

CJ-Sécurité Privée

Dossier Projet

Sommaire

I - Présentation personnelle	3	III - Prestation attendues	16
		Charte graphique et charte éditoriale	16
		Maquettage	20
		Développement	23
II - Présentation de mon projet	4	IV - Remerciements	34
Contexte	4		
Les cibles	5		
Concurrences et positionnement	9		
Objectif du site	12		
Arborescence	13		
Fonctionnalités	14		

Qui suis-je?

Je m'appelle Gaël POUGETOUT j'ai 27 ans. J'aime la boxe et l'airsoft.

Bac techno STL PLPI

Gendarme Adjoint Volontaire

Policier municipal

Simplon

2011

2012-2017

2017-2018

2018-2019

Contexte

Pour une société de sécurité il peut être parfois difficile de recruter des agents afin de pouvoir accepter de nouveau contrat. Elle doit gagner en visibilité afin de se faire connaître.

J'ai rencontré le responsable d'une entreprise de sécurité qui travaille à l'échelle nationale avec différent partenaire. Cette société se nomme CJ-Sécurité Privée.



Nous avons discuté ensemble et il m'a demandé de lui faire un site internet car il n'en avait pas alors que ses principaux concurrents ont tous un site internet. Je lui ai demandé quels étaient ses attentes d'un site. Il m'a alors exprimé son souhait d'accroître sa visibilité, d'augmenter le nombre d'agents de sécurité qui travaille dans son entreprise afin de pouvoir accepter de nouveau contrat et permettre à ses clients de demander des devis en ligne.

Les cibles

Les utilisateurs et visiteurs

CJ-Sécurité Privée vise avant tout des professionnels de la sécurité qui permettrons à l'entreprise de grandir et d'évoluer.

Les profils principaux seraient des agents de sécurité expérimentés à la recherche de mission à la hauteur de leur ambition ; des anciens gendarmes, policiers et militaires à la recherche d'une reconversion professionnelle dans le secteur de la sécurité.

Les profils secondaires seraient des chefs d'entreprise, gérant de magasin, directeur de magasin de luxe ou particulier voulant bénéficier d'une ou plusieurs prestations proposées par la société CJ-Sécurité Privée.

Les cibles

Rôle: Visiteur



Léa

Statut : Responsable des partenariats chez Total.

Age : 20 ans

Action : Rechercher un nouveau partenaire

Besoin : Elle voudrait trouver une société de sécurité capable de sécuriser tous les événements proposés par son entreprise.

Scénario : CJ-Sécurité Privée met à disposition ses tarifs ainsi que leur numéro de téléphone afin de pouvoir prendre contact.

Les cibles

Rôle: Utilisateur



Boris

Statut : Agent de sécurité cynophile

Age : 30 ans

Action : Rechercher un emploi

Besoin : Il voudrait trouver une meilleure opportunité.

Scénario : CJ-Sécurité Privée l'aidera dans sa recherche en mettant à sa disposition un formulaire de candidature.

Les cibles

Rôle: Utilisateur



Mathieu

Statut : PDG

Age : 41 ans

Action : Établir un devis

Besoin : Il souhaiterait bénéficier d'une protection rapprochée après avoir subis une menace de mort.

Scénario : CJ-Sécurité Privée met à sa disposition la possibilité d'établir un devis en ligne.

Concurrences et positionnement



➞ Charte graphique :

RMS utilise du bleu, du blanc et du noir.

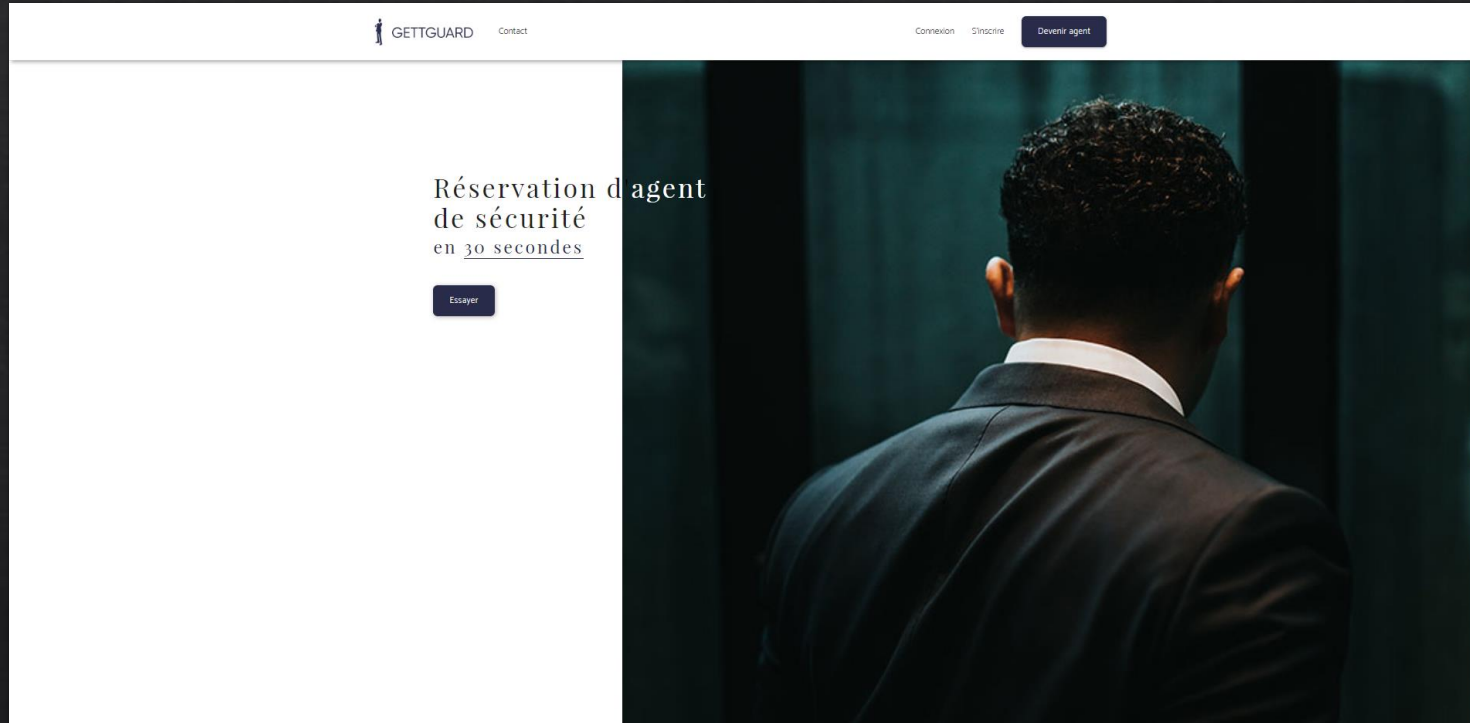
➞ Les plus:

RMS rassure le client, on peut faire des devis en ligne et il y a de jolies animations.

➞ Les moins:

Aucune possibilité de recrutement, redondance entre services et nos métiers ex : vigile magasin.

Concurrences et positionnement



➡ Charte graphique :

GUETTGUARD utilise du bleu foncé, blanc et noir.

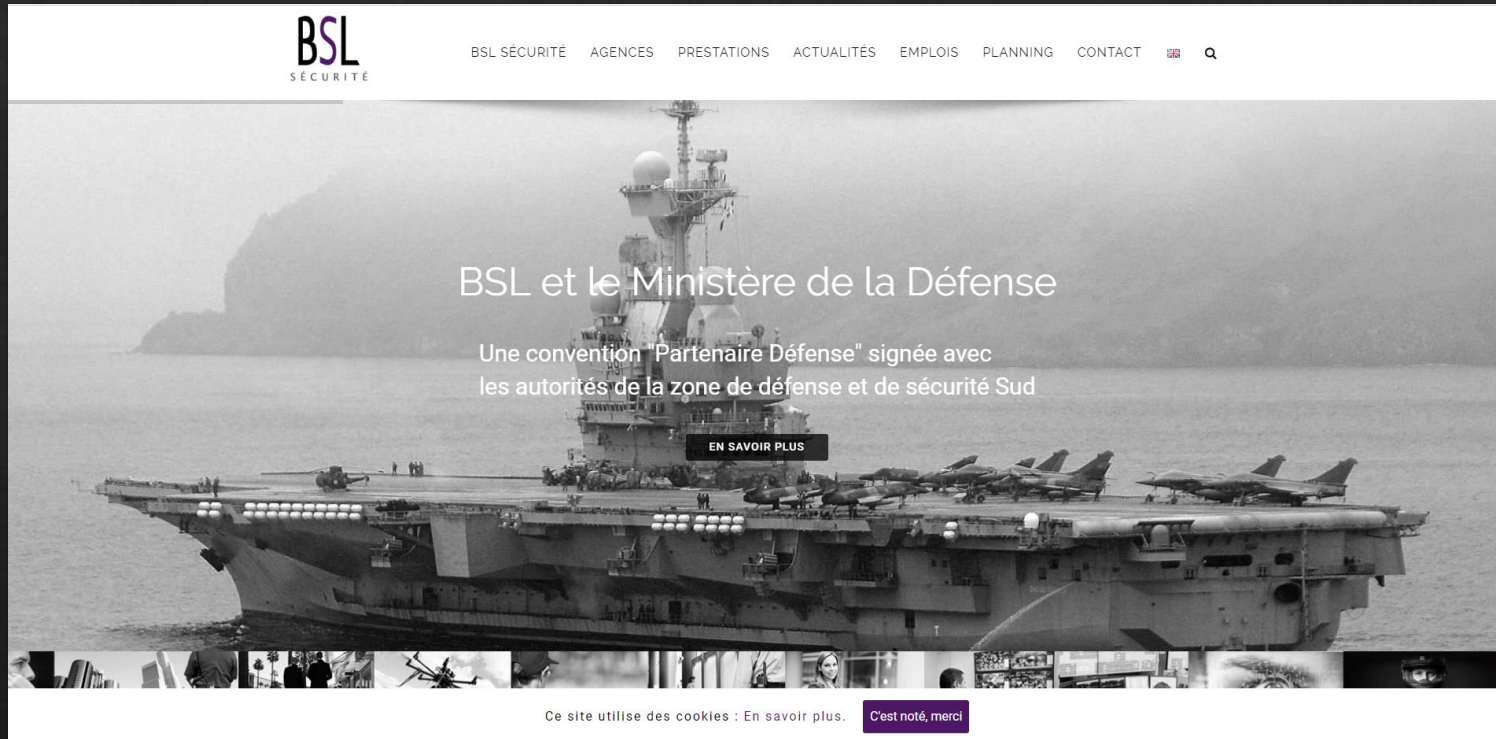
➡ Les plus:

Visibilité sur les prix/domaine de compétence, possibilité de recrutement en ligne, recommandation dans le footer donc visible sur toutes les pages. Header devenir agent, devenir client devenir partenaire.

➡ Les moins:

Le footer est mal réparti.

Concurrences et positionnement



➡ Charte graphique :

BSL utilise du blanc, noir et du violet.

➡ Les plus:

Traduction en anglais possible.
Première page : partenaire, activité.
Possibilité de consulter son planning en ligne et en temps réel.

➡ Les moins:

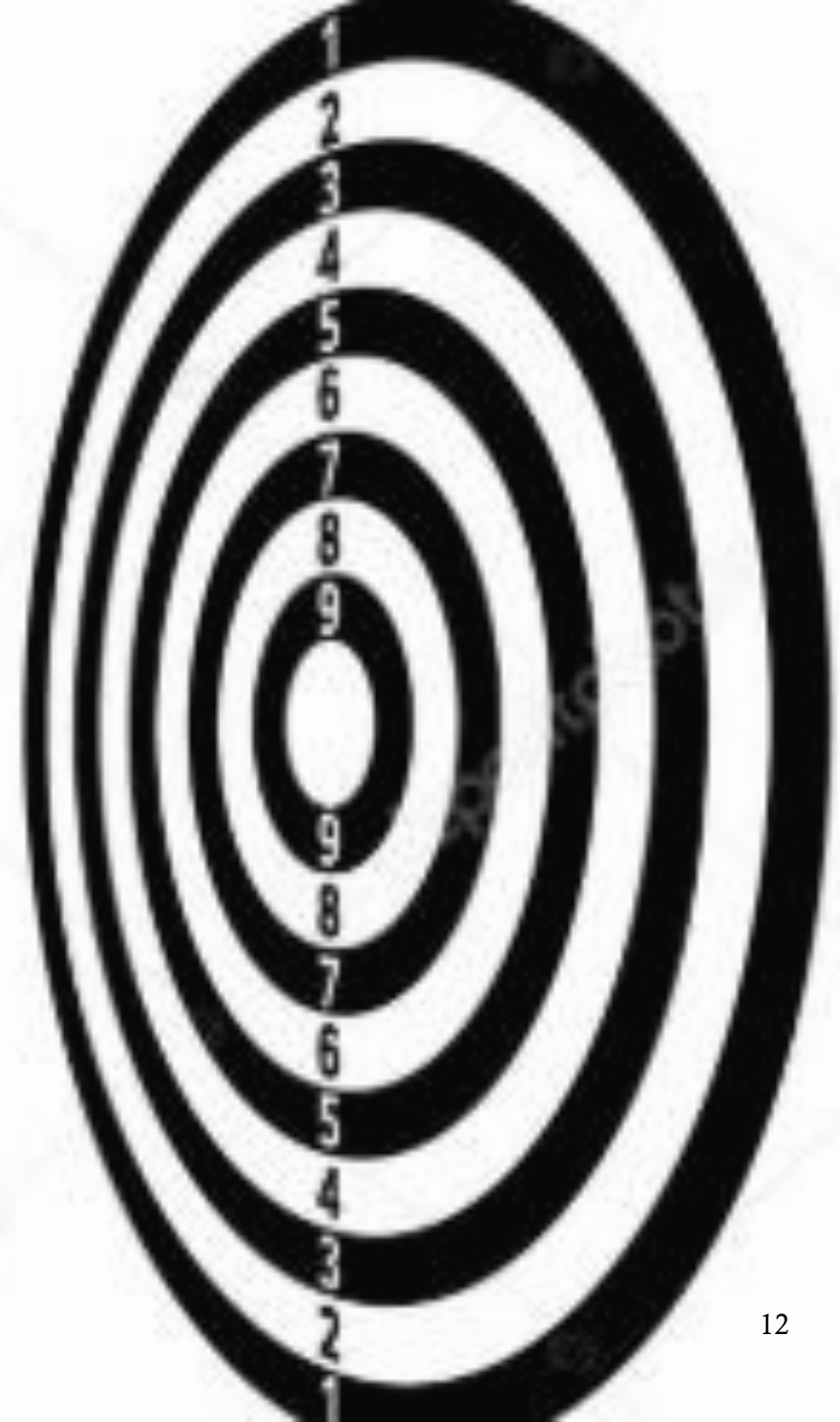
Les mots sont mal choisis pour parler des métiers. Le footer est surchargé d'information.

Objectif du site

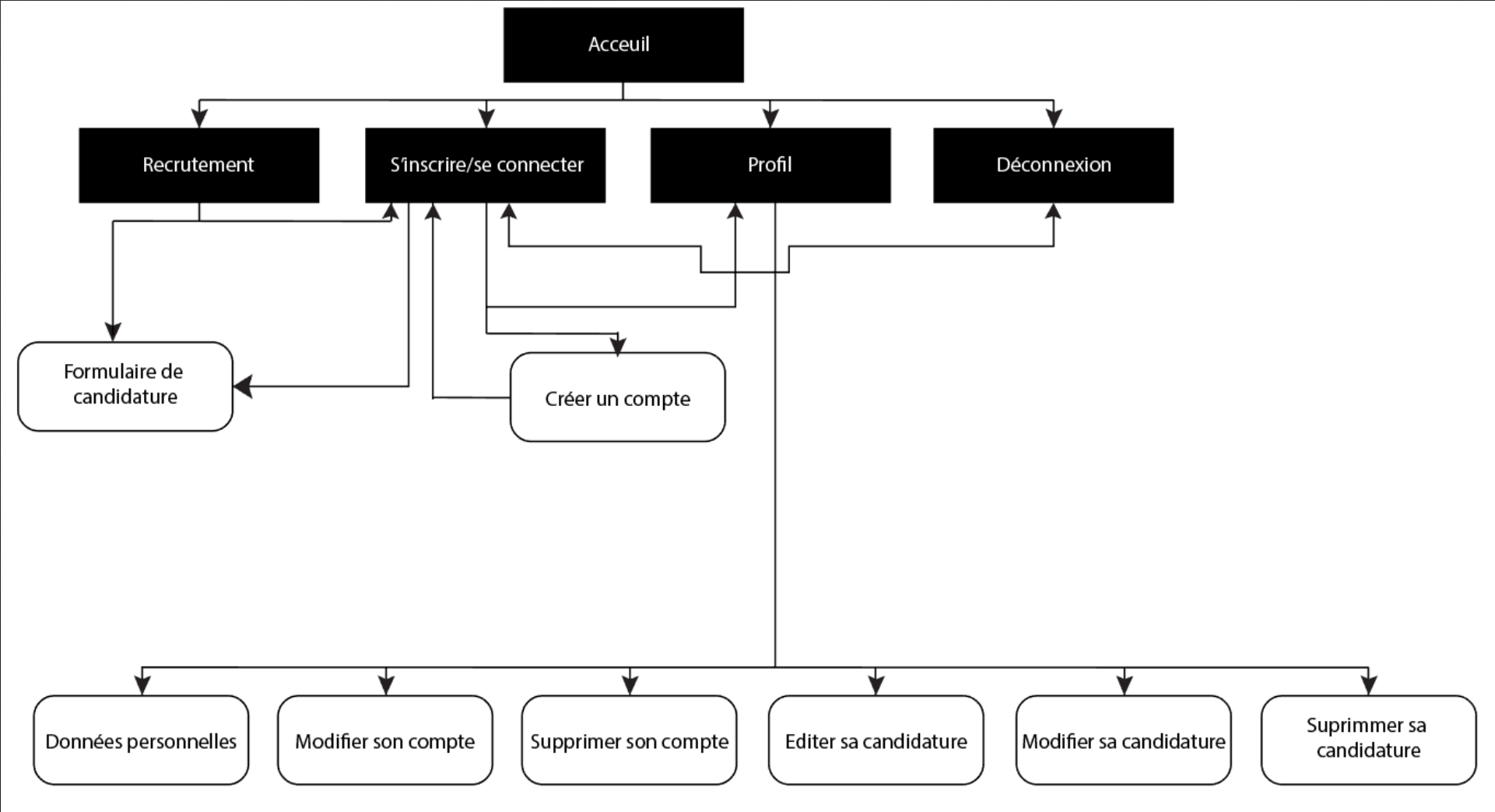
Permettre à CJ-Sécurité Privée de recruter plus d'agents de sécurité.

Accroître la visibilité de l'entreprise à l'échelle nationale.

Permettre à ses clients de contracter des devis en lignes.



Arborescence

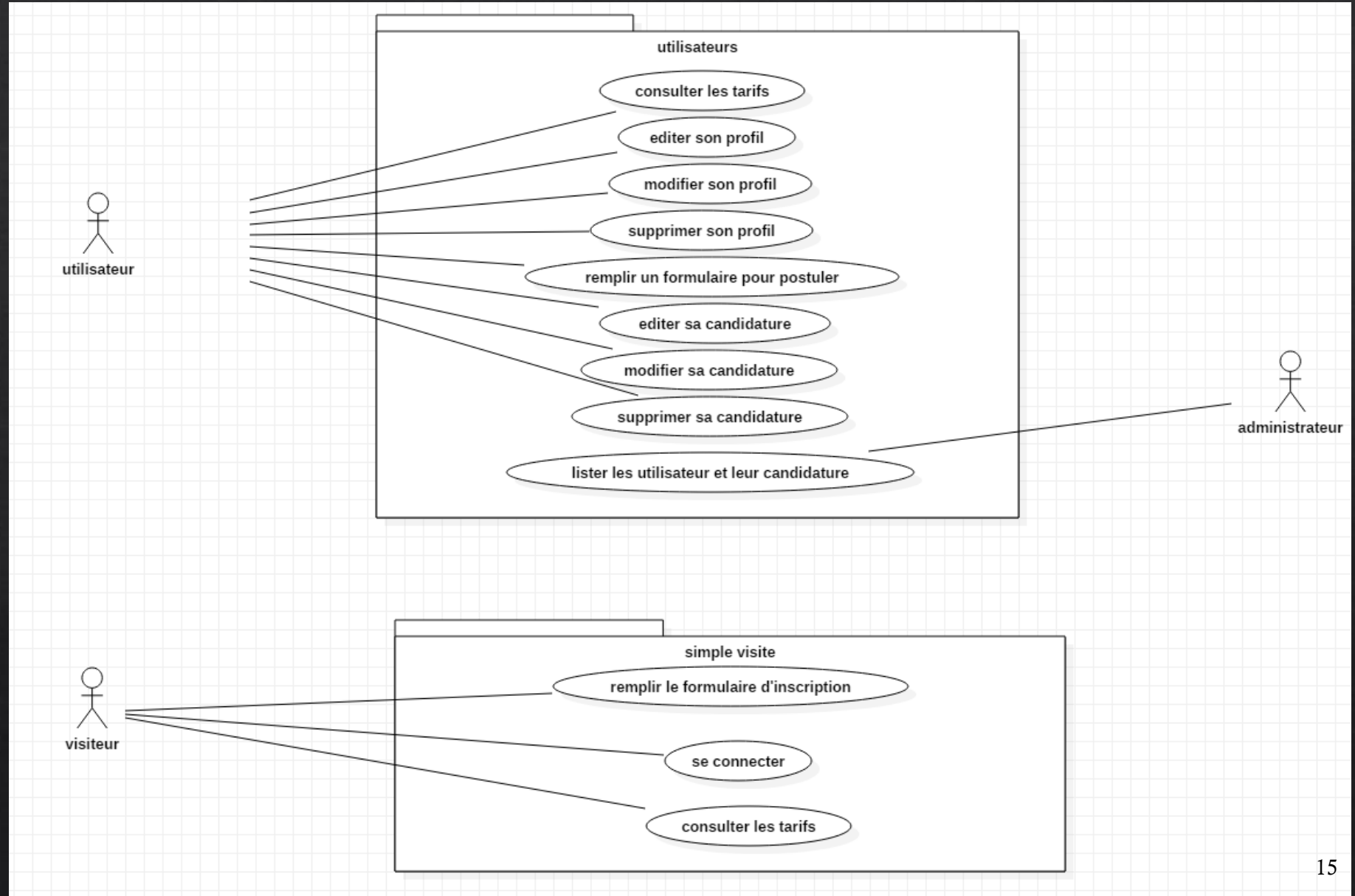


Fonctionnalités


- Création de compte utilisateur
- Espace utilisateur accessible par mot de passe
- Édition de son profil
- Modification de son profil
- Suppression de son profil
- Postuler
- Éditer sa candidature
- Modifier sa candidature
- Supprimer sa candidature

Fonctionnalités

Use Case



Charte graphique et charte éditoriale



Sécurité
Armes
Communication
Service
Discrétion

Protection
Surveillance

Sûreté
Prestation

Médiation
Prevention

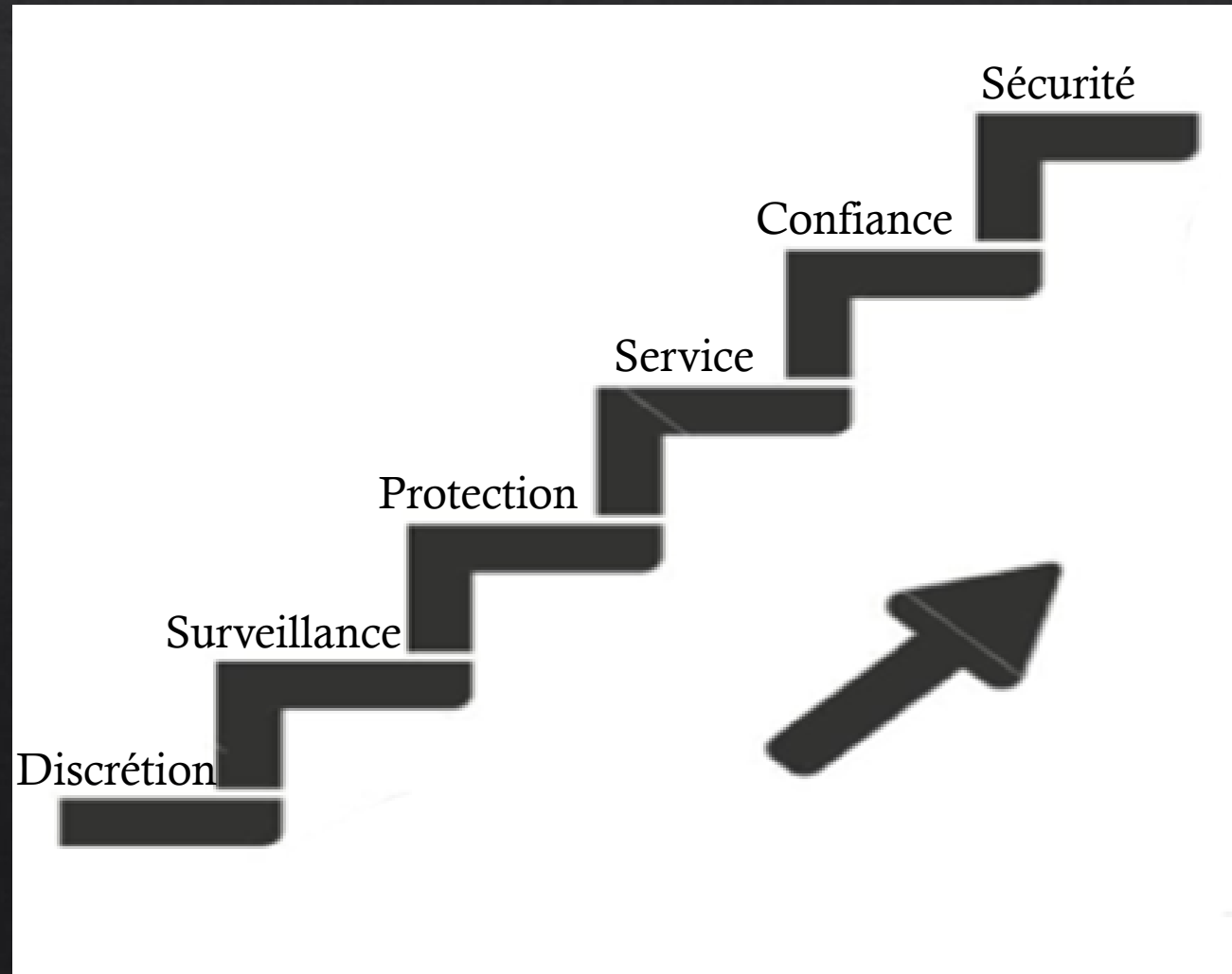
Equipe
Client

Agent
Confiance

Brainstorming :

Charte graphique et charte éditoriale

Echelle d'Osgood :




Le ton est donné par une
Sélection de mot clés:

[illegible]

Charte graphique et charte éditoriale

Style tile



Style tile

Adjectifs

Securitée

Confiance

Discretion

Protection

Service

Surveillance

r: 255
v: 255
b: 255

#: ffffff





r: 0
v: 0
b: 0




#: 000000



r: 68
v: 140
b: 191

#: 448cbf

Icons







Fonts

Roboto

Buttons

Default

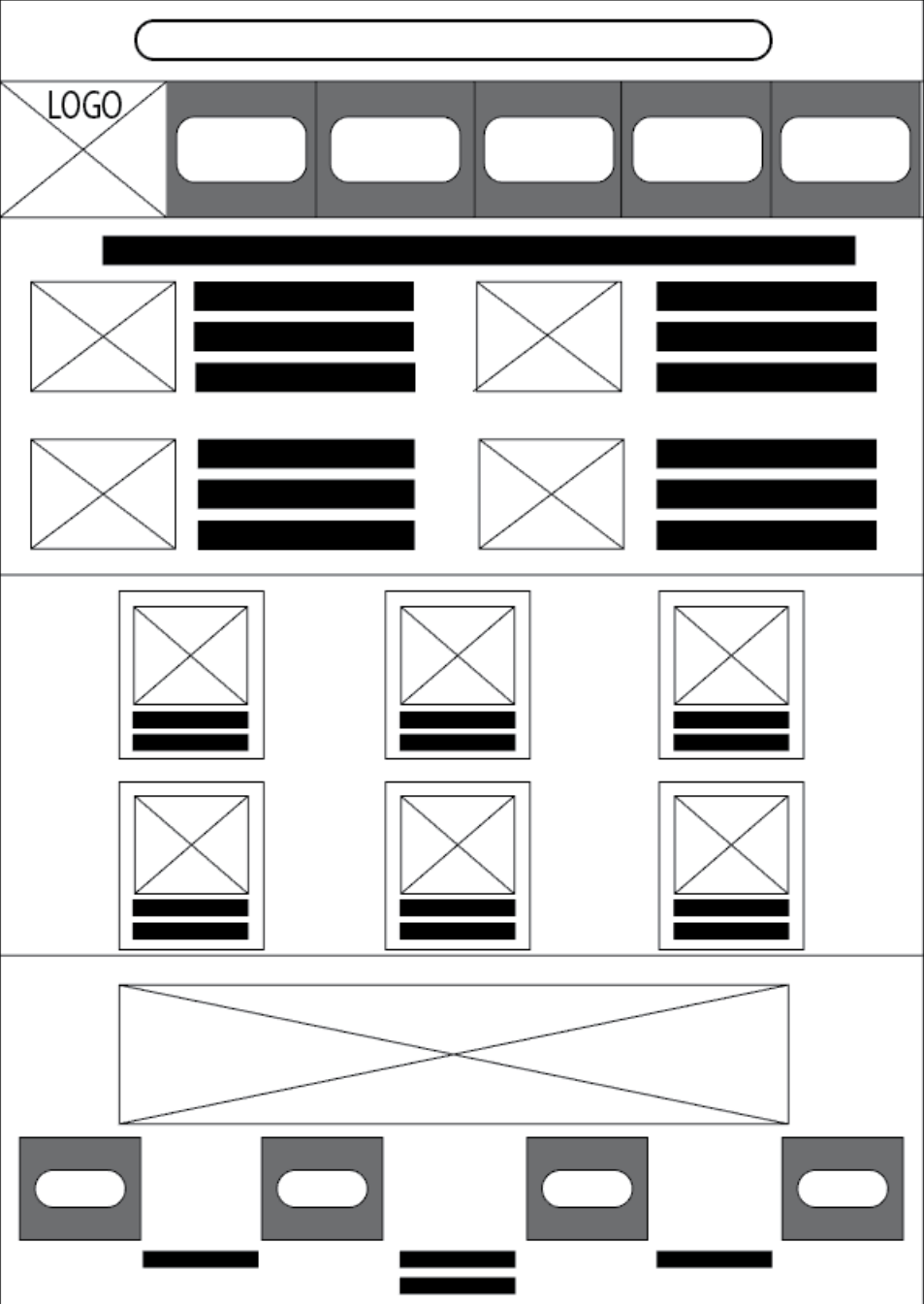
Maquettage

Zoning

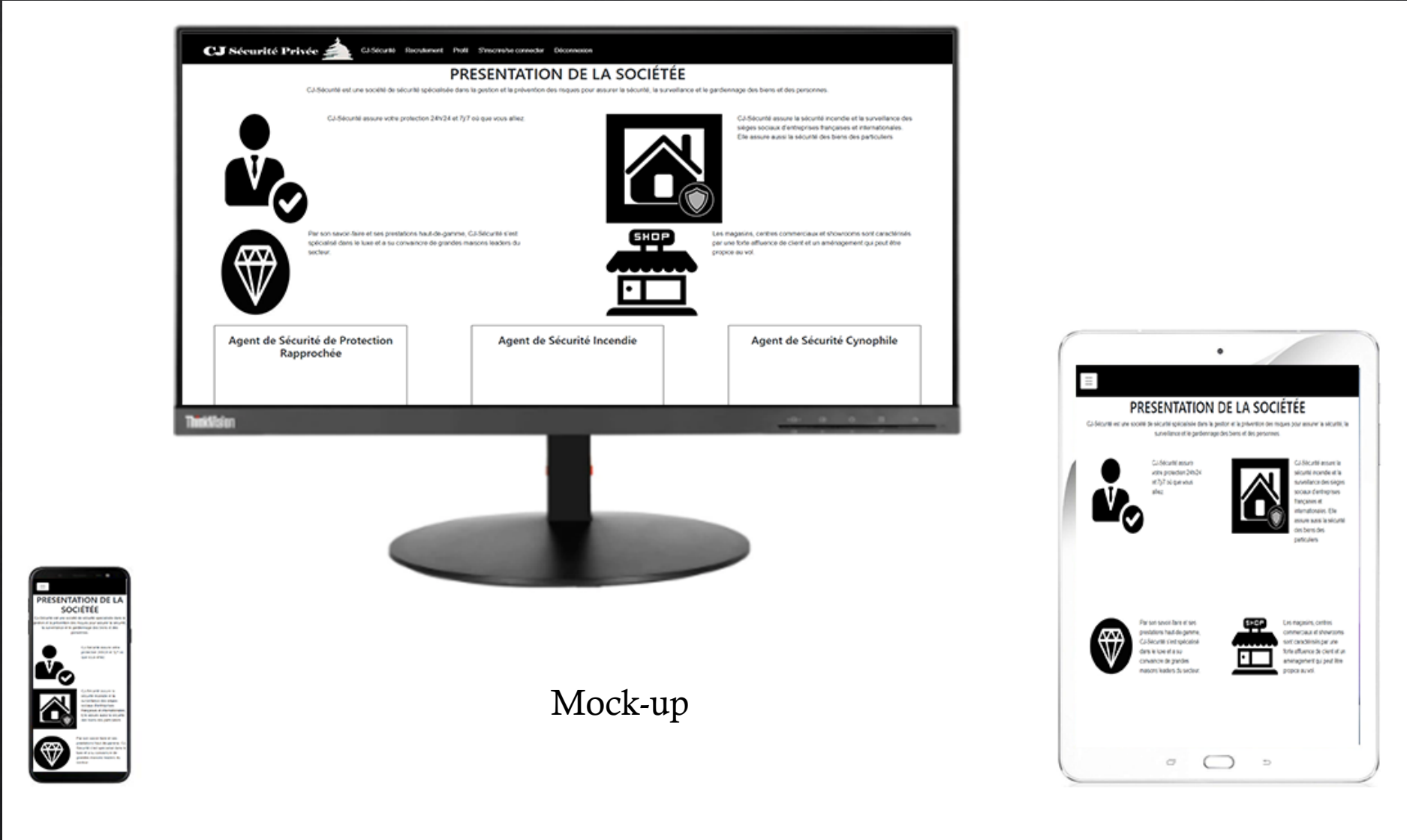


Maquettage

Wireframe



Maquettage



Mock-up

Développement

Technologie

Issue de la formation de développement full stack JavaScript/Java, ils deviennent tout deux les langages principaux utilisés pour cette plateforme.

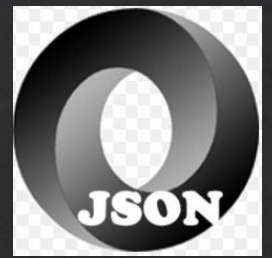
Le framework Vue.js est un framework qui s'appuie sur javascript; il permet de rendre les pages web dynamiques à travers la manipulation du DOM (le binding). Il permet de faire des requêtes et recevoir des réponses HTTP vers un serveur.

Pour le back-end, Java permet de créer un serveur simple d'utilisation et optimale pour la communication entre client/serveur et base de donnée/serveur. J'ai utilisé le framework springboot pour faciliter la configuration.

Le système de gestion de base de donnée relationnelle est MySQL car c'est le seul que j'ai vu pendant ma formation et j'ai donc plus de facilité avec.

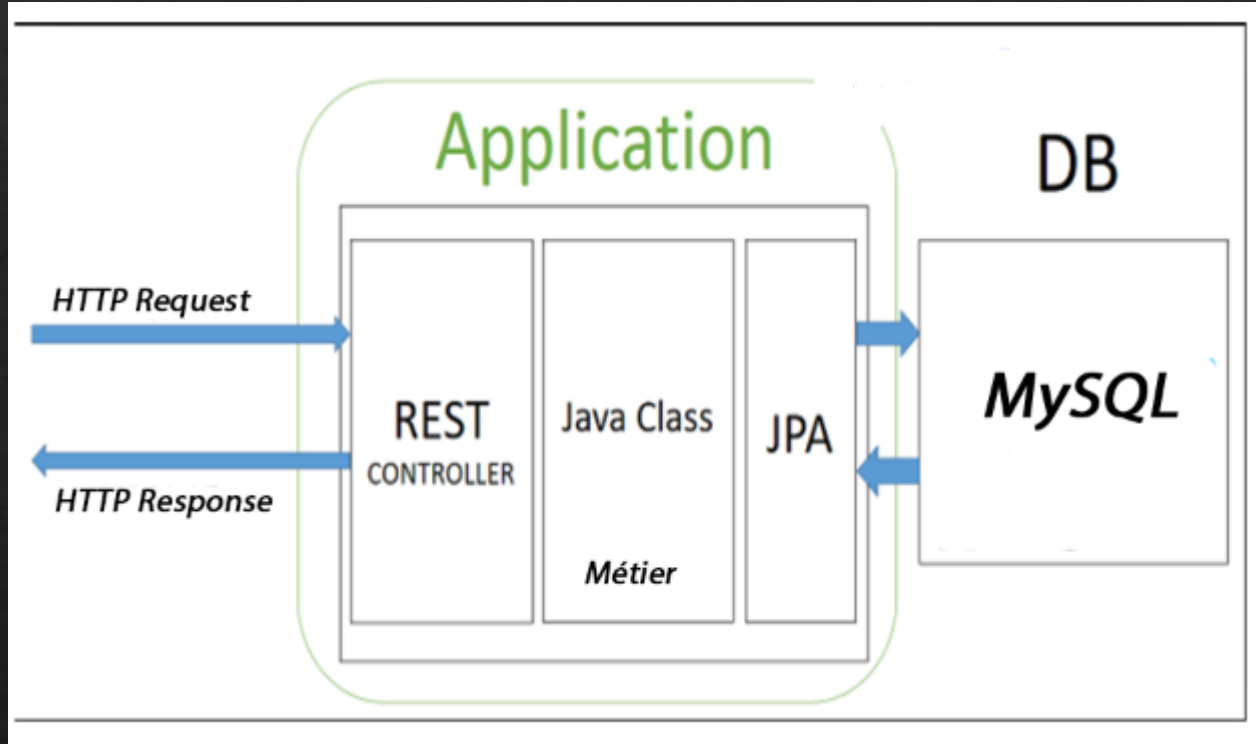
Le format Json est utilisé pour le transfert de données car il est léger, pris en charge par de nombreux langages et permet de stocker des données de différents types.

Pour sécuriser mon application j'ai choisi SpringSecurity et le Jason Web Token car ce sont les outils que je maîtrise le mieux afin de sécuriser une application.



Développement

Architecture

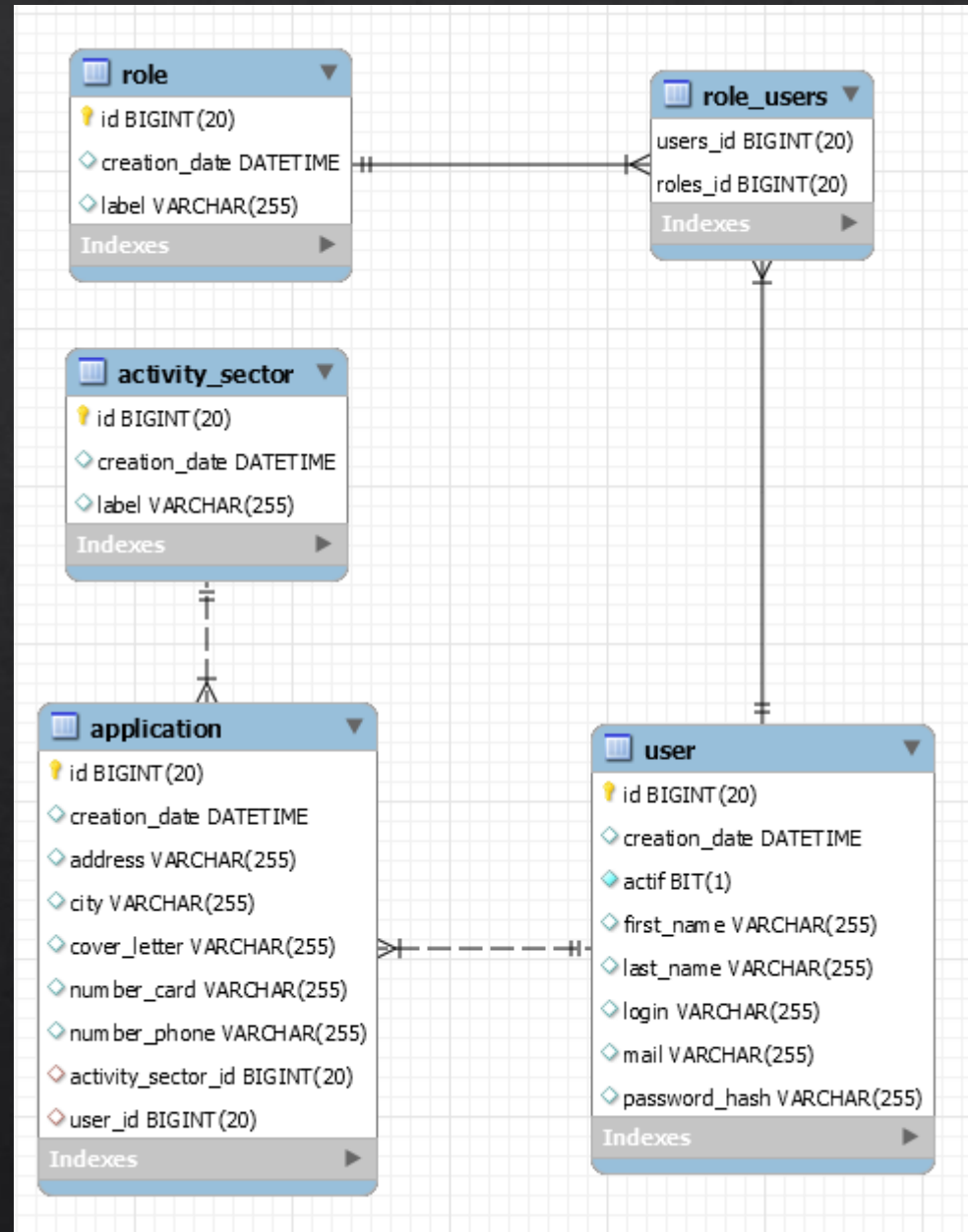


Une HTTP Request est envoyée du front-end à mon back-end, elle passe la chaîne de filtre configuré avec SpringSecurity, la requête arrive à mon REST (Representational State Transfer) Controller. Mon Controller interagit avec ma couche service. Mon service interagit avec mon repository (JPA). Mon repository interagit avec ma base de donnée. Une fois ma base de donnée sollicitée par JPA mon Controller retourne une HTTP Response.

Développement

Base de donnée

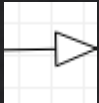
4 entités avec 1 table de jointure



Développement

Diagramme UML

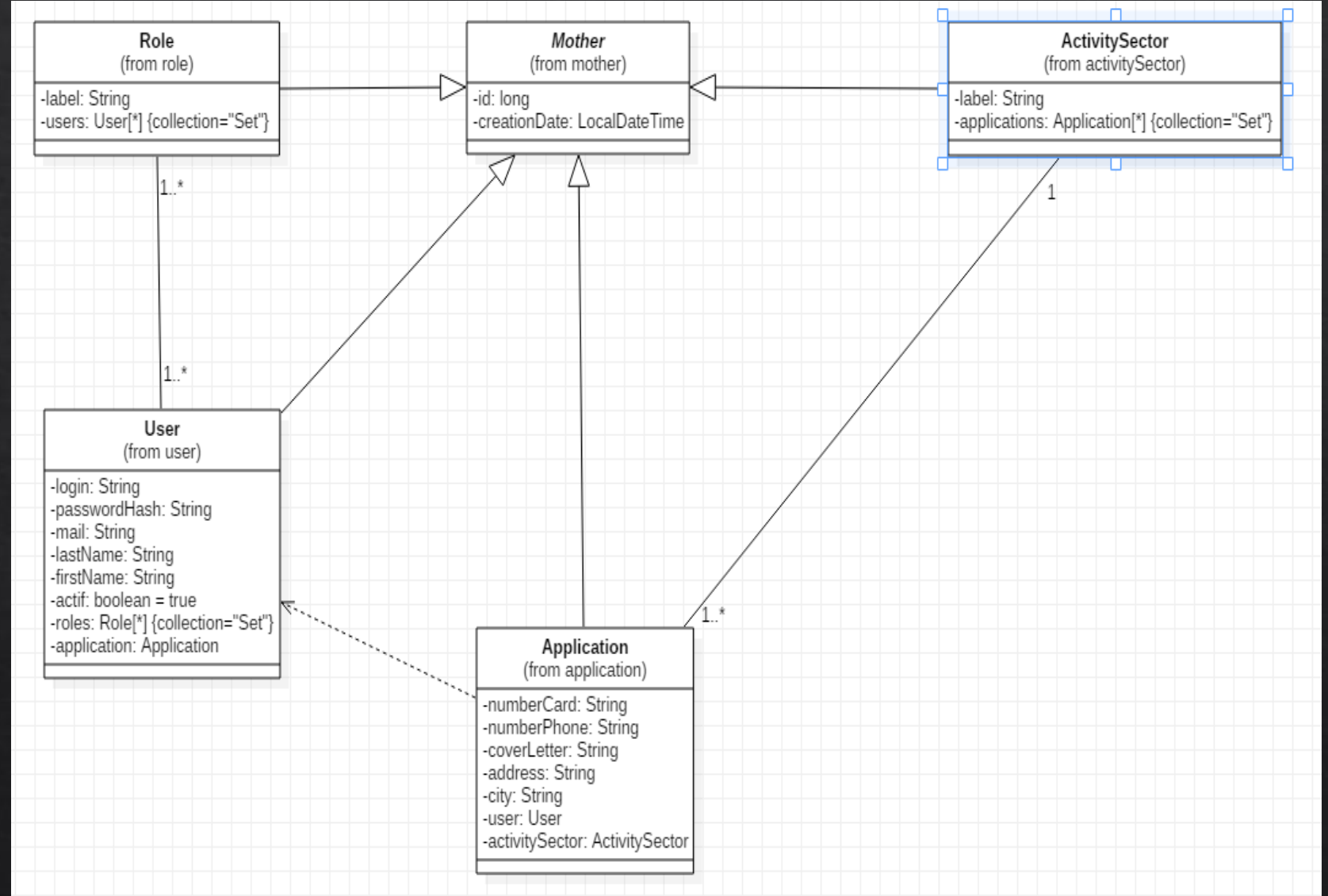
5 classes et 7 liaisons



Symbolise l'héritage, tous mes objets héritent de la classe « Mother » ce qui leur permet d'avoir le champ « id » et « creationDate ».



Symbolise la dépendance, mon entité « Application » dépend de mon entité « User ». Si mon Objet « User » est détruit alors l'objet « Application » qui lui est associé est détruit aussi.



Développement

Application Responsive

Smartphone

```
@media screen and (max-width: 780px) {
  figure{
    margin-top: 20px;
    width: 100%;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-evenly;
    margin-left: -30px;;
  }

  .numero {
    width: 20%;
    margin-left: 32%;
  }

  .num {
    width: 25px;
    height: 25px;
    color: ■ white;
  }
}
```

Tablette

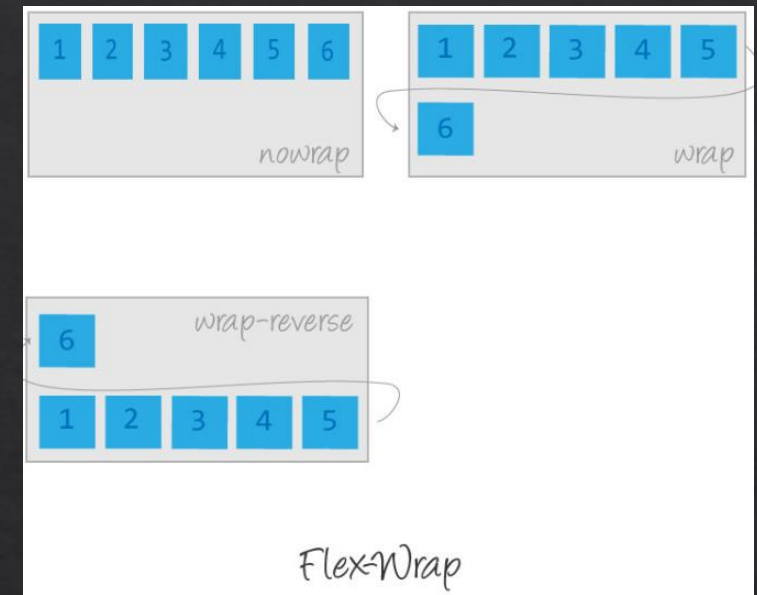
```
@media screen and (max-width: 1200px) {
  .main {
    width: 100%;
    margin-top: 80px;
  }

  p.presentation {
    text-align: center;
  }

  h1 {
    text-align: center;
  }

  section.un {
    margin-top: 50px;
    width: 100%;
    margin-bottom: 100px;
  }

  section.un article {
    flex-wrap: wrap;
    display: flex;
    width: 100%;
    height: 300px;
    justify-content: space-around;
  }
}
```



J'ai utilisé des media queries et Flexbox afin que le contenu de mon site soit compatible avec tous les supports numériques (écran d'ordinateur, tablette numérique, smartphone).

Les media queries permettent de changer mon code CSS (Cascading Style Sheets) en fonction de la largeur du support numérique.

Flexbox me permet de rapidement et simplement disposer mes éléments où je souhaite en précisant la largeur de leur conteneur. Il permet à mes éléments d'adapté leur positionnement en fonction de la largeur du support numérique.

Développement

Web dynamique

```
27  mounted() {  
28      |   |   |   axios  
29      |   |   |   .get("user/get/all")  
30  |   |   |   .then(res => {  
31      |   |   |   |   this.users = res.data;  
32      |   |   |   |   })  
33  |   |   |   .catch(err => {  
34      |   |   |   |   console.error("error ", err);  
35      |   |   |   |   });  
36      |   |   |   },  
37  |   |   |   data() {  
38  |   |   |   |   return {  
39      |   |   |   |   |   users : [],  
40      |   |   |   |   |   user : {},  
41      |   |   |   |   |   }  
42      |   |   |   |   },  
43      |   |   |   |  
44      |   |   |   }  
45      |   |   |   </script>
```

La fonction « mounted » me permet d'exécuter l'instruction entre {} seulement une fois le composant est créé et chargé par le navigateur. Cette instruction interroge le RestController de mon back-end qui répond à l'URL (« /user »). La méthode associée à l'URL « get/all » me permet de récupérer tous les utilisateurs stocké dans ma base de donnée ainsi que les candidatures qui leurs son associées puis j'incrémente mon tableau « users » des objets que je viens de récupérer.

Développement

Web dynamique

```
1  <template>
2    <div>
3      <table>
4        <thead>
5          <tr>
6            <th v-for="(prop, p) in usersProp" :key="p">{{prop}}</th>
7          </tr>
8        </thead>
9        <tbody>
10         <tr v-for="(prop, p) in users" :key="p">
11           <td v-for="(user, n) in prop" :key="n">{{user}}</td>
12         </tr>
13       </tbody>
14     </table>
15   </div>
16 </template>
17
18 <script>
19 import axios from "axios";
20 export default {
21   computed: {
22     usersProp() {
23       return this.users[0] ? Object.keys(this.users[0]) : [];
24     }
25   },
26 }
```

La présence de la directive v-for nous permet de rendre le DOM dynamique en fonction des données.

Développement

Création de la base de donnée

```
package cjSecurity.model.mother;

import java.time.LocalDateTime;

@Getter
@MappedSuperclass
public abstract class Mother {

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

    private LocalDateTime creationDate = LocalDateTime.now();
}
```

Grâce à Hibernate un ORM (Object Relational Mapping) de JPA (Java Persistence API) (Application Programming Interface) en utilisant l'annotation « @Entity » je peux créer une table dans ma base de donnée correspondant à mon objet « Application ».

J'ai créé une classe abstraite « Mother » dont vont hériter toutes mes autres classes afin de partager le champ « id » et « creationDate ». L'annotation « @MappedSuperclass » permet à Hibernate de récupérer le champ « id » et « creationDate » et de l'insérer dans la table de tous mes objets qui héritent de « Mother ».

Les annotations « @OneToOne » et « @ManyToMany » permettent de générer une clef étrangère et de relier ma table Application avec User en One To One et Application avec ActivitySector en Many To One.

```
16 @Getter@Setter
17 @Entity
18 public class Application extends Mother{
19
20     private String numberCard;
21
22     private String numberPhone;
23
24     private String coverLetter;
25
26     private String address;
27
28     private String city;
29
30     @JsonIgnore
31     @OneToOne(cascade=CascadeType.DETACH)
32     private User user;
33
34     @JsonIgnore
35     @ManyToOne
36     private ActivitySector activitySector;
37
38     public Application(String numberCard, String numberPhone, String coverLetter, String address, String city,
39         User user, ActivitySector activitySector) {
40         super();
41         this.numberCard = numberCard;
42         this.numberPhone = numberPhone;
43         this.coverLetter = coverLetter;
44         this.address = address;
45         this.city = city;
46         this.user = user;
47         this.activitySector = activitySector;
48     }
49
50     public Application() {
51         super();
52     }
53
54 }
```

```
CREATE TABLE application (
    id bigint(20) NOT NULL AUTO_INCREMENT,
    creation_date datetime DEFAULT NULL,
    address varchar(255) DEFAULT NULL,
    city varchar(255) DEFAULT NULL,
    cover_letter varchar(255) DEFAULT NULL,
    number_card varchar(255) DEFAULT NULL,
    number_phone varchar(255) DEFAULT NULL,
    activity_sector_id bigint(20) DEFAULT NULL,
    user_id bigint(20) DEFAULT NULL,
    PRIMARY KEY (id),
    CONSTRAINT FOREIGN KEY (user_id) REFERENCES user (id),
    CONSTRAINT FOREIGN KEY (activity_sector_id) REFERENCES activity_sector (id)
) ENGINE=InnoDB ;
```

Développement

Interaction avec la base de donnée

```
1 package cjSecurity.repository.application;
2
3
4 import java.util.List;
9
10 public interface IApplicationRepository extends JpaRepository<Application, Long>{
11
12     List<Application> findAll();
13 |
14 }
```

Pour interagir avec ma base de donnée j'utilise Hibernate. Je crée une interface qui hérite de JpaRepository; je mets en paramètre mon objet et l'id (Application, Long). J'ai créé une méthode afin de pouvoir récupérer toutes les candidatures contenues dans ma base de donnée.

« List <Application> findAll(); » correspond à « select * from application; »

Développement

Services

```

1 package cjSecurity.service.application;
2
3 import java.util.List;
4
5
6
7
8
9
10
11 public interface IApplicationService {
12
13     Application createApplication(UserApplicationDTO applicationDTO);
14     Application updateApplication(UserApplicationDTO application);
15     List<Application> allApplication();
16     ApplicationDTO getApplication(Long id);
17     void removeApplication(Long id);
18 }
19

```

Je crée une interface puis j'instancie les méthodes que je vais utiliser dans mon Controller.

Je crée une classe qui implémente mon interface. J'annote cette classe `@Service` afin que springboot puisse l'utiliser.

Dans cette classe grâce à l'annotation `@Autowired` je fais une injection de dépendance de mes repository de mes objets ActivitySector, Application et User. J'implémente ma méthode `createApplication` qui prend en argument un objet de type `UserApplicationDTO`.

J'instancie un utilisateur et lui attribue la valeur de l'utilisateur contenu dans ma base de donnée possédant l'id de l'utilisateur contenu dans mon objet `UserApplicationDTO`. Je fais la même chose avec l'objet `ActivitySector`.

J'instancie un objet `apply` de type `Application`; je lui attribue les champs de mon objet « `applicationDTO` » qui est en paramétré de ma méthode.

Je sauvegarde l'application dans mon utilisateur afin de les liées.

Puis je sauvegarde mon objet dans ma base de donnée.

```

1 package cjSecurity.service.application;
2
3 import java.util.List;
4
5
6
7
8
9
10
11 @Service
12 public class ApplicationService implements IApplicationService {
13
14     @Autowired
15     private IActivitySectorRepository activitySector;
16
17     @Autowired
18     private IApplicationRepository applications;
19
20     @Autowired
21     private IUserRepository users;
22
23     @Override
24     public Application createApplication(UserApplicationDTO applicationDTO) {
25         User john = users.findById(applicationDTO.getUserId()).get();
26
27         Application apply = new Application();
28
29         ActivitySector activity = null;
30         activity = activitySector.findByLabel(applicationDTO.getSectorLabel());
31
32         apply.setActivitySector(activity);
33         apply.setAddress(applicationDTO.getAddress());
34         apply.setCity(applicationDTO.getCity());
35         apply.setCoverLetter(applicationDTO.getCoverLetter());
36         apply.setNumberCard(applicationDTO.getNumberCard());
37         apply.setNumberPhone(applicationDTO.getNumberPhone());
38         apply.setUser(john);
39
40         john.setApplication(apply);
41
42         return applications.save(apply);
43     }
44 }
45

```

Développement

RestController

```

1 package cjSecurity.controller.application;
2
3 import java.util.ArrayList;
4
24
25 @CrossOrigin(value = "*")
26 @RestController
27 @RequestMapping("/apli")
28 @PreAuthorize("hasRole('ROLE_admin') or hasRole('ROLE_user')")
29 public class ApplicationController {
30
31     @Autowired
32     IApplicationService applications;
33
34     @PostMapping("/post")
35     public ResponseEntity<?> createApplication(@RequestBody UserApplicationDTO applicationDTO) {
36
37         return new ResponseEntity<Application>(applications.createApplication(applicationDTO), HttpStatus.OK);
38     }
39

```

Je crée une classe que j’annote de `@RestController` afin que springboot sache que c’est un Controller de type REST. L’annotation `@RequestMapping`(« /apli ») permet à mon Controller de répondre à l’URL « /apli ». L’annotation `@PreAuthorize` permet de restreindre l’accès, seul un utilisateur avec le rôle « user » ou « admin » peut avoir accès aux méthodes de ce Controller.

Je fais une injection de dépendance du service dont je vais avoir besoins.

L’annotation `@PostMapping`(« /post ») au-dessus de ma méthode permet de faire matcher ma méthode avec l’URL « /post » et seulement pour une requête de type Post et Option. Je crée une méthode qui retourne un objet de type `ResponseEntity` afin de retourner mon objet et une réponse HTTP.

Développement

Sécurité

Afin de sécuriser mon application j'utilise SpringSecurity et le Json Web Token. Lorsqu'une requête HTTP part de mon front elle passe par ma chaîne de filtre puis si tout est en ordre alors un contexte de sécurité est créé. Celui-ci accède à mon Controller.

Mon application est hébergée sur Pivotal cloud foundry, celui-ci me permet de bénéficier de HTTPS (HyperText Transfer Protocol Secure) et donc de sécuriser les requêtes provenant du front en chiffrant mes données en SSL ou TLS.

Remerciement

Je souhaite remercier Simplon.co de m'avoir permis d'accéder à cette formation.

Je tiens à remercier Monsieur Soufiane ROUASS pour toute l'aide qu'il m'a apporté.

Je remercie Stéphane, Jérémy et Guillaume ; de m'avoir appris tout ce que je sais aujourd'hui.

Je remercie aussi mes camarades de promotions qui m'ont énormément apporté.

Je vous remercie de votre attention.

CJ Sécurité Privée

