



2022 / 2023

Rapport de stage d'initiation

Module gestion des utilisateurs et des
ressources en utilisant le Framework
« Spring Boot »

Réalisé par : Timoumi Mahmoud

Effectué au sein de la société :



Société Tunisienne de l'Ingénierie et de l'Informatique

Adresse: 1, Avenue du Dollar, résidence Sidi Mansour III Bloc A, Les Jardins du Lac,
Lac II 1053 Tunis

TEL: 71 195 300- FAX: 71 195 301

Email : st2i@st2i.com.tn

Remerciement

Tout d'abord, je tiens à remercier toute l'équipe du service développement informatique de Société Tunisienne de l'Ingénierie et de l'Informatique, pour leur accueil et leur collaboration.

Plus précisément, je tiens à remercier sincèrement :

- Mon encadrant de stage, Monsieur Mondher Abidli, pour m'avoir pris en charge, fait confiance, conseillé tout au long la période de stage.

SOMMAIRE

Introduction.....	1
I – Présentation De La Société	2
1- Introduction.....	2
2- Histoire	3
3- Secteurs d'activité.....	3
II – Projet réalisé durant le stage.....	4
1- Introduction Général	4
2- Diagramme de cas d'utilisation	6
3- Diagramme de classe	7
IV - Environnement de travail.....	9
1- Environnement matériel (Hardware).....	9
2- Environnement logiciel (Software)	9
3- Technologies et langages utilisées.....	10
III- Réalisation	11
1- Back End.....	11
A) Couche Entity	11
B) couche Repository.....	16
C) couche service	16
D) couche Controller	18
E) couche Security.....	22
2- Front End.....	23
A) Intégration de Template	23
A.1) INTÉGRATION DE TEMPLATE.....	23
A.2) INTÉGRATION DES FRAGMENT À PARTIR DE Template.....	23
B) L'INTERFACE GESTION DES DÉPARTEMENT	24
C) L'INTERFACE GESTION DES UTILISATEUR	26
D) l'interface gestion des Rôles	31
E) l'interface gestion des Fonctions	32
IV- Conclusion.....	33
Table des figure et des code	34

INTRODUCTION

Du 25/07/2022 au 09/07/2022, j'ai effectué un stage au sein de l'entreprise **Société Tunisienne de l'Ingénierie et de l'Informatique (ST2I)** qui se spécialise dans le domaine de l'informatique et des nouvelles technologies.

Au cours de ce stage au département Informatique, j'ai pu m'intéresser au module de gestion des utilisateurs et gestion de l'accès aux ressources.

Plus largement, grâce à cette expérience pratique, j'ai eu l'opportunité à la fois :

- D'appréhender le métier de développeur des application web et de le pratiquer pendant six semaines par la réalisation d'un projet FullStack en Spring backend et en bootstrap ,theymleaf front end .
- De mettre en pratique les notions que j'ai acquises durant mon troisième année comme (Jave, git, modélisation UML ...).
- De développer des nouvelles compétences cotées personnel comme l'organisation et gestion du temps. Coté technique découvrir des nouvelles framework et langage come (Spring boot(MVC),Spring Data, Spring Security, Theymleaf, javascript, JasperRepports ..etc.)

Annnonce de plan

Pour présenter mon expérience de stage au sein de la société ST2I, il apparaît pertinent de présenter à titre préalable l'entreprise qui m'avait accueilli pendant ces derniers semaines et de parler de son histoire et ces secteurs d'activité (**chapitre I**). Le **chapitre II** envisager la conception de projet réalisé : on commence par une petite introduction sur le projet et ces atouts, ensuite on fourni la conception d'application avec détaillés d'abord le digramme de cas d'utilisation ensuite le digramme des classes.

Dans le **chapitre III** on précisera l'environnement du travail : d'abord l'environnement matériel, ensuite on présente les logiciels utiliser (inteliide, xampp, jasperReport) et on fini par lister les différentes technologies utilisé dans ce projet (bootstrap,spring, thyemleaf...).

Dans le **chapitre IV** l'avant dernière, on s'occupait de la réalisation de projet, ce chapitre se divise en deux parties dans la première on s'intéresse de la partie Backend on présente quelques exemples de code de projet dans chaque package et la logique derrière. Dans la deuxième partie, on s'occupe de la coté Frontend.

En termine ce rapport par une conclusion.

I – PRESENTATION DE LA SOCIETE

1- INTRODUCTION

ST2i est un bureau de consulting se positionnant sur des créneaux à forte valeur ajoutée dans le domaine de l'Informatique, de la géomatique et des nouvelles Technologies de l'Information et de la Communication.



FIGURE 1: SOCITE ST2I

- Adresse: GP1 1, Avenue du Dollar, résidence Sidi Mansour III Bloc A, Les Jardins du Lac, Lac II 1053 Tunis
- Téléphone : 71 195 300
- FAX : 71 195 301
- Email : st2i@st2i.com.tn
- Site web : www.st2i.com.tn

2- HISTOIRE

Fondé en 1992 et membre d'un groupe international d'ingénierie multidisciplinaire, le Groupe STUDI, ST2i est un bureau de consulting se positionnant sur des créneaux à forte valeur ajoutée dans le domaine de l'Informatique, de la géomatique et des nouvelles Technologies de l'Information et de la Communication.

Forte d'une équipe dirigeante, ST2i a su être à l'écoute constante de ses clients et à l'affût des innovations technologiques lui garantissant une amélioration continue et une meilleure adaptation aux changements. Ces atouts ont permis à ST2i de mener à bien multiples projets d'envergure dans plus de quinze pays : en Tunisie, en Algérie, au Mali, au Côté d'Ivoire, au Sénégal, à Burkina Faso, au Bénin, au Togo, au Cameroun, en Guinée, en RDC, en Mauritanie, en Arabie Saoudite, au Canada, etc.

3- SECTEURS D'ACTIVITE

ST2i intervient particulièrement dans les domaines suivant :

- **Systèmes d'information** : avec une très grande expertise dans l'audit des systèmes existants et la conception de schémas directeurs stratégiques et opérationnels de l'Administration et des Établissements publics;
- **Géomatique et systèmes d'information géographique** : avec une équipe pluridisciplinaire constituée des experts géomaticiens, des développeurs et des techniciens hautement qualifiés et expérimentés, maîtrisant les techniques et technologies de pointe et couvrant une large gamme de solutions logicielles spécialisées
- **Télécommunications et TIC** : avec une maîtrise des solutions et réseaux de télécommunications et une aptitude à assister les clients dans l'audit des réseaux existants, le pilotage des projets en cours et le choix et la définition des spécifications de nouvelles solutions et de nouveaux réseaux ;
- **Informatique pour l'ingénierie** : en offrant une gamme de produits standards, ouverts et respectant la réglementation en vigueur ainsi que les méthodes de travail des différents corps de métier et en proposant les outils et les configurations les plus adaptées aux clients.

II – PROJET REALISE DURANT LE STAGE

1- INTRODUCTION GENERAL

La sécurité et l'un des principaux aspects pour juger de l'efficacité et de la qualité des applications à notre époque.

Il est nécessaire de fournir aux utilisateurs finaux une application fortifiée et armer et c'est pourquoi nous avons choisi Spring comme un cadre de travail en raison de ce qu'il offre avec Spring Security 5, comme des classes préfabriquées (userDetails, userDetailsService .etc.) et la flexibilité des ces configurations.

Dans notre projet, nous avons tiré parti de ce framework et nous avons essayé d'utiliser et de personnaliser ses configurations pour nous donne un module de gestion de l'utilisateur et des ressources selon les rôles qui peuvent être mis en place dans n'importe quelle application.

Le contrôle d'accès basé sur les rôles (ou RBAC sous une forme abrégée) signifie autoriser ou refuser d'effectuer une tâche ou d'accéder à une ressource en fonction du rôle de l'utilisateur authentifié. Prenons un exemple (figure 2 : RBAC):

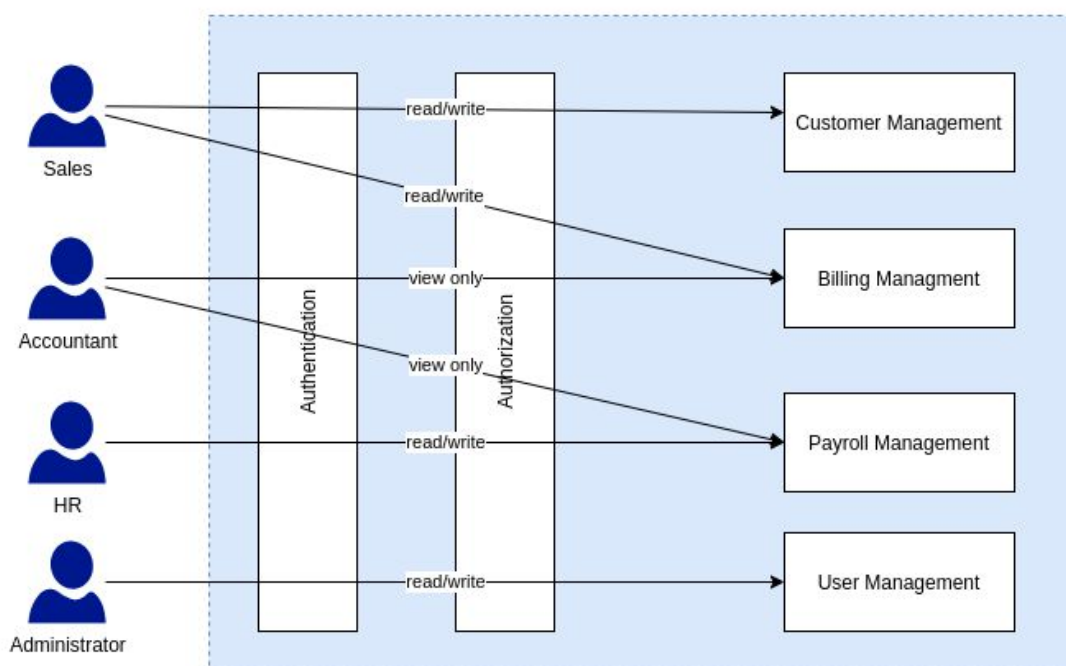


FIGURE 2:RBAC

Avec la réalisation des CRUD (Create , update, Read, Delete) des différent tables dans notre system on a concentré aussi d'appliquer des filtre d':

- **Authentification** — C'est le processus qui permet de s'assurer qu'un utilisateur est bien la personne qu'il prétend être. Traditionnellement, des combinaisons de nom d'utilisateur (dans notre projet c'est l'email) et de mot de passe sont utilisées pour garantir cela. Lorsqu'un utilisateur fournit une combinaison correcte de nom d'utilisateur et de mot de passe, nous le considérons comme un utilisateur authentifié en partant du principe que "seul cet utilisateur connaît la bonne combinaison nom d'utilisateur/mot de passe".
- **Autorisation** — S'applique généralement après qu'un utilisateur est authentifié avec succès. Donc après qu'il passe le premier filtre on décide si l'utilisateur authentifié dispose des autorisations nécessaires pour consulter une ressource demandé lors de la demande de cette ressource.

Pour résumer notre module de gestion des utilisateurs et d'accès basé sur les rôles consiste à :

- Création des profiles
- Affecter des rôles à ces profiles
- Affecter des ressources à des rôles
- Ajouter des nouvelles : ressources, rôles, département.
- Modifier les informations des utilisateurs, des départements, des rôles et des fonctions et également modifier l'affectation des rôles et des ressources.
- Configuration dynamique de la parité Security (autorisation).
- Ajouter des fonctionnalités avancées comme l'réinitialisation de mots de passe avec JavaMail, PDF avec JasperReports, pagination ...etc.

2- DIAGRAMME DE CAS D'UTILISATION

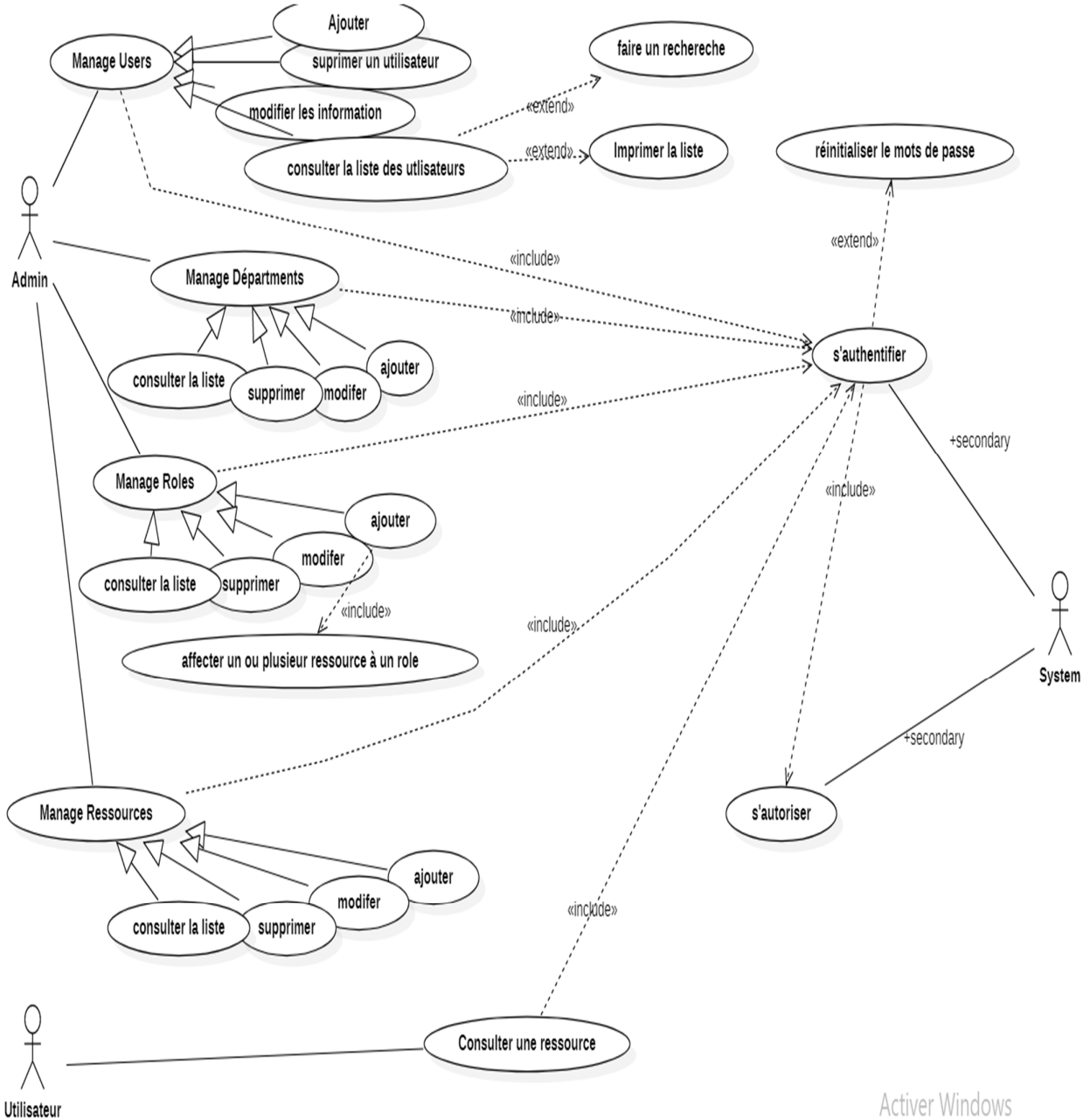


FIGURE 3:USE_CASE_DIAGRAMME

3- DIAGRAMME DE CLASSE

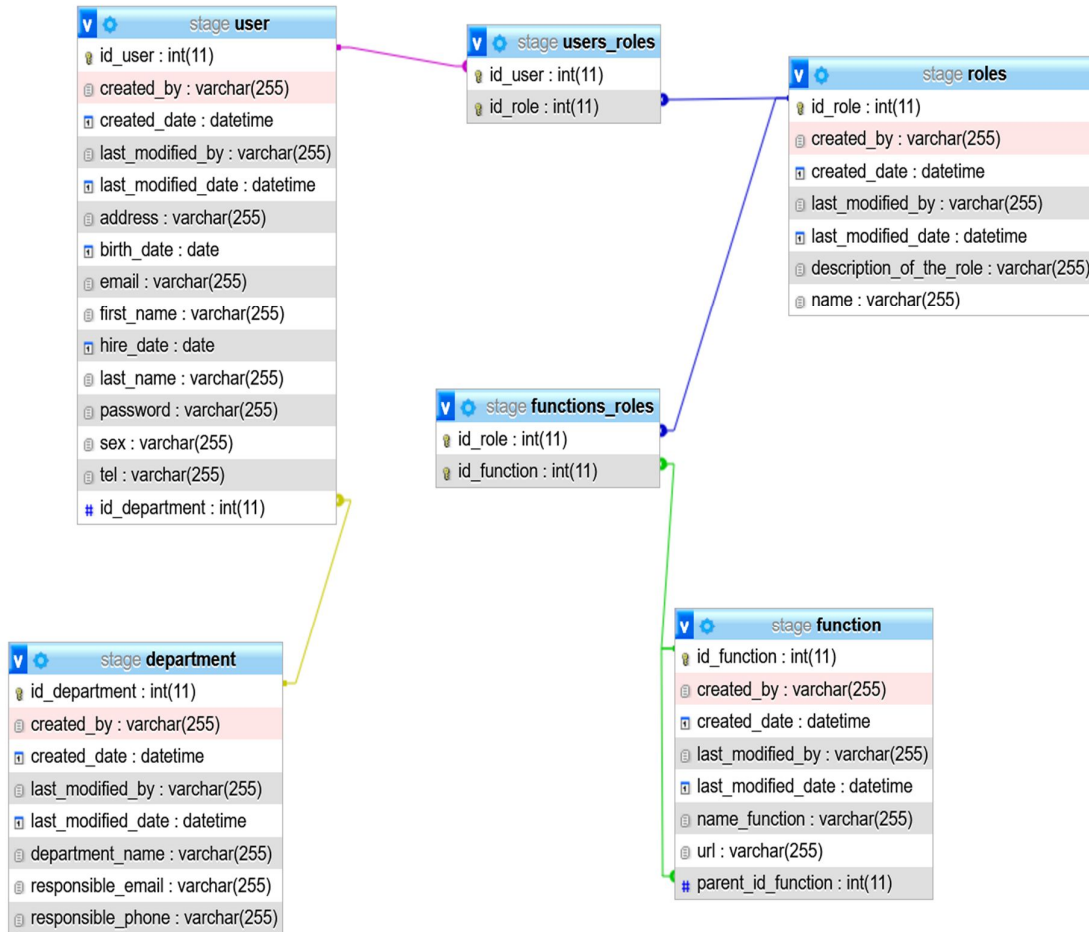


FIGURE 4:CLASS DIAGRAME

Notre application possède quatre classes (USER, DEPARTMENT, ROLES, FUNCTION) et des autres classes générées automatiquement par Hibernate (users_roles et fontions_roles) :

- la table USER possède les champs suivant (firstName, lastName, birthDate, hireDate, phonenumber, gender , idDepartmentetc.) cette tableau à une relation ManyToOne avec la table DEPARTMENT ce qui signifie que un utilisateur peut avoir une seul département et une département peut avoir plusieurs employés .
- Cette dernière possède les champs suivant (department_name, responsable_name, responsable_phone).

- La table ROLES possède les champs suivants (id_role, name, description_of_role), aussi avait une relation ManyToMany bidirectionnelle avec la table USER c'est-à-dire un utilisateur peut avoir plusieurs rôles et vice-versa un rôle peut être affecté a plusieurs utilisateur. Plus de la relation avec la table USER la table ROLES a aussi une relation ManyToMany avec la table FUNCTION.
- La table FUNCTION à les champs suivant (id_function, name_function, url, parent_id_function) avec deux relation :
 - Avec la table ROLES ManyToMany
 - Avec lui-même (self références bidirectionnelle)
OneToMany une fonction peut avoir plusieurs filles
et ManyToOne des fonctions filles a une seule fonction mère)

Plus des champs au-dessus, on trouve d'autres champs des historisation comme CreatedBy, CreatedAt, UpdatedBy, UpdatedAt qui sont présent dans les quatre tables.

IV - ENVIRONNEMENT DE TRAVAIL

1- ENVIRONNEMENT MATERIEL (HARDWARE)

- Disque Dur : 512 GB (ssd)
- Mémoire RAM : 8GB
- Processeur : 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
- Système d'exploitation : Windows10

2- ENVIRONNEMENT LOGICIEL (SOFTWARE)

Ci-dessous la liste des logiciels utilisés durant ce stage :

- **IntelliJ IDEA (ultimate edition)** : est un IDE intelligent et sensible au contexte pour travailler avec Java et d'autres langages JVM comme Kotlin, Scala et Groovy sur toutes sortes d'applications. De plus, IntelliJ IDEA Ultimate peut aider à développer des applications Web complètes, grâce à ses puissants outils intégrés, à la prise en charge de JavaScript et des technologies associées, et à la prise en charge avancée des frameworks populaires tels que Spring, Spring Boot.
- **XAMPP** : est une distribution logicielle qui fournit le serveur Web Apache, la base de données MySQL (en fait MariaDB), Php et Perl dans un seul package. Il est disponible pour les systèmes Windows, MAC et Linux.
- **Git Bash** : est une application pour les environnements Microsoft Windows qui fournit une couche d'émulation pour une expérience de ligne de commande Git.
- **Jasper Report studio** : est un outil de reporting Java open source qui peut écrire sur une variété de cibles, telles que : un écran, une imprimante, dans des fichiers PDF, HTML, Microsoft Excel, RTF, ODT, des valeurs séparées par des virgules (CSV) ou des fichiers XML. Il peut être utilisé dans des applications compatibles Java, y compris Java EE ou des applications Web, pour générer du contenu dynamique. Il lit ses instructions à partir d'un fichier XML ou .jasper.

3- TECHNOLOGIES ET LANGAGES UTILISEES

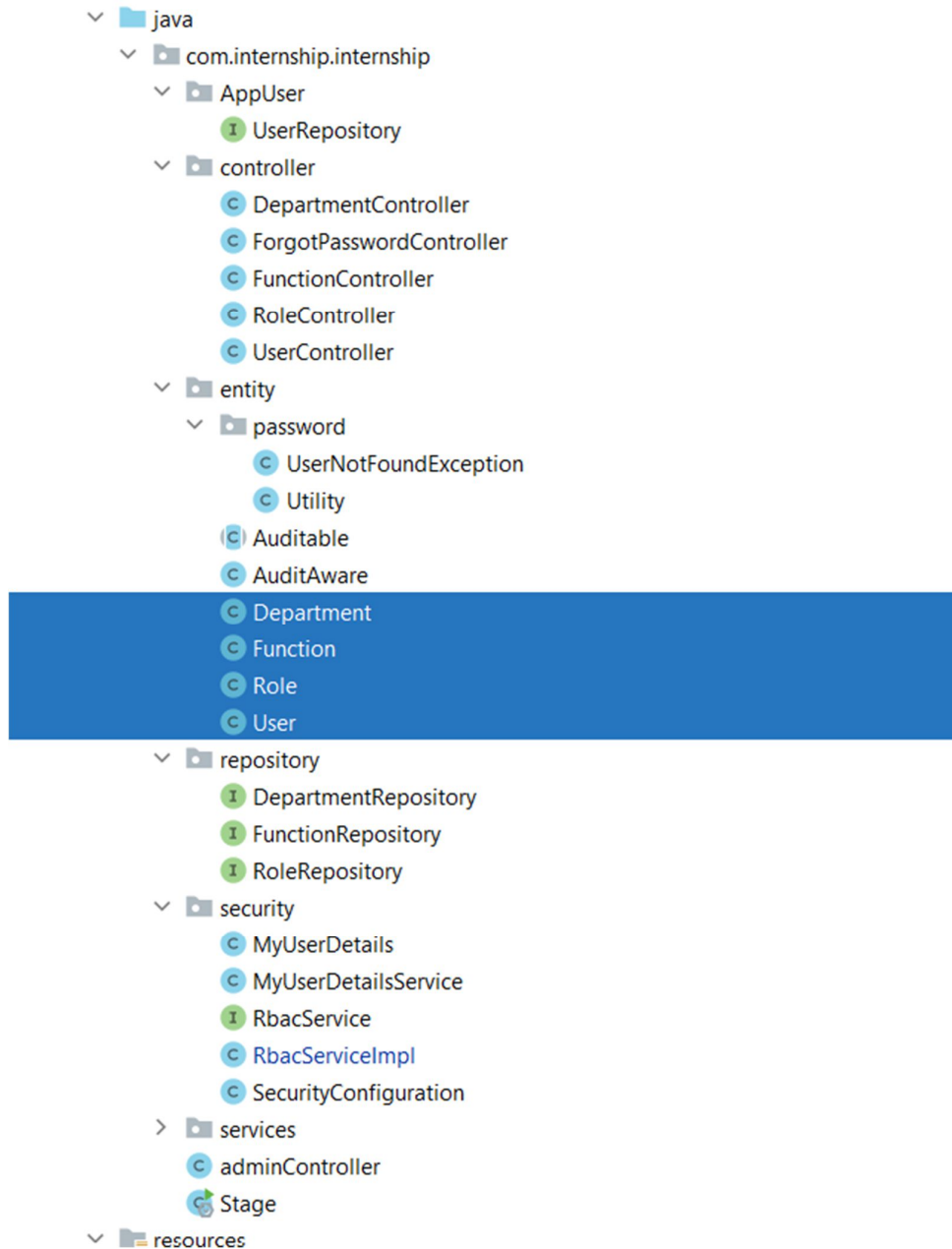
Ci-dessous la List des langages et framwork utilisé :

- **Spring** (Back-end) est un micro framework open source géré par une société appelée Pivotal. Il fournit aux développeurs Java une plate-forme pour démarrer avec une application Spring de production auto-configurable. Avec lui, les développeurs peuvent démarrer rapidement sans perdre de temps à préparer et configurer leur application Spring. Spring Boot est construit sur le framework Spring.
- **Bootstrap** : est un framework développé par l'équipe du réseau social Twitter. Proposé en open source (sous licence MIT), ce framework utilisant les langages HTML, CSS et JavaScript fournit aux développeurs des outils pour créer un site facilement. Elle est pensée pour développer des sites avec un design responsive, qui s'adapte à tout type d'écran.
- **Thymeleaf** : est un moteur de modèle Java côté serveur qui peut fonctionné à la fois dans des environnements Web et non Web. Il est mieux adapté pour servir XHTML/HTML5 au niveau de la couche d'affichage des applications Web basées sur MVC.
- **HTML** : est le langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom.
- **CSS** : est un langage informatique qui décrit la présentation des documents.

III- REALISATION

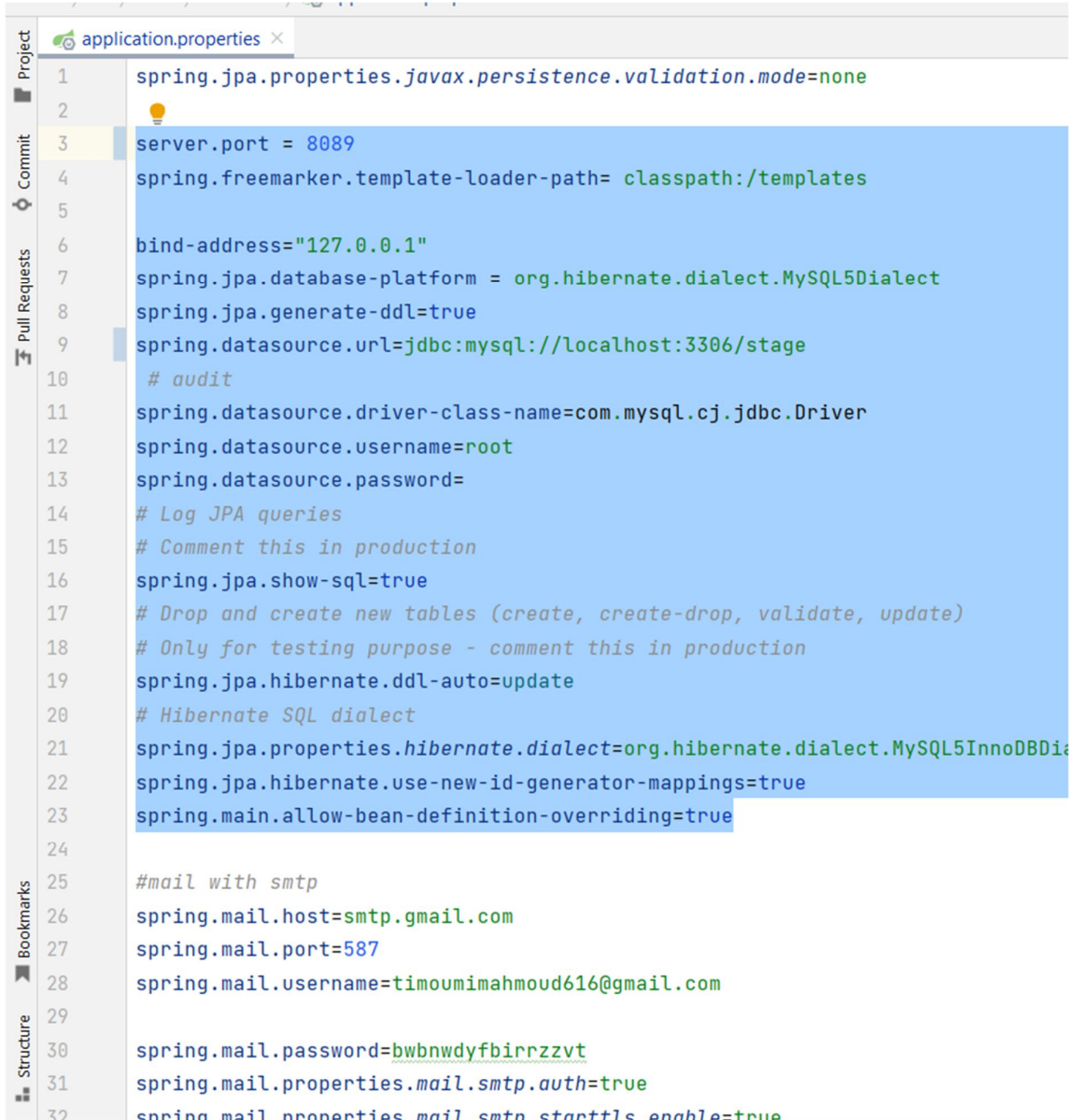
1- BACK END

A) COUCHE ENTITY



Spring offre le framework Hebernate qui est une implémentation de JPA ce qui permet de gérer le mapping entre les objets de l'application (enity) et la base de données.

L'idée est d'écrire les champs de chaque classe et les relations entre elle, ensuite après quelque configuration dans le fichier application.properties tel que le nom de base de données, le username et password de base de données (voir FIGURE 5:CONNECTION AVEC MYSQL). Il y aura donc une migration, un mappage de ces champs vers la base de données.



```
1 spring.jpa.properties.javax.persistence.validation.mode=none
2
3 server.port = 8089
4 spring.freemarker.template-loader-path= classpath:/templates
5
6 bind-address="127.0.0.1"
7 spring.jpa.database-platform = org.hibernate.dialect.MySQL5Dialect
8 spring.jpa.generate-ddl=true
9 spring.datasource.url=jdbc:mysql://localhost:3306/stage
10 # audit
11 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
12 spring.datasource.username=root
13 spring.datasource.password=
14 # Log JPA queries
15 # Comment this in production
16 spring.jpa.show-sql=true
17 # Drop and create new tables (create, create-drop, validate, update)
18 # Only for testing purpose - comment this in production
19 spring.jpa.hibernate.ddl-auto=update
20 # Hibernate SQL dialect
21 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
22 spring.jpa.hibernate.use-new-id-generator-mappings=true
23 spring.main.allow-bean-definition-overriding=true
24
25 #mail with smtp
26 spring.mail.host=smtp.gmail.com
27 spring.mail.port=587
28 spring.mail.username=timoumimahmoud616@gmail.com
29
30 spring.mail.password=bwbnwdyfbirrzzvt
31 spring.mail.properties.mail.smtp.auth=true
32 spring.mail.properties.mail.smtp.starttls.enable=true
```

FIGURE 5:CONNECTION AVEC MYSQL

Il faut aussi ajouter les dépendances nécessaire ci-dessous au niveau fichier pom.xml afin de utiliser Jpa hibernate et mysql :


```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
```

Ci-dessous quelque exemples des entity avec les différent annotations comme (@Entity , @Table , @generated , ManyToOne, @JoinTable...etc.) :

- USER classe (voir code 1: USER_CLASS)
- ROLES classe (voir code 2 : ROLES_CLASS)
- FUNCTION classe (voir code 3 : FUNCTION_CLASS)

```

Entity
@Table(name = "user")
public class User extends Auditable<String> {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_user")
    private int idUser;
    @Email(message = "Email is not valid", regexp = "(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*/+=?^_`{|}~-]+)*|\\\"(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-\\x5b\\x5d-\\x7f]|\\\\\\\\[\\x01-\\x09\\\\\\\\]))\"")
    @NotEmpty(message = "Email cannot be empty")
    private String email;
    @Size(min = 6, message = "Password must be greater than Six characters")
    private String password;
    @NotEmpty(message = "first name cannot be empty")
    @Column(name = "first_name")
    private String firstName;
    @NotEmpty(message = "last name cannot be empty")
    private String lastName;

    @Pattern(regexp = "male|female", flags =
Pattern.Flag.CASE_INSENSITIVE)
    @NotBlank(message = "please provide a gender ")
    private String sex;
    private Date birthDate;
    private Date hireDate;
    @Column(name = "reset_password_token")
    private String resetPasswordToken;
    @NotBlank(message = "mobileNumber is required")
    @Size(min = 8, max = 12)
    private String tel;
    @NotEmpty(message = "address cannot be empty")
    private String address;

    @ManyToMany(targetEntity = Role.class, cascade =
{CascadeType.PERSIST,
CascadeType.DETACH, CascadeType.MERGE, CascadeType.REFRESH} ,
    fetch = FetchType.EAGER )
    @JoinTable(
        name = "users_roles",
        joinColumns = @JoinColumn(name = "id_user"),
        inverseJoinColumns = @JoinColumn(name = "id_role")
    )
    private Set<Role> roles = new HashSet<>();

    @ManyToOne( fetch = FetchType.LAZY, optional = true)
    @JoinColumn(name = "id_department", nullable = true)
    private Department department;//getters , setters and constructor

```

```

@Entity
@Table(name = "roles")
public class Role extends Auditable<String> {
    @Id
    @Column(name = "id_role")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotBlank(message = "this field can't be empty")
    private String name;
    @Size(min = 10, max = 200, message
        = "Description must be between 10 and 200 characters")
    private String descriptionOfTheRole;

    @ManyToMany(targetEntity = User.class, mappedBy = "roles", cascade =
    {CascadeType.PERSIST,
    CascadeType.DETACH, CascadeType.MERGE, CascadeType.REFRESH})
    private Set<User> userss;
    @PreRemove
    private void removeRolesFromUsers() {
        for (User u : userss) {
            u.getRoles().remove(this);
        }
    }
    @ManyToMany(targetEntity = Function.class, cascade =
    {CascadeType.PERSIST, CascadeType.DETACH, CascadeType.REFRESH})
    @JoinTable(
        name = "Functions_roles",
        joinColumns = @JoinColumn(name = "id_role"),
        inverseJoinColumns = @JoinColumn(name = "id_function")
    )

    private Set<Function> RolesFunction ;
    //getters , setters and constructor

```

CODE 2: ROLES_CLASS

```

@Entity
public class Function extends Auditable<String> {
    @Id
    @Column(name = "id_function")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idFunction;
    private String nameFunction;
    private String url;

    @ManyToMany(targetEntity = Role.class, mappedBy = "RolesFunction", cascade =
    {CascadeType.PERSIST, CascadeType.DETACH, CascadeType.REFRESH} , fetch =
    FetchType.EAGER)
    private Set<Role> RolesF ;
    @PreRemove
    private void removeFunctionFromRoles() {
        for (Role r : RolesF) {
            r.getRolesFunction().remove(this); } }

    @ManyToOne(cascade = {CascadeType.PERSIST, CascadeType.DETACH,
    CascadeType.REFRESH})
    private Function parent;

    @OneToMany(mappedBy = "parent", cascade=CascadeType.ALL, fetch = FetchType.EAGER)
    private Set<Function> children ;

```

CODE 3: FUNCTION_CLASS

B) COUCHE REPOSITORY

C'est la couche qui communique directement avec la base de données.

Elle offre des méthodes prédéfinies comme `findAll()`, `findById()`, `save()`, `saveAndflush()`...etc. Ces différentes méthodes sont utilisées dans ce projet afin de faire les CRUD (créer, mettre à jour, supprimer, lire) de différentes tables.

Pour consommer ces méthodes dans la couche service ou bien dans le Controller on injecte le Repository avec l'annotation `@Autowired`.

Il y a aussi la possibilité de customiser des requêtes. Pour ce projet on a fait plusieurs requêtes par exemple :

- `loadUserByEmail(String email)` pour obtenir toutes les informations d'utilisateur au cours de login, ces informations sont utilisées dans le filtre d'authentification.
- Requête qui permet de connaître la liste des URLs qui sont reliées à l'utilisateur connecté selon son username afin de filtrer l'accès à différentes URLs (ressources) voici un exemple :

```
@Query( value ="select * from function f JOIN functions_roles fr  
on f.id_function= fr.id_function"+  
        " JOIN roles r on fr.id_role = r.id_role " +  
        " JOIN users_roles ur on ur.id_role=r.id_role" +  
        " JOIN user u on u.id_user=ur.id_user and u.email  
like :username "  
        , nativeQuery = true  
        )  
List<Function> urlsFinder( @Param("username") String username);
```

C) COUCHE SERVICE

La couche service joue le rôle d'un intermédiaire entre le Repository (couche qui communique directement avec la base de données) et le Controller (responsable de servir des requêtes entrantes et appeler les vues)

Dans cette couche on trouve les classes suivantes : `UserService`, `RoleService`, `DepartmentService`, `FunctionService`.

Ci-dessous un exemple de l'une de ces classes (code 7 : USER_SERVICE CLASS)

```

@Service
public class UserService {

    @Autowired    //call of the user Repository interface
    private UserRepository userRepository;

    //used in the pagination
    public Page<User> findPage(int pageNumber) {
        Pageable pageable = PageRequest.of(pageNumber - 1,5);
        return userRepository.findAll(pageable);}

    public List<User> search(String value) {
        return userRepository.search(value);
    }

    public void delete(int idUser) {
        userRepository.deleteById(idUser);  }

    public User findById(int idUser) {
        return userRepository.findById(idUser).get(); }
    public User save(User user) {
        /////Decrypt password in the registration process with BCrypt
        PasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();
        String hashedPassword =
passwordEncoder.encode(user.getPassword());
        user.setPassword(hashedPassword);
        return userRepository.save(user);
    }
    public User update(User user, int idUser) {
        User oldUser = userRepository.findById(idUser).get();
        oldUser.setFirstName(user.getFirstName());
        oldUser.setLastName(user.getLastName());
        oldUser.setHireDate(user.getHireDate());
        oldUser.setAddress(user.getAddress());
        oldUser.setSex(user.getSex());
        oldUser.setPassword(user.getPassword());
        oldUser.setTel(user.getTel());
        oldUser.setBirthDate(user.getBirthDate());
        oldUser.setRoles(user.getRoles());
        oldUser.setEmail(user.getEmail());
        oldUser.setDepartment(user.getDepartment());
        userRepository.save(user);
        return user;  }

    /////Work with Jasper report to generate PDF file
    public List<Map<String, Object>>getAllUsers(){
        List<Map<String, Object>> users= new ArrayList<>();
        users=userRepository.allUserDetails(); //Call a query form
userRepository to use it's return value as dataSource for the pdf file .

        return users    }

```

```

//////////////////////////////////////Forget password ////////////////////////////////////////

public void updateResetPasswordToken(String token, String email)
throws UserNotFoundException {
    User user = userRepository.findByEmail(email);
    if (user != null) {
        user.setResetPasswordToken(token);
        userRepository.save(user);
    } else {
        throw new UserNotFoundException("Could not find any customer with
the email " + email);
    }
}

public User getByResetPasswordToken(String token) {
    return userRepository.findByResetPasswordToken(token);
}

public void updatePassword(User user, String newPassword) {
    BCryptPasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();
    String encodedPassword = passwordEncoder.encode(newPassword);
    user.setPassword(encodedPassword);

    user.setResetPasswordToken(null);
    userRepository.save(user);
}}

```

CODE 4: USER_SERVICE CLASS

D) COUCHE CONTROLLER

La classe contrôleur est responsable du traitement des requêtes entrantes de l'API REST, de la préparation d'un modèle et du renvoi de la vue à rendre en réponse. Les classes de contrôleur dans Spring sont annotées soit par l'annotation `@Controller` soit par l'annotation `@RestController`. Voici un exemple (CODE 9: USERCONTROLE)

```

@RequestMapping(value = "/User")
@RestController
public class UserController {
    @Autowired
    private UserService userService;
    @Autowired
    private RoleService roleService;
    @Autowired
    private DepartmentService departmentService;
    @Autowired
    private RoleRepository roleRepository;
    @Autowired
    private FunctionService functionService;

    @RequestMapping(value = "/list/{id}")
    public User findById(@PathVariable int id) {
        return userService.findById(id);
    }

    //-----delete a user-----
    @GetMapping("delete/{id}")
    public RedirectView remove( @PathVariable int id){
        userService.delete(id);
        return new RedirectView("/User/userList");    }

    //-----list of users with pagination also call search
    //method from userService-----
    @GetMapping( path={"/userList","/search"})
    public ModelAndView getAllPages(String keyword){
        Map<String, List<User>> model = new HashMap<String,
List<User>>();
        String viewName = "/user/userList";
        if(keyword != null){
            model.put("Users",userService.search(keyword));
            return new ModelAndView(viewName , model);
        } return getPage( 1);}

    @GetMapping("/userList/page/{pageNumber}")
    public ModelAndView getPage( @PathVariable("pageNumber") int
currentPage){
        Page<User> page = userService.findPage(currentPage);
        int totalPages = page.getTotalPages();
        long totalItems = page.getTotalElements();
        List<User> Users = page.getContent();
        ModelAndView mav = new ModelAndView("/user/userList");
        mav.addObject("currentPage", currentPage);
        mav.addObject("totalPages", totalPages);
        mav.addObject("totalItems", totalItems);
        mav.addObject("Users", Users);
        return mav;    }

```

```

//////////--add new user--//////////
@GetMapping("/addUser")
public ModelAndView addForm() {
    ModelAndView mav = new ModelAndView("user/registration");
    mav.addObject("departs", departmentService.findAll());
    mav.addObject("listRoles", roleService.findAll());
    User user = new User();
    mav.addObject("user", user);
    Map<String, List<Department>> departs = new HashMap<String,
List<Department>>();
    mav.addObject("Function", functionService.findAll());
    return mav; }

@RequestMapping(value = "/addUser",method = RequestMethod.POST)
public ModelAndView saveUser(@Valid User user, BindingResult
bindingResult) {
    if (bindingResult.hasErrors()) {
        ModelAndView mav = new ModelAndView("user/registration");
        mav.addObject("departs", departmentService.findAll());
        mav.addObject("listRoles", roleService.findAll());
        return mav;
    }
    userService.save(user);
    try {
        Thread.sleep(1000); } catch (InterruptedException e) {
        e.printStackTrace(); }
    RedirectView redirectView = new RedirectView();
    redirectView.setUrl("/User/userList");
    return new ModelAndView(redirectView);
}

//////////--update a user--//////////
@GetMapping("/edit/{id}")
public ModelAndView editUser(@PathVariable("id") int id) {
    User user = userService.findBy(id);
    ModelAndView mav = new ModelAndView("role_user/giveRole");
    mav.addObject("user", user);
    mav.addObject("u", userService.findBy(id));
    List<Role> listRoles = roleService.findAll();
    mav.addObject("listRoles", listRoles);
    return mav;
}

@RequestMapping(value = "/edit/{id}" , method =
RequestMethod.POST)
public ModelAndView saveUser(@Valid @ModelAttribute("user") User
user, @PathVariable("id") int id) {
    userService.update(user, id);
    RedirectView redirectView = new RedirectView();
    redirectView.setUrl("/");
    return new ModelAndView(redirectView);
}

```



```

//////////--login page--//////////
@GetMapping("/login")
public ModelAndView loginPage() {
    ModelAndView mav = new ModelAndView("user/login");
    return mav;
}

//////////--generate pdf file with jasperReport--//////////

@Autowired
ApplicationContext context;
@GetMapping(path = "/pdf")
@ResponseBody
public void getPdf(HttpServletResponse response) throws Exception
{
    //Get JRXML template from resources folder
    Resource resource =
context.getResource("classpath:userDetails.jrxml");

    //Compile to jasperReport
    InputStream inputStream = resource.getInputStream();
    JasperReport report =
JasperCompileManager.compileReport(inputStream);
    //Parameters Set
    Map<String, Object> params = new HashMap<>();
    //Data source Set
    List<Map<String, Object>> users = (List<Map<String, Object>>)
userService.getAllUsers();
    JRDataSource dataSource = new
JRBeanCollectionDataSource(users);
    params.put("datasource", dataSource);
    //Make jasperPrint
    JasperPrint jasperPrint =
JasperFillManager.fillReport(report, params, dataSource);
    //Media Type
    response.setContentType(MediaType.APPLICATION_PDF_VALUE);
    //Export PDF Stream
    JasperExportManager.exportReportToPdfStream(jasperPrint,
response.getOutputStream());
}

//////////

```

CODE 5: USER_CONTROLLER

E) COUCHE SECURITY

Dans ce package on trouve les class suivant (voir figure 6 : PACKAGE_SECURITY), voici un exemple de l'une des ces classe SecurityConfiguration le main classe de configuration de security qui utilise les autre classe comme MyUserDetails, MyUserDetailsService, RBAC (CODE 6: SECURITYCONFIGURATION)



FIGURE 6: PACKAGE SECURITY

```
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter
{
    private MyUserDetailsService myUserDetailsService;

    public SecurityConfiguration(MyUserDetailsService myUserDetailsService)
    {
        this.myUserDetailsService = myUserDetailsService;
    }

    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) {
        auth.authenticationProvider(authenticationProviderBean());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/login").permitAll()
            .antMatchers("/css/**", "/js/**", "/images/
            **").permitAll().antMatchers("/login",
            "/forgot_password", "/reset_password").permitAll()
            .anyRequest().access("@rbacService.hasPermission(request, a
            uthentication)") // call Rbac class
            .and().formLogin().loginProcessingUrl("/signin")
            .loginPage("/login").permitAll()
            .usernameParameter("txtUsername")
            .passwordParameter("txtPassword")
            .and().logout().logoutRequestMatcher(new
            AntPathRequestMatcher("/logout")).logoutSuccessUrl("/login")
            .and().rememberMe().tokenValiditySeconds(2592000).key("m
            ySecret!").userDetailsService(userDetailsService)

            .and().exceptionHandling().accessDeniedPage("/accessDenied"); ;
    }

    @Bean
    DaoAuthenticationProvider authenticationProviderBean() {
        DaoAuthenticationProvider daoAuthenticationProvider=new
        DaoAuthenticationProvider();
        daoAuthenticationProvider.setPasswordEncoder(passwordEncoder());
        daoAuthenticationProvider.setUserDetailsService(this.userDetailsService);
        return daoAuthenticationProvider;
    }

    @Bean
    PasswordEncoder passwordEncoder() {return new BCryptPasswordEncoder();}
```

CODE 6: SECURITYCONFIGURATION

2- FRONT END

A) INTEGRATION DE TEMPLATE

A.1) INTÉGRATION DE TEMPLATE

Dans ce projet on a :

- choisi ce Template (à partir de site <https://colorlib.com/etc/bootstrap-sidebar/sidebar-01/>)

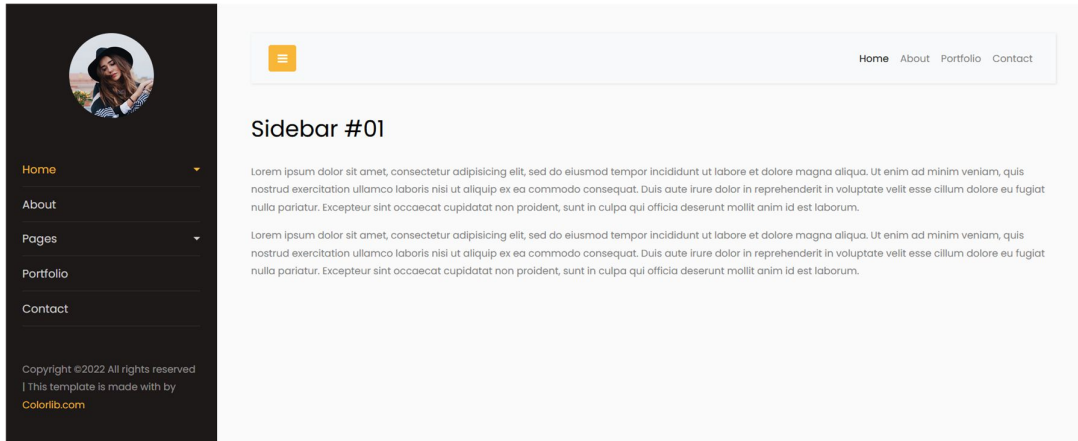


FIGURE 7: TEMPLATE

- Télécharger le .rar et extraire, couper les ressources (les dossier javascript, css , bootstrap, images) et le placé dans la dossier src/main/Ressource/Static
- Copier l'index.html
- Diviser le en différent fragments avec `th:fragment` et `th:block` fourni par thymeleaf.

A.2) INTÉGRATION DES FRAGMENT À PARTIR DE TEMPLATE

Avec thymeleaf il y a la possibilité de diviser la contenu d'un page html sous des fragments et l'appelé ces fragment dans les différents pages de notre application pas de répétition de code. Pour ce projet on a divisé l'index.html en trois statiques et une dynamique.

Le fragment header, js, sidebar sont statique ne change pas et un dynamique c'est-à-dire on peut le changer son contenu qui est le fragment contents.

Donc il suffi de récrire seulement l'appelé a ces fragments dans n'importe quel .html de notre application (voir CODE 7: FRAMENTS_THYEMLEAF) ces ligne de code seront appeler au différent page de notre vue seulement qui change c'est le contenu de fragment contents.

```

<!DOCTYPE html>
<html lang="en"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="fragments/adminTemplate.html" >
<head th:insert="fragments/adminTemplate.html :: headfragment ">
<!-- the headFragment call-->

</head>
<body>
<th:block th:insert="fragments/adminTemplate.html::nav"><!-- the
sidebar call-->

    <div layout:fragment="content">
        <!-- page content -->
        here we can put what we want, this is the dynamic part of our
        template.
        <!-- end page content -->
    </div>
</th:block>
<div th:insert="fragments/adminTemplate.html ::js"></div><!-- the js
script call-->
</body>
</html>

```

CODE 7: FRAMENTS_THYEMLEAF

B) L'INTERFACE GESTION DES DÉPARTEMENT

Dans l'interface Admin de gestion des départements on trouve la liste du département avec les différentes informations comme l'email de responsable et numéro de téléphone et aussi l'action qu'en souhaite applique sur ce département (ajout, suppression et modification)

List des départements dans un tableau avec la pagination (cinq ligne dans un seul de tableau dans un seul page avec des boutons fait avec font awesome au-dessous de table qui nous permettant de passer d'un page à un autre ou d'aller au premier ou la dernier page (la pagination aussi présent dans la table user et la table des rôles).

Name	Email of the responsible	Phone	Actions
admin	admin@admin.com	21456879	
Rh	Rh@rh.com	21431055	
Finance	finance@finance.com	21431055	
IT	it@it.com	21431055	
Management	manger@manger.com	21431055	

Total Items 7 : Page 1 of 2
 < 1 2 >

FIGURE 8: DEAPRTEMNT_TABLE_FIRST_PAGE

Name	Email of the responsible	Phone	Actions
Sales	sales@sales.com	21431055	
Marketing	marketing@marketing.com	21431055	

Total Items 7 : Page 2 of 2
 < 1 2 >

FIGURE 9: SECONDE_PAGE_OF_TABLE

Ajout d'une nouvelle département avec l'utilisation de Model de Bootstrap et aussi le contrôle de sassai avec jQuery.

New department ✕

Name of Department:
 ✓

Email of responsible:
 ✕
 Please provide a valid email address.

Phone of responsible:
 ✕
 phone number field cannot be empty.

Close Save

FIGURE 10: ADD NEW DEPRATMENT WITH INPUT CONTROLLE IN CASE OF FAIL

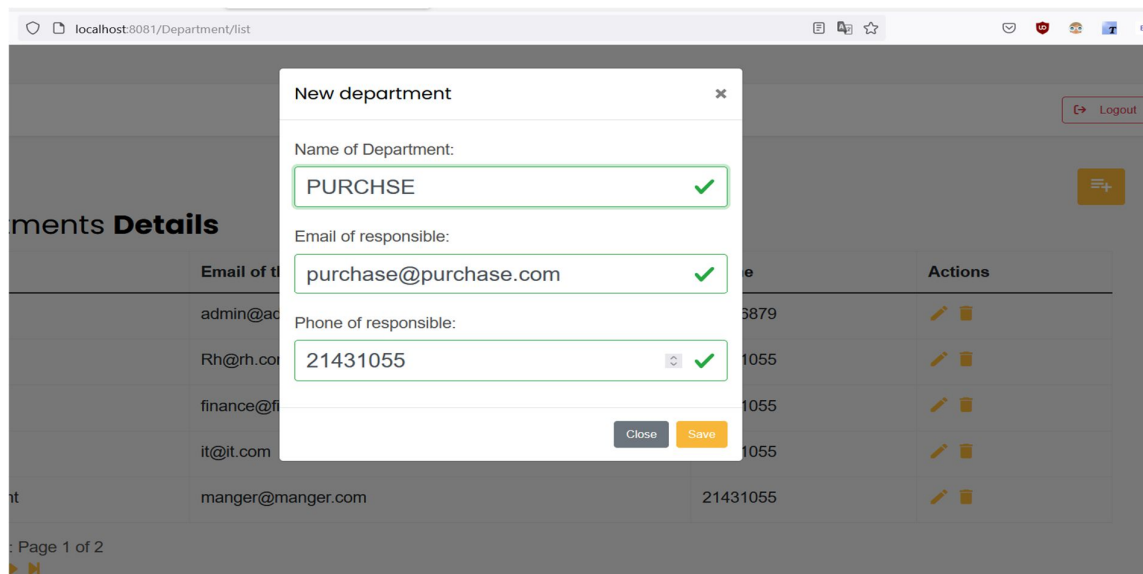


FIGURE 11: IN CASE OF SUCCESS

C) L'INTERFACE GESTION DES UTILISATEUR

Please Login

Email :

Email

Password :

Password

☐ Remember me?

Log in

[Forgot your password?](#)

FIGURE 12: LOGIN_PAGE

mahmoud timoumi
(ROLE_TEST, ROLE_ADMIN)
Manage Users
List of employees
Add new User
Role
Function
Department
Parent One
Parent two
Parent three
Parent four

Logout

User Details

Full name	Email	phone	department	hiring date	birth date	address	role	Action
mahmoud mahmoud	mahmoud.timoumi@esprit.tn	21431054	admin	2022-09-05	1997-10-15	Ariana, Raouedd	ADMIN TEST	
admdds admdds	admin@admin.com	21431055	admin	2022-08-30	2022-08-29	Jaafers	ADMIN TEST	
test test	ali@ali.com	21431055	admin	2022-08-07	2022-10-05	yhtgfrds		
test test	test@test.com	555555554	admin	2022-08-31	2022-08-30	gtfrdes		

Total users 4 : Page 1 of 1

FIGURE 13: MANAGE_USERS

Please provide correct information :

Full Name * :

Email * :

Password * :

Phone number*:

Gender * :

☐ Male
☐ Female

Birth Date * :

Hire date * :

Adresse * :

Adresse * :

Complete address ...

Departemnt * :

-----Chose----- ▾

Roles :

☐ ADMIN
☐ TEST

Submit

FIGURE 14: ADD_NEW_USER_FORM

Please provide correct information :

Full Name * :

first name ...

last name ...

last name cannot be empty
first name cannot be empty

Email * :

valid email address ...

Email cannot be empty
Email is not valid

Password * :

A solid password more than six characters ..

Password must be greater than Six characters

Phone number*:

phone number ...

size must be between 8 and 12
mobileNumber is required

Gender * :

☐ Male
☐ Female

please provide a gender

Birth Date * :

mm/dd/yyyy

Hire date * :

mm/dd/yyyy

Adresse * :

Complete address ...

address cannot be empty

Departemnt * :

-----Chose----- ▾

Roles :

☐ ADMIN
☐ TEST

Submit

FIGURE 15: INPUT_VALIDATION

Logout

Edit user information ::

Back to users List

Monsieur

: admdds adminzzz

Phone

: 21431055

Email

: admin@admin.com

Address

: Jaafers

Department

: admin

birth date

: 2022-08-29

hiring date

: 2022-08-30

Created

: at 2022-09-02 16:50:47.0 by mahmoud.timouni@esprit.tn

last modification

: at 2022-09-03 13:13:51.0 by mahmoud.timouni@esprit.tn

Full Name * :

admdds

adminzzz

Email * :

admin@admin.com

Phone number*:

21431055

Gender * :

☒ Male

☐ Female

Birth Date * :

08 / 29 / 2022

Hire date * :

08 / 30 / 2022

Adresse * :

Jaafers

Roles :

☒ ADMIN

☒ TEST

Update

FIGURE 16: UPDATE_USER

Des pop-up après succès dans l'ajout et la modification, aussi un pop up avant la suppression pour la confirmation.(FIGURE 17: POP_PUP_DELETE_CONFERMATION et FIGURE : 18 POP_PUP_DELETE_CONFERMATION) (ces pop-up sont présent aussi dans les autre interface Function, département et rôle).

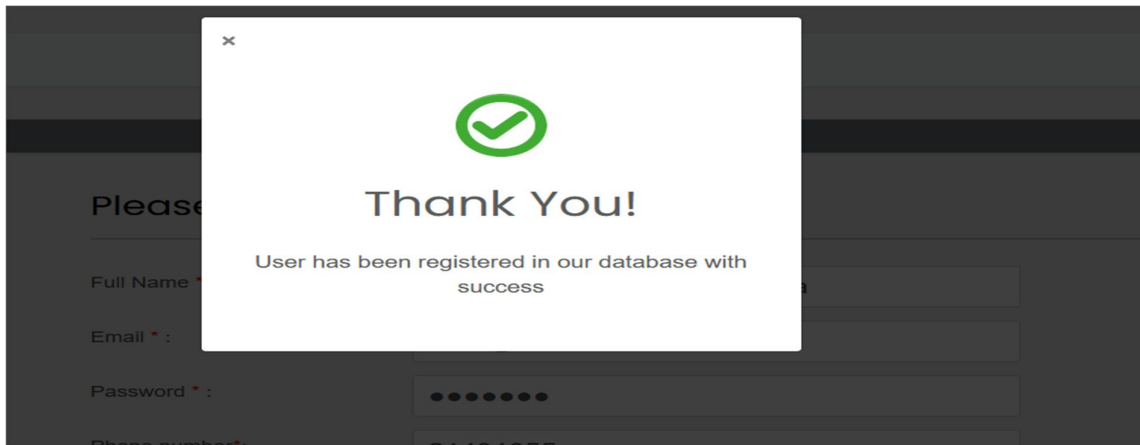


FIGURE 17: POP_UP_ADD/UPDATE

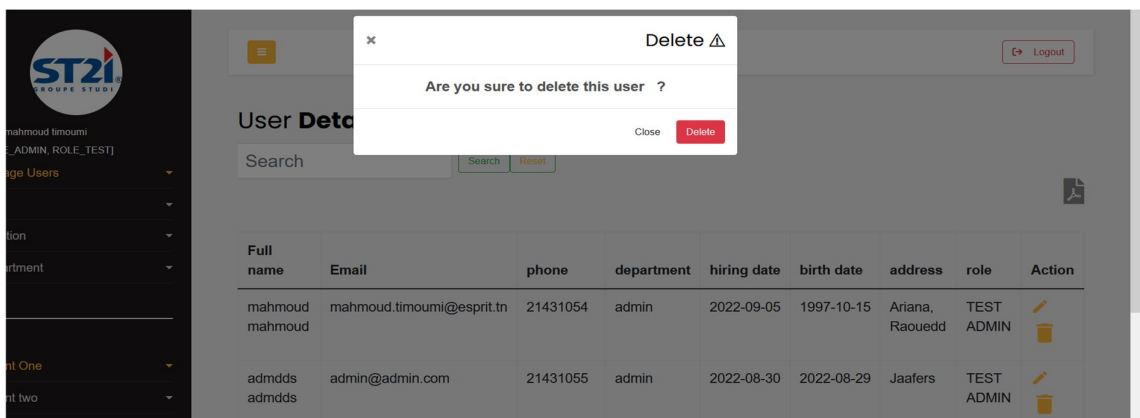


FIGURE 18: POP_PUP_DELETE_CONFIRMATION

Plus des quatre fonctions CRUD on a ajouté la fonctionnalité de mot de passe oublier au-dessous des capteurs de processus de réinitialisation de mots de passe.

Forgot your password?

Change your password in three easy steps. This will help you to secure your password!

1. Enter your email address below.
2. Our system will send you a temporary link
3. Use the link to reset your password

Enter your email address

Enter your email address . Then we'll send email which contains a link to this address.

Get New Password
Back to Login

FIGURE 19: FORGET_PASSWORD_FORM

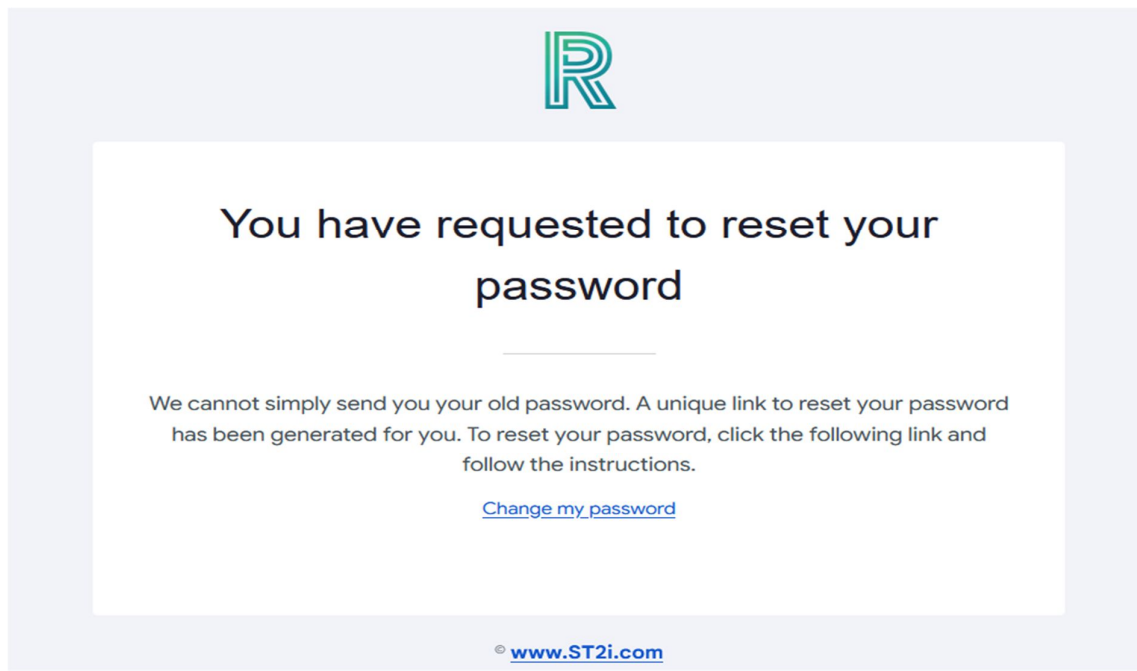


FIGURE 20: RESET_PASSWORD_LINK_EMAIL

The image shows a web form titled 'Reset Your Password' in a bold, black font. The form is contained within a light gray rectangular box. Inside this box, there are two white input fields. The first field is labeled 'Enter your new password' and the second field is labeled 'Confirm your new password'. Below these two fields is an orange button with the text 'Change Password' in white.

FIGURE 21: RESET_PASSWORD_FORM

D) L'INTERFACE GESTION DES ROLES

Add a new role::

Role name * :

Description of the role * :

Manage Users ☒

Manage Roles ☒

- update a role ☐
- add new role ☒
- list of roles ☐
- delete a role ☒

Manage function ☐

Manage Department ☐

dep ☒

- list dept ☒

Add

FIGURE 22: ADD_NEW_ROLE

E) L'INTERFACE GESTION DES FUNCTIONS

La List des fonctions (ressources) sous forme d'un arbre interactif (avec javascript) si on click sur la fonction père les fils apparaissent, après si on click une autre fois sur la même fonction la liste des enfants disparaît.

Aussi on trouve la différente action (suppression, modification)

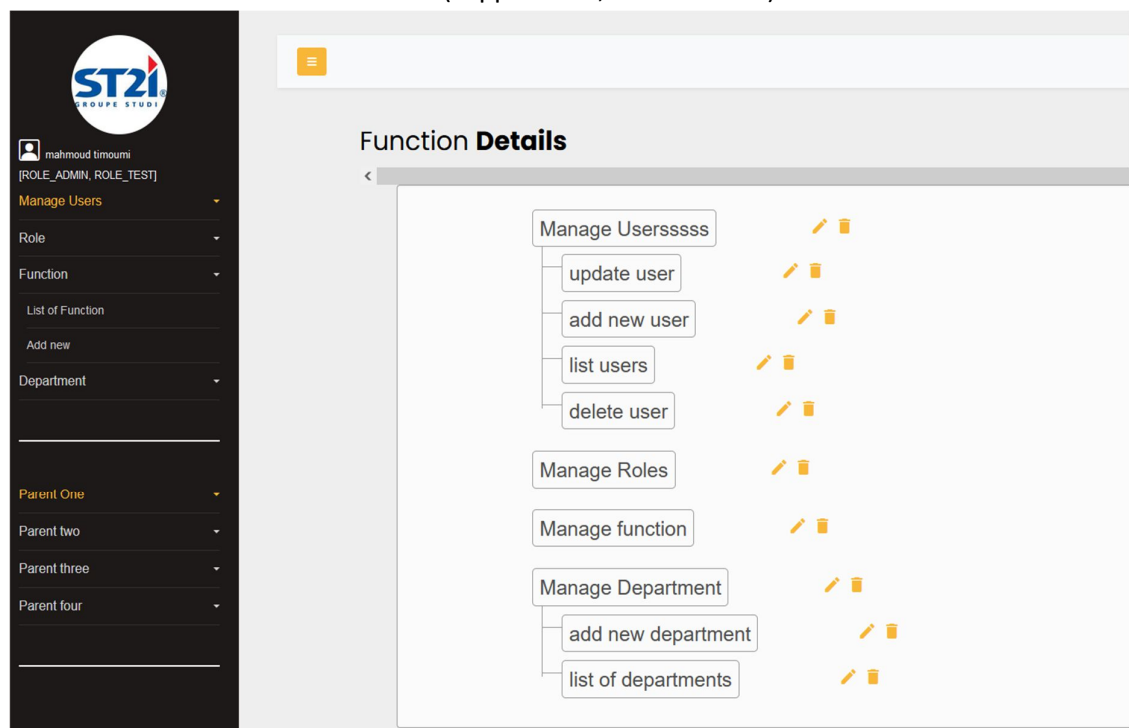


FIGURE 23: MANAGE_FUNCTION

IV- CONCLUSION

Je tiens avant tout à remercier l'équipe et mon encadrant qui m'a vraiment très bien accueilli durant ces semaines. Il a toujours été présente lorsque je rencontrais des problèmes, et toujours prêt à répondre à mes questions.

Ce stage a parfaitement répondu à mes attentes car je souhaitais découvrir des nouvelles technologies. Il m'a permis de découvrir un univers que je ne connaissais beaucoup mais pour lequel je porte un immense intérêt.

Ce stage a vraiment confirmé mes ambitions futures d'exercer dans le domaine de développement des applications web, même s'il me reste encore beaucoup à apprendre.

TABLE DES FIGURE ET DES CODE

Figure 1: Société ST2i	2
Figure 2:Rbac	4
Figure 3:use_case_diagramme	6
Figure 4:class Diagramme	7
Figure 5:connection avec mysql	12
Figure 6: package Security	22
Figure 7: template	23
Figure 8: Deaprtemnt_table_first_page.....	25
Figure 9: seconde_page_of_table	25
Figure 10: add new depratment with input controle in case of fail	25
Figure 11: in case of success.....	26
Figure 12: login_page	26
Figure 13: Manage_users.....	27
Figure 14: add_new_user_Form	28
Figure 15: input_validation	28
Figure 16: update_user.....	29
Figure 17: pop_up_add/update.....	30
Figure 18: pop_pup_delete_confermation	30
Figure 19: forget_password_form.....	30
Figure 20: reset_password_link_email	31
Figure 21: reset_password_form.....	31
Figure 22: Add_new_role.....	32
Figure 23: manage_function.....	32
CODE 1: USER_CLASS.....	14
CODE 2: Roles_class	15
CODE 3: function_class	15
CODE 4: USER_SERVICE CLASS	18
CODE 5: USER_CONTROLLER	21
CODE 6: SECURITYCONFIGURATION	22
CODE 7: framents_thyemleaf	24



ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siège Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : 1-2 rue André Ampère - 2083 - Pôle Technologique - El Ghazala - Tél +216 70 250 000 - Fax +216 70 685454