package fr.formation.devs.controllers;

@RestController

public class DevRessourceController{

private final DevService service;

public DevController(DevService service)
{this.service=service)

@PostMapping ("/devs")
public void create(@RequestBody DevCreate)
{service.createRessource}

@GetMapping("/devs/{pseudo}
public DevCreate getByPseudo (@Valid @PathVariable String pseudo)
{return service.getByPseudo()}
p RessourceView getByPseudo(@Valid @PathVariable String pseudo)
{ service.getByPseudo() }

@PatchMapping("/devs/{pseudo}")
public void updateBirthDate(@PathVariable String pseudo, @Valid
@RequestedBody DevUpdate partial)
{ service.updateField (partial, pseudo); }

@GetMapping("/devs/find")
p IdevView find()
{return service.find()}

package fr.formatio.devs.services;

public interface Dev service {

DevView getByPseudo(String pseudo);
DevView getByPseudo(String Pseudo);
void create (DevCreate ressource);
void updateBirthDate(String pseudo, devUpdate partial);
IDevView find();
}

package fr.formation.devs.repositories;

@Repository

public interface DevRepository extends JpaRepository <Dev, Long>

{
Optional <Dev> findByPseudo (String Pseudo);

Optional <IDevView> findByFirstNameAndLastName (String fstName,

package fr.formation.skills.repositories;

@Repository

public interface SkillRepository extends JpaRepository <Skill, Long>

{
Optional <Skill> findByName (String name);
boolean existsByName (String value);
}

package fr.formation.developers.services;

@Service

public class DevServiceImpl implements Ressource Service {

```
private final  DevRepository devRepo;
private final  SkillRepository skillRepo;
public  DevServiceImpl ( DevRepository devs, SkillRepository skills )
{    this.devs= devs;   this.skills = skills }
```

```
@Override
public void create ( DevCreate dto ) {
Dev entity= devsRepo.findByPseudo ( pseudo ).get( );
    devEntity.setPseudo( dto. getPseudo( ));
    devEntity.setBirthDate( dto. getBirthDate( ));
    devEntity.setFirstName( dto. getFirstName( ));
    devEntity. setLastName( dto. getLastName( ));
Long mainSkillId = dto.getMainSkillId( );
 Skill skill = skillsRepo.getOne (mainSkill( Id ));
  devEntity.setMainSkill ( skill );
  devsRepo.save ( devEntity );
}
```

```
@Override
DevView getByPseudo ( String pseudo ) {
 Dev devEntity= devRepo. findByPseudo( pseudo). get();
 DevView view = new DevView ( );
       view.setPseudo ( devEntity. getPseudo( ));
       view.setFirstName (devEntity. getFirstName( ));
       view.setLastName (devEntity. getLastName( ));
       view. setBirthDate ( devEntity. getBirthDate( ));
 return view;
 }
```

```
@Override
public void updateBirthDate (String pseudo, DevUpdate partial)
{
 Dev devEntity = devRepo. findByPseudo( pseudo ). get();
 devEntity. setBirthDate (partial. getBirthDate ( ));
devRepo. save (devEntity);
}
```

```
@Override
public IDevView find ( )
{
   String firstName = "....";
    String lastName=".....";
return devRepo. findByFirstNameAndLastName (firstName, lastName). get();
 }
```

package fr.formation.devs.domain.entities;

```
@Entity
@Table (name="developers" )
```

public class Dev {

```
@Id
@GeneratedValue ( strategy= GenerationType. IDENTITY )
private Long id;

@Column ( name = "pseudonyme", unique = true; nullable = false )
private String pseudo;

@Column ( name = "first_name", nullable=false)
private String firstName;

@Column ( name = "last_name", nullable=false)
private String lastName;

@Column ( name = "birth_date", nullable=false)
private LocalDate birthDate;

@ManyToOne  // many devs to one skill
@JoinColumn ( name= "main_skill_id", nullable = false)
private Skill mainSkill;
```

public dev()
{ // }

***** getters: Long getId(), String getPseudo(), String getFirstName(), String getLastName(), LocalDate getBirthDate(), Skill getMainSkill();**

package.fr.formation.devs.validation

```
@Retention (RUNTIME)
@Target (FIELD)
@Constraint (validatedBy= {
UniqueSkillNameValidator.class})
```

MyConstraint annotation class

```
public @interface UniqueSkillName {
 String message() default "Doit être unique";
Class<?>[] groups() default {};
Class<? extends Payload>[]payload() default {};
```

package fr.formation.devs.domain.dtos

```
public class DevView {
private String Pseudo;
private String firstName;
private String lastName;
private LocalDate birthDate;

public DevView()
{       /  /       }
```

*** getters: String getPseudo(), String getFirstName(), String getLastName(), LocalDate getBirthDate();

***** setters: setPseudo(String pseudo), setPseudo(String pseudo), setFirstName(String fstName), setLastName(String

LogiqueOf MyConstraint

package.fr.formation.devs.validation

```
public class UniqueSkillNameValidator implements
ConstraintValidator<UniqueSkillName, String> {

private final SkillRepository skillRepo;

public UniqueSkillValidator( SkillRepository skillRepo)
   {   this.skillRepo = skillRepo}

@Override
public boolean isValid (String value, ConstraintValidatorContext context)
 {
  if (value==null) { return true; }
   return !skills. existsByName (value);
```

package fr.formation.devs.dtos;

```
public class DevUpdate {
@NotNull
private LocalDate birthDate;

public DeveloperUpdate()
 {          / /        }

// getters  & setters:
 LocalDate getbirthDate();
setBirthDate (LocalDate birthDate);
```

un DTO representant tous les inputs du client.
Contraints de validation par champ ou par classes (pour acceder aux plsrs champs)

```
package fr.formation.domain.dtos;
public class DevCreate {
@NotBlank
private String Pseudo;
@NotBlank
private String firstName;
@NotBlank
private String lastName;
@NotNull
private LocalDate birthDate;
@NotNull
private Long mainSkillId;

public DevCreate()
    {    //    }

*** getters: Long getId(),  String getPseudo(),  String getFirstName(),  String getLastName(),  LocalDate getBirthDate(),  Skill getMainSkill();

     ***** setters:   setId(Long id),   setPseudo(String pseudo),  setFirstName(String fstName), setBirthDate(LocalDate date),
                     setMainSkill( Skill mainSkill), ;

toString( ) { return "..............................."; }
```