
InstructorRepository
Instructor
InstructorController

1 import com.etu.cow.repository.InstructorRepository;
2 import org.springframework.beans.factory.annotation.Autowired;
3 import org.springframework.http.HttpStatus;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.*;
6
7 import java.util.List;
8 import java.util.Optional;
9
10 @RestController
11 public class InstructorController {
12     @Autowired
13     private InstructorRepository instructorRepository;
14
15     @PostMapping("/instructors")
16     public ResponseEntity<Instructor> createInstructor(@Valid @RequestBody Instructor instructor) {
17         Instructor tempInstructor = new Instructor();
18         tempInstructor.setInstructorFirstName(instructor.getInstructorFirstName());
19         tempInstructor.setInstructorLastName(instructor.getInstructorLastName());
20         tempInstructor.setInstructorEmail(instructor.getInstructorEmail());
21         tempInstructor.setInstructorPassword(instructor.getInstructorPassword());
22         return instructorRepository.save(tempInstructor);
23     }
24
25     @GetMapping("/instructors")
26     public ResponseEntity<List<Instructor>> getInstructors() {
27         List<Instructor> instructorList = instructorRepository.findAll();
28         if (instructorList == null) {
29             return new ResponseEntity<>((null), HttpStatus.NOT_FOUND);
30         }
31         return new ResponseEntity<>(instructorList, HttpStatus.OK);
32     }
33
34     @GetMapping("/instructors/{instructorEmail}")
35     public ResponseEntity<Instructor> searchInstructor(@PathVariable(value = "instructorEmail") String instructorEmail) {
36         Instructor result = instructorRepository.findByInstructorEmail(instructorEmail);
37         if (result == null) {
38             return new ResponseEntity<>((null), HttpStatus.NOT_FOUND);
39         }
40         return new ResponseEntity<>(result, HttpStatus.OK);
41     }
42
43     @PostMapping("/instructors/login")
44     public ResponseEntity<Instructor> login(@RequestParam(value = "instructorEmail") String instructorEmail,
45                                           @RequestParam(value = "instructorPassword") String instructorPassword) {
46         Instructor result = instructorRepository.findByInstructorEmailAndInstructorPassword(instructorEmail, instructorPassword);
47         if (result == null) {
48             return new ResponseEntity<>((null), HttpStatus.NOT_FOUND);
49         }
50         return new ResponseEntity<>(result, HttpStatus.OK);
51     }
52 }

InstructorRepository createInstructor()
Spring Terminal Java Enterprise Version Control
19:33 LF UTF-8 4 spaces Git-master Friday July 26, 2019
```

```

1 package com.etu.cow.controller;
2
3 import org.springframework.web.bind.annotation.*;
4
5 @RestController
6 public class StatusController {
7
8     @Autowired
9     private StatusRepository statusRepository;
10
11     @Autowired
12     private CourseRepository courseRepository;
13
14     @GetMapping("/courses/{courseId}/status")
15     public ResponseEntity<List> getStatus(@PathVariable Long courseId) {
16         List<Status> statusList = statusRepository.findByCourseId(courseId);
17         if (statusList == null) {
18             return new ResponseEntity<>((null), HttpStatus.NOT_FOUND);
19         }
20         return new ResponseEntity<>(statusList, HttpStatus.OK);
21     }
22
23     @PostMapping("/courses/{courseId}/status")
24     public Status addStatus(@PathVariable Long courseId,
25                             @RequestBody Status status) {
26         return courseRepository.findById(courseId).map(course -> {
27             status.setCourse(course);
28             return statusRepository.save(status);
29         }).orElseThrow(() -> new ResourceNotFoundException("Course not found with id " + courseId));
30     }
31
32     @DeleteMapping("/courses/{courseId}/status/{id}")
33     public ResponseEntity<> deleteOwner(@PathVariable Long courseId,
34                                         @PathVariable Long id) {
35         if (!courseRepository.existsById(courseId)) {
36             throw new ResourceNotFoundException("Course not found with id " + courseId);
37         }
38         return statusRepository.findById(id)
39             .map(status -> {
40                 statusRepository.delete(status);
41                 return ResponseEntity.ok().build();
42             }).orElseThrow(() -> new ResourceNotFoundException("Status not found with id " + id));
43     }
44 }

```

InstructorRepository (JpaRepository)

The screenshot shows the IntelliJ IDEA IDE with the following details:

- Title Bar:** IntelliJ IDEA - [Project Path] - InstructorRepository.java [cow.main] - IntelliJ IDEA
- Menu Bar:** Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help
- Toolbar:** Includes icons for search, run, debug, and other IDE functions.
- Project Explorer (Left):** Shows the project structure with folders like src, main, java, com, etsi, cow, repository, and the file InstructorRepository.java.
- Code Editor:**

```

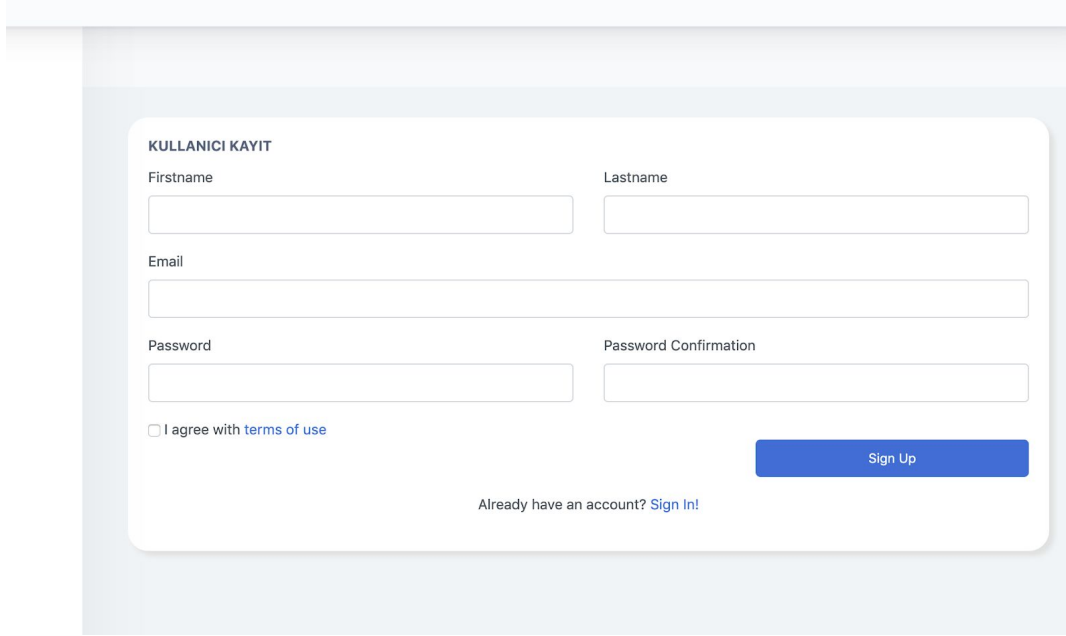
package com.etsi.cow.repository;

import java.util.List;

@Repository
public interface InstructorRepository extends JpaRepository<Instructor, Long> {
    Instructor findbyInstructorMail(String mail);
    Instructor findbyInstructorMailAndInstructorPassword(String mail, String passwd);
}

```
- Status Bar (Bottom):** Shows "12:1", "UTF-8", "4 spaces", "Git-master", and "Friday, July 26, 2024".

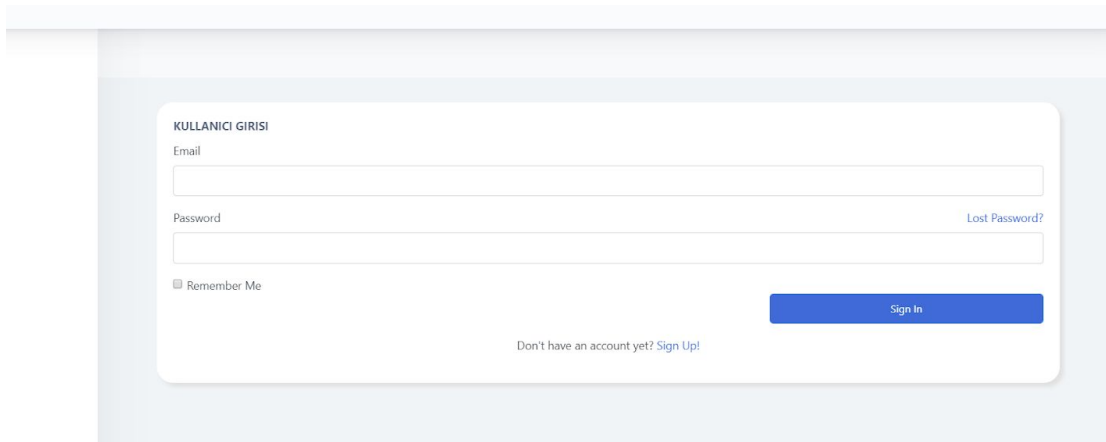
7- Admin Modülü



The screenshot shows a sign-up form titled "KULLANICI KAYIT". It includes input fields for "Firstname", "Lastname", "Email", "Password", and "Password Confirmation". Below the password fields, there is a checkbox labeled "I agree with [terms of use](#)". A blue "Sign Up" button is positioned to the right of the checkbox. At the bottom, there is a link: "Already have an account? [Sign In!](#)".

Figür 1.1 - Instructor Sayfası Sign-Up İşlemi

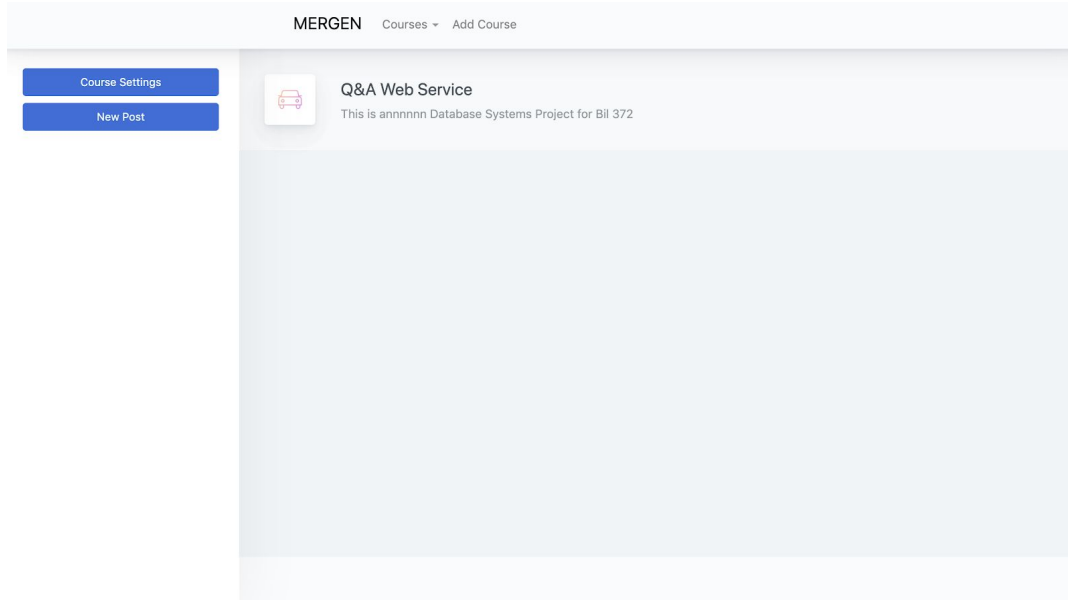
Figür 1.1 'deki şekil Instructor olarak kayıt olan kişinin Sign-Up sayfasını göstermektedir. Sistemdeki adminimiz Instructor'lardır. Admin modülü sistemdeki her fonksiyonalityi tamamen kullanma yetkisi olan kullanıcılardır. Bu tanım bizim sistemimizdeki Instructor'a denk gelmektedir. Bu kısmın devamında Instructor'ın yapabildiği tüm özellikler görseller ile tanıtılacaktır.



The screenshot shows a log-in form titled "KULLANICI GIRISI". It includes input fields for "Email" and "Password". To the right of the password field, there is a link: "Lost Password?". Below the password field, there is a checkbox labeled "Remember Me". A blue "Sign In" button is positioned to the right of the checkbox. At the bottom, there is a link: "Don't have an account yet? [Sign Up!](#)".

Figür 1.2 - Instructor Sayfası Log-In İşlemi

Figür 1.2'deki şekil Instructor olarak kayıt olan kişinin Log in sayfasını göstermektedir.Email ve password bilgileri ile sisteme giriş yapabilmektedirler.



Figür 1.2 - Instructor Course Sayfası

Figür 1.2' de Instructor'ın ders anasayfasını görmekteyiz. New Post ve Course Setting butonlarından sırasıyla sisteme herkesin görebileceği duyurular paylaşabilmektedir. Course Setting kısmı altında yapılan özellikler aşağıda sıralanmıştır.

Figür 1.4 - Instructor Course Edit Sayfası

Bu sayfa adminimiz eklediği ders ile ilgili bilgileri değiştirebilmektedir. Resources kısmına Amazon S3 servisinden destek aldığımız storage kısmına dosya eklemeyi Figür 1.4'te görmektesiniz. Eklenebilen maksimum dosya boyutunu 100MB olarak belirledik. Instructor herhangi bir dosyanın 100 MB ve aşağısında olduğu sürece istediği ders notunu Resources kısmına ekleyebilmektedir.

Figür 1.5 'te görüldüğü Course Edit sayfasının devamını görmekteyiz. Bu kısımda derse kayıtlı öğrencilerin listesini tutmaktayız ve Instructor bu kısımda listeyi güncellediği zaman derse kayıtlı öğrencileri listesini görebilmektedir. Aynı zamanda derse manual olarak kendisi de öğrenci ekleyebilmektedir.

The screenshot displays the 'MERGEN' interface for editing a course. At the top, there's a navigation bar with 'Courses' and 'Add Course' links. The main content area is divided into two sections. The first section, titled 'STUDENT LIST', contains a table with three columns: 'School Id', 'First Name', and 'Last Name'. Below the table is a blue button labeled 'Refresh List'. The second section, titled 'ADD STUDENT TO COURSE', contains three input fields for 'Student School Id', 'Student Name', and 'Student Lastname', followed by a blue button labeled 'Add Student to Course'.

Figür 1.5 - Course Edit Sayfası Devamı

The screenshot shows the 'Q&A Web Service' interface for creating a new post. It features a header with a car icon and the text 'Q&A Web Service' and 'This is aninnnn Database Systems Project for Bil 372'. The main content area is titled 'CONTROLS TYPES' and contains two input fields: 'Post Title' and 'Post Message'. Below these fields is a blue button labeled 'Add Post'.

Figür 1.6 - Instructor New Post Sayfası

Figür 1.6'da gösterilen sayfa ile instructor olarak sistemde bulunan kişiler post atacıkları zaman bu sayfaya yönlendirilmektedirler. Post title ve post message alanları doldurulduktan sonra add post ile postlarını paylaşabilmektedirler.

Q&A Web Service
This is annnnnnn Database Systems Project for Bil 372

ADD COURSE

Course Code

Course Name

Course Year
Select

Course Semester
Select

Submit

Figür 1.7 - Instructor Add Course Sayfası

Figür 1.7’de gösterilen sayfa ile instructorlar yeni bir ders ekleyebilmektedirler.Ders kodu, dersin adı, dersin açılacağı yıl ve dönem bilgileri girildikten sonra submit’lendiğinde yeni bir ders sisteme eklenmiş olacaktır.

8- Kullanıcı Arayüzü

MERGEN Courses Resources

New Post

SINAV TARIHI HK.
Sınav tarihi ne zaman belli olacak ?

PROJE ERTELEME TALEBİ HK.
Hocam projelerimizin çok yoğun olması nedeni ile proje süresini uzatabilir misiniz?

FINAL HK.
Hangi konulardan sorumluyuz?

PROJE TESLİMİ

Q&A Web Service
This is a Database Systems Project for Bil 372

Figür 1.8 - Ders Sayfası

Figür 1.8’de sisteme eklediğimiz Programlama-1 dersinin anasayfası görülmektedir. Bu arayüz kullanıcı yani Student tarafından sağlanmaktadır. Bu başlık altında kullanıcının yapabildikleri tanıtılacaktır.

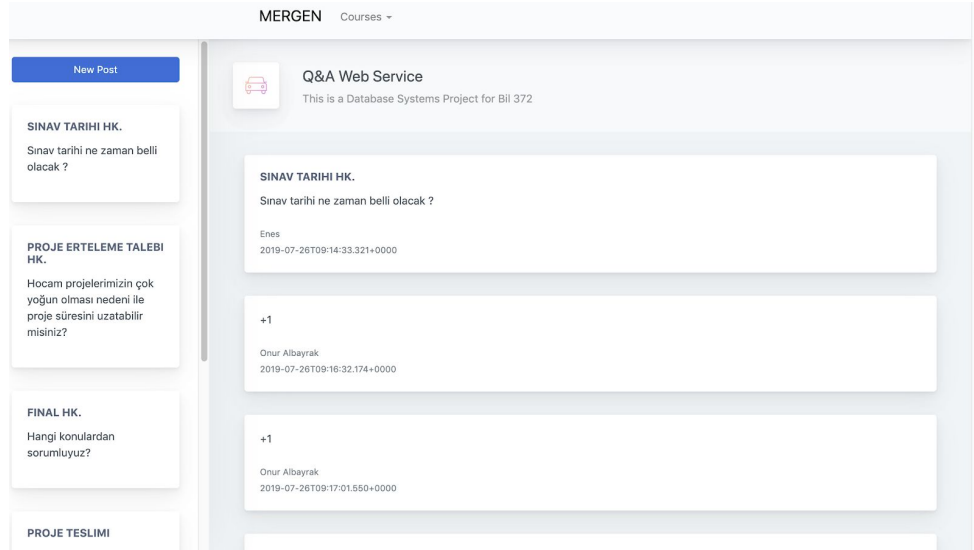
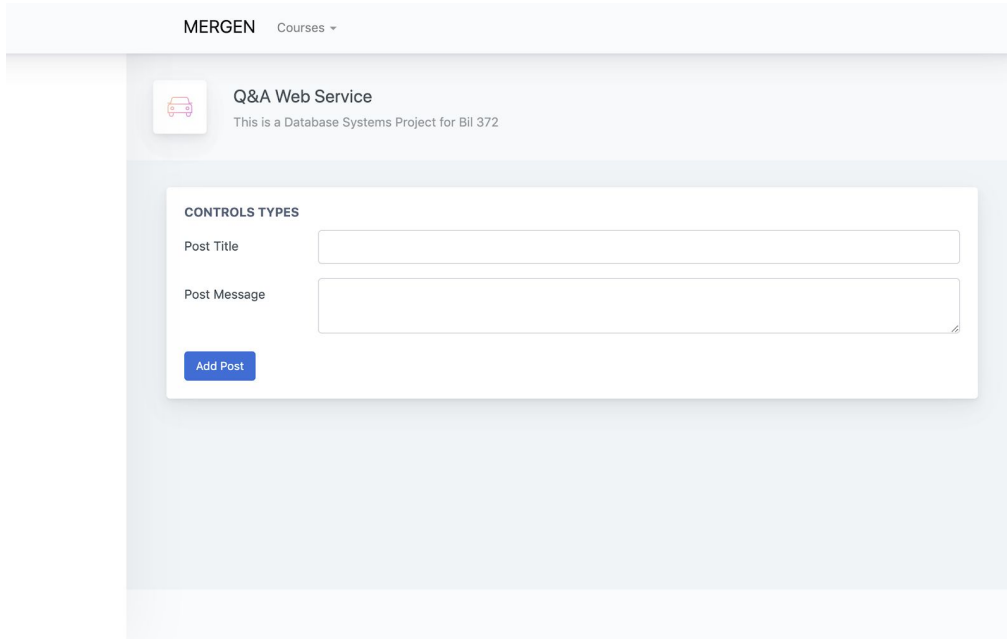
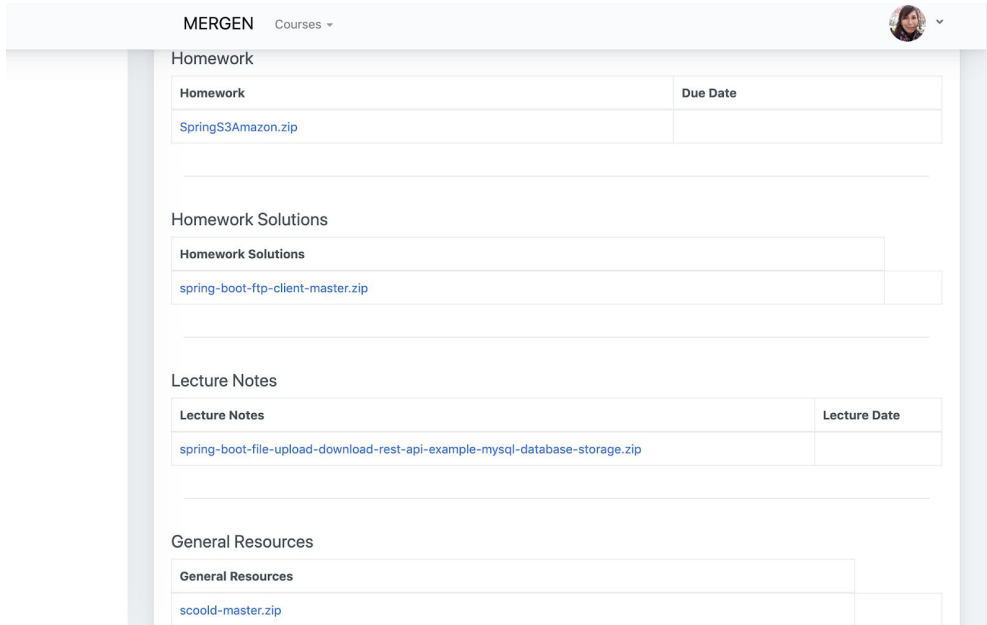


Figure 1.9 - Post Sayfası



Figür 1.10 - Student Post Atma Sayfası

Figür 1.10’da gözüken New Post sayfası Admin kullanıcısı ile aynıdır. İkisi de aynı şekilde sistemde kullanıcılara soru sorabilmekte ve cevaplayabilmektedir.



The screenshot shows the 'MERGEN' website with a 'Courses' dropdown menu and a user profile icon. The main content area is divided into four sections: Homework, Homework Solutions, Lecture Notes, and General Resources. Each section contains a table with resource names and due dates.

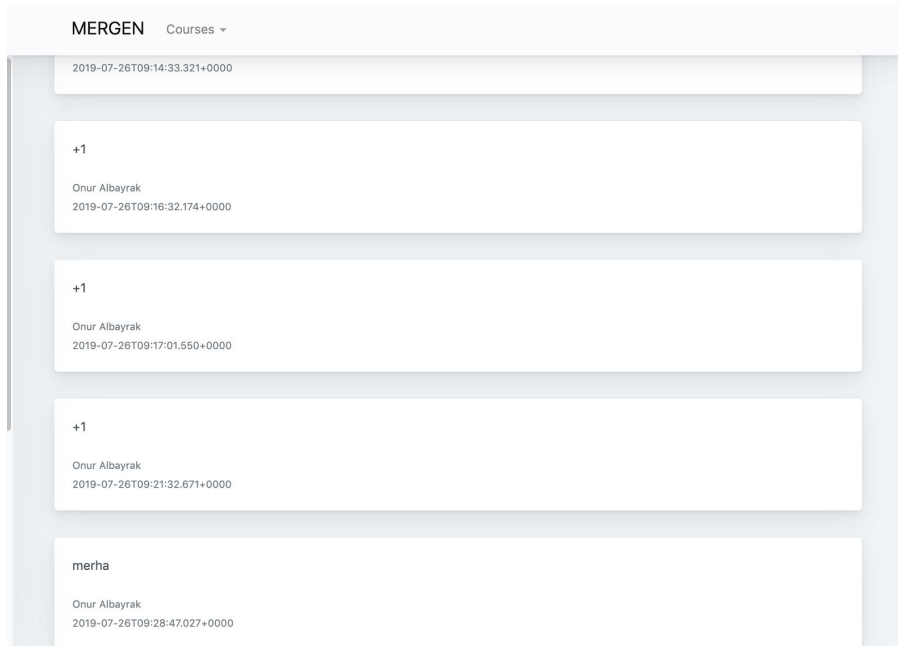
Homework	
Homework	Due Date
SpringS3Amazon.zip	

Homework Solutions	
Homework Solutions	
spring-boot-ftp-client-master.zip	

Lecture Notes	
Lecture Notes	Lecture Date
spring-boot-file-upload-download-rest-api-example-mysql-database-storage.zip	

General Resources	
General Resources	
scoold-master.zip	

Figür 1.11 - Resources Sayfası



The screenshot shows the 'MERGEN' website with a 'Courses' dropdown menu. The main content area displays a list of posts with their timestamps and comments. Each post has a '+1' button and a comment by 'Onur Albayrak'.

MERGEN Courses	
2019-07-26T09:14:33.321+0000	
+1	
Onur Albayrak	
2019-07-26T09:16:32.174+0000	
+1	
Onur Albayrak	
2019-07-26T09:17:01.550+0000	
+1	
Onur Albayrak	
2019-07-26T09:21:32.671+0000	
merha	
Onur Albayrak	
2019-07-26T09:28:47.027+0000	

Figür 1.12 - Postların Altındaki Yorumlar

9- Veritabanı Yapısı

Answers:

id	created_at	updated_at	author	message	question_id
1	2019-07-26T10:21:51.446Z	2019-07-26T10:21:51.446Z	ediz@gmail.com	Teşekkürler	2
2	2019-07-26T10:25:42.306Z	2019-07-26T10:25:42.306Z	ozyer@etu.edu.tr	Rica Ederim	2
3	2019-07-26T10:26:06.725Z	2019-07-26T10:26:06.725Z	ozyer@etu.edu.tr	Windowsta sanal makine kullansak olur mu?	4

Courses:

id	created_at	updated_at	course_code	course_name	course_semester
1	2019-07-26T10:19:23.745Z	2019-07-26T10:19:23.745Z	Bil 113	Bilgisayar Programlama I	2018-2019/Fall
2	2019-07-26T10:20:42.906Z	2019-07-26T10:20:42.906Z	Bil 211	Bilgisayar Programlama II	2018-2019/Summer
3	2019-07-26T10:20:55.697Z	2019-07-26T10:20:55.697Z	Bil 372	Veritabanı Sistemleri	2018-2019/Summer

Instructors:

id	created_at	updated_at	instructor_first_name	instructor_last_name	instructor_mail	instructor_password
1	2019-07-26T10:19:07.786Z	2019-07-26T10:19:07.786Z	Onur	Albayrak	onuralbayrak@etu.edu.tr	1
2	2019-07-26T10:24:54.344Z	2019-07-26T10:24:54.344Z	Tansel	Ozyer	ozyer@etu.edu.tr	1

Questions:

id	created_at	updated_at	author	message	title	course_id
1	2019-07-26T10:19:41.424Z	2019-07-26T10:19:41.424Z	onuralbayrak@etu.edu.tr	Dersimizin sayfası açılmıştır	Ders Sayfası	1
2	2019-07-26T10:21:25.264Z	2019-07-26T10:21:25.264Z	onuralbayrak@etu.edu.tr	Derslerimiz Bitmiştir. Proje demo tarihleri duyurulacaktır.	Dersler Hk.	3
3	2019-07-26T10:21:30.471Z	2019-07-26T10:21:30.471Z	ediz@gmail.com	Sınav tarihi ne zaman belli olacak?	Sınav Tarihi Hk.	3
4	2019-07-26T10:21:47.493Z	2019-07-26T10:21:47.493Z	onuralbayrak@etu.edu.tr	Linux kullanma zorunluluğu vardır.	Lab Sınavı Hk.	2

Resources:

id	created_at	updated_at	filename	tag	course_id
1	2019-07-26T10:22:20.58Z	2019-07-26T10:22:20.58Z	SpringS3Amazon.zip	General Resources	3
2	2019-07-26T10:22:42.264Z	2019-07-26T10:22:42.264Z	scoold-master.zip	Homework	2
3	2019-07-26T10:23:05.583Z	2019-07-26T10:23:05.583Z	Arsiv.zip	Lecture Notes	1

Students:

id	created_at	updated_at	school_id	student_first_name	student_last_name	student_mail	student_password
1	2019-07-26T10:20:59.378Z	2019-07-26T10:20:59.378Z	151101070	ediz	uslu	ediz@gmail.com	1

10-Sonuç ve Değerlendirmeler

Bu projede Spring Boot'un REST API özelliğinden faydalanarak takımdaki kişilerin olabildiğince bağımsız çalışmasına önem verildi. Grup üyelerinden bazıları projenin Backend kısmı ile uğraşırken diğer kısmı Front end kısmı ile uğraştı. Front end kısmında sayfa tasarımları yapıldıktan sonra REST API ile backend tarafından sağlanan endpointlere istek atarak sistemi inşa ettik. Proje takımı olarak bu projedeki en büyük kazanımımız Git ile açık kaynak kodlu proje geliştirerek ekip çalışmasının ve grup içi iletişimin önemini kavramaktı böylece bu konuda kendimizi geliştirme fırsatı bulduk. Daha öncesinde Bootstrap ve Spring Boot tecrübemiz olmadığı için bu projede sıfırdan bir sistem inşa ederek bu konu hakkında tecrübe sahibi olduk.

Geliştirilen sistemin benzeri şuan okulda Piazza ismi ile kullandığımız üniversite bilgi paylaşım platformudur. Bu projede başlangıç aşamasında ve ara raporlarda Piazza temel alınarak yapılmıştır.

11-Ekler

Bu kısımda ise son olarak teslim ettiğimiz Revised Miniworld Assumption'ımızı görmekteyiz.

BİL 372 VERİTABANI PROJE ÖNERGESİ DÜZENLEMESİ

Günümüzde teknolojinin gelişmesi ve yaygınlaşması ile haberleşme imkanları da son derece ulaşılabilir olmuştur. Hayatın her alanında birçok farklı şekilde haberleşme türü

kullanılmaktadır. Eskiden haberleşme için kullanılan fiziksel materyaller günümüzde uygun şartlar sağlandığı zaman yerini elektronik sistemlere bırakabiliyor. Bu sayede fiziksel materyalin ulaşımı esnasında harcanan zaman ortadan kalkıyor, haberleşme süresi kısalıyor ve kolaylık sağlanmış oluyor. Örnek olarak üniversitelerde kullanılan öğrenci-üniversite arasındaki ilişkiyi kuran bir elektronik sistem verilebilir.

Ders kapsamında yapmamız gereken proje için bu tür bir yazılımı geliştirmek istedik. Projemizin amacı, dönem içinde ders veren akademisyenler ve asistanlar ile dersi alan öğrenciler arasında bir köprü olarak görev yapacak, üniversite içi verileri güvenli bir şekilde muhafaza edip yüksek oranda erişilebilir bir sistem inşa etmektir.

Kullanılması Planlanan Teknolojiler

- Java
- Oracle Database
- MongoDB
- Hibernate
- Spring Boot
- Html/Css, Javascript, Ajax, JQuery vs.

Bulunması Planlanan Özellikler

- **Her Ders İçin Döneme Özel Sayfa:** Üniversitede bulunan her ders için açıldığı döneme özel sayfa bulunmaktadır.
- **Ders Bilgileri:** Her dersin sayfasında ders ile ilgili
 - Dersi veren akademisyenin iletişim bilgileri ve ofis saatleri
 - Ders asistanının iletişim bilgileri ve ofis saatleri
 - Ders Saatleri
- **Dosya Paylaşımı:** Derse kayıtlı bulunan akademisyen ve asistanlar ders ile ilgili dosya paylaşımı yapabileceklerdir. Paylaşılan bu dosyalar derse kayıtlı olan öğrenciler tarafından erişilebilir olacaktır.
- **Soru & Cevap Bölümü:** Öğrenciler ve Akademisyenlerin ortak kullanımıyla derse ekli olan herkes sorular sorabilir ve akademisyen tarafından duyurular paylaşılabilir.
- **Öğrencilerin Dönemlik Ders Programı:** Öğrencilerin kayıtlı olduğu dersler kullanılarak her öğrenciye kendisine ait dönemlik ders programı gösterilecektir.

Uygulama Detayları

Uygulamada bir login ekranı olacaktır.Bu login ekranından 2 farklı şekilde giriş yapılabilecektir(Instructor, Student).User'lar ise unique id'lere sahip olacaktır ve bu şekilde erişilebileceklerdir. User'ların online olup olmadığı bilgileri de tutulmakta olup, uygulamada bu bilgi tüm user'lar ile paylaşılacaktır. Bu iki farklı user'lar ise student ve instructor'lardır. Verilmekte olan derslerin bir arada tutulduğu courses tablosu bulunmaktadır. Dersler courses tablosunda unique course id'ye bağlı tutulacaktır.Student'lar alacakları dersleri

ekleyeceklerdir ve instructor'lar ise vereceği dersleri ekleyeceklerdir. Bir student birden fazla ders seçebilecektir. Bir instructor ise birden fazla dersi verebilmektedir. Dersleri alan studentlar ve dersleri veren instructorlar enrollment tablosunda unique enrollment id 'ye bağlı olarak tutulacaktır. Uygulamada soru/cevap bölümü bulunmaktadır. Bu soru/cevap bölümü iki alt kısımdan oluşmaktadır. Bunlar post ve comment'dir. İki user da post ve comment atabileceklerdir. Q&A tablosunda da user'lar tarafından atılan post ya da comment'ler unique qa id'ye bağlı olarak tutulacaktır.

Tablolar

- Uygulamada USER'lar iki alt sınıftan meydana gelmektedir. Bunlar STUDENT ve INSTRUCTOR'dır.
- COURSES tablosu ile USER'lar arasında ilişkiler bulunmaktadır. STUDENT ve COURSES arasında TAKES (N,M) ilişkisi bulunmaktadır. INSTRUCTOR VE COURSES arasında GIVES (N,M) ilişkisi bulunmaktadır.
- ENROLLMENT tablosu ile dersler ve bu derslere kayıtlı olan kişiler tutulacaktır. COURSES ve ENROLLMENT arasında CONSISTS (N,1) ilişkisi bulunmaktadır.. STUDENT ile ENROLLMENT arasında ASSIGNS (N,M) ilişkisi bulunmaktadır.
- Uygulama içinde USER'lar için Q&A bölümü yer alacaktır. USER'lar post ve comment atabileceklerdir. Q&A'nın de iki alt sınıfı bulunmaktadır. Bunlar POSTS ve COMMENTS'dir. USER ile Q&A arasında CAN USE (N,M) ilişkisi bulunmaktadır.
- Bunlarla birlikte ENROLLMENT ile Q&A arasında da bir ilişki bulunmaktadır. Bu HAS (N,N) ilişkisidir.
- Yukarıda sayılan user story'de toplamda 8 adet tablo ve 6 adet relationship bulunmaktadır.
- Her USER'ın kendine özel bir id'si bulunmaktadır. Ayrıca name, surname, email, password, isOnline attribute'larına da sahiptir.
- Her POST'un kendine özel bir post_id'si bulunmaktadır. Ayrıca post_message, post_author, post_date attribute'larına da sahiptir.
- Her COMMENT'in kendine özel bir comment_id'si bulunmaktadır. Ayrıca comment_message, comment_author, comment_date, post_id attribute'larına da sahiptir.
- Her COURSE'un kendine özel bir course_id'si bulunmaktadır. Ayrıca course_name'e de sahiptir.
- Her ENROLLMENT'ın kendine özgü bir enrollment_id'si bulunmaktadır. Ayrıca course_id, course_name, student_list ve qaId attribute'larına da sahiptir.
- Her Q&A kendine özgü bir qaId'ye sahiptir.

