

Yacine
MAHMOUDI
BTS SIO 1^{er} année



Rapport de stage

Janin Consulting

Développeur Back-End et Front-End : Démarche
pour la création d'une application téléphone

13/05/2024 au 14/06/2024

Sommaire

Introduction	3
Développement	4
1. Développement Web (Front-End)	4
1.1 Création des Pages Dashboard	4
1.2 Programmation des pages (Angular et HTML/CSS)	5
1.3 Débogage du Front-End	10
2. Développement Java (Back-End)	11
2.1 Créations des Services et Controllers de l'API	11
2.2 Débogage de l'API	11
3. Répartition du Travail	12
3.1 Tâches Assignées	12
Conclusion	13
Annexe	14

Introduction

Dans le cadre de ma formation en BTS Services Informatiques aux Organisations, spécialité Solutions Logicielles et Applications Métiers (SLAM), j'ai effectué un stage au sein de l'entreprise Janin Consulting, du 13 mai au 14 juin 2024.

Les objectifs de ce stage étaient d'acquérir une expérience professionnelle afin de mettre en pratique et de consolider les connaissances obtenues tout au long de ma première année de BTS SIO. Grâce à cette expérience professionnelle, j'ai pu me rapprocher de mon objectif de devenir développeur web.

Ce stage m'a permis de m'exercer, notamment en créant un site web que j'ai ensuite converti en application mobile. Pour cela, j'ai travaillé en binôme avec un étudiant de ma promotion, mon camarade de classe Luc, et nous avons dû nous répartir le travail de manière efficace.

Le site ModelAgencyHub est une plateforme innovante qui simplifie la collaboration entre mannequins, agences de mannequins, et marques dans l'industrie de la mode et de la publicité. La plateforme vise à optimiser les interactions, la gestion et le recrutement parmi ces entités grâce à des outils intuitifs et puissants.

L'une des principales missions qui nous a été confiée était de créer les différents tableaux de bord (pages de gestion) et de faciliter l'accessibilité pour les administrateurs, managers, modèles et créateurs d'offres.

Développement

1. Développement Web (Front-End)

1.1 Création des Pages Dashboard

Tout d'abord, l'une des missions qui nous a été donnée afin de créer notre application web en Angular est de produire un prototype de page HTML et CSS. L'objectif est de créer une sorte de réseau social spécialisé dans le marché professionnel du mannequinat. Le but de ce site est de mettre en relation des mannequins et des agences en recherche de profils.

Il est important de rendre le réseau social ergonomique et facile d'utilisation pour les fonctionnalités proposées. Ces pages correspondent à des Dashboards (des tableaux de bord/gestion) pour différents rôles : admins, modèles, managers et créateurs d'offres.

Nous devons donc créer 7 pages différentes correspondant aux rôles mentionnés précédemment, en plus d'une page d'Accueil, une page pour modifier son profil et une page pour créer une offre.

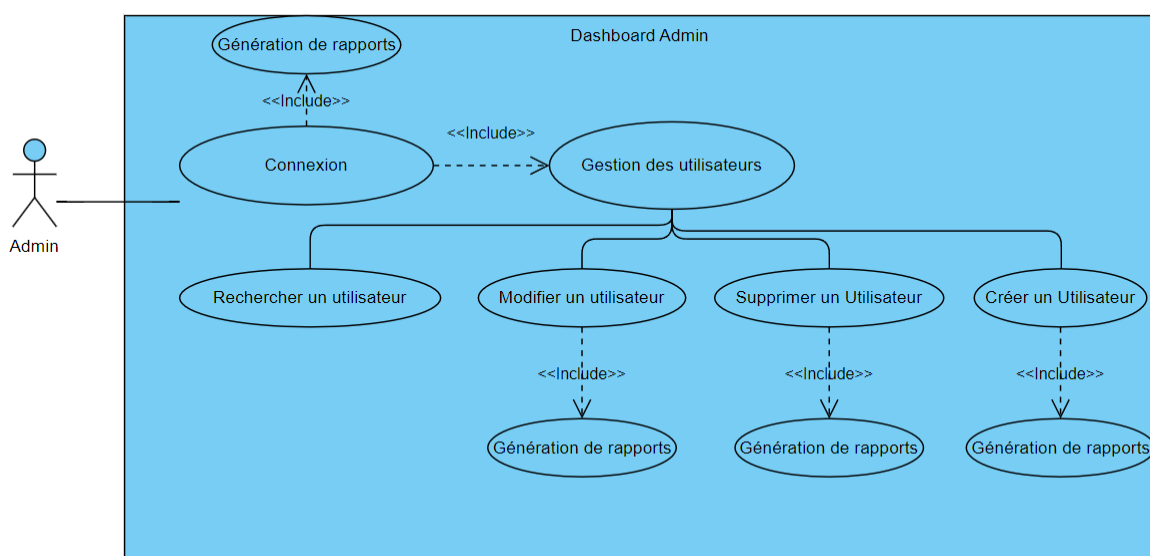
Pour les pages Admins (accès à tous les utilisateurs) et Managers (accès seulement aux utilisateurs modèles), nous avons les fonctionnalités d'ajout d'un utilisateur, la possibilité de le modifier ainsi que de le supprimer et pouvoir chercher l'utilisateur via son adresse mail ou nom. Nous avons les mêmes possibilités avec les offres et les réponses d'offres.

Avec la page Modèle, on peut voir les offres disponibles et les accepter ou non.

La page Offres permet de créer des demandes d'offres, les modifier ainsi que les supprimer selon les offres créées par l'utilisateur actuellement connecté, et pouvoir en créer d'autres en étant redirigé vers la page pour créer une offre.

Lors de la première semaine de notre stage, il nous a été demandé de rédiger un use case d'un utilisateur admin afin de créer l'admin Dashboard. Ce use case (diagramme de cas d'utilisation) va permettre d'expliquer les droits d'administrateurs.

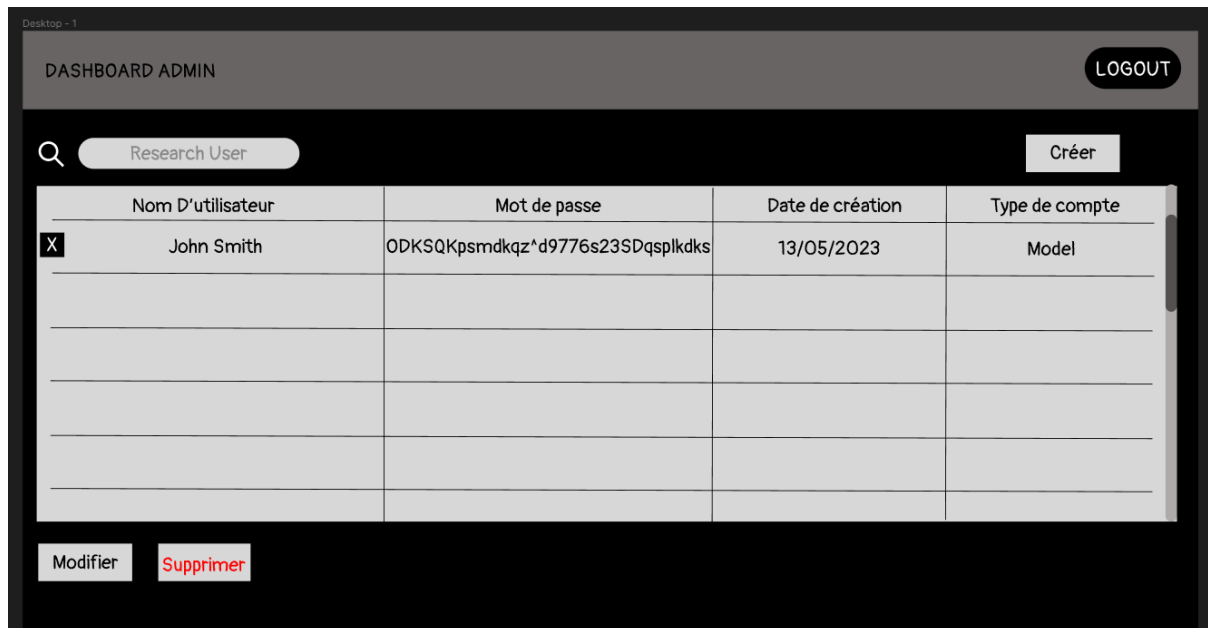
Voici un exemple d'use case produit :



1.2 Programmation des pages (Angular et HTML/CSS)

Au départ, notre tuteur nous avait proposé de laisser libre cours à notre imagination pour la page Admin. Cette liberté nous a aidés à répondre à la demande du client. Par la suite, pour créer nos pages, nous avons dû suivre un modèle offert par le tuteur. Il était indispensable de s'inspirer du design violet et moderne. Grâce à nos cours de HTML et CSS que nous avons suivis durant l'année, nous avons pu mettre en œuvre nos compétences et les améliorer.

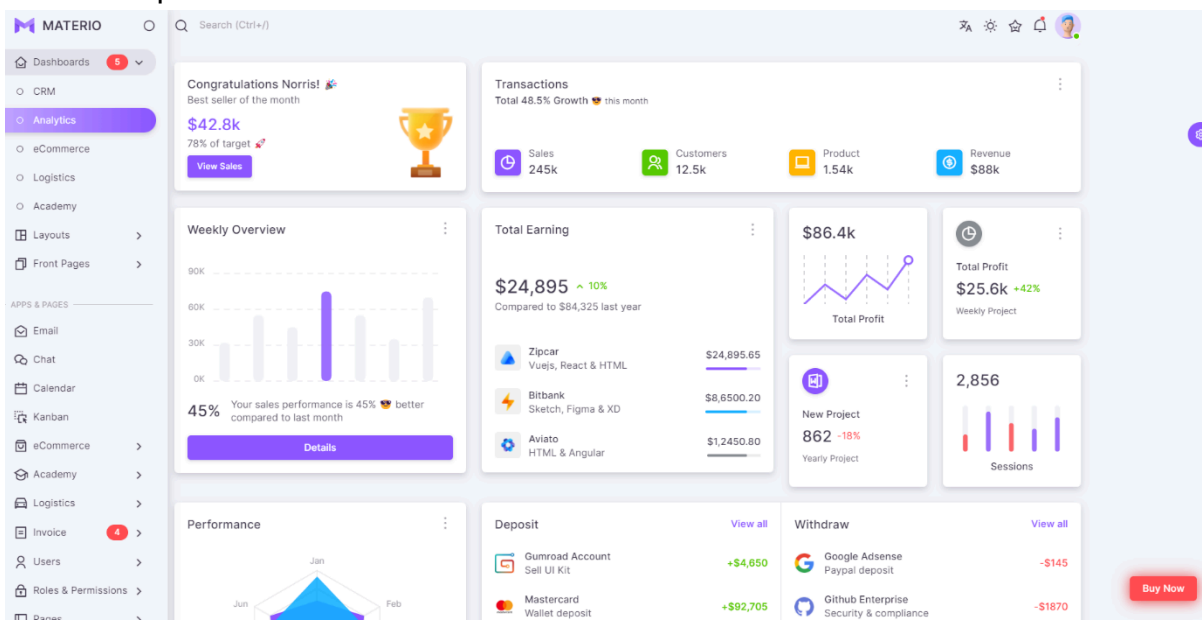
Notre base :



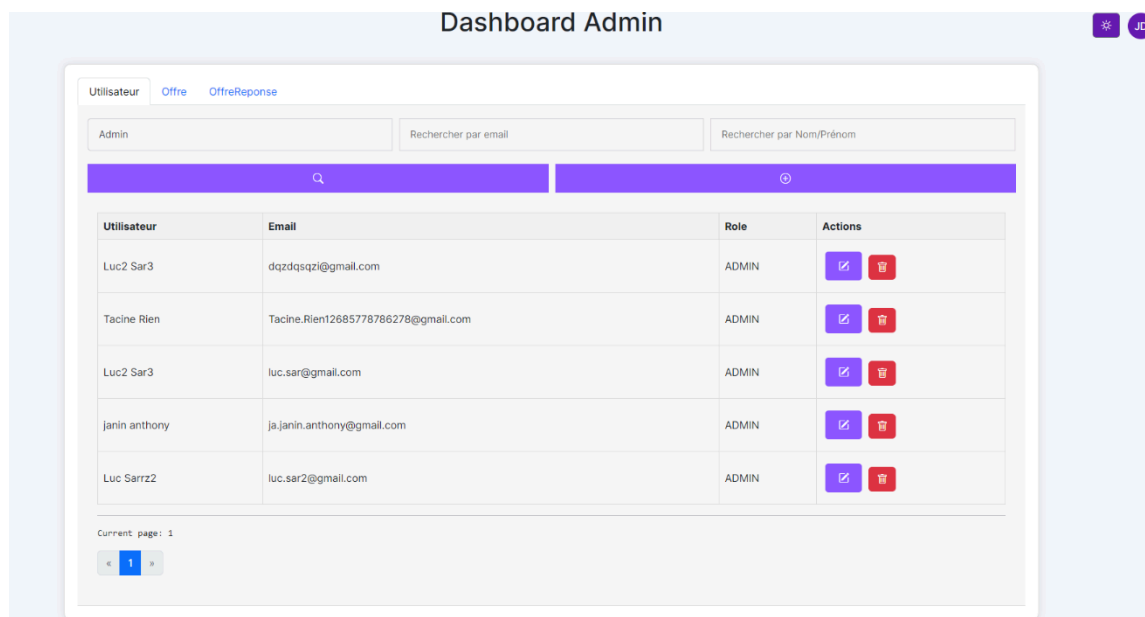
Voici la page HTML de la base :



Le template :



La page admin final :



Pour créer les différentes pages, nous avons utilisé Visual Studio Code, ce qui nous a permis de manipuler le HTML, le CSS et Angular (qui utilise TypeScript). Sur l'image ci-dessus, on retrouve plusieurs onglets donnant accès à différents tableaux permettant de voir les utilisateurs, les offres et les réponses aux offres.

Nous avons utilisé un framework, Bootstrap, qui a facilité la création des onglets et l'esthétique, notamment avec les icônes de la loupe et du "+" qui permet d'ajouter un utilisateur. On retrouve également une pagination pour les tableaux, réalisée grâce à une ligne de code intégrée dans Bootstrap avec une fonction permettant d'effectuer une action au changement de page.

```
<hr tabindex="1" />
<pre>Current page: {{ pageOffer }}</pre> <ngb-pagination
[(page)]="pageOffer"
[pageSize]="pageSize"
[collectionSize]="totalElementsOffer"
(pageChange)="onPageChangeOffers($event - 1)">
</ngb-pagination>
```

L'implémentation a été compliquée et nous avons dû chercher de l'aide dans les documentations, car lorsque nous appuyions sur les boutons permettant de passer d'une page à l'autre, le bon nombre d'utilisateurs dans la base de données ne s'actualisait pas correctement.

Les boutons de modification (que l'on retrouve dans la colonne "Action") et le "+" ont été créés grâce à des modals de Bootstrap, réalisés avec des classes et des divs.

Il y a des formulaires pour tous les modals afin d'envoyer les informations à la base SQL, que ce soit pour modifier ou créer un utilisateur en Angular :

```
this.setform = this.fb.group({
  firstName: ['', Validators.required],
  lastName: ['', Validators.required],
  email: ['', [Validators.required, Validators.email]],
  phoneNumber: this.fb.group({
    countryCode: ['', Validators.required],
    areaCode: ['', Validators.required],
    number: ['', Validators.required]
  }),
  address: this.fb.group({
    addressLine1: ['', Validators.required],
    addressLine2: [''],
    city: ['', Validators.required],
    state: ['', Validators.required],
    country: ['', Validators.required],
    postalCode: ['', Validators.required]
  }),
  organization: ['ROOT', Validators.required],
  roles: [], Validators.required
});
```

On retrouve donc sur cette image tous les champs à remplir sur le formulaire. Une fois que nous avons fini la page Admin, elle nous a servi de base pour toutes les autres. Ainsi, nous avons pu réutiliser toutes les méthodes Bootstrap et Angular créées pour cette page sur les autres.

Nous avons également mis en place un design responsive en utilisant du CSS grâce aux classes et ID, ce qui permet d'ajuster les éléments de la page pour le format téléphone. Les cours sur le CSS durant l'année nous ont permis d'acquérir des connaissances sur le responsive, ce qui nous a permis de les mettre en œuvre durant le stage.

Voici la page Modèle :

The screenshot displays a web application interface. At the top right, there are user avatars for 'JP' and another user. Below this, a search bar is labeled 'Rechercher des offres...'. A notification box on the left says 'Bienvenue Yacine Luc. Il y a un total de 12 offres disponibles. Vous avez répondu à 11 offres.' Below the notification is a table of offers with columns: Titre de l'offre, Statut, Modèle, Date de création, and Date de mise à jour. The table contains 10 rows of data. On the right, there is a larger table with columns: Statut, Lieu, Prix, Date d'expiration, Date de création, Date de mise à jour, and Actions. This table contains 8 rows of data, each with a corresponding 'Acc' button in the Actions column.

Statut	Lieu	Prix	Date d'expiration	Date de création	Date de mise à jour	Actions
CLOSED	tete	4	2024-06-22T00:00:00.000Z	2024-05-30T17:06:00	2024-06-07T16:04:11	Acc
OPEN	rtrtrtrtrt	99999	2024-07-07T00:00:00.000Z	2024-05-30T17:02:43	2024-06-04T17:40:56	Acc
OPEN	qzdsq	142	2024-06-04T00:00:00.000Z	2024-05-30T17:01:26	2024-05-30T17:01:26	Acc
OPEN	setdst	1242	2024-05-30T00:00:00.000Z	2024-05-28T17:55:29	2024-05-28T17:55:29	Acc
REFUSED	Paris	1242	2024-06-07T00:00:00.000Z	2024-05-30T17:01:01	2024-05-30T17:01:01	Acc
OPEN	qkez*pkdda	1251	2024-07-07T00:00:00.000Z	2024-06-06T11:30:15	2024-06-06T11:30:15	Acc

Voici la page Manager :

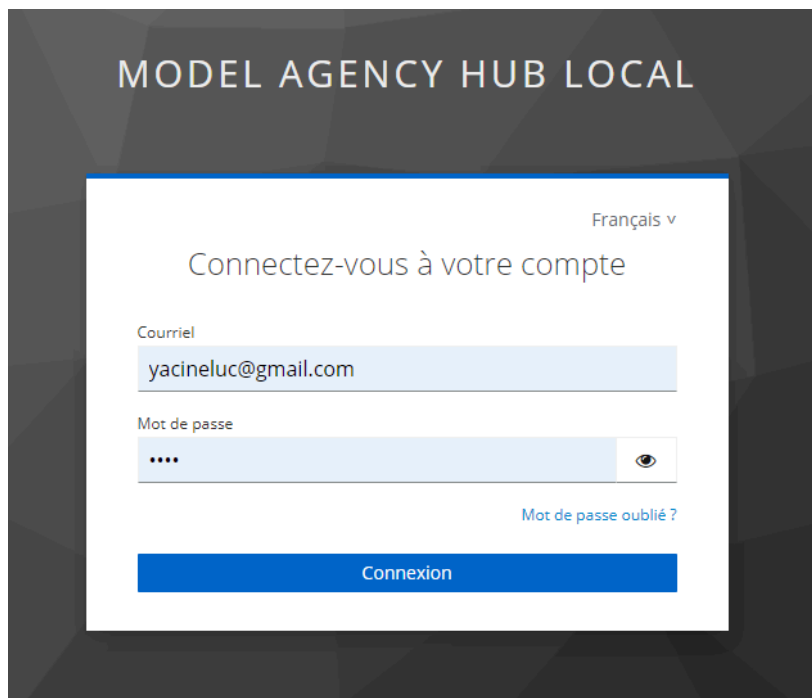
Voici la page Offer :

8

Nous avons également ajouté le mode sombre qui permet de rendre le fond blanc en noir grâce à des classes. Nous avons pu modifier la couleur de chaque composant, car sur toutes les pages, la structure est à peu près la même.

Il y a également une bulle "JD" (initiales du prénom et du nom de l'utilisateur) qui ouvre un petit menu fait avec des divs, permettant d'accéder à un bouton de déconnexion et un autre pour notre profil.

Voici le système de keycloak :



Ceci est le système qui nous permet de nous connecter à nos comptes et d'accéder à nos pages, qui sont accessibles uniquement selon nos rôles. Par exemple, avec ce compte qui est un compte Modèle, je serais redirigé vers la page Modèle.

Nous avons mis en place tout un système d'authentification et de redirection de pages intégré dans notre programme.

```
const routes: Routes = [
  { path: '', component: ModelDashboardComponent },

  { path: 'admin-dashboard', component: AdminDashboardComponent, canActivate: [AuthGuard], data: { roles: ['admin'] } },
  { path: 'model-dashboard', component: ModelDashboardComponent, canActivate: [AuthGuard], data: { roles: ['model'] } },
  { path: 'profil', component: ProfilComponent, canActivate: [AuthGuard]},
  { path: 'offer-dashboard', component: OfferComponent, canActivate: [AuthGuard], data: { roles: ['offer'] } },
  { path: 'create-offer', component: CreateOfferComponent, canActivate: [AuthGuard]},
  { path: 'manager-dashboard', component: ManagerDashboardComponent, canActivate: [AuthGuard], data: { roles: ['manager'] } },

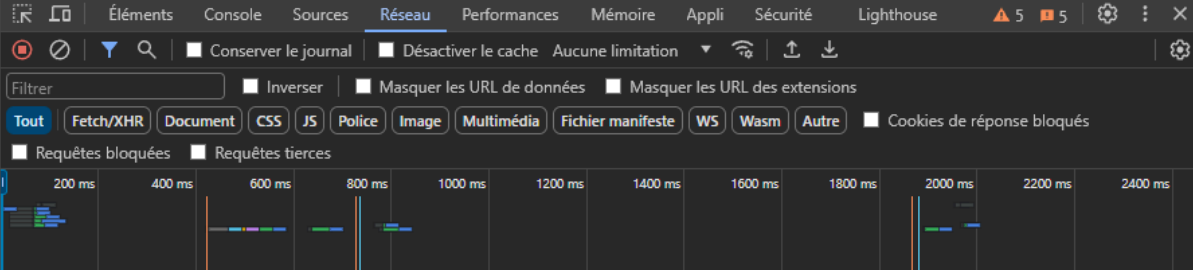
  // 404
  { path: '404.html', component: NotFoundPageComponent },
  { path: '**', redirectTo: '/404.html' },

  // otherwise redirect to home
  { path: '**', redirectTo: '' }
];
```

Ainsi, pour chaque nouvelle page créée, il faut définir une route en la reliant à Keycloak. Cela nous permet, lorsqu'on se connecte, d'atterrir sur la bonne page correspondant à notre rôle.

1.3 Débogage du Front-End

Le stage nous a appris à inspecter une page dans notre navigateur internet, ce qui nous a grandement aidés à identifier les problèmes et erreurs dans le code.



Nom	État	Type	Initiateur	Taille	Durée	Cascade
localhost	304	document	Autre	233 B	4 ms	
runtime.js	304	script	(index):11	234 B	7 ms	
polyfills.js	304	script	(index):11	235 B	10 ms	
vendor.js	304	script	(index):11	236 B	65 ms	
main.js	304	script	(index):11	235 B	28 ms	
styles.js	304	script	(index):11	235 B	27 ms	
css2?family=Inter:wght@100..900&displa...	200	stylesheet	styles.css:1	(cache mé...)	0 ms	
UcCO3FwrK3iLTeHuS_fvQtMwCp50KnMw...	200	font	css2	(cache mé...)	0 ms	
UcCO3FwrK3iLTeHuS_fvQtMwCp50KnMw...	200	font	css2	(cache mé...)	0 ms	
UcCO3FwrK3iLTeHuS_fvQtMwCp50KnMw...	200	font	css2	(cache mé...)	0 ms	
UcCO3FwrK3iLTeHuS_fvQtMwCp50KnMw...	200	font	css2	(cache mé...)	0 ms	
ng-cli-ws	101	websocket	polyfills.js:1	0 B	En attente	
step1.html	200	document	vendor.js:100555	2.6 kB	132 ms	
step2.html	200	document	security.wasterzy.co...	992 B	39 ms	
login-status-iframe.html	200	document		4.4 kB	34 ms	
favicon.ico	304	vnd.micros...	Autre	233 B	6 ms	
init?client_id=model-agency-hub&origin...	204	fetch	login-status-iframe.h...	163 B	33 ms	
1.png	200	png	home.page.compon...	(cache mé...)	1 ms	
2.png	200	png	home.page.compon...	(cache mé...)	0 ms	
3.png	200	png	home.page.compon...	(cache mé...)	0 ms	
data:image/svg+xml,...	200	svg+xml	styles.css	(cache mé...)	0 ms	
bootstrap-icons.woff2?24e3eb84d0bcaf8...	304	font	styles.css	235 B	7 ms	

En allant dans l'onglet Réseau, nous avons accès à toutes les informations entrantes et sortantes du site internet, telles que les logos Bootstrap ou les requêtes de recherche d'utilisateurs.

Nous avons également utilisé cet outil pour récupérer le token d'authentification, nécessaire pour obtenir l'identifiant de l'utilisateur connecté. Chaque utilisateur génère automatiquement un token sécurisé de manière aléatoire.

Dans l'onglet Console, nous pouvons voir toutes les erreurs ou avertissements de programmation grâce aux fonctions comme `console.log` ou `console.warning`. Cela fonctionne un peu comme un `"System.out.println"` en Java, utilisé pour le débogage ou pour afficher des messages informatifs pour l'utilisateur.

Pour résoudre les différentes erreurs rencontrées, cette interface nous a permis de copier les messages d'erreur pour effectuer des recherches sur internet ou demander de l'aide à notre tuteur afin de trouver des solutions.

On a aussi nettoyé le front pour une meilleure visibilité des erreurs.

2. Développement Java (Back-End)

2.1 Créations des Services et Controllers de l'API

Durant notre stage, nous devons créer une API pour toutes les fonctionnalités des tableaux de bord, telles que la recherche par nom/mail ou la modification/suppression/ajout d'un utilisateur, d'une offre ou d'une réponse à une offre.

Pour cela, notre tuteur nous a fourni un code GitHub comprenant le début d'une API basée sur Java Spring Boot, que nous avons ensuite complété en développant des contrôleurs (controllers) et des services.

Exemple de controller :

```
@GetMapping("/models")
public ResponseEntity<UserModelDto> findModelByEmail(@RequestParam String email) {
    return ResponseEntity.of(Optional.of(email) Optional<String>
        .flatMap(userService::findModelByEmail) Optional<UserModel>
        .map(userModelDtoMapper::toDto));
}
```

Exemple de service :

```
public Optional<UserAdmin> findAdminByEmail(String email) { 1 usage
    return Optional.ofNullable(email) Optional<String>
        .flatMap(userAdminJpaRepository::findByEmail) Optional<UserAdminEntity>
        .map(userAdminEntityMapper::toModel);
}
```

Le contrôleur est l'un des codes qui nous permettra de communiquer entre le Front et le Back. Le service est la ligne de code qui sera exécutée lorsque le contrôleur le sera. Par exemple, lorsque dans le contrôleur, findModelByEmail sera exécuté par le front, le contrôleur va exécuter la ligne dans le service findAdminByEmail et va la transmettre au contrôleur, qui la transmettra au front.

2.2 Débogage de l'API

Notre tuteur nous a enseigné l'art du débogage avec IntelliJ IDEA, qui dispose d'une fonctionnalité intéressante : les breakpoints. Ces derniers permettent de suspendre l'exécution du programme à des points spécifiques, afin de pouvoir avancer pas à pas dans le code.

```
Hibernate: select l1_0.user_model_entity_id,l1_0.languages from user_model_entity_langua
Hibernate: select oe1_0.id,oe1_0.compensation,oe1_0.creation_date,oe1_0.description,oe1_
Hibernate: select count(oe1_0.id) from offer oe1_0 where 1=1
```

3. Répartition du Travail

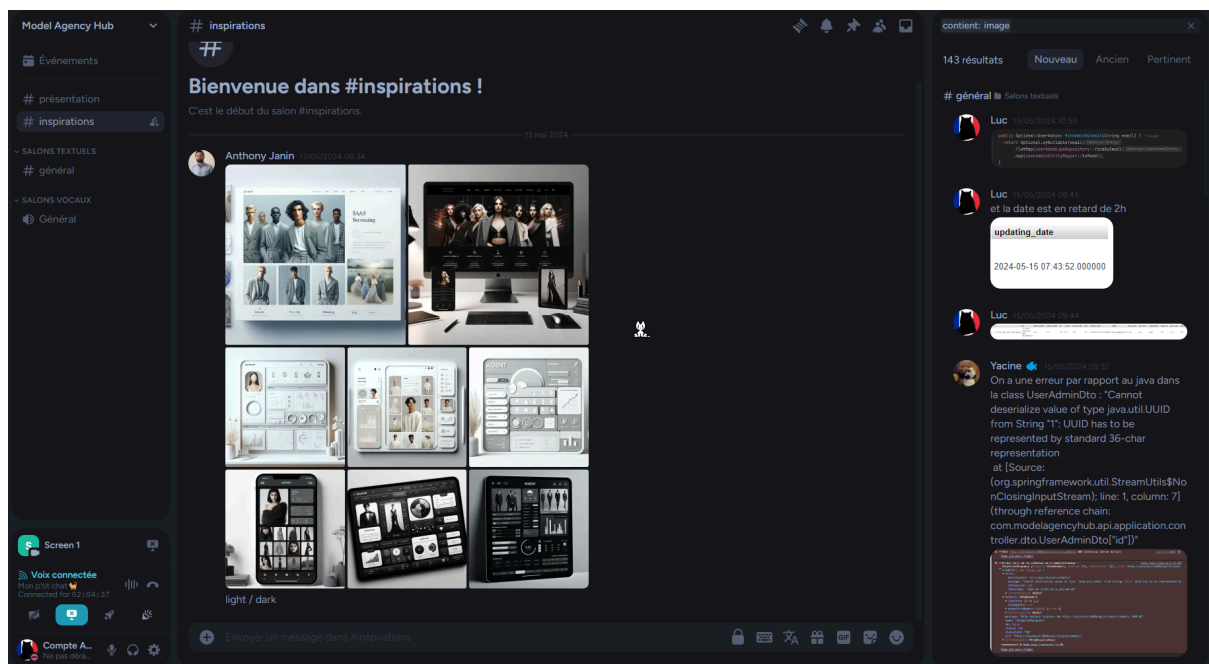
3.1 Tâches Assignées

Pendant le stage, nous étions deux, donc nous avons dû répartir les tâches et nous entraider. Pour nous organiser, nous avons utilisé Visual Studio Code et IntelliJ IDEA afin de travailler sur les mêmes fichiers et de modifier le code simultanément.

Certains d'entre nous se sont concentrés davantage sur certaines pages que d'autres. Par exemple, j'étais responsable de la page Manager tandis que Luc travaillait sur la page Admin. En cas de besoin ou de problèmes, nous pouvions nous entraider. Le fait d'être deux sur ce projet a renforcé notre travail d'équipe et nous a permis d'être plus efficaces et rapides dans nos tâches, ce qui a conduit à une finalisation plus rapide des projets (Web et Mobile).

Nous avons utilisé l'application Discord pour faciliter le partage de fichiers et la communication entre nous. Un serveur créé par le tuteur nous a été mis à disposition, présentant le réseau social et ses différentes fonctionnalités. De plus, le tuteur nous a fourni quelques images d'inspiration pour nous aider à visualiser le produit final.

Voici une image du serveur discord :



Pendant le stage, j'ai développé les pages Manager, Model, Create Offer, et la page d'accueil en collaboration avec Luc. Quant à Luc, il s'est occupé des pages Admin, Offer, Profil et également de la page d'accueil.

Le seul inconvénient était que les fichiers étaient stockés sur l'ordinateur de Luc, ce qui signifiait que je n'avais pas accès à eux sans lui. Par exemple, la base de données SQL était différente, donc nous devons constamment synchroniser nos

fichiers. Cela nécessitait une rigueur dans le partage des fichiers, heureusement, nous avons tous les deux accès à un GitHub partagé par notre tuteur.

Conclusion

Finalement, ce stage d'un mois m'a permis de découvrir le monde professionnel et les différents aspects d'un projet. Pouvoir aborder l'informatique dans ce genre d'environnement discipliné m'a fait gagner en maturité et en responsabilité.

Sans parler des connaissances acquises tout au long du développement en codant dans différents langages : HTML, CSS, Angular, Java Spring Boot, MySQL, Bootstrap, et Ionic. J'ai pu mettre en pratique toutes les connaissances et compétences acquises lors de ma première année de BTS SIO. Ce projet m'a également appris la création d'un site web et la gestion d'une base de données. J'ai acquis de nouvelles compétences et notions utiles comme le débogage, les API, les tokens, et le push sur GitHub. Cette expérience a également renforcé mon autonomie à travers la recherche de solutions à des problèmes donnés. J'ai appris à gérer plusieurs logiciels : IntelliJ IDEA, Visual Studio Code, XAMPP (PhpMyAdmin, Apache).

Nous avons pu aussi commencer à mettre en place une application mobile à l'aide de Ionic, mettant en pratique notre capacité à nous adapter à un nouveau langage.

De plus, le fait d'être avec un camarade de classe m'a confronté au travail d'équipe et m'a appris à m'organiser afin de mener à bien un projet. Savoir structurer le travail pour construire un projet est l'une des choses les plus importantes.

Enfin, cette expérience a conforté mon projet professionnel de devenir développeur web à travers les différentes connaissances que j'ai pu acquérir.

Je remercie sincèrement mon tuteur Anthony Janin de nous avoir accueillis durant ce stage d'un mois, d'avoir été là pour nous aider et nous faire progresser.

Annexe

Logiciels utilisés

- Visual studio Code
- IntelliJ Idea
- XAMPP
- Visual Paradigm

Langage de programmation utilisés

- HTML/CSS
- Java Spring BOOT
- Angular (Typescript)
- MySQL

Framework utilisés

- Bootstrap
- Ionic

Compétences Acquis

- Autonomie
- Recherche de solutions
- Organisation
- Travail d'équipe
- Adaptabilité

Liens utiles (Documentation) :

- Template : <https://demos.themeselection.com/materio-bootstrap-html-admin-template/html/vertical-menu-template/index.html>
- Bootstrap Docs : <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
- Bootstrap Icon : <https://icons.getbootstrap.com/>
- Ionic Docs: <https://ionicframework.com/docs>
- Géolocalisation : <https://www.freecodecamp.org/news/how-to-get-user-location-with-javascript-geolocation-api/>
- Keycloak on Ionic: <https://medium.com/@derjoba/using-keycloak-with-ionic-capacitor-9235be71bdf5>
- Explication Token JWT: <https://grafikart.fr/tutoriels/json-web-token-presentation-958>
- UUID in Java Spring Boot: <https://www.baeldung.com/java-uuid>

Page Home du site en Ionic et Angular :

Bienvenue sur ModelAgencyHub

Votre porte d'entrée vers les meilleurs mannequins du monde entier.



Description

ModelAgencyHub

C'est une plateforme innovante qui simplifie la collaboration entre mannequins, agences de mannequins, et marques dans l'industrie de la mode et de la publicité. La plateforme vise à optimiser les interactions, la gestion et le recrutement parmi ces entités grâce à des outils intuitifs et puissants.

Nos services

Gestion des modèles

Nous gérons les meilleurs talents et les mettons en relation avec des marques et des agences de premier plan.

On live

La fonctionnalité "On Live" est conçue pour répondre aux besoins immédiats des marques, permettant la création d'offres qui nécessitent une action rapide, comme trouver un mannequin pour une session d'essayage dans l'heure. Utilisant la géolocalisation, cette fonctionnalité permet de matcher les mannequins disponibles et proches pour une réactivité sans précédent.

Interface personnalisée

Pour chaque profession, il y a une interface spécifique pour correspondre aux besoins de l'utilisateur afin de le satisfaire et lui faciliter la tâche.

Rencontrer notre équipe



John Doe

CEO



Jane Smith

Directeur créatif



Mike Johnson

Responsable du marketing

Nous contacter

Envoyer