Spring sécurité ?

- **Gestion de l'authentification et des autorisations**
- **Protection contre les attaques de type:**
  - **XSS :**
  - **CSRF :**

La librairie « Spring security» propose des mécanismes d'authentification et d'autorisations

Gestion de l'authentification et des autorisations

- Le plus simple pour démarrer un projet Spring Security consiste à le charger dans une configuration Spring Boot.
  - Commencez par créer un projet Spring Boot.
  - Voici la dépendance Maven (le starter Spring Boot) à rajouter dans votre fichier pom.xml pour y injecter Spring Security.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Introduction Spring Security

- Il existe plusieurs types d'attaques pouvant compromettre la sécurité d'une application Web. Citons notamment :
  - Les attaques par injection SQL : on injecte dans une requête SQL un bout de code non prévu pouvant compromettre la sécurité.
  - Les attaques de type XSS (Cross-Site Scripting)
  - Les attaques de type CSRF (Cross-Site Request Forgery)
- Il est important de se prémunir de ces types d'attaques.
  - La librairie « Spring Security » propose aussi des outils qui vont dans ce sens.
  - **ATTENTION** : Spring Security ne prend pas en charge les problèmes d'injection SQL. Pour ces sujets, il faut impérativement utiliser des requêtes JPQL à base de « named parameters ».

Etude de cas

- **Authentification JWT et Gestion des Rôles**
  - **Partie 1: Front-end Angular**
  - **Partie 2: Back-end Spring Boot**

Description d l'étude de cas

- **Authentification et gestion de rôles avec Angular/SpringBoot :**

- Ce sera une pile complète, avec **Spring Boot** pour le back-end et **Angular** pour le front-end.

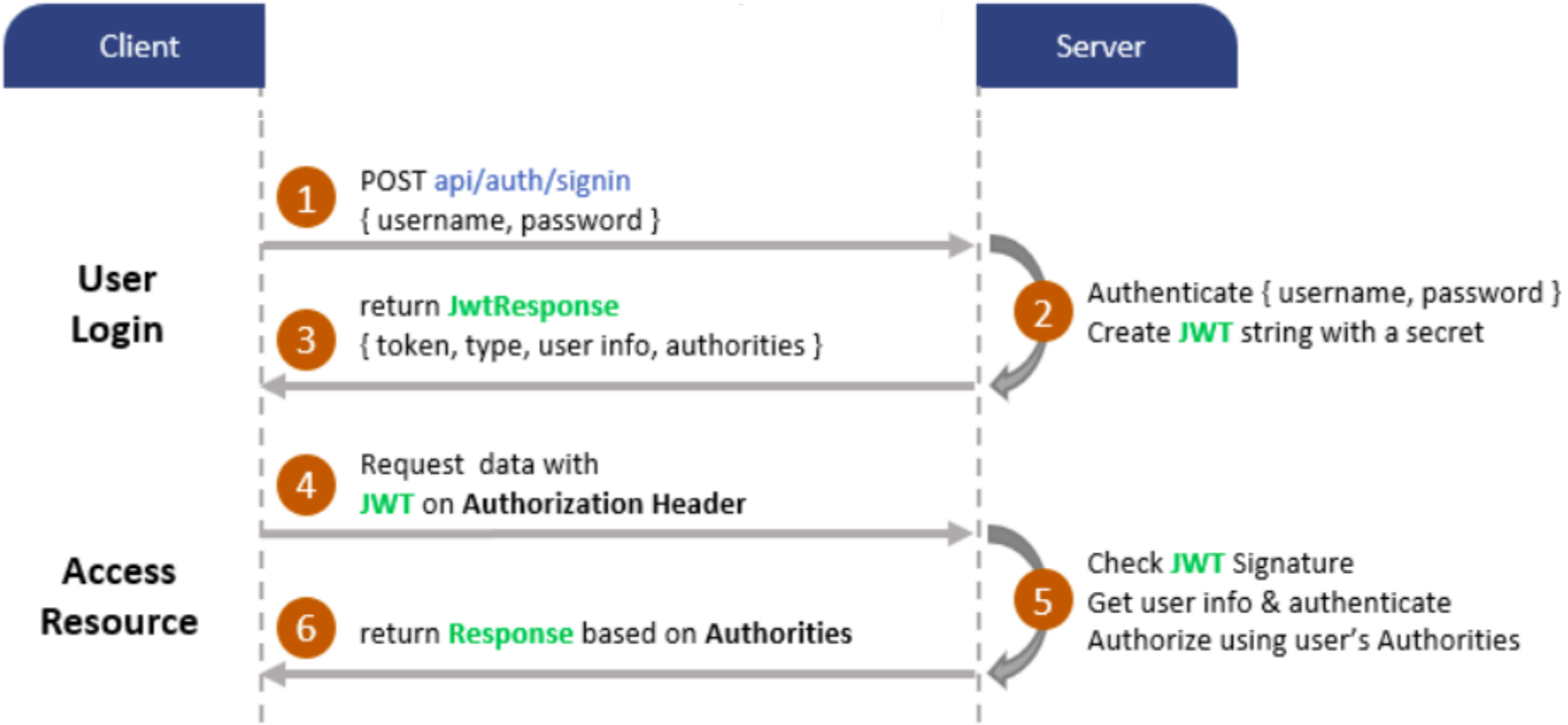
- Le système est **sécurisé** par Spring Security avec ***authentification JWT (Json Web Token).***

- L'utilisateur peut **se connecter** avec un **nom d'utilisateur** et un **mot de passe**.

- **Autorisation** par le **rôle** de l'**Utilisateur (admin, modérateur, utilisateur)**

Scénario de connexion d'un utilisateur

- Si le **client** souhaite envoyer une demande à des **données/points de terminaison** protégés, un **JWT** légal doit être ajouté à **l'en-tête d'autorisation HTTP**.

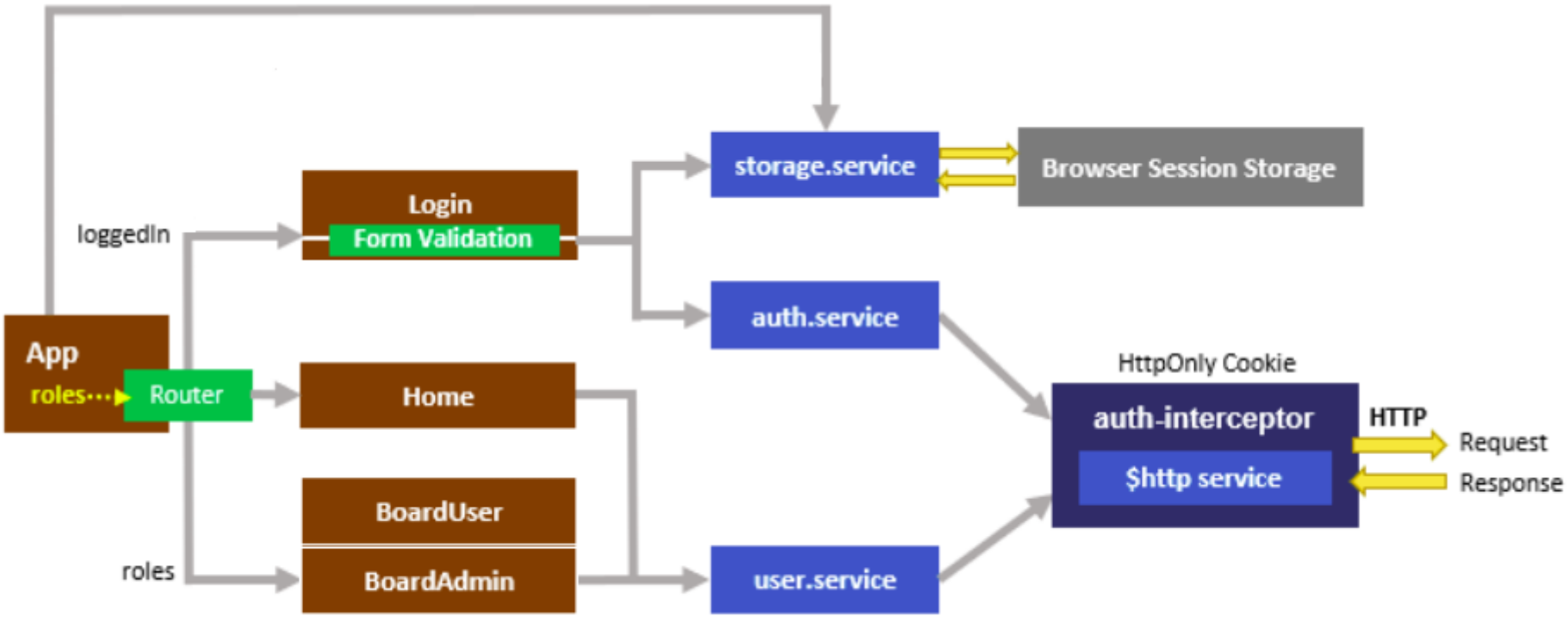


Etude de cas

- **Authentification JWT et Gestion des Rôles**

Partie 1: Front-end Angular

Architecture frontend



Frontend Angular : création de squelette

- Générer les composants, les services et la guard suivants :

- **ng generate component login**
- **ng generate component home**
- **ng generate component board-admin**
- **ng generate component board-user**
- **ng generate service services/auth**
- **ng generate service services/TokenStorage**
- **ng generate interceptor helpers/auth**
- **ng g guard services/auth**

Front-end Angular : exécution

C:\WINDOWS\system32\cmd.exe

```
√ Compiled successfully.
^C
C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate component home
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.spec.ts (585 bytes)
CREATE src/app/home/home.component.ts (194 bytes)
CREATE src/app/home/home.component.css (0 bytes)
UPDATE src/app/app.module.ts (1096 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate service services/auth
CREATE src/app/services/auth.service.spec.ts (347 bytes)
CREATE src/app/services/auth.service.ts (133 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate service services/TokenStorage
CREATE src/app/services/token-storage.service.spec.ts (388 bytes)
CREATE src/app/services/token-storage.service.ts (141 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng generate interceptor helpers/auth
CREATE src/app/helpers/auth.interceptor.spec.ts (404 bytes)
CREATE src/app/helpers/auth.interceptor.ts (409 bytes)

C:\Users\x\Documents\micro-frontend-exemple\marchapp>ng g guard services/auth
? Which type of guard would you like to create? CanActivate
CREATE src/app/services/auth.guard.spec.ts (331 bytes)
CREATE src/app/services/auth.guard.ts (457 bytes)
```

Architecture front-end : explication (1)

– Le **composant App** est un conteneur utilisant Router. Il obtient les informations utilisateur du **stockage de session** du navigateur via ***storage.service***. Ensuite, la barre de navigation peut maintenant s'afficher en fonction de **l'état de connexion** de l'**utilisateur** et des **rôles**.

- Le **composant *Login*** ont un formulaire pour soumettre des données (avec prise en charge de la **validation** de formulaire). Il utilise les services:
  - ***storage.service*** pour vérifier l'**état** de l'utilisateur
  - ***auth.service*** pour envoyer des **demandes** de connexion.

- *auth.service* utilise le service **$http** d' Angular pour effectuer des demandes d'**authentification**.

– chaque requête **HTTP** envoyer par le service **$http** sera inspectée et transformée avant d'être envoyée par le service **auth-interceptor**.

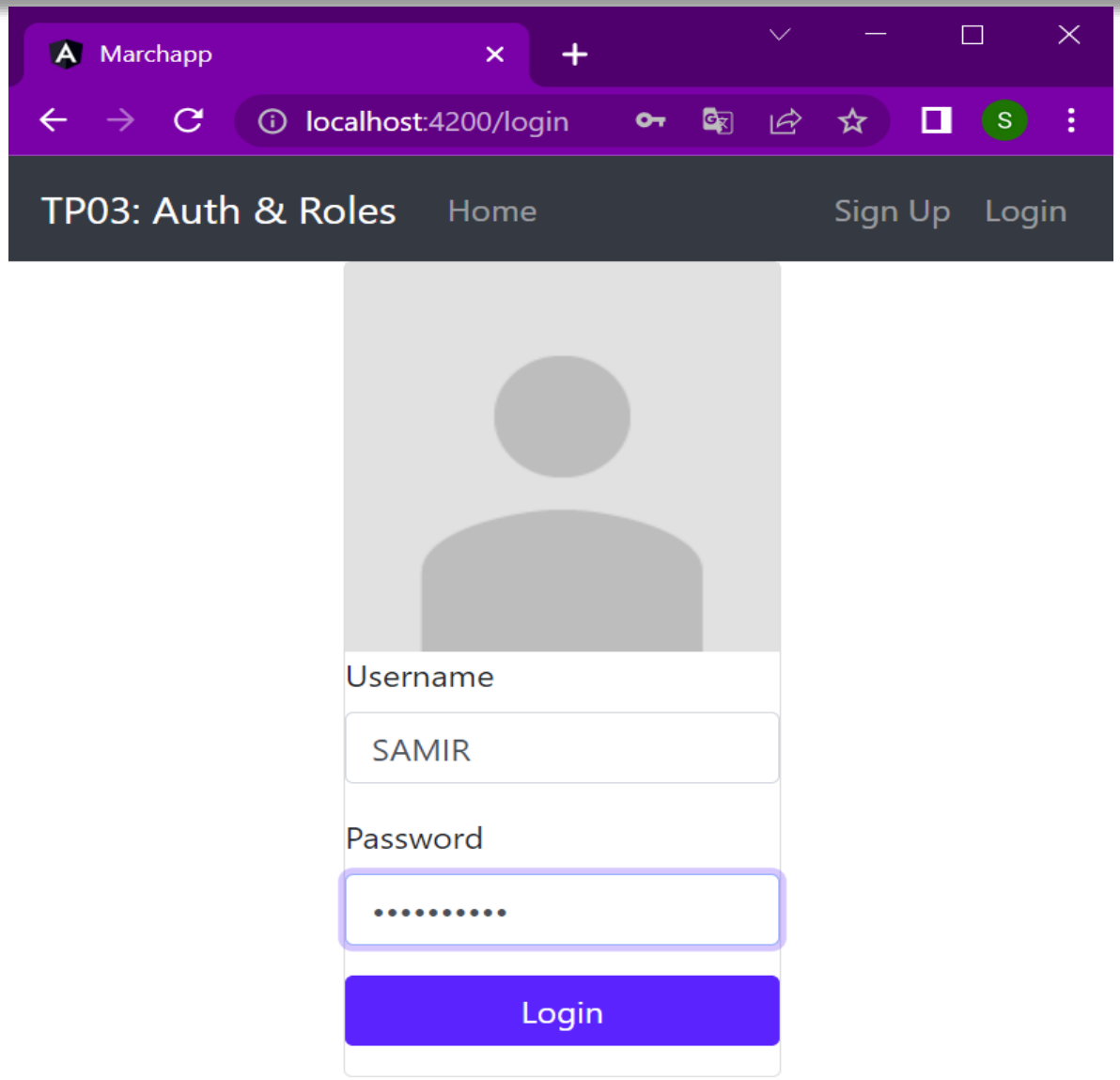
Architecture front-end : explication (2)

– Le **composant** d'accueil (**Home**) est public pour tous les visiteurs.

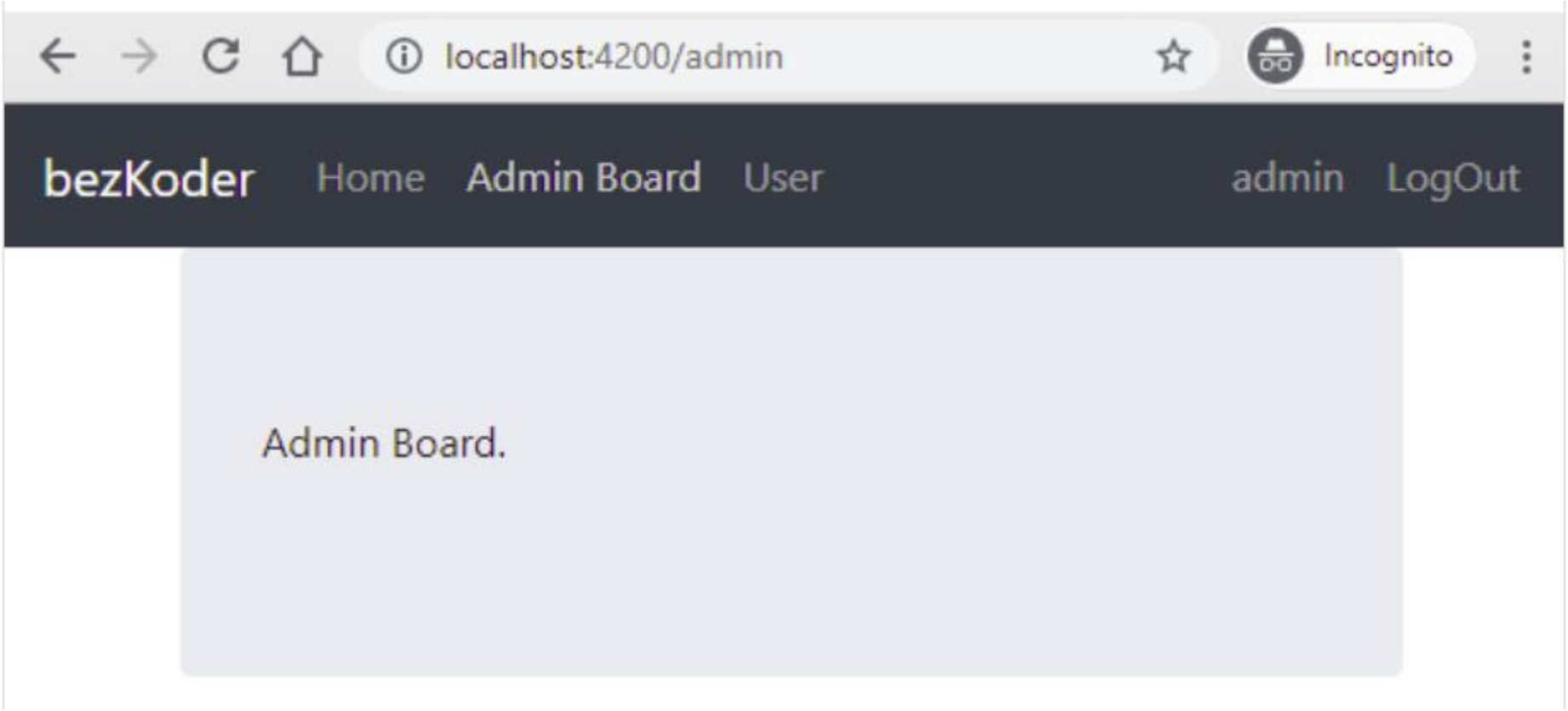
– Les composants ***BoardUser***, ***BoardAdmin*** seront affichés en fonction des ***rôles*** de la session (**Session Storage**). Dans ces composants, nous utilisons ***user.service (httpClient.service)*** pour obtenir des ressources **protégées** de l'API (avec **JWT** dans **HttpOnly** Cookie).

- **Les tokens JWT** permettent **d'échanger** entre un **serveur** et différentes applications **clientes des informations d'utilisateur et de rôles de manière stateless.** Ils sont **signés** et **cryptés** pour éviter d'être **modifiés** côté client**,**
- **Attention** à l'endroit où vous décidez de stocker **JWT!**
- Il est recommandé d'utiliser les **Cookies HttpOnly** pour le transmettre, couplé au mécanisme de contrôle de **token xsrf, vous** serez protégés de manière efficace contre les **attaques XSS (**car **HTML 5 Web Storage est vulnérable** et le **XSS** bien plus fréquent que les attaques **CSRF).**

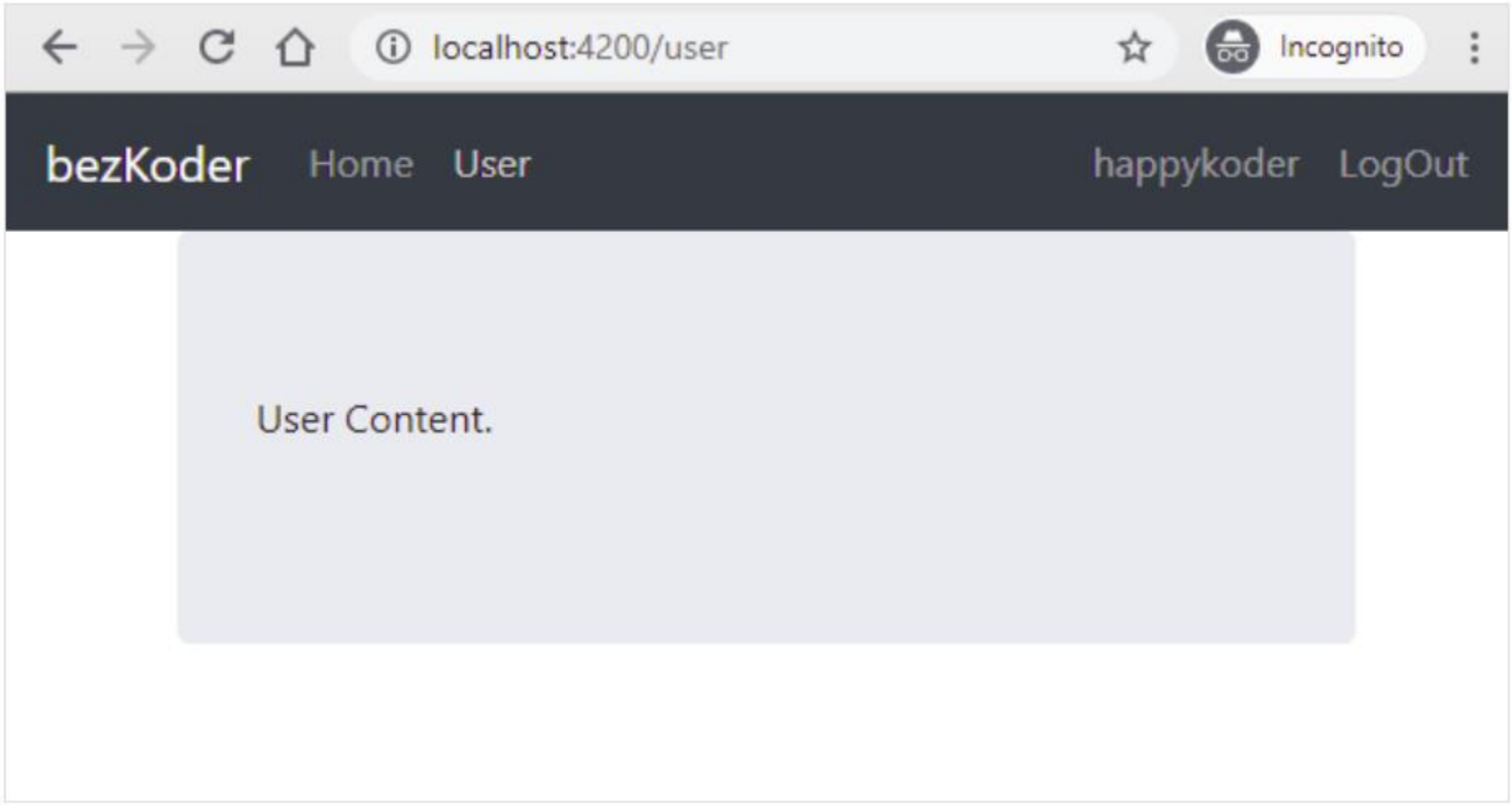
Composant Login



Admin Board


localhost:4200/admin
Incognito
bezKoder
Home
Admin Board
User
admin
LogOut
Admin Board.

User Board

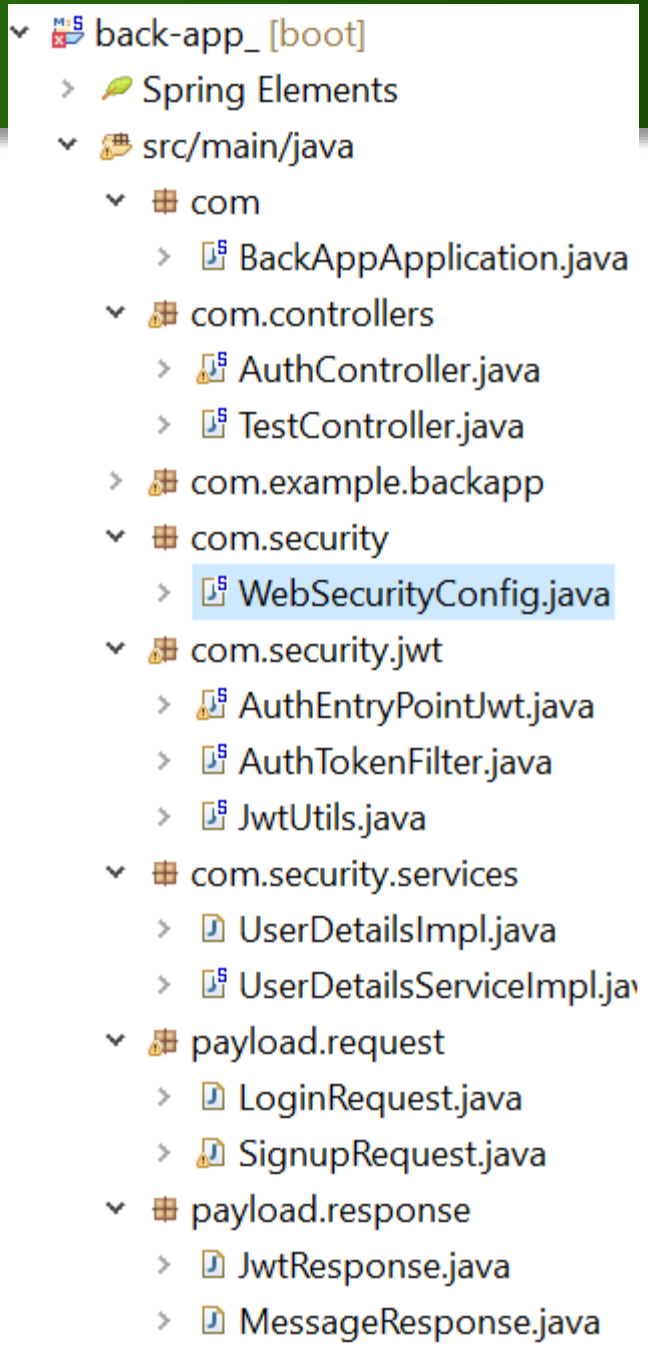

localhost:4200/user
Incognito
bezKoder
Home
User
happykoder
LogOut
User Content.

Etude de cas

- **Authentification JWT et Gestion des Rôles**

Partie 2: Backend spring boot, spring security

Structure du projet (1)



Structure du projet (2)

Package sécurité : nous configurons Spring Security et implémentons des objets de sécurité.

- **WebSecurityConfig**
- ***UserDetailsServiceImpl*** implémente ***UserDetailsService***
- ***UserDetailsImpl*** implémente **UserDetails**
- ***AuthEntryPointJwt*** implémente ***AuthenticationEntryPoint***
- ***AuthTokenFilter*** étend ***OncePerRequestFilter***
- ***JwtUtils*** fournit des méthodes pour générer, analyser, valider ***JWT***

Les contrôleurs *gèrent les demandes* ***de connexion*** *et les demandes* ***autorisées.***

- AuthController : @PostMapping('/signin'),

- TestController : @GetMapping('/api/test/all'), @GetMapping('/api/test/[role]'

Structure du projet (3)

Repository: a des interfaces qui étendent **Spring Data JPA JpaRepository** pour interagir avec la base de données PostgreSQL**.**

- **UserRepository** étend **JpaRepository<User, Long>**
- **RoleRepository** étend **JpaRepository<Role, Long>**

Les modèles définissent deux modèles principaux pour **l'authentification (utilisateur) et l'autorisation (rôle).** Ils ont une relation *plusieurs à plusieurs***.**

- Utilisateur : identifiant, nom d'utilisateur, mot de passe, rôles..

- **Rôle : identifiant, nom,…**

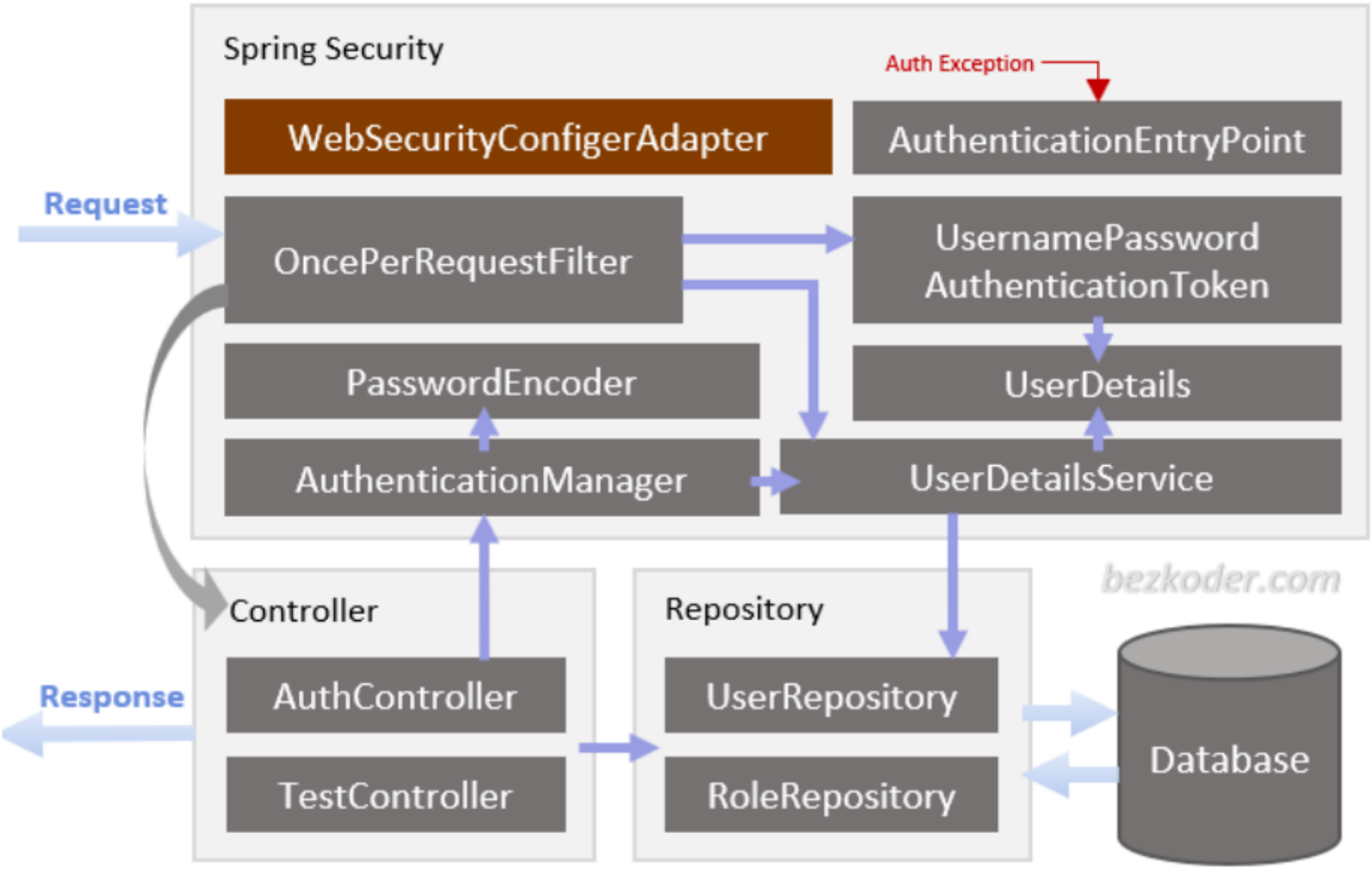
Pakage Payload :définit les classes pour les objets **Request** et **Response**

Ajouter les dépendances au fichier POM.xml

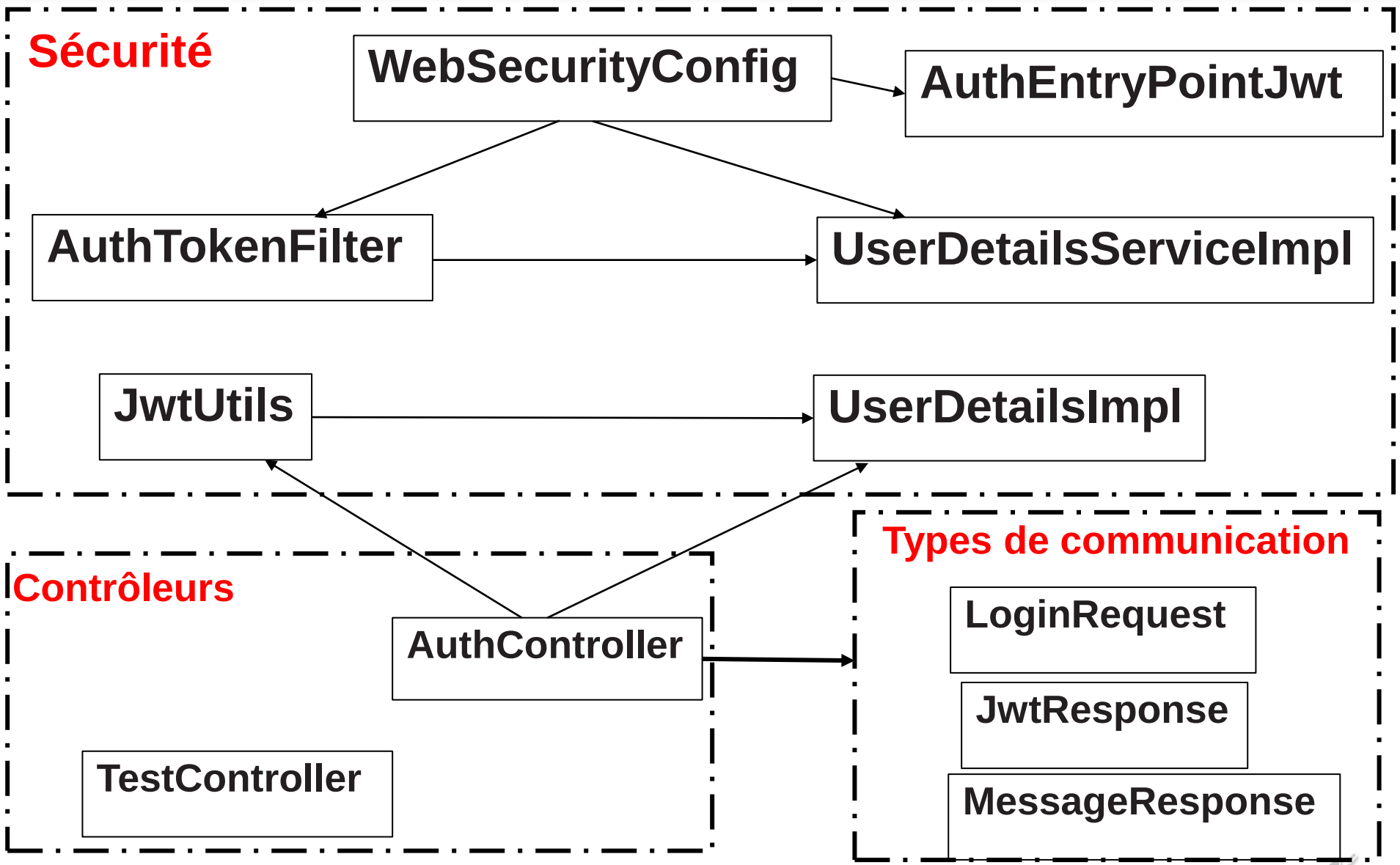
back-app_/pom.xml

```
49 <dependency>
50     <groupId>org.springframework.boot</groupId>
51     <artifactId>spring-boot-starter-security</artifactId>
52 </dependency>
53 <dependency>
54     <groupId>io.jsonwebtoken</groupId>
55     <artifactId>jjwt</artifactId>
56     <version>0.9.1</version>
57 </dependency>
58 <dependency>
59   <groupId>org.springframework.boot</groupId>
60   <artifactId>spring-boot-starter-validation</artifactId>
61 </dependency>
62 <dependency>
63     <groupId>jakarta.xml.bind</groupId>
64     <artifactId>jakarta.xml.bind-api</artifactId>
65 </dependency>
66     </dependencies>
```

Architecture backend



Dépendance entre les classe Java



Architecture Backend: spring security (1)

- Class **WebSecurityConfig** : est au cœur de notre implémentation de la sécurité.
- Elle fournit des configurations **HttpSecurity** pour configurer :
  - **CORS**
  - **CSRF**,
  - la gestion des **sessions**
  - les **règles** pour les **ressources protégées**.

- Note : Nous pouvons également **étendre** et personnaliser la configuration par **défaut** qui contient les éléments ci-dessous.

Architecture Backend: spring security (2)

- L'interface **UserDetailsServiceImpl** a une méthode permettant de:
  - Charger un utilisateur par **son nom**
  - et renvoie un **objet** de type ***UserDetailsImpl*** que Spring Security peut l'utiliser pour l'***authentification*** et la ***validation*** d'un utilisateur.

- **UserDetailsImpl** contient les **informations** nécessaires pour créer un objet d'**authentification,** telles que :
  - **Nom d'utilisateur**
  - **Mot de passe**
  - **Rôles**

Architecture Backend: spring security (3)

- ***UsernamePasswordAuthenticationToken*** obtient {***username, password***} à partir de la demande de connexion, ***AuthenticationManager*** l'utilisera pour authentifier un compte de connexion.

- ***AuthenticationManager*** a un ***DaoAuthenticationProvider*** (avec l'aide de UserDetailsService & PasswordEncoder) pour valider l'objet **UsernamePasswordAuthenticationToken**. En cas de succès, **AuthenticationManager** renvoie un objet **Authentication** entièrement rempli (y compris les **Roles** accordées).

Architecture Backend: spring security (4)

- **AuthTokenFilter:** effectue une seule exécution pour chaque requête adressée à notre **API**.
- Il fournit une méthode **doFilterInternal()** que nous allons implémenter pour :
  - Analyser et valider un **JWT**,
  - Charger les ***détails*** de l'utilisateur (à l'aide de ***UserDetailsServiceImpl***),
  - Vérifier les ***autorisations*** (à l'aide de ***UsernamePasswordAuthenticationToken***).

- **AuthEntryPointJwt** détectera une **erreur** non autorisée et renverra un **401** lorsque les clients accèdent à des ressources protégées sans authentification.

Architecture Backend: Controller et Repository

- **Repository** contient **UserRepository** et **RoleRepository** pour fonctionner avec la base de données, sera importé dans le contrôleur.

- Le contrôleur reçoit et traite la demande après qu'elle a été filtrée par **AuthTokenFilter (extends OncePerRequestFilter)**

- **AuthController** gère les demandes de connexion

- **TestController** a accès à des méthodes de ressources **protégées** avec des **validations** basées sur les **rôles**.

- **L’hébergement de sites Web et Le référencement Internet**
  - Etude de cas : Déployer une application sur Heroku

Introduction : cycle de vie d'un site web

- **Elaboration**, **conception** , **préparation au référencement**

- **Hébergement** : Une fois site « **terminé** », il faut trouver un **hébergeur** afin de rendre le site **visible** sur le web.
  - L'hébergeur **stocke** le site sur internet et attribue au propriétaire une **adresse internet** (ou **URL**).

- **Référencement :** A cette étape, le **webmaster** s'engage à **référencier** le site et à améliorer son **positionnement** dans les **moteurs de recherche**.
  - **Par exemple: Indexer** le site auprès des **moteurs de recherche** : suggérer le site à **Google**

- **Mettre à jour** , **rafraîchir le site**…., **faire vivre le site web**

L'Hébergeur Web

L'hébergeur peut s'occuper d'un site à ***divers niveaux***:

- **Gestion du (des) domaine(s) (DNS)**
  - Entrées DNS à configurer
  - Sous-domaines (par ex. « sous.domaine.tn »)
- **Gestion des emails du domaine**
  - Stockage des emails
  - Redirection des emails
  - Interfaces d'accès aux emails
  - Protocoles de consultation (POP, IMAP)
- **Mise à disposition d'un serveur**
  - Espace de stockage de fichiers
  - Architecture technique (scripts, frameworks, logiciels)

Pourquoi l'hébergement Web ?

Les raisons de l'hébergement Internet :

- la nécessité de **sécuriser** le service hébergé,
- la mise à disposition par le prestataire de **ressources conséquentes** (bande passante en téléchargement,),
- le **conseil** et les **services** de support associés.

Comment héberger un site ? (1)

Pour héberger un site, il faut :

- Un **nom de domaine**
- Un serveur
- Du contenu

Nom de domaine :

Permet d'associer un **nom** facile à retenir à une **adresse IP**.

MYaPPweb.dz → **5.135.181.163** (IPv4)

MYaPPweb.dz → 2001:41d0:8:bca3::1 (IPv6)

Comment héberger un site ? (2)

Acheter un nom de domaine

- Il faut passer par un **bureau d'enregistrement** (registrar) **agréé**.
- Le **paiement** est annuel.
- Le tarif dépend du domaine de premier niveau choisi (.fr, .com,).
- Certains domaines sont réservés pour des usages précis.

Choisir son nom de domaine

- Domaines **génériques** : .com, .net, .org, .pro, .edu, .name
- Domaines **géographiques** : .fr, .de, .ru, .alsace, .bzh
- Domaines **professionnels** : .aero, .archi, .xxx

Annuaire des bureaux d'enregistrement : http://s.strasweb.fr/e

Comment héberger un site ? (3)

Les hébergeurs commerciaux:

- **OVH** (ovh.com)
- **Gandi** (gandi.net)
- **1and1** (1and1.com)
- **PHPNET** (phpnet.org)
- **Dedibox** (online.net)
- **SDV Plurimédia** (sdv.fr)

Et plein d'autres...

Activité de l’hébergeur Web

La principale **activité** de l'hébergeur web consiste à :

- **Installer** les serveurs et les **sécuriser** (par une alimentation électrique **ondulée**, secourue par un **groupe électrogène**, une **salle climatisée** équipée de dispositifs anti-incendie),
- Tenir les serveur **à jour** en installant les **mises à jour** de sécurité pour éviter les at**taques malveillantes**.
- **Réparer** les serveurs en **cas de panne**,
- Installer les **technologies logicielles** souhaitées par les **clients** ou qu'il souhaite leur offrir (comme les langages de programmation et les modules supplémentaires).

Types d'hébergement

Dédié (dedicated)

- Une machine entière à disposition
- Plus rapide, plus disponible, plus de stockage
- Plus cher

Mutualisé (mutualized)

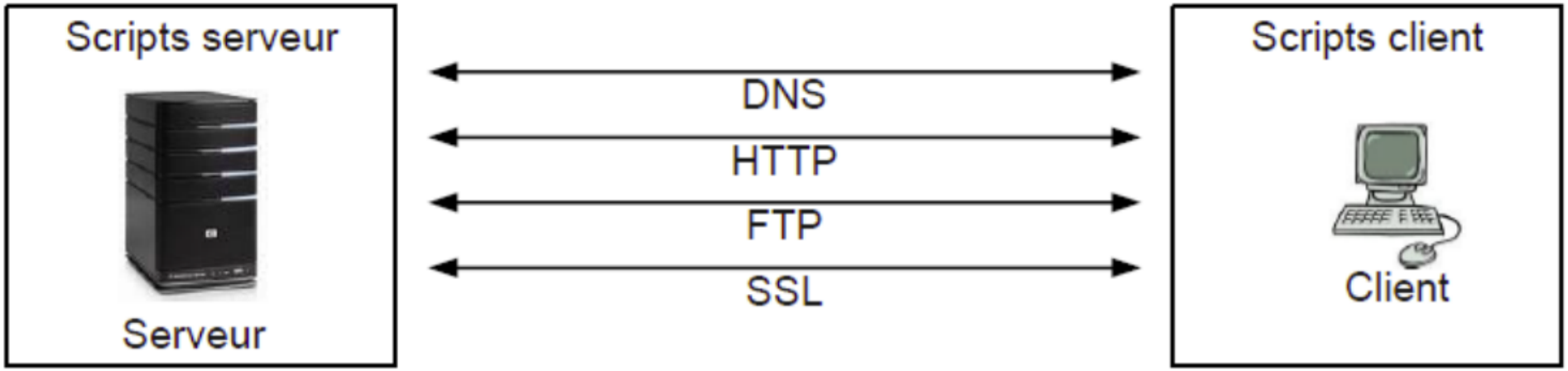
- Une « portion » de machine à disposition
- Moins rapide, plus de risques de sécurité
- Beaucoup moins cher

Colocation

- La machine appartient au client
- L'hébergeur fournit l'infrastructure, le réseau, la maintenance

Serveur, propose des services « orientés web »

- **DNS** : noms de domaine
- **HTTP** : pages web (HTML, XML, JSON, etc.)
- **FTP** : serveur de fichiers
- **SSL** : transactions cryptées
- Principe **client** / **serveur**



- *Comment font les **internautes** pour **trouver** le site web?*

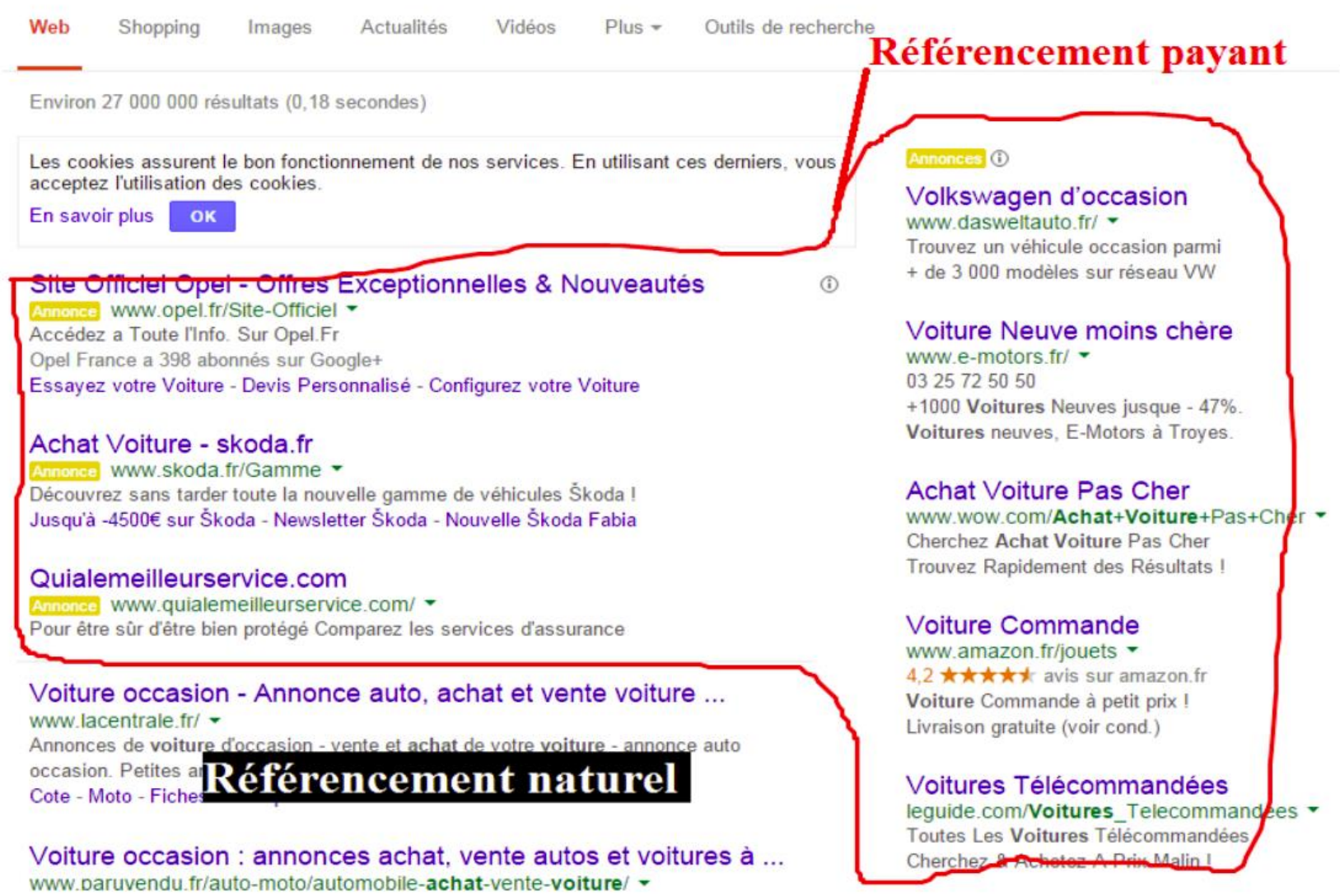
- *Quelle est l'importance du **positionnement** du site web?*

Définition:

- Le référencement est le fait d'**indexer** un **site web** dans les **moteurs de recherche** ou dans les annuaires.

- Par extension: l'ensemble des actions et techniques visant **à améliorer la position du site internet** dans ces **résultats de recherche** (positionnement) et à en optimiser la **visibilité**.

Types de Référencement



Types de Référencement

Référencement naturel

• Toutes les **techniques** utilisées pour améliorer le **positionnement** du site dans les **moteurs de recherche** et accroitre sa popularité, c'est un effort de **conception et de développement** du site

Référencement payant

• L'utilisation de **publicités** comme les campagnes **GoogleAdwords**, achat de bannières

Domaines de Référencement

- **SEA (Search Engine Advertising)** : achat de liens ***sponsorisés*** afin de se positionner sur des mots clés spécifiques.

- **SEM (Search Engine Marketing)** : mise en place de liens sponsorisés, notamment sur les moteurs de recherche, afin de gagner en **visibilité** tout en augmentant son trafic.

- **SEO (Search Engine Optimisation)** : optimisation d'une page ou d'un site pour les moteurs de recherche destinée à faire remonter la page ou le site dans les pages de résultats.

Conclusion Référencement

- L'**hébergement** du site web doit s'accompagner par un effort de **référencement** pour augmenter la **visibilité** du site.

- Le référencement web est tout une **stratégie** qui doit être suivie dès la **conception** du site web.

- Les administrateurs des sites s'engagent à assurer la **mise à jour** régulière et la meilleure qualité de leurs ressources en ligne: Impératifs de qualité

- **Déployer une application sur Heroku**

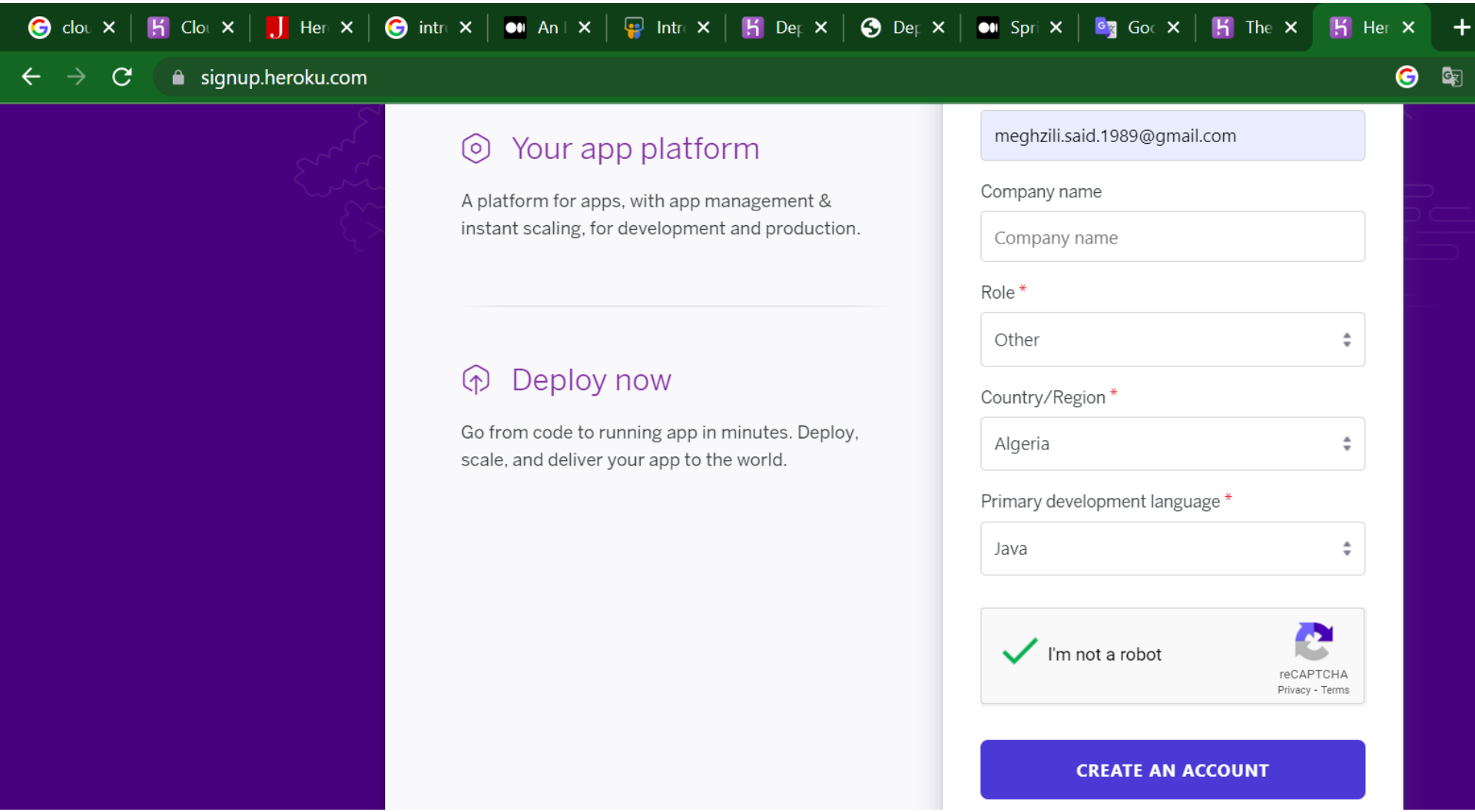
Etapes de déploiement

1) **Préparation de l'envirenement**
   - Création d'un compte Heroku
   - Télécharger et installer Herouku CLI:

   https://devcenter.heroku.com/articles/heroku-cli#download-and-install
   - Installer **GIT**

2) **Préparation de l' application pour la production**
3) **Déploiement de l'application**

Création d'un compte Heroku

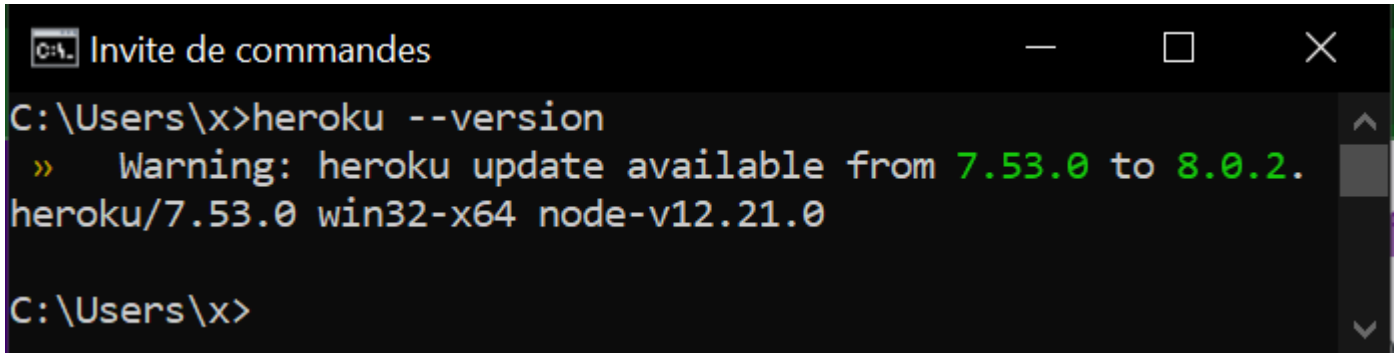


Git

- **Un système de gestion de versions : est un programme ou un ensemble de programmes qui effectue le suivi des changements apportés à une collection de fichiers.**
- **Son objectifs est de rappeler facilement les versions antérieures de fichiers ou d'un projet entier.**
- **Outil très utile**
  - Pour la gestion de version
  - Permet "facilement" le travail collaboratif
  - Partage et gestion de vos fichiers TEXTES vers un serveur distant
- **Installation : https://git-scm.com/downloads**
- **dépôt (repository) :Quelques forges :**
  - GitHub
  - GitLab

Déploiement de l'application en utilisant Heroku CLI (1)

1- Tester l'installation



2- Se Connecter à heroku en utilisant

```
$ heroku login -i
heroku: Enter your login credentials
Email: <your-email>
Password: <your-secret-password>
Logged in as <your-email>
```

Déploiement de l'application en utilisant Heroku CLI (2)

3- Créer l'application

```
$ heroku create
Creating app... done, ⬢ salty-reaches-63504
https://salty-reaches-63504.herokuapp.com/ | https://git.heroku.com/salty-reaches-63504.git
```

4- Renommer l'application : (spring-redit-clone)

```
$ heroku apps:rename --app salty-reaches-63504 spring-reddit-clone
Renaming salty-reaches-63504 to spring-reddit-clone... done
https://spring-reddit-clone.herokuapp.com/ | https://git.heroku.com/spring-reddit-clone.git
```

Déploiement de l'application en utilisant Heroku CLI (3)

5- Créer une branche distante dans git appelée heroku: en utilisant cette branche ***Heroku*** démarrera automatiquement la construction et le déploiement de votre application, une fois que vous aurez envoyé le code source à **Git**.

```
$ git remote add heroku https://git.heroku.com/spring-reddit-clone.git
```

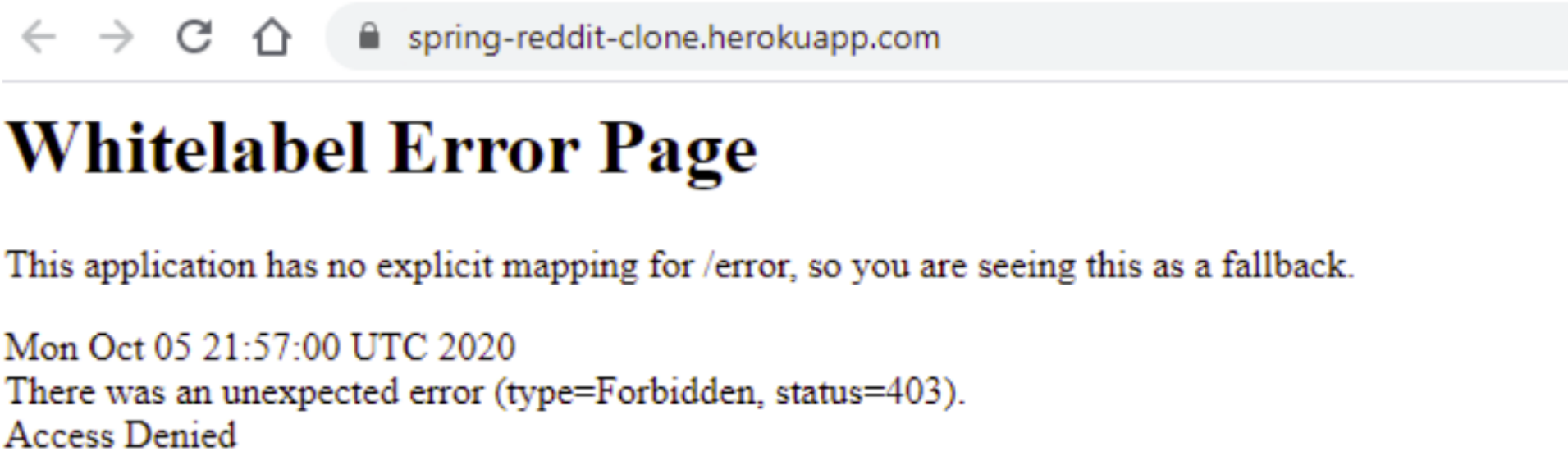
Déploiement de l'application en utilisant Heroku CLI (4)

6- Déploiement de l'application: Maintenant que notre application est prête à être déployée, exécutez la commande ci-dessous en vous assurant que vous êtes dans le répertoire racine de l'application.

```
$ git push heroku master
Enumerating objects: 560, done.
Counting objects: 100% (560/560), done.
Delta compression using up to 12 threads
Compressing objects: 100% (184/184), done.
Writing objects: 100% (560/560), 445.74 KiB | 44.57 MiB/s, done.
Total 560 (delta 284), reused 560 (delta 284)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Java app detected
remote: -----> Installing JDK 1.8... done
remote: -----> Executing Maven
remote:        $ ./mvnw -DskipTests clean dependency:list install
```

Déploiement de l'application en utilisant Heroku CLI (5)

7- Ouvrez le lien https://spring-reddit-clone.herokuapp.com/, vous verrez la page suivante.



Bibliographie (1)

1- Web Application Security: Exploitation and Counter measures for Modern Web Applications
1rst edition, Andrew Hoffman ,Oreilly editions , 2020.

2- Web Security for Developers: Real Threats, Practical Defense, Malcolm McDonald , No Starch Press ,2020.

3-https://www.bezkoder.com/spring-boot-security-postgresql-jwt-authentication/

4- https://www.bezkoder.com/angular-spring-boot-jwt-auth/

5- https://owasp.org/ : Open Web Application Security Project,

Bibliographie (2)

6- INJECTION SQL ÉNS Vieux Kouba Département d'Info, Présenté par : -Bentami Billel -Mesbaiah Ahmed Dhia Eddine Avril 2017

7- Cours CSS, Moutasem Hamour, New York Institute of Technology (NYIT)-Amman.

8- Security of Web Applications, Vitaly Shmatikov

9- https://slideplayer.fr/slide/14876272/

10- Cours Emna ABIDI, conception d'un site web marchant, url: https://www.nassimbahri.ovh/docs/conception/Chapitre-4.pdf

11- https://programmingtechie.com/2020/10/10/deploy-spring-boot-and-angular-application-to-heroku/