TP 02 : Développement d'une application Web CRUD (P2- SpringBoot- Back-end)

1- Créer le modèle représente par la classe Java **User1** et le compléter avec le constructeur + les **getters** et **setters** :

```
package com.example.backapp;
import java.util.Date;
import jakarta.persistence.*;

@Entity
@Table(name = "user1")
public class User1 {
	@Id
  @GeneratedValue(strategy = GenerationType.AUTO)
	private Long numUser;

	@Column(name = "user_name")
	private String userName;

	@Column(name = "nom")
	private String nom;

	@Column(name = "prnom")
	private String prnom;

	@Column(name = "date_de_naissance")
	private Date dateDeNaissance;

	@Column(name = "lieu_de_naissance")
	private String lieuDeNaissance;

	@Column(name = "mot_passe")
	private String motPasse;

	@Column(name = "role")
	private int role;

	@Column(name = "num_service")
	private int numService;
```

2- Créer l'interface Java **UserRepository**:

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface UserRepository extends JpaRepository<User1, Long> {
}
```

3- Créer le contrôleur avec la classe Java **UserController**:

```
package com.example.backapp;
import java.util.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import jakarta.validation.Valid;
@CrossOrigin(origins = "http://localhost:4200")
@RestController
@RequestMapping("employees")
public class UserController {
        @Autowired
          private UserRepository UserRepository;
          @GetMapping
          public ResponseEntity<List<User1>> getAllUsers() {
            List<User1> returnUsers = UserRepository.findAll();
            return new ResponseEntity<>(returnUsers, HttpStatus.OK);
          }
```

```
@GetMapping("{id}")
public ResponseEntity<User1> getBlogPostById(@PathVariable Long id) {
  if (UserRepository.existsById(id)) {
      User1 user1 = UserRepository.findById(id).get();
     return new ResponseEntity<>(user1, HttpStatus.OK);
  }

  return new ResponseEntity<>(HttpStatus.BAD_REQUEST);   }

@PostMapping
public ResponseEntity<User1> createUser(@RequestBody User1 user1) {
      System.out.println("Je suis dans la procedure CreateUser! ");
      UserRepository.save(user1);
  return new ResponseEntity<>(HttpStatus.CREATED);
}
@PutMapping("{id}")
public ResponseEntity<User1> updateUserById(@PathVariable Long id,
@RequestBody User1 user) {
  Optional<User1> userToFind = UserRepository.findById(id);
  if (userToFind.isPresent()) {
      User1 userToUpdate = userToFind.get();
      userToUpdate.setUserName(user.getUserName());
      userToUpdate.setNom(user.getNom());
```

```
            userToUpdate.setPrnom(user.getPrnom());
            userToUpdate.setDateDeNaissance(user.getDateDeNaissance());
            userToUpdate.setLieuDeNaissance(user.getLieuDeNaissance());
            userToUpdate.setMotPasse(user.getMotPasse());
            userToUpdate.setRole(user.getRole());
            userToUpdate.setNumService(user.getNumService());
           UserRepository.save(userToUpdate);
              return new ResponseEntity<>(HttpStatus.OK);
       } else {
          UserRepository.save(user);
          return new ResponseEntity<>(HttpStatus.CREATED);
       }
    }

    @DeleteMapping("{id}")
    public ResponseEntity<User1> deleteBlogPostById(@PathVariable Long id) {
       if (UserRepository.existsById(id)) {
          UserRepository.deleteById(id);
          return new ResponseEntity<>(HttpStatus.OK);
       } else {
          return new ResponseEntity<>(HttpStatus.NOT_FOUND);
       }
    }
}
```