# ITÉRATION 3

## Cas pratique

## Réalisation d'une application SPA Angular

### Modalités

- Travail seul ou en équipe
- 4,5 jours en présentiel

### Objectifs

Savoir lire et analyser un cahier des charges fonctionnels, préparer la conception du projet et réaliser une application SPA communiquant avec une API Rest réalisée en Java.

### Compétences

- Intégrer une interface selon des conceptions
- Créer des pages web responsives
- Développer une application multi-couche
- Sécuriser une application multi-couche
- Prendre en compte les contraintes des applications multilingues
- Documenter un code source
- Connaître et appliquer les éléments de qualité logicielles
- Maîtriser les règles d'accessibilité

## 1.1 — Introduction

4h — Présentiel

Pour cette partie, nous nous mettons dans la configuration suivante : votre entreprise à répondu un appel d'offre d'un Client et vous partage son cahier des charges qui reprend ses besoins et attentes fonctionnelles et techniques.

Vous devez prendre connaissance de ce document, et apporter une réponse conceptuelle qui prendra la forme d'une application SPA réalisée avec Angular (qui communique avec un backend Java et une base de données).

**Attention : dans le cahier des charges, une partie est indiquée "optionnelle" : les fonctionnalités décrites dans cette partie ne sont pas à gérer / prendre en compte pour le moment.**

Le client a entendu parler de plusieurs technologies (entre VueJS, React, Angular, Knockout, ...) : il vous demande de motiver et argumenter le choix d'Angular pour son besoin.

### TODO

- ➔ **Lire le cahier des charges du Client** (document à part).
- ➔ **Analyser le besoin et effectuer la conception** afin de proposer une solution technique à ce besoin.
- ➔ **Préparer un argumentaire** sur le choix d'Angular pour la solution technique.

**RESSOURCES**

- Document "Cahier des charges client"
- Comparatif ReactJS vs Angular vs VueJS https://www.ambient-it.net/reactjs-vs-angular-vs-vuejs/
- Comparatif ReactJS vs Angular vs VueJS https://blog.dyma.fr/quel-framework-choisir-angular-vue-js-ou-react/

## 1.2 — Backend - Application Java Spring Boot

6h — Présentiel

La réponse au besoin du Client implique la création d'une application serveur JAVA API REST en Spring Boot, disposant de plusieurs ressources REST, afin de pouvoir :

- récupérer la liste des experts et leurs informations ;
- récupérer le nombre de crédits d'un client ;
- formuler une demande de réservation d'un expert avec les données associées.
- toute autre ressource que vous jugerez utile pour votre application.

Par ailleurs, l'appel aux API devra être sécurisée par jeton JWT

### TODO

➔ **Conceptualiser le modèle de données** que vous aurez à construire (vous avez le choix de la base de données à utiliser pour stocker les informations)

➔ **Réaliser une application Java Spring Boot** sécurisée par jeton JWT permettant de fournir les ressources API nécessaires à votre application SPA.

➔ **Sauvegarder votre projet dans un repository GIT** (GitLab ou GitHub)

**RESSOURCES**

- Cf. kits précédents du socle CDA Java.

## 1.3 — Frontend - Application SPA Angular

18h — Présentiel

Maintenant que vous disposez de la partie serveur et base de données, il est temps de s'attaquer à votre application SPA Angular.

### TODO

➔ Initier un nouveau projet Angular (cf. Kit 2) ;

➔ Sauvegarder votre projet dans un repository GIT (GitLab ou GitHub) ;

➔ Réaliser votre application SPA grâce aux concepts et notions vues dans les Kits 1 et 2, et des ressources de ce Kit.

Quelque soit la conception que vous proposez, votre application devra nécessairement gérer les aspects suivant :

- gestion de la navigation ;
- gestion de l'authentification et de l'accès aux différentes pages (Route Guards) ;
- gestion des appels HTTP à vos ressources API Rest ;
- gestion de l'internationalisation (i18n) ;
- gestion du *responsive design* (adaptation de l'affichage en fonction de la taille de l'écran) ;

Par ailleurs, votre application devra également respecter les règles élémentaires d'accessibilité.

Vous pouvez vous aider des sections suivantes au fur et à mesure de votre progression dans la réalisation de votre application.

### 1.3.1 - Gestion de la navigation

La navigation dans votre application s'effectue de manière "classique" grâce au module natif *RouterModule* d'Angular.

**TODO**

➔ Configurer les *routes* de votre application grâce au *RouterModule* (cf kit 2) (vous pouvez également consulter le tutoriel Angular dans la liste des ressources).

**RESSOURCES**

- Tutoriel angular sur le routing (navigation) https://angular.io/guide/router-tutorial

### 1.3.2 - Gestion des accès (Route Guards)

La gestion des droits d'accès dans la navigation s'effectue au niveau de la configuration des *routes*, par l'ajout de la propriété *canActivate*.

### TODO

- ➔ Consulter la ressource https://guide-angular.wishtack.io/angular/routing/route-guards pour comprendre la manière dont Angular gère les droits d'accès dans la navigation et l'appliquer à votre projet.
- ➔ Adapter votre configuration des *routes* pour protéger les pages en fonction de l'utilisateur et ses droits.

**RESSOURCES**

- Explication du Route Guards : https://guide-angular.wishtack.io/angular/routing/route-guards

#### 1.3.3 - Gestion des appels HTTP au backend (ressources API Rest) ;

La navigation dans votre application s'effectue grâce au module angular *HttpClientModule*.

### TODO

- ➔ Consulter la ressource https://angular.io/guide/http pour comprendre comment envoyer une requête HTTP et communiquer avec votre backend Java.

- ➔ Utiliser le client *http* angular pour communiquer avec votre backend.

**RESSOURCES**

- Guide angular pour communiquer avec un backend https://angular.io/guide/http

#### 1.3.4 - Gestion de l'internationalisation

La gestion de l'internationalisation dans Angular se fait grâce à la dépendance *localize*, qu'il faut rajouter

### TODO

➔ Ajouter la dépendance permettant de gérer l'internationalisation au projet : `ng add @angular/localize`.

➔ Consulter la ressource https://angular.io/guide/i18n-overview pour comprendre la manière dont Angular gère l'internationalisation, et l'utiliser dans votre projet.

**RESSOURCES**

- Guide angular sur l'internationalisation https://angular.io/guide/i18n-overview

### 1.3.5 - Gestion du responsive design

Afin de faciliter la gestion et l'adaptation de la présentation des contenus en fonction de la taille des écrans, et plus globalement pour faciliter la construction de la partie *frontend*, nous pouvons utiliser la librairie *boostrap* (https://getbootstrap.com/).

Il existe plusieurs façons d'intégrer bootstrap dans une application Angular (consulter la ressource associée), mais pour faciliter son intégration nous pouvons utiliser la dépendance *ngx-bootstrap* (à noter que d'autres dépendances existent, comme *ng-bootstrap*).

### TODO

➔ Consulter la documentation de *bootstrap* et de *ngx-boostrap* pour comprendre les possibilités d'utiliser les fonctionnalités de bootstrap dans votre application.

➔ Ajouter la dépendance *ngx-bootstrap* grâce à la commande `ng add ngx-bootstrap` et suivre la documentation pour configurer correctement votre application.

➔ Consulter la documentation *Grid Layout* de bootstrap pour comprendre comment gérer les aspects *responsive design* dans votre application : https://getbootstrap.com/docs/5.3/layout/grid/

**RESSOURCES**

- Page du plugin *ngx-bootstrap* https://valor-software.com/ngx-bootstrap/#/
- Différentes façon d'intégrer bootstrap dans une application Angular https://www.knowledgehut.com/blog/web-development/install-bootstrap-in-angular

### 1.3.6 - Règles d'accessibilité

Avoir un site / SPA accessible consiste à faciliter son accès et son utilisation à tous, notamment aux personnes handicapées.

Pour celà, le W3C (consortium international qui définit les standards du web) a établi un certain nombre de règles à suivre et respecter dans la conception et le code d'un site internet.

**TODO**

➔ Consulter les ressources pour vous familiariser avec les attendus en termes d'accessibilité.

➔ Suivre autant que possible les règles d'accessibilité dans votre code et la conception de l'application SPA Angular.

➔ Utiliser une extension Chrome pour tester l'accessibilité de vos pages (comme IBM Equal Access Accessibility Checker https://chrome.google.com/webstore/detail/ibm-equal-access-accessib/lkcagbfjnkomcinoddgooolagloogehp/related)

**RESSOURCES**

- Page W3C sur les fondamentaux de l'accessibilité web : https://www.w3.org/WAI/fundamentals/accessibility-intro/fr

- Lien officielle pour la déclaration d'accessibilité (sous conditions) : https://design.numerique.gouv.fr/accessibilite-numerique/declaration-accessibilite/
- Formulaire W3C permettant de vérifier l'accessibilité d'un site internet : https://validator.w3.org/
- Article présentant des outils de vérification de l'accessibilité d'un site internet : https://geekflare.com/fr/test-web-accessibility/
- Article présentant 10 règles d'accessibilité https://cinqmars.fr/rgaa-10-regles-daccessibilite-pour-votre-site-internet/
- Article présentant 5 façons de rendre son site accessible : https://www.jimdo.com/fr/blog/5-facons-rendre-site-web-accessible-a-tous/
- Article présentant des outils pour les développeurs (extensions Chrome) https://code-garage.fr/blog/a11y-les-meilleurs-outils-pour-tester-l-accessibilite-dun-site-web/

## 1.4 — Frontend - Application SPA Angular (partie optionnelle) - — Présentiel

Cette partie concerne les fonctionnalités "optionnelles" du cahier des charges et s'adresse aux apprenants qui sont en avance sur le programme.

**Cette partie n'est pas obligatoire, et n'est pas à démarrer avant d'avoir terminé et validé les compétences du Kit.**

Si vous avez terminé et finalisé les parties précédentes, vous pouvez approfondir et compléter votre application SPA en tentant d'intégrer les fonctionnalités optionnelles du cahier des charges.

### TODO

➔ Compléter les fonctionnalités de votre solution en gérant les fonctionnalités back office.

## Livrables

Voici les livrables attendus à la fin de ce kit :

- ➔ Document d'analyse et conception du besoin client ;
- ➔ Argumentaire Angular vs VueJS vs React ;
- ➔ API Rest réalisée en JAVA Spring Boot en respectant les bonnes pratiques ;
- ➔ Application SPA réalisée en Angular respectant les contraintes techniques et fonctionnelles du cahier des charges (notamment *responsive design*, accessibilité, ...) ;