Semester Project

IBRAHIM OUNON

PROFESSOR: BENOIT GARBINATO

ASSISTANT: MELIKE GECER

Summary

- Background reminders
- Mockup User Interfaces
- Difficulties encountered
- Demonstration
- Conclusion

Background reminder(1)

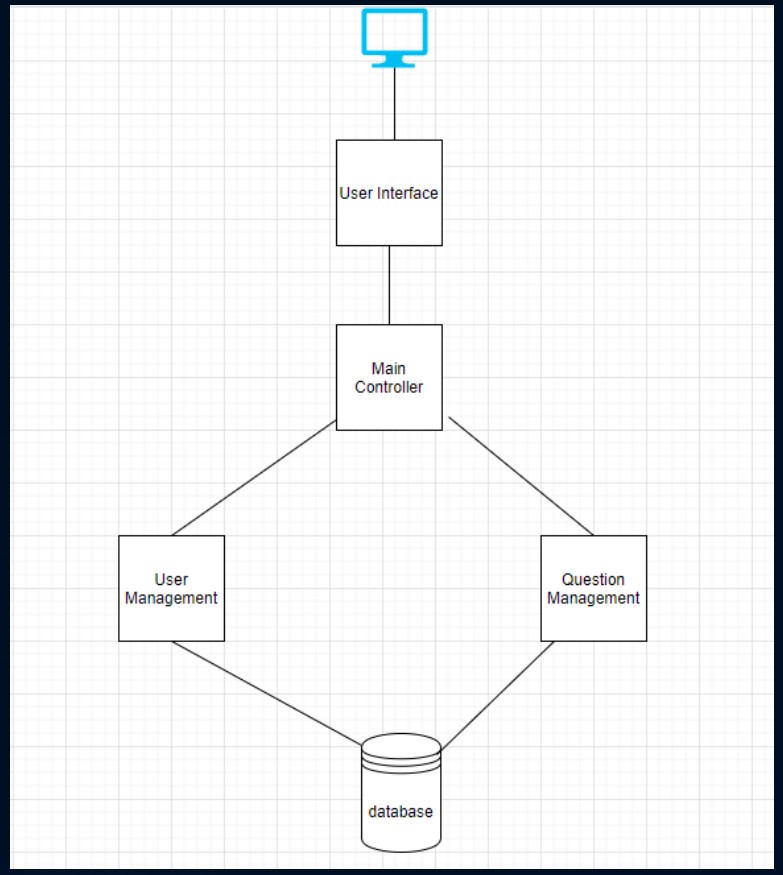
- Problem
  - Impossible to check students solution one by one.
  - The JupyterHub solution is not optimal.

- Solution to be implemented
  - developing an online coding environment.

Background reminder(1/2)

- Backend
  - Two microservices.
    - Microservice of users.
    - Microservice of exercises.

Background reminder(1/3)



Structure of microservices

- Entities
- DAO
- Services

Structure of microservices

- Entities
  - JPA entities that represent a table in our database.
  - The fields make up the attributes of the table.
  - Classes are annotated with @Entity which indicates that it is a persistent class.

Structure of microservices

```
@Entity
@Table(name = "student")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Student.findAll", query = "SELECT s FROM Student s"),
    @NamedQuery(name = "Student.findByStudentId", query = "SELECT s FROM Student s WHERE s.studentId = :studentId"),
    @NamedQuery(name = "Student.findByFirstName", query = "SELECT s FROM Student s WHERE s.firstName = :firstName"),
    @NamedQuery(name = "Student.findByLastName", query = "SELECT s FROM Student s WHERE s.lastName = :lastName"),
    @NamedQuery(name = "Student.findByEmail", query = "SELECT s FROM Student s WHERE s.email = :email"),
    @NamedQuery(name = "Student.findByPassword", query = "SELECT s FROM Student s WHERE s.password = :password"),
    @NamedQuery(name = "Student.findIfExist", query = "SELECT s FROM Student s WHERE s.password = :password AND s.email = :email")})
public class Student implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "student_id")
    private Short studentId;
    @Basic(optional = false)
    @Column(name = "first_name")
    private String firstName;
    @Basic(optional = false)
    @Column(name = "last_name")
    private String lastName;
    @Basic(optional = false)
    @Column(name = "email")
    private String email;
    @Basic(optional = false)
    @Column(name = "password")
    private String password;
    @ManyToMany(mappedBy = "studentList")
    private List<Course> courseList;

    public Student() {
    }
```

- @NameQuery allows you to specify a query
- @Id specifies the primery Key
- @Column specifies the attribute of Table

Structure of microservices

```
public Student(Short studentId, String firstName, String lastName, String email, String password) {
    this.studentId = studentId;
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
}

public Student(String firstName, String lastName, String email, String password) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
}

public Short getStudentId() {
    return studentId;
}

public void setStudentId(Short studentId) {
    this.studentId = studentId;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}
```

- Constructors, setters and Getter

Structure of microservices

```
public interface StudentRepository extends JpaRepository<Student, Short> {

    @Query(value = "SELECT * FROM Student s WHERE s.email = :email", nativeQuery = true)
    Student findByEmail(@Param("email") String email);

}
```

```
findAllById(Iterable<Short> itrbl)  List<Student>
findByEmail(String email)                  Student
findById(Short id)                Optional<Student>
findOne(Example<S> exmpl)               Optional<S>
flush()                                       void
getClass()                               Class<?>
getOne(Short id)                           Student
hashCode()                                    int
notify()                                      void
notifyAll()                                   void
save(S s)                                        S
saveAll(Iterable<S> itrbl)                List<S>
saveAndFlush(S s)                                S
toString()                                 String
wait()                                        void
wait(long timeout)                            void
wait(long timeout, int nanos)                 void
```

```
public interface CourseRepository extends JpaRepository<Course, Short>{


}
```

- JPArepository is an interface made available by SpringBoot.
- CRUD methods and other

Structure of microservices

Demonstration

Conclusion (1/2)

- Views on the professors are missing.
- Application of the concepts seen in the course.
- Learning and discovery of new technology.
- Good experiences