





Titre professionnel  
Développeur web et  
web mobile



faire un sommaire a la fin

# Remerciement

Je tiens à remercier Mich  al Zani ainsi que Fran  ois Autheman de leurs confiance et qui m'ont accueillis    bras ouverts dans leurs entreprise, la bonne humeur et   galement l'ann  e pass  e avec eux et tous les associ  s.

## Résumé

Dans le cadre de ma reconversion professionnelle, j'ai entrepris une formation en tant que développeur web et web mobile durant un an, avec l'école [WebForce3](#).

J'ai eu l'occasion de réaliser mon alternance avec l'entreprise [AMX One Solutions](#).

Au sein d'une équipe de deux développeurs, d'un alternant développeur, un responsable de la sécurité, ainsi que deux experts en finance s'occupant de la partie client, j'ai eu pour mission d'effectuer de nouvelles features\* créées par des demandes clients ou des idées sortant de réunion permettant l'amélioration de l'expérience d'utilisateur, d'améliorer le code existant et de résoudre des bugs rencontrés.

Lors de ce rapport d'alternance je vous présenterai deux projets :

[Le logiciel AMX1](#)\*\* permet de simplifier et d'automatiser tous les processus de reporting d'une société de gestion de portefeuilles.

Principalement développer grâce au langage de programmation Angular ce site web / application est une solution Saas de reporting adressée aux sociétés de gestion, banques privées qui permettent de créer des reportings avec une qualité professionnelle et de façon simple et intuitive.

**Tâches à réaliser visant une amélioration\***

**AMX One Solutions\*\***

pour la suite de cette présentation j'utiliserais l'abréviation AMX1

Et puis un 'intranet' ([eQuinox](#)) à la demande de l'entreprise avec comme objectif de réduire leurs processus tels que : permettre de faire des bons de commande et générer l'environnement sur le logiciel ci-dessus, afficher les différents prospects/clients et pouvoir suivre l'avancement afin de ne pas relancer un client qui a déjà été contacter, avoir un regard sur l'état des serveurs et des certificats permettant la connexion au logiciel [AMX1](#). Cet intranet a été développer avec le langage de programmation [Angular](#) également pour la partie front-end avec la librairie [Angular Materials](#), et également les contrôleurs permettant les requêtes et les appels à la base de données utilisant des fichiers principalement en [TypeScript](#).

Ces deux projets sont liés dans le sens ou quand un prospect devient un client, ce dernier se connectera sur eQuinox afin de fournir tous les dossiers nécessaires au bon paramétrage du premier logiciel AMX1. Il aura également accès à un tchat où il pourra discuter avec la personne en charge désignée si quelconques problèmes intervient.

J'aborderais ces sujets un peu plus tard dans cette présentation pour que cela soit plus évident.

# I. Contexte

## 1 - Présentation de l'entreprise

AMX One Solutions est une start-up de cinq employés et deux associés dont le siège social est situé à Paris et des bureaux dans le sud de la France à Lambesc. La société a été créée en octobre 2020 et la solution (Que l'on pourrait appeler un logiciel) a été développée la même année.

AMX One Solutions propose ses services aux sociétés de gestion de portefeuilles leur permettant d'automatiser, et de faciliter les processus de reporting financier devenu aujourd'hui très long avec un coût opérationnel très élevé. En proposant leurs services les processus de reporting deviennent simples et intuitifs, source de création de valeurs, accessibles n'importe où et configurables en quelques clics.

### Les services proposés



#### Rapport de gestion

- ✓ Reporting des fonds
- ✓ Relevé périodique des mandats
- ✓ Reporting des profils



### Rapport de performance

- ✓ Calcul des performances
- ✓ Calcul des contributions
- ✓ Calcul des frais



### Rapport de risques

- ✓ Calcul des indicateurs
- ✓ Suivi des ratios de risque
- ✓ Suivi des expositions

## II. Général

### 1 - Les utilisateurs

Les utilisateurs finaux sont les analystes financiers de société de gestion de portefeuilles. Parmi ces derniers nous pouvons retrouver des contrôleurs des risques, middle office, chargé de reporting, des gérant de portefeuille également des RCCI (Responsable du contrôle interne et conformité) etc. Néanmoins le logiciel se veut simple et intuitif à l'ensemble des collaborateurs de la société.

### 2 - Les besoins fonctionnels

#### 2.1 - La connexion

Un certificat de connexion au logiciel est nécessaire pour y accéder.



Chaque membre ayant un certificat de connexion a leurs adresses email ainsi que leurs informations personnelles telles que leurs :

- Prénoms
- Noms
- Poste (Job occupé au sein de l'entreprise)
- (Adresse email)

Toutes ces informations sont demandées afin de réaliser la création d'un utilisateur (user) en base de données, à la suite ils recevront un email d'invitation pour se connecter.

En amont, après la signature du contrat, on utilise l'intranet "[eQuinox](#)" qui sert "d'onboarding client", nous affichons explicitement les documents où nous avons besoin au bon paramétrage du logiciel AMX1 ces derniers sont :

- Les informations (nom de l'entreprise et adresse)
- Un profil administrateur (qui aura le droit de changer le nom, supprimer ou ajouter plusieurs services et d'en changer les droits)
- Fournir les adresses de tous ses collaborateurs

Les documents importants au set-up du logiciel sont plus précisément :

- Une liste des clients avec les informations comme le titulaire du compte, la civilité, l'adresse, le mail et le profil de risque
- La liste des portefeuilles contenant plusieurs informations ;

Dans le cas d'un [mandat](#), on doit avoir... Le nom du mandat, le numéro de compte, le/la titulaire du compte, le/la dépositaire du compte, la devise, le type de gestion, la date de début de gestion et la compagnie d'assurance ainsi que des informations qui pourrait sembler importantes.

Pour les [fonds](#) (financier) on doit avoir... le nom du fonds, le nom des parts, le code ISIN\* (du fonds et/ou des parts), la devise, l'indice de référence, le SRRI\*\*, le domicile du fonds ainsi que des informations qui pourrait sembler importantes.

Et enfin pour la liste des portefeuilles nous finissons sur les [profils](#), pour ceux-là nous avons besoin du nom du profil, l'ISIN, la devise, la compagnie d'assurance, la date de début de gestion, les frais annualisés et également les informations qui pourraient sembler importantes.

Nous continuons sur les documents importants au set up. Nous avons aussi besoin de :

- La liste à jour des VL\*\*\* de vos fonds
- Les reportings de vos fonds, mandats et profils (ou un modèle de reporting qu'ils voudraient paramétrer)

### ISIN\*

Le code ISIN (International Securities Identification Numbers). Il est utilisé pour identifier un instrument financier (action, obligation, OPCVM ...) lors d'une transaction.

### SRRI\*\*

Synthétic Risk and Reward Indicator, que l'on peut traduire en Français par indicateur synthétique de risque et de rendement. Le SRRI évalue le profil de risque d'un placement.

- DICI\* de vos fonds nécessaires pour le paramétrage des ratios
- La liste et composition des indices de référence composites (un indice donne de précieuses indications. Pour avoir une vision plus large et plus juste.)
- Un exemple des fichiers fournis par leurs valorisateurs pour la valorisation des fonds
- Les identifiants du ou des fournisseur(s) de leurs données financières

## 2.2 Les clients

L'utilisateur (de la société de gestion) doit pouvoir (suivant ses droits et ses services attribués) :

Administrateur = \*

- Ajouter/supprimer un utilisateur/collaborateur\*
- Pouvoir importer des listes de clients, des mandats, fonds, profils(financiers), des mouvements ainsi que des opérations, des VL et cours
- Pour les contrôleurs des risques ils doivent avoir la possibilité de prévenir et/ou d'envoyer un dépassement qu'il y aurait eu sur un mandat, un fond, ou un profil

### **VL\*\*\***

La valeur liquidative correspond à la valeur des actifs nets d'un organisme de placement collectif divisée par le nombre de parts ou d'actions en circulation, une fois ajoutés ou défalqués les droits d'entrée ou de sortie.

### **DICI\***

Le DICI est un document pré-contractuel qui doit être remis à l'investisseur préalablement à sa souscription. Il lui permet de prendre sa décision d'investissement en toute connaissance de cause.

### 2.3 Les contraintes

Les principales contraintes que nous avons côté développement et la réutilisabilité du code, pouvoir avoir des "templates" de pages pour une uniformité de l'UI, des composants ayant la même UI mais différents parmi les paramètres d'entrée. La facilité de compréhension du code et l'optimisation.

### 2.4 Les difficultés

La principale difficulté a été de coder en même temps de découvrir le langage Angular.

D'apprendre et de comprendre le code existant, le code ayant très peu de commentaire, je remercie François d'avoir toujours été présent pour m'expliquer lors des premières semaines.

### 2.4 Les livraisons

En fonctionnant en méthode agile, sur une période de 4 semaines (un sprint), nous (devs) avons des tickets attribués avec une estimation de temps sur la période du sprint ensuite nous poussons notre code (push) sur une branch de développement et les livraisons se font tous les Jeudis sur un serveur de "test", dont tous les développements vont être testés, en condition utilisateur.

Une fois les features, bugs, développés, résolus ils seront push(é) le dernier jeudi de la 4e semaine. Dans le cas où il y aurait des bugs suite aux devs précédents, la 4e semaine sert à la résolution de bugs et/ou de veille technologique.

# Explication d'un ticket

AMX One  
solutions

## Qu'est-ce qu'un ticket ?

### Une FEATURE / BUG:

- Ecrite par la TEAM PRODUCT
- Doit-être claire, compréhensible et non interprétable par la TEAM DEV

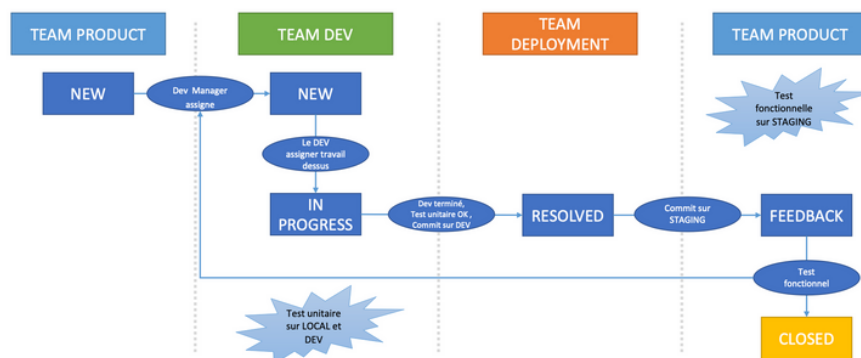
### Une/des USER STORY:

- C'est la FEATURE coupé en tâches de développement
- Ecrite par le développeur qui prend le ticket
- Faire un commit unique par user story
- Terminée quand toutes ses tâches sont terminées et testées

DONE quand toutes les USER STORY sont terminées et testées

AMX One  
solutions

## Vie du ticket



AMX One  
solutions

## Qu'est-ce qu'un commit ?

### Granularité:

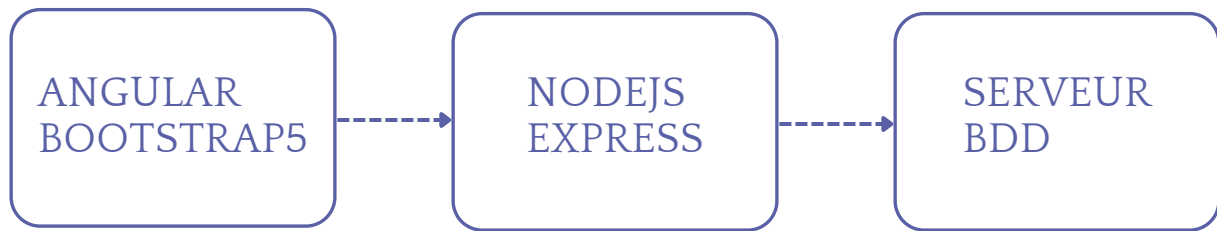
- USER STORY
- TECHNICAL STORY
- BUG

### Commentaire:

- [BUG] #NUM\_TICKET:Nom USER STORY => explications
- [FEATURE] #NUM\_TICKET:Nom USER STORY => explications
- [WIP] #NUM\_TICKET:Nom USER STORY => explications
  - Commit intermédiaire:
    - Développement non fini
    - Tâche du USER STORY
  - Interdit dans l'arbre de commit de STAGING (seront écrasé par le commit final)

## II. Les technologies utilisées

### 1 - Modèle de l'application et environnement technique (AMX1)



BDD = Base de données

#### Pour le frontend

Nous utilisons principalement Angular, avec Bootstrap 5 pour la création de l'interface utilisateur. Les outils :

- Angular
- Bootstrap 5
- HTML / SCSS

#### Pour le backend

Nous utilisons principalement "NodeJs" et "Express" permettant de faire des contrôleurs. Ces derniers font le lien entre la demande via l'interface utilisateur et la requête SQL sur la base de données.

Les outils :

- NodeJs
- Express
- MySQL
- TypeScript
- PostgreSQL

## Outils de collaboration et versionning

- Git (SourceTree, Gitlab, Bash, Cygwin)
- Teams
- Slack

## Outils de test

- Postman
- SGBD (Système de gestion de base de données)
- Serveur de test pour tous les devs en cours ou future

Nous avons également une personne ce serveur, faisant la passerelle entre le serveur de développement et celui du client comme j'expliquais un peu plus haut dans mon point "2.4 Les livraisons".

## Déploiement

- Docker
- CI/CD Gitlab

Quand on pousse le code (push) sur une branche en particulier gitlab créer un docker avec une image allégée de type alpine, gitlab récupère (pull) et fait le premier job.

Nous avons le build dans un autre conteneur (docker) trion-ng qui contient tous les outils Angular.

Et pour finir un troisième conteneur (alpine) qui déploie via rsync (over SSH).

## Quelques définitions

Docker (conteneur) conviennent aux situations où vous souhaitez exécuter plusieurs applications sur un seul système d'exploitation. À l'inverse si vous avez des applications ou des serveurs qui doivent s'exécuter sur différentes versions de système d'exploitation, le choix se portera vers une machine virtuelle.

Trion-ng c'est une image docker pour Angular permettant de créer un container.

Rsync (pour "remote synchronization" ou synchronisation à distance), est un logiciel de synchronisation de fichiers.

Alpine Linux est une distribution Linux ultra-légère, orientée sécurité, principalement conçue pour un « utilisateur intensif qui apprécie la sécurité, la simplicité et l'efficacité des ressources.

SSH (Secure Shell) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion.



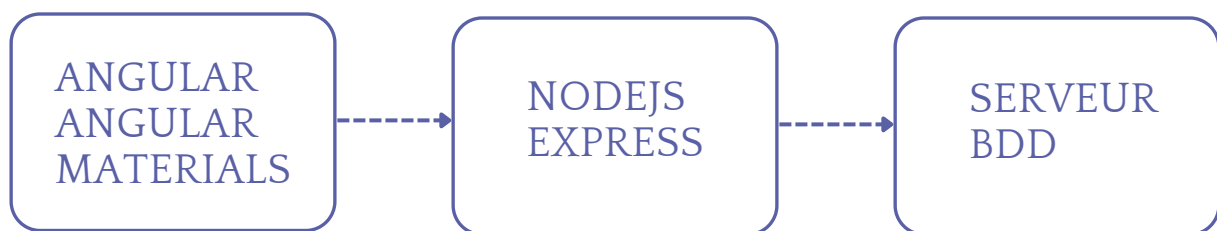
Le choix du développement en frontend s'est porté sur la bibliothèque [Angular](#). Celle-ci est devenu un choix de choix pour les applications Web, ayant une architecture MVC ([modèle dans la conception de logiciels](#)) et des outils robustes, en particulier pour les applications à page unique.

[Angular](#) a été un choix pour François qui était le seul développeur à la création, étant le langage le plus courant, et soutenu par Google.

[Angular](#) facilitant l'affichage rapide des composants et avantage la fluidité de la navigation au sein du logiciel.

Le but est de charger des composants les plus génériques possible qui selon leurs entrées auront des états, des affichages différents.

## 2 - Modèle de l'application et environnement technique (eQuinox)



### Pour le frontend

Pour l'intranet nous avons utiliser également Angular et la librairie Angular Materials pour les mêmes raisons énoncées plus haut. Les outils :

- Angular
- Angular Materials
- HTML / SCSS

### Pour le backend

"NodeJs" et "Express" sont également présents pour la création de contrôleurs ainsi que TypeScript. Les outils sont les mêmes que pour AMX1.

## III. Le développement

### 1 - Environnement humain

Pour la partie développement, j'ai travaillé avec Nathan Girard un autre alternant en développement et Estelle Leonetti, en charge de tester les développements. Nous étions sous le manager de la "team dev", tuteur d'alternance et développeur, François Autheman.

#### 1.1 - Premiers temps dans l'alternance

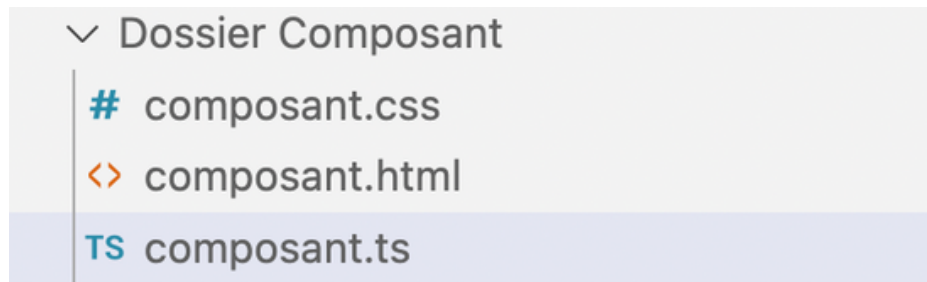
Le projet d'intranet a été amené lors de l'entretien de recrutement. Dans un premier temps nous (Nathan et moi-même) avons établi les besoins de l'intranet avec l'entreprise qui nous a permis de faire un premier schéma de base de données, notre travail commun a été de développer cet intranet au cours de l'apprentissage en ayant un rythme 60% AMX1, 40% eQuinox.

Pour les besoins du logiciel je suis rapidement passé sur le projet d'AMX1, début Mars mes premières missions ont été de :

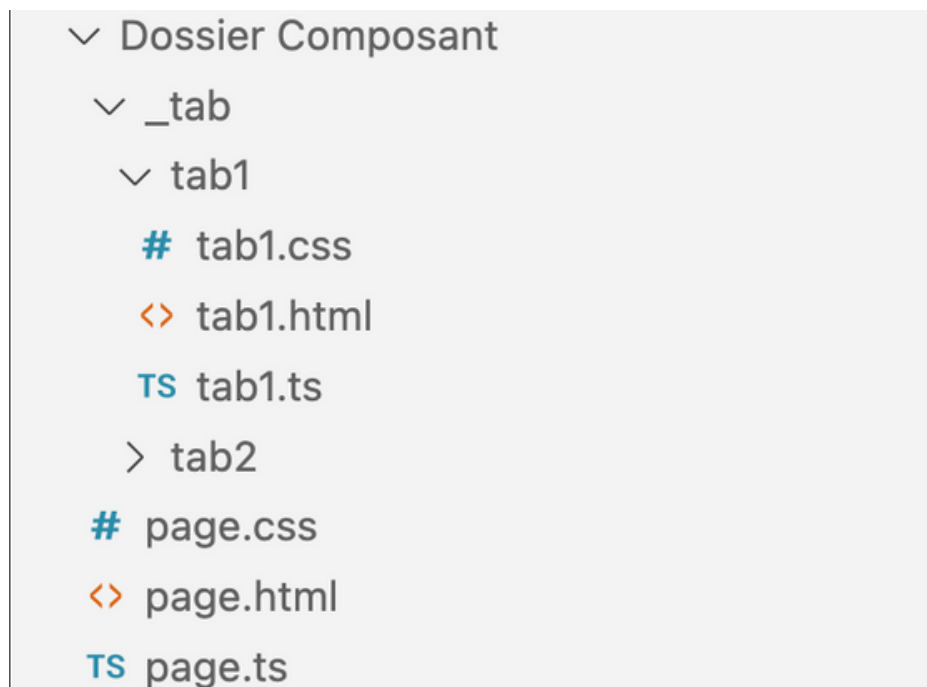
- Faire une passe sur tout le site (interface utilisateur) afin de bien comprendre le public visé.
- Améliorer le code existant, supprimant les parties inutilisées, réduire les répétitions, clarifier l'architecture du code.

Un exemple de clarification du code existant :

Avoir ceci



Transformer en ceci



Séparer le code d'une page qui tenait sur une seule page HTML en plusieurs "tabs", onglets la composant pour une meilleure lisibilité du code, pouvoir retrouver plus facilement la page ou l'onglet lors d'une future feature. On parle d'ici "d'éclatement" des pages. Cette première mission m'a permis de me familiariser avec le code existant ainsi que de connaître les différents composants existants.

Ensuite, une des autres parties de code sur lesquelles je suis intervenu a été la refonte ainsi que l'amélioration de tous les droits et services.

## Explications des droits et services

Un utilisateur ayant la case "OUI", aura accès à la page, à l'action concernée.

L'inverse est également valable, un utilisateur qui n'aurait pas le droit d'accéder à une page ou à action spécifique aura un "NON" dans la case concernée.

Un utilisateur peut avoir un seul service.

Un service peut avoir plusieurs droits.

Plusieurs utilisateurs peuvent faire parti du même service.

Seul l'administrateur du logiciel [AMX1](#) à accès à la modification des droits, des services, et leurs accès à travers tout le logiciel.

## Onglet des autorisations

The screenshot shows the 'Autorisations' (Permissions) tab in the AMX1 software. The interface includes a sidebar with navigation options like 'Clients', 'Bases portefeuilles', 'Bases valeurs', 'Données', 'Calculs', 'Charte graphique', 'Portail web', 'Entreprise', 'Profil personnel', and 'Logiciel'. The main content area displays a table of permissions for various services, categorized into 'Autorisations transverses à tout AMX1', 'Autorisations de la page principale d'AMX1', and 'Autorisations de la page principale d'AMX1'. Each row lists a specific action and its status for 'Gestion', 'Middle-Office', and 'Risque' roles.

Autorisations transverses à tout AMX1	Gestion	Middle-Office	Risque
Règler les bases de données	Non	Non	Non
Import de données	Non	Non	Non
Modification des tables de données	Non	Non	Non
Création d'un ticket dans le support client	Non	Non	Non
Valeurisation des Fonds	Non	Non	Non
Valeurisation des Profits	Non	Non	Non

Autorisations de la page principale d'AMX1	Gestion	Middle-Office	Risque
Modification de la page 'Calcul'	Non	Non	Non
Modification de la page 'Charte Graphique'	Non	Non	Non
Modification de la page 'Profil Personnel'	Non	Non	Non
Modification de la page 'Logiciel'	Non	Non	Non
Modification de la page 'Portefeuille'	Non	Non	Non

Autorisations de la page principale d'AMX1	Gestion	Middle-Office	Risque
Modification de la page 'Analyse des Frais'	Non	Non	Non
Modification de la page 'Expositions et Risques'	Non	Non	Non
Modification de la page 'Contrôle des Ratios'	Non	Non	Non
Modification de la page 'Commentaires'	Non	Non	Non
Modification de la page 'Reporting'	Non	Non	Non
Modification de la page 'Documents'	Non	Non	Non

## Exemple de quelques droits

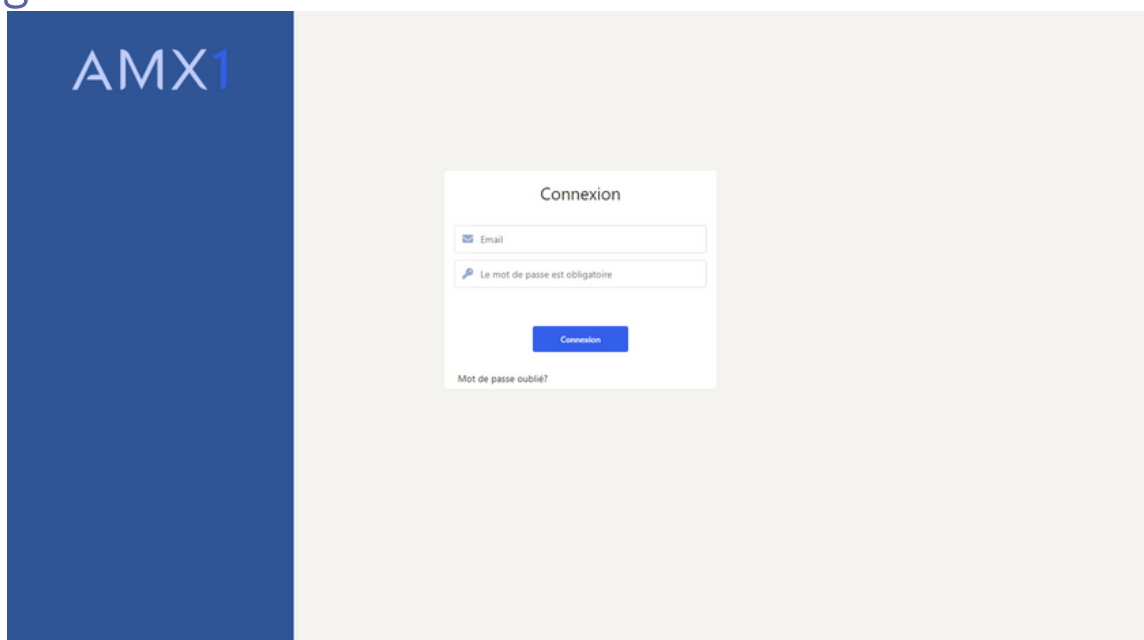
- Un utilisateur peut consulter un reporting financier qu'il n'a pas créé mais ne peut pas l'éditer.
- Le droit "Modification des tables de données" donne accès comme son nom l'indique à la modification de tables de données et l'export des tables située dans les onglets du site : Clients, Base portefeuilles, Base valeurs.
- Le droit "Modification de la page Performances", donne accès à la validation de performance des mandats dans l'onglet "Calendaire". Egalement à l'import et à l'ajout manuel de performances calendaires pour les mandats et les profils.

## 2 - Les projets

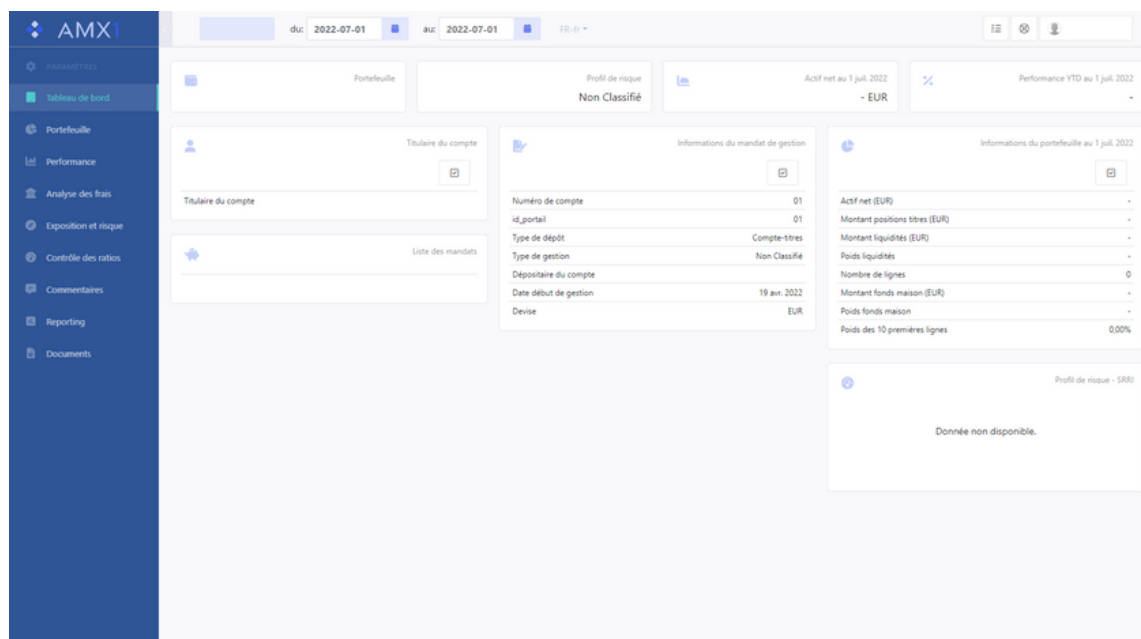
### 2.1 - L'interface utilisateur

Voici une présentation globale de l'interface utilisateur d'AMX1 :

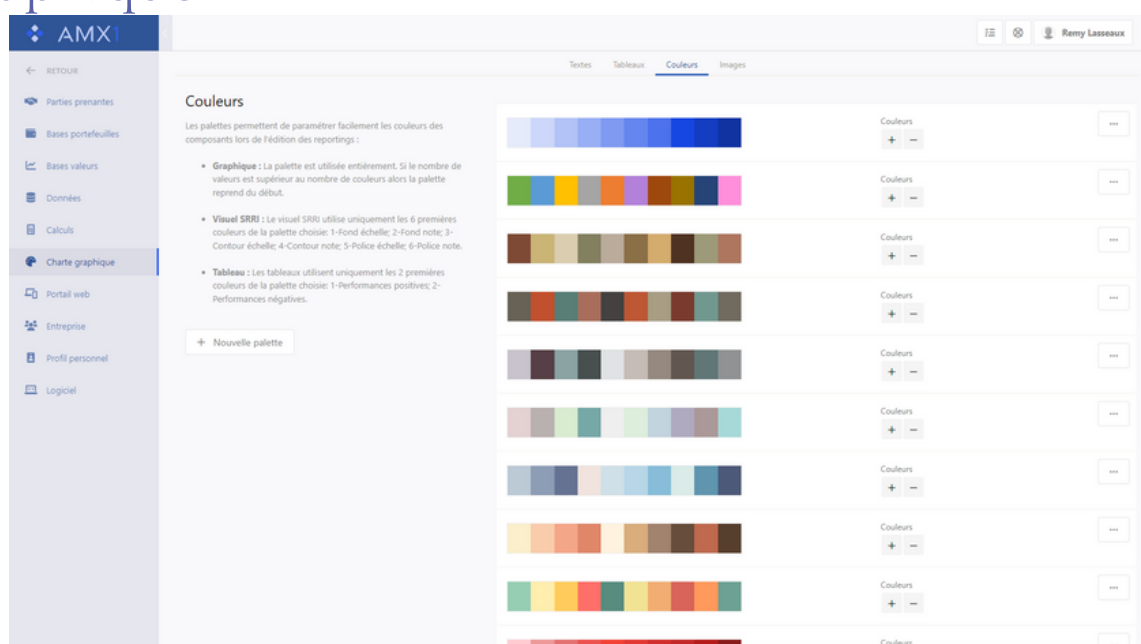
### Page de connexion

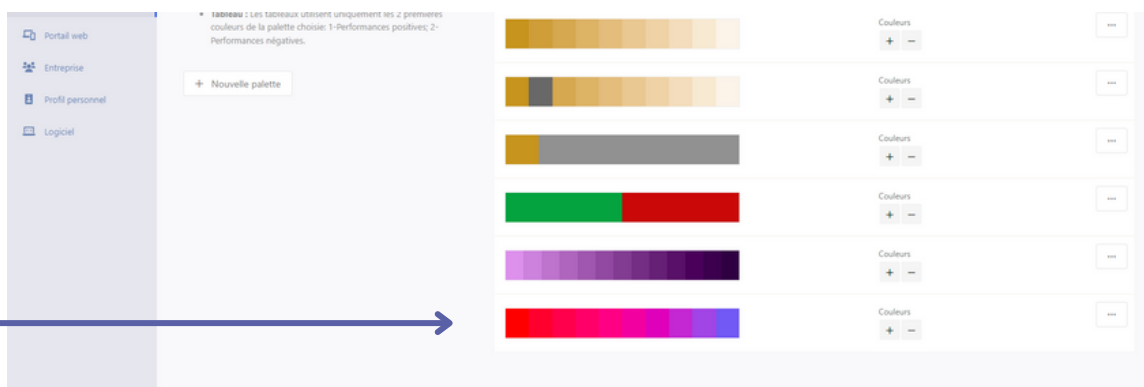
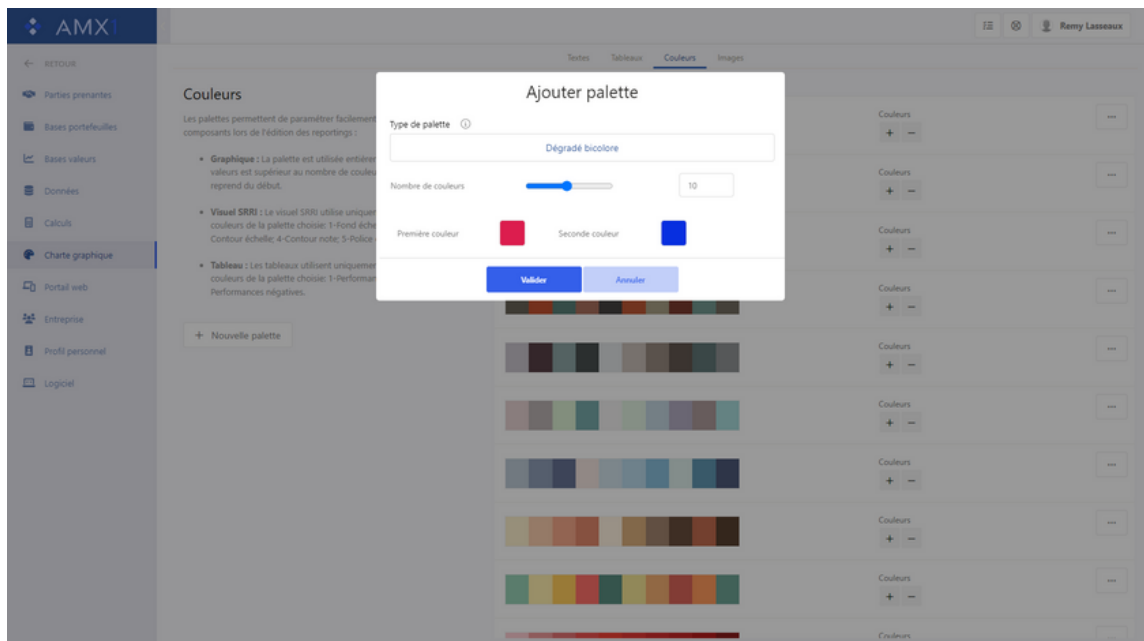


## Le dashboard



## La création et modification des palettes de couleurs disponible pour les reportings financier et les graphique

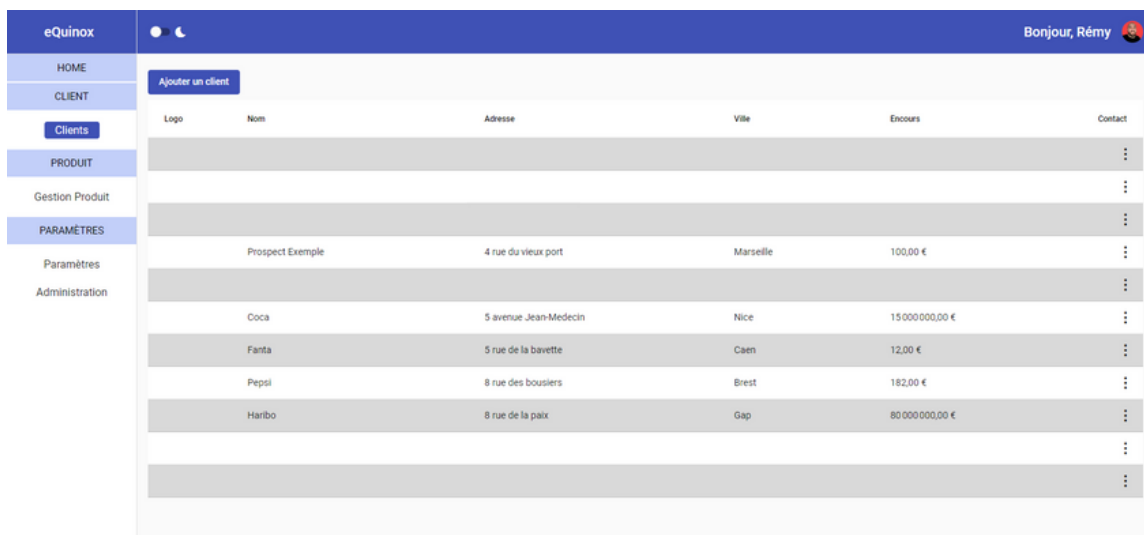




## 2.1 bis - L'interface utilisateur

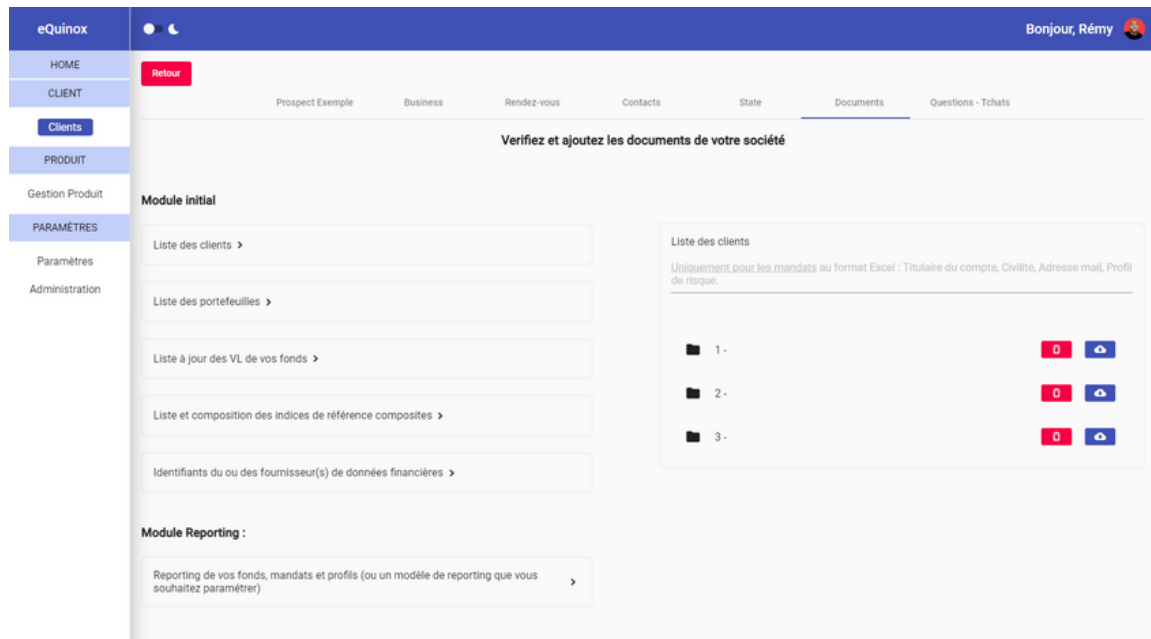
Egalement quelques screenshot de l'interface utilisateur (intranet) d'eQuinox :

### Ajout d'un client

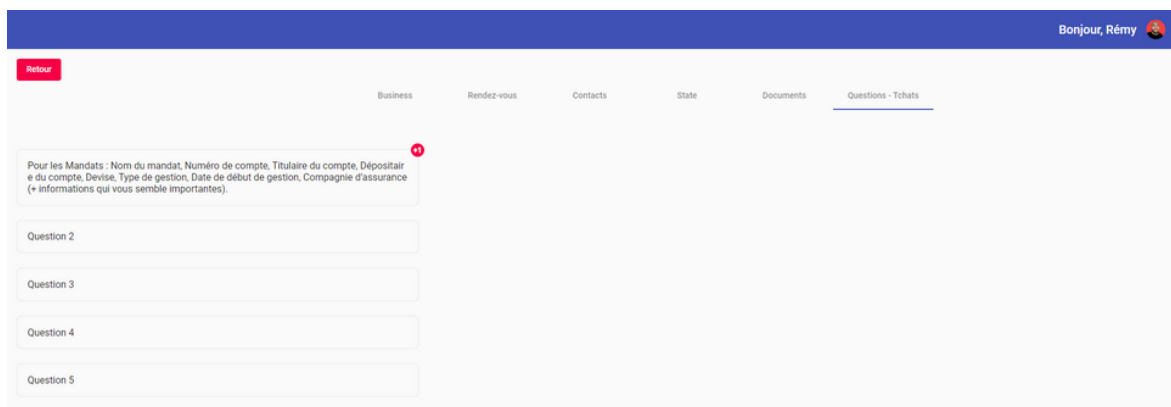




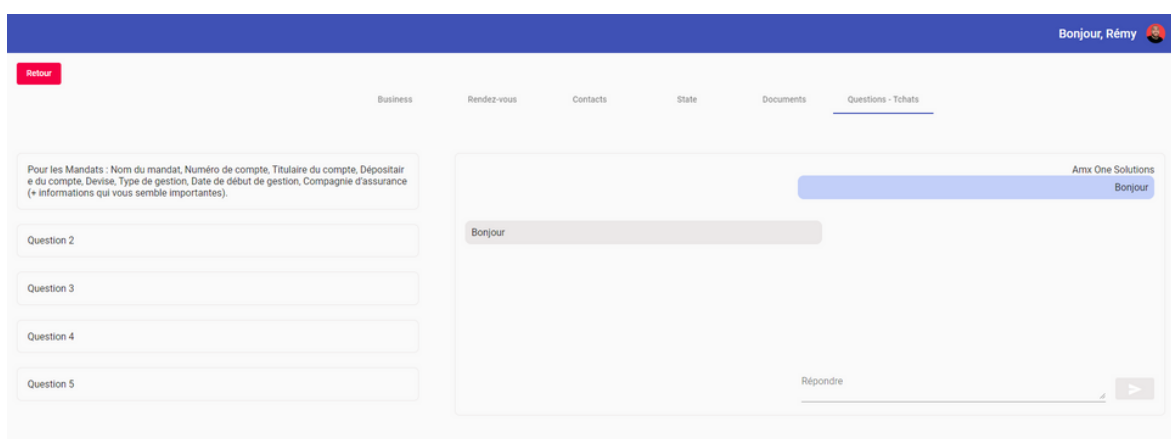
## Gestion du client en cliquant sur l'icone ci-dessus



Et le tchat de discussion quand on a un nouveau messages



L'interface client est la même. La seule différence est que les messages sont inversés



### 3 - L'interface coding

Voici quelques exemples de code par rapport aux pages ci-dessus.

Rappel : Sur le langage de programmation Angular, (et la manière de coder dans l'entreprise)

l'architecture des composants sont constitués :

Nous avons la liste des différents modules qui contiennent X composants, pages...

> <code>_Customer</code>	←	Module en rapport avec les clients
> <code>_Dashboard</code>		
> <code>_Department</code>		
> <code>_guard</code>		
> <code>_Onboarding</code>	←	Gestion des fichiers, interface clients
> <code>_Pricing</code>		
> <code>_Product</code>		
> <code>_Setting</code>	←	Préférences, modification du profil
> <code>_Shared</code>		

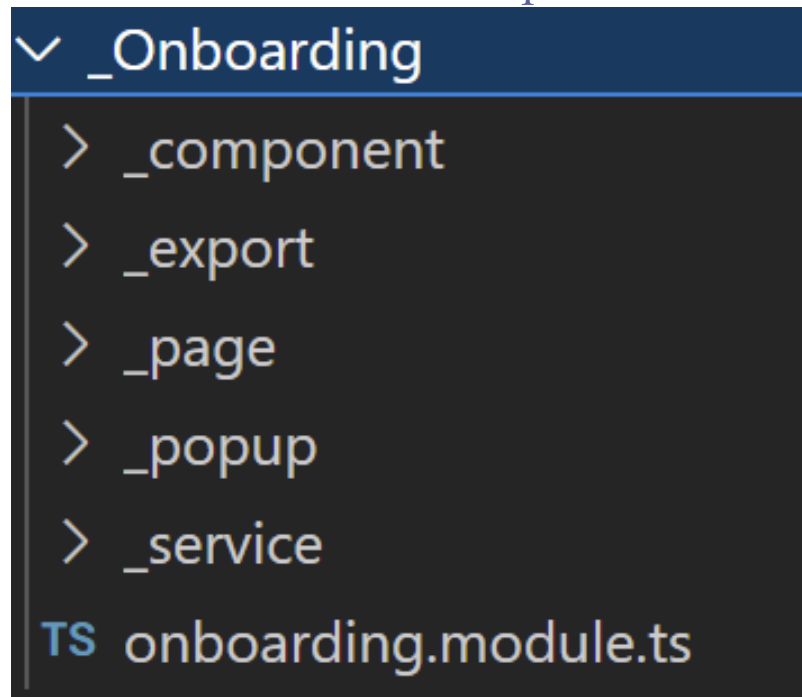
(Toujours dans notre manière de fonctionner) Les modules ont chacun, (si nécessaire), un sous dossier :

- `_component` : qui contient un fichier `.html` pour l'affichage, un fichier `.ts` pour déclarer les variables et les fonctions, et un fichier `.scss` pour appliquer différents style visuel au balises `.html`
- `_export` : pour les interfaces de type\*

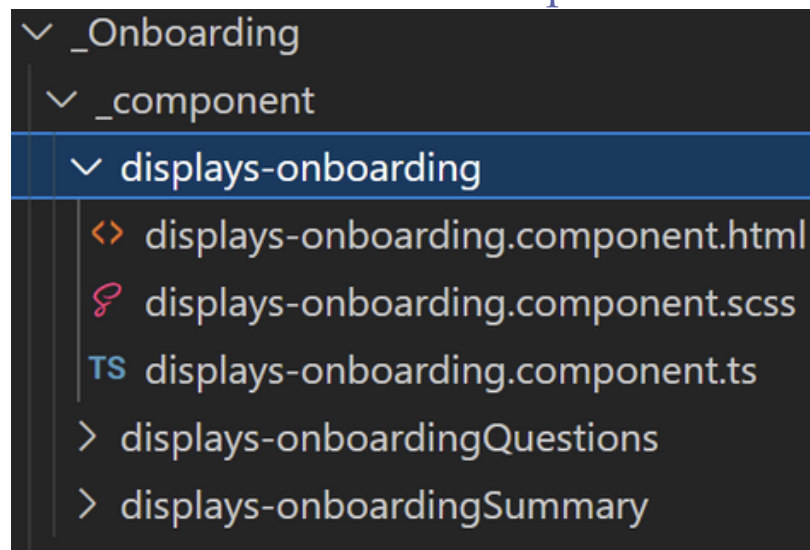
- `_page` : la page, aussi un composant sera constituée de différentes balises html ou, de l'imbrication d'un ou plusieurs composants
- `_popup`: les popups de ce module
- `_service` : le sous-dossier qui va contenir "l'intelligence du module, la récupération, de données du backend ainsi la modification et la suppression à envoyé également

Voici à quoi ça ressemble :

### Module complet



### Sous dossier composant



## Entièrete déroulée

```

  ▾ _Onboarding
    ▾ _component
      > displays-onboarding
      > displays-onboardingQuestions
      > displays-onboardingSummary
    ▾ _export
      TS onboarding.ts
      TS onboardingQuestions.ts
      TS tchats.ts
    ▾ _page \ page-onboarding
      <> page-onboarding.component.html
      🐛 page-onboarding.component.scss
      TS page-onboarding.component.ts
    ▾ _popup
      > popup-add-onboardingfile
      > popup-rejected
    ▾ _service
      TS back-onboarding.service.ts
      TS back-onboardingQuestions.service.ts
      TS back-tchats.service.ts
      TS onboarding.service.ts
      TS onboardingQuestions.service.ts
      TS onboarding.module.ts

```

\* Les interfaces de type : Comme son nom l'indique, elle servent à créer un type, qui sera attribuée lorsque qu'on l'on va créer une variable

```
// -----TCHAT-----
export interface Tchats {
  id_onboardingQuestions: number,
  id_customer: number,
  texts: any,
}

export const TchattingGlobales = {
  defaultTchatting:
  {
    id_onboardingQuestions: -1,
    id_customer: -1,
    texts: [],
  }
}

// -----/ TCHAT-----

// -----MSG DATA-----
export interface MsgData {
  id: number,
  isAMX: boolean,
  msg: string,
  date: Date,
  read: boolean,
}

// -----/ MSG DATA-----
```

Exemple: le paramètre de cette fonction qui permet d'envoyer un message, est de type OnboardingQuestion, la variable message et de type MsgData et param de type Tchats

```
sendMessage(item: OnboardingQuestion): void {
  if(this.msg !== '') {
    let message: MsgData = { id: this.userService.user.id!, isAMX: this.isAMX, msg: this.msg, date: new Date(), read: false };
    item.data?.push(message);
    let param: Tchats = { id_onboardingQuestions: item.id!, id_customer: undefined!, texts: item.data! };
    this.onboardingQuestionService.sendMessage(param);

    this.snackbarService.openSnackBar('Message envoyé');
    this.msg = '';
    setTimeout(this.scrollToBottom, 100);
  }
}
```

## 3.1 - Déroulement de A à Z

Frontend - Le composant de l'affichage des clients et de l'ajout d'un client

Je vais reprendre cette page pour l'exemple



The screenshot shows the eQuinox application interface. On the left is a sidebar menu with options: HOME, CLIENT, Clients (highlighted), PRODUIT, Gestion Produit, PARAMÈTRES, Paramètres, and Administration. The main content area has a header with 'Bonjour, Rémy' and a button 'Ajouter un client'. Below is a table with columns: Logo, Nom, Adresse, Ville, Encours, and Contact. The table contains three rows of data: 'Prospect Exemple', 'Coca', and 'Fanta'.

Logo	Nom	Adresse	Ville	Encours	Contact
	Prospect Exemple	4 rue du vieux port	Marseille	100,00 €	
	Coca	5 avenue Jean-Medecin	Nice	15 000 000,00 €	
	Fanta	5 rue de la bavette	Caen	12,00 €	

Le code de cette page ressemble à ceci :

Le fichier .html avec les différentes balises :

Les div sont une sorte de conteneur, avec angular on utilise aussi parfois la balise `<ng-container>...</ng-container>` la différence ce que le ng-container "n'existe pas" en interface, cela veut dire, qu'il n'a, ni de marge, ni de marge interne, aucune interférence avec le design

```
Go to component
1 <div class="buttonUpTable">
2   <button mat-flat-button class="validButton" (click)="openDialogAddCustomer()">
3     Ajouter un client
4   </button>
5 </div>

<table mat-table [dataSource]="this.datas" *ngIf="this.datas && this.datas.length; else noDataBloc">

  <!-- Name Column -->
  <ng-container matColumnDef="company">
    <th mat-header-cell *matHeaderCellDef>Nom</th>
    <td mat-cell *matCellDef="let element">{{element.company}}</td>
  </ng-container>

  <!-- Address Column -->
  <ng-container matColumnDef="address">
    <th mat-header-cell *matHeaderCellDef>Adresse</th>
    <td mat-cell *matCellDef="let element">{{element.address}}</td>
  </ng-container>

  <!-- City Column -->
  <ng-container matColumnDef="city">
    <th mat-header-cell *matHeaderCellDef>Ville</th>
    <td mat-cell *matCellDef="let element">{{element.city}}</td>
  </ng-container>

  <!-- Outstanding Column -->
  <ng-container matColumnDef="outstanding">
    <th mat-header-cell *matHeaderCellDef>Encours</th>
    <td mat-cell *matCellDef="let element">{{element.outstanding | currency: 'EUR'}}</td>
  </ng-container>

  <!-- Contact Column -->
  <ng-container matColumnDef="contact">
    <th mat-header-cell *matHeaderCellDef class="contactButton">Contact</th>
    <td mat-cell *matCellDef="let element" class="contactButton" (click)="setCustom(element.customer)"><a<i class="fal fa-ellipsis-v"></i></a></td>
  </ng-container>

  <!-- Columns & Rows -->
  <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
  <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>

</table>
```

Le fichier .ts, ce que vous pouvez voir tout au début de la capture d'écran, c'est les imports. Les imports sont des références à d'autres fichiers de l'ensemble du projet, une variable, une fonction, une classe (qui correspond souvent à un fichier entier comme on peut voir ci dessous), la seule condition pour pouvoir importer, un composant dans une page, une variable dans une classe ou une fonction en particulier, il faut avoir le **ceci** devant la déclaration

```
import { Component, Input, OnInit } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { Subscription } from 'rxjs';
import { Customer, TypeDisplay } from '../_export/types';
import { PopupAddCustomerComponent } from '../_popup/popup-add-customer/popup-add-customer.component';
import { CustomerService } from '../_service/customers.service';
import { MainService } from '../_service/main.service';

@Component({
  selector: 'app-displays-customer',
  templateUrl: './displays-customer.component.html',
  styleUrls: ['./displays-customer.component.scss']
})
export class DisplaysCustomerComponent implements OnInit {

  @Input() typeDisplay: TypeDisplay = TypeDisplay.ALL;

  private subscriptions: Array<Subscription> = []
  constructor(public customerService: CustomerService, public mainService: MainService, public dialog: MatDialog) { }

  public datas: Array<any> = [];
  public displayedColumns: Array<string> = ['company', 'address', 'city', 'outstanding', 'contact'];

  ngOnInit(): void {
    // this.subscriptions.push(this.customerService.cntRdy.subscribe(() => this.refresh()));
    this.refresh();
    this.customerService.cntRdy.subscribe(() => {
      this.refresh();
    });
    this.customerService.cntRdy.subscribe(() => {
      this.refresh();
    });
  }

  ngOnDestroy(): void {
    this.subscriptions.forEach((subscription) => subscription.unsubscribe());
  }
}
```

Ensuite nous déclarons le sélecteur, qui servira d'appel de ce composant, et son fichier .html et .scss en référence et le nom de la classe qui va contenir le code

```
@Component({
  selector: 'app-displays-customer',
  templateUrl: './displays-customer.component.html',
  styleUrls: ['./displays-customer.component.scss']
})
export class DisplaysCustomerComponent implements OnInit {
```





### 3.1 - Déroulement de A à Z

Je vais reprendre cette page pour l'exemple

eQuinox	Bonjour, Rémy					
HOME	Ajouter un client					
CLIENT						
Clients	Logo	Nom	Adresse	Ville	Encours	Contact
PRODUIT						⋮
Gestion Produit						⋮
PARAMÈTRES						⋮
Paramètres	Prospect Exemple					⋮
Administration	4 rue du vieux port					⋮
	Marseille					⋮
	100,00 €					⋮
	Coca					⋮
	5 avenue Jean-Medecin					⋮
	Nice					⋮
	15 000 000,00 €					⋮
	Fanta					⋮
	5 rue de la bavette					⋮
	Caen					⋮
	12,00 €					⋮

Quand on clique sur ce bouton cela nous ouvre une popup qui est la suivante

Client

Création d'un client

Raison sociale \*

VilleAdresse \*

EncoursChiffre potentiel

Certificat :  
☐ Oui ☐ Non

ClosingStage

ValiderAnnuler



## 2.2 Librairie sur AMX1

Pour réaliser la partie front-end, la librairie Bootstrap été déjà présente.

Ce choix s'est fait ainsi en fonction des connaissances des membres de l'équipe de ces librairies et pour leur facilité d'utilisation.

## 2.2 Librairie sur eQuinox

Pour réaliser la partie front-end, la librairie Angular Materials a été choisie pour sa facilité, sa rapidité et sa facilité de prise en main. Nathan et (...)

(...) moi même étions d'accord pour partir de zéro avec cette librairie pouvant afficher des tableaux, utiliser la pagination, des "modals" avec facilité lors de notre apprentissage du langage Angular.

### 3 - Exemples de Bootstrap

## Buttons

[View on GitHub](#)

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

### Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

Primary Secondary Success Danger Warning Info Light Dark [Link](#)

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

[Copy](#)

### Basic example

Wrap a series of buttons with `.btn` in `.btn-group`.

Left Middle Right

```
<div class="btn-group" role="group" aria-label="Basic example">
  <button type="button" class="btn btn-primary">Left</button>
  <button type="button" class="btn btn-primary">Middle</button>
  <button type="button" class="btn btn-primary">Right</button>
</div>
```

[Copy](#)

Image cap

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Go somewhere

```

<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up th
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>

```

Copy

## 3.bis - Exemples d'Angular Materials

Greetings

hello

hello world

**Autocomplete**  
 Suggests relevant options as the user types.

1

6

18

**Badge**  
 A small value indicator that can be overlaid on another object.

Favorite

Share

**Bottom Sheet**  
 A large interactive panel primarily for mobile devices.

Button

Button

Button

Button

**Button**  
 An interactive button with a range of presentation options.

☰

☰

☰

**Button toggle**  
 A groupable on/off toggle for enabling and disabling options.

**Card**  
 A styled container for pieces of itemized content.

All

☒ Draft

☒ In Progress

☐ Submitted

**Checkbox**  
 Captures boolean input with an optional indeterminate mode.

**Amenities**  

Washer/Dryer Dogs ok Cats ok Kitchen

Fireplace Wheelchair accessible Elevator

**Chips**  
 Presents a list of items as a set of small, tactile entities.

⚙️

**Core**  
 Reusable parts used by other components in the library.

MAY

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31

**Datepicker**  
 Captures dates, agnostic about their internal representation.

Discard draft?

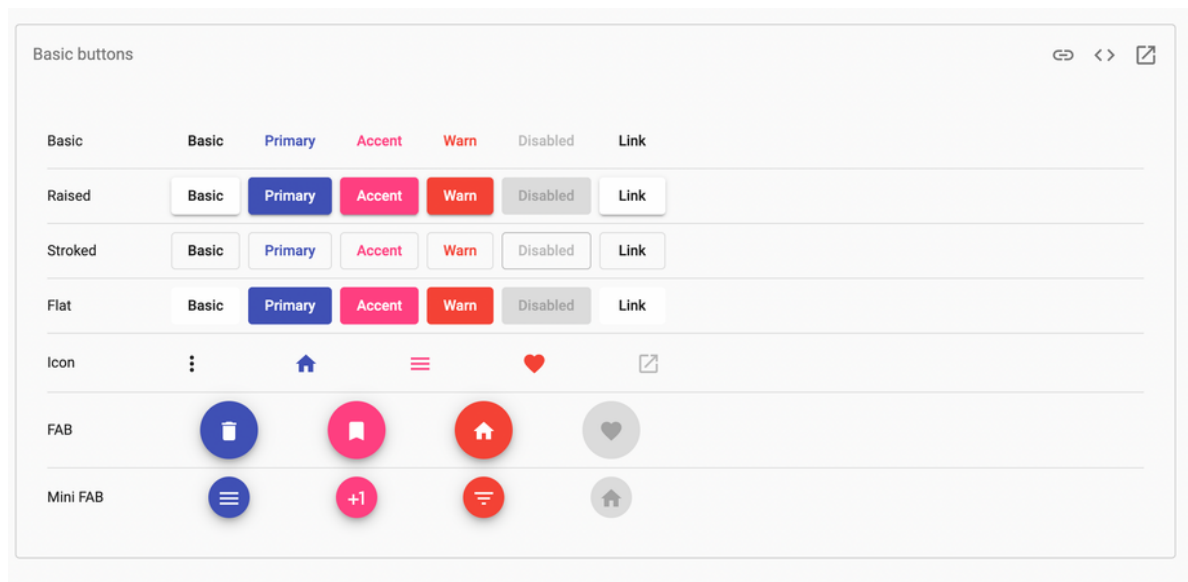
Cancel Discard

**Dialog**  
 A configurable modal that displays dynamic content.

Contact 1

Contact 2

**Divider**  
 A vertical or horizontal visual divider.



```
<section>
  <div class="example-label">Basic</div>
  <div class="example-button-row">
    <button mat-button>Basic</button>
    <button mat-button color="primary">Primary</button>
    <button mat-button color="accent">Accent</button>
    <button mat-button color="warn">Warn</button>
    <button mat-button disabled>Disabled</button>
    <a mat-button href="https://www.google.com/" target="_blank">Link</a>
  </div>
</section>
<mat-divider></mat-divider>
<section>
  <div class="example-label">Raised</div>
  <div class="example-button-row">
    <button mat-raised-button>Basic</button>
    <button mat-raised-button color="primary">Primary</button>
    <button mat-raised-button color="accent">Accent</button>
    <button mat-raised-button color="warn">Warn</button>
    <button mat-raised-button disabled>Disabled</button>
    <a mat-raised-button href="https://www.google.com/" target="_blank">Link</a>
  </div>
</section>
<mat-divider></mat-divider>
<section>
  <div class="example-label">Stroked</div>
  <div class="example-button-row">
    <button mat-stroked-button>Basic</button>
    <button mat-stroked-button color="primary">Primary</button>
    <button mat-stroked-button color="accent">Accent</button>
    <button mat-stroked-button color="warn">Warn</button>
    <button mat-stroked-button disabled>Disabled</button>
    <a mat-stroked-button href="https://www.google.com/" target="_blank">Link</a>
  </div>
</section>
```

## 4 - Quelques outils

### 1. Outils de développement

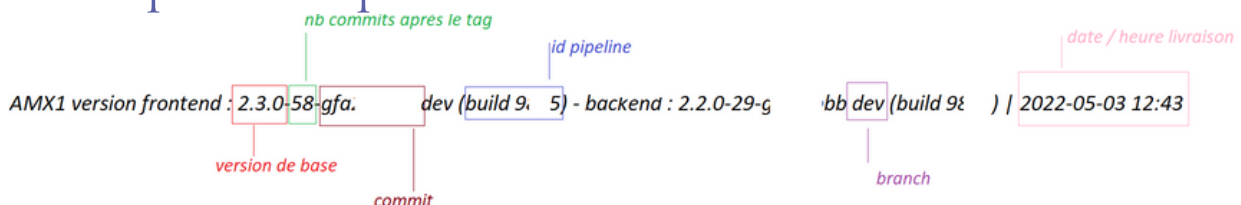
Tant pour la partie front et back Visual Studio Code à été utilisé.

## 2. Outil de versionning et de collaboration

Nous avons créé un répertoire (repository) distant sur Gitlab pour le front-end et pour le back-end. L'utilisation de git s'est faite via l'application SourceTree afin de gérer les différents commits. Les éventuels conflits étaient réglés soit en local, soit via l'interface de SourceTree ou encore sur l'éditeur Visual Studio Code.

Commits : Mot que l'on emploie pour exprimer une validation, une soumission d'une partie de code.

### Exemple d'un push



## 2.1 Méthode agile et retrospective

### Comment se déroule la journée rétrospective

9h00 – 11h00

- Rétrospective sprint précédent:
  - Définir les axes d'améliorations
  - Définir les axes qui ont fonctionnés

11h00 – 12h00

- Point entreprise

14h00 – 16h30

- Définition du futur sprint
  - Explication des tickets
  - Estimation des tickets
  - Affectation des tickets


# Réunion d'innovation

- Mise en commun des idées d'innovation
  - Découverte sur les sites annexes
  - Découverte chez les concurrents
  - Invention

Ce qui en découle ceci

A jeter	A réutiliser	A acheter	Comment ?
La gestion de projet	Manque de commentaires dans le code	Mise en place de	Ajouter un Chef de projet
Pouvoir livrer	Message de commit sans numéro de tickets	Bonne communication	Passer les projets en AGILE
Lundi matin : mise en place d'un point de début de semaine autour d'un café d'équipe.	Améliorer le remplissage des tickets (redmine : 1) Ajouter un champs "Dev dédié" 2) Ne pas pouvoir changer le status si certains champs ne sont pas remplis.	Processus via du ticket marche bien	pas de projet à l'abandon
Faire des templates de tickets redmine de l'idée jusqu'à la livraison	Gestion des bureaux : sortir les poubelles, ménage, ranger les bureaux, achats...	Peu de retour client négatif (en fin je crois)	Réunion du mercredi matin (cofé)
Aucun commentaire dans les tickets, pas de suivi	Communication entre équipes services/personnels sur les tickets à régler	Noms de sites dans les livraisons, avec contour clair	Formation sur les branches
Manque formation architecture AIX	Design (form/cst) des dev - Finalisation des tickets par le team dev	Food / Ranzan « présents ce mois ci, (stock ou présents)	Prendre un R pour les clés
Utilisation des branches	Ne pas affecter les bugs qui ne sont pas dans le scope	Productivité en télétravail : tout le monde est joignable, présence au bureau.	Formation sur les scripts
Changer la deadline du sprint	Garder les bugs pour vous	Tous présent aux réunions	Réunion du mercredi matin à 9h00 (Dinner)
Build casse et personne s'en rend compte	Les dev deviennent experts d'un périmètre : Graphiques, Templates, Un design...	Ambiance au travail : cohésion d'équipe et bonne ambiance quand on vient au bureau	Création d'un template de description
Organisation générale, communication et outils : Email, OneDrive, Teams, Slack			pour l'EAUUEG
			Ajouter un champs développeur
			Bloquer le changement de status
			Utiliser des branches pour les bugs clients;
			Formation sur les branches
			Ajouter une nouvelle case





Dans la dernière case "**Comment ?**" on peut voir les actions qui devront être mise en place jusqu'à la prochaine rétrospective, là on pourra débattre si oui ou non cela a été mis en place et si c'est une bonne habitude ou non.