MEYNADIER Renaud

Développeur Web et Web Mobile.

Dossier projet

REMERCIEMENTS :

Je voudrais remercier toutes les personnes sans qui ce projet n'aurait pas pu aboutir.

Mes remerciements vont donc à Human Booster, Pôle Emploi, Be-Ys Software ainsi que ses développeurs qui m'ont accueilli au sein de leur équipe.

Plus précisément l'équipe Human Booster qui a fourni les locaux, le matériel, la connexion à notre disposition.

A mes collègues de promotion de cette formation, avec qui on a passé une très bonne année tous ensemble. (Remerciements spéciaux à Lydie Bertrand pour la correction de certaines fautes)

Aux collaborateurs de l'entreprise Be-Ys Software, dans laquelle le cadre de travail est vraiment stimulant et paradisiaque pour évoluer auprès de professionnels.

Grâce à toutes les parties prenantes, une alternance chez Be-Ys Software est envisageable.

Bien sûr à Pôle Emploi sans qui rien de tout cela ne serait possible sans leur investissement dans cette formation.

Et à ma compagne bien évidemment !

Le projet doit couvrir les compétences suivantes :

Le stage chez Be-Ys Software couvre la partie front du projet :

CCP1 - Pour la partie « Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité »

- Maquetter une application
- Réaliser et développer une interface utilisateur web adaptable statique et dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Mon projet annexe "fil rouge" couvre la partie back du projet :

CCP2 - Pour la partie « Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité »

- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

Les technologies abordées en formation sont les suivantes :

HTML - CSS - JS - TYPESCRIPT - ANGULAR - SYMFONY - PHP - SQL - DQL

Précision :

Mon stage ne couvrant que la partie ‘'front" du métier de développeur, j'ai conçu en parallèle un site de e-commerce, permettant de compléter la partie fournie par le stage. Ce site sera présenté en toute fin, avec une navigation dans ses rubriques.

SOMMAIRE

Résumé :

Be-Almerys, est une entité de plus de 2000 collaborateurs dont 400 en France, qui propose des solutions et services pour le traitement de données, gestion de conformité, distribution de produits, délégation de gestion et la mise en œuvre des services différenciant. C'est un expert de haute technologie appliqué à l'assurance et à la santé. L'entreprise est composée de sous-entités :

Beys Trusted Solutions, Beys Pay, Beys Health Solutions, Beys Search, Beys Software et la Citadelle.

Ses sous-entités ont pour objectifs de :

- Sécuriser le bien le plus précieux de l'entreprise : ses données.
- Digitaliser les processus métiers de l'entreprise pour lui permettre de réduire ses coûts et d'améliorer ses ventes.
- Sécuriser juridiquement le client lors des prises d'engagement.
- Réduire et sécuriser les coûts d'archivage et de transfert de documents.

Ils ont différents clients : La Banque Postale (assurance), Generali, Mutuelle Générale, Swisslife…

J'ai donc fait un stage au sein de Be-Ys Software, dans l'équipe produit IAM (Identity, Access, Management) avec les collaborateurs spécialisés dans le développement front. (Ils sont présents aussi à Paris, Toulouse, Annecy, au Luxembourg et ils possèdent une filiale à Tunis).

J'ai travaillé sur une application Angular IAM-FRONT, qui permet la gestion de rôles, de données pour les professionnels de la santé. Mais pour être plus précis, l'application permet pour chaque domaine d'un client de :

- Gérer les comptes d'authentification/habilitation et leurs données personnelles associées pour un domaine.
- Analyser l'évolution des données d'un domaine (via des graphiques).
- Tracer les différentes créations/modifications/suppressions des données du domaine.

En ce moment l'application est refondue en version 2.0. Les tâches sont divisées en plusieurs Sprint de trois semaines, qui sont eux même divisés en plusieurs tickets. J'ai eu la chance de pouvoir prendre quelques tickets front, des correctifs de modal, de la correction de CSS et l'utilisation de leur librairie Angular.

J'ai eu l'opportunité de corriger quelques problèmes de traduction, de rendre l'interface plus intuitive et facile d'accès. J'ai eu recours à des appels d'API, et des dashboard pour gérer les comptes et identifiants.

Mes tickets sont ensuite analysés (pipeline gitlab), traités et validés (avec Jira), s'ils ont passé une première série de tests. J'ai utilisé MatterMost, Skype Entreprise et Outlook qui permettent de faire des retours très rapide.

Rappel : Le stage ne couvrant pas la partie ''back'' j'ai conçu en parallèle un site de e-commerce : La Nimes'Alerie, que je vous présenterai juste après.

PRÉSENTATION DU DOSSIER PROJET

Après un premier résumé ainsi qu'un sommaire, je vais dans cette section vous expliquer comment le cheminement de ce dossier va s'effectuer. C'est en quatre parties majeures que cette épreuve va se dérouler, avec l'ajout d'une petite conclusion et de quelques captures d'écran.

PARTIE I - Le Stage à Be-Ys Software Almerys

Dans cette partie, je vais pouvoir parler du stage que j'ai eu la chance de faire dans une société à la pointe de la technologie. J'ai principalement travaillé sur une interface front qu'on appelle IAM-FRONT.

PARTIE II - Projet fil rouge : La Nimes'Alerie alias "business case"

Cette partie est importante, car elle a été faite selon un énoncé et un besoin d'un client. C'est construire un site complet de e-commerce. J'ai énuméré les tâches à effectuer, décortiqué le besoin, déconstruit le projet, pour pouvoir gravir étape par étape l'énorme montagne qu'il représente. Ce fut au fur et à mesure du temps, de la montée en compétence que le site de la Nimes'Alerie a pu voir le jour.

CONCLUSION DE LA FORMATION À L'APPRENTISSAGE DU MÉTIER

Dans cette partie je tire la conclusion de cette expérience de la formation avec l'équipe Human Booster et des nouveaux collègues que j'ai pu rencontrer pendant ce cursus, cela a vraiment été un sacré défi à relever mais, je me rappelle un slogan disant un jour : ''Ensemble tout devient possible''. Par ailleurs je sais qu'une formation en alternance s'ouvre prochainement avec Human Booster et Be-Ys semble ok pour me prendre en alternant. J'ai donc évidemment postulé !

LES CAPTURES D'ÉCRANS

Ici je mets quelques captures d'écrans que j'ai pu faire pendant le stage (captures qui ont été vérifiées par la société, protection des données sensibles…) Heureusement à côté j'ai pu mettre mes captures sur mon projet fil rouge qui lui n'a rien de confidentiel.

PARTIE I - Le Stage à Be-Ys Software (Almerys)

• Cahier des charges (présentation du projet, des besoins de l'entreprise, la problématique, etc.)

Be-Ys Software, comporte plusieurs équipes produit notamment l'équipe IAM qui implémente des applications front et back telle que celle sur laquelle j'ai travaillé, IAM Front.

En ce moment l'équipe front (IAM) que j'ai intégré effectue une refonte totale de l'interface vers une deuxième version. Nous faisons de la R&D d'interfaces sécurisées destinées à des clients sensibles, protégeons leurs données. L'interface front doit être intuitive, claire, complète et facilement accessible. Pour le moment elle est juste usée en interne.

Durant mon stage on m'a apporté de l'aide et notamment une note pour des recherches efficaces écrite par mon tuteur de stage :

Il est important de rechercher par mots-clés et je te conseille la structure suivante : Nom de la techno + élément technique OU erreur.

Ensuite, il est important d'identifier les points ci-dessous :

Identifier la technologie de travail en cours, que ce soit un langage, un framework ou un autre type de techno, il faut toujours commencer ta recherche par ce mot-clé (Exemple : Javascript, Angular, CSS...)

Citer l'élément technique ou l'erreur affichée à l'écran.

- Cela peut être un concept propre à la techno si tu cherches à te documenter dessus
  (Ex : BehaviourSubject en Angular)
- Ou bien l'erreur affichée sur ta console de la page
  (Ex : Cannot read "length" of undefined)

Privilégier la doc officielle de la technologie ou StackOverflow.

Ce sont les premières ressources d'informations à privilégier, ne pas hésiter à détailler la recherche comme ceci :

Angular FormArray

Typescript cannot read "length" of undefined

Java NullPointerException…

• Spécifications techniques du projet (l'organisation de travail, environnement, outils, techno, etc.)

Webstorm : Un IDE pour les langages Web, adéquat pour Angular. (VS Code également)

Firefox Developer : Une version pour développeur, avec des fonctions inédites.

Notepad : Pour la prise de notes diverse, les journaux de bords, mémo et liste à faire.

Git Fork : Permet de visualiser ses branches en temps réel et les collaborateurs.

Keycloak : Gestionnaire d'identification et d'habilitation basée sur le protocole oauth2.

Adobe XD : Aide à la construction de maquette pour le développement web.

Planit Poker : Aide à l'estimation des tâches pour les sprints à venir.

PowerShell : Permet d'exécuter des lignes de commandes, indispensable ici.

Chocolatey Software : Gestionnaire de package en ligne de commande et installation.

Mattermost : C'est un open source de slack, il est similaire à Discord avec des salons.

Jira : Système de suivi de bug, gestion d'incident, prise et assignation de tickets.

Skype entreprise : Nécessaire pour les réunions journalières, Teams est aussi utilisé.

Outlook Pro : Messagerie professionnelle pour gérer les retours client, tâches, erreur...

Phpstorm : Un IDE cousin de Webstorm, utilisé pour le php et pour le site du fil rouge.

Ces outils sont utilisés quotidiennement pour les tâches effectuées à Be-Ys Software.

Du simple Outlook message jusqu'à l'IDE VS code (Webstorm.dans mon cas).

De la prise d'informations jusqu'à la création de lignes de code, tout est suivi et mis à jour en permanence avec les équipes front.

A côté de ça nous utilisons du matériel de pointe, plusieurs écrans, une connexion à toute épreuve et pour l'anecdote, les ordinateurs sont branchés sur une alimentation parallèle, ainsi vous pouvez continuer à travailler normalement même pendant une coupure de courant.

Une journée type à Be-Ys Software peut être schématisée ainsi :

- Prise de poste à 9 heures.
- Vérification des messages sur Mattermost, des ‘’daily’’ (réunions à 10h45 jusqu'à 11h pour faire le point) avec Skype Entreprise, des e-mails automatiques (pipeline gitlab acceptée, refusée, à modifier…), des retours des collègues en distanciel, avec Outlook Pro.
- Pendant la définition d'un nouveau sprint on utilise planitpoker qui aidera le PO et l'équipe à estimer la difficulté d'une tâche dans ledit sprint.
- Avec Jira, prise de nouvelles tâches à faire, de ticket, de commentaire à ajouter à l'équipe sur un travail à faire, réponse aux collègues qui veulent parfois des précisions sur votre travail.
- Prise de note pendant les réunions des fonctionnalités à faire avec bloc note (un cahier fonctionne aussi), utile aussi pour transmettre du code, faire des tâches…
- Parfois on doit utiliser PowerShell et Chocolatey Software pour installer certains logiciels (blocage strict de l'entreprise sur certains domaines), même Google Docs n'est pas toléré.
- On peut utiliser occasionnellement Adobe XD mais Be-Ys Software dispose d'un bureau à côté composé d'une équipe UX/UI.
- Webstorm et Firefox developer nous permettent le plus gros du travail, le développement web bien sûr. Avec l'aide aussi de Google et internet.
- Utilisation de keycloak pour la gestion d'identification et d'habitation.
- Gitfork je l'utilise personnellement, mais l'équipe utilise des commandes dans le terminal. (Git checkout -b ou -D, git status, git add -A, git commit -m, git push origin, « mais aussi » : git log, git fetch origin alpha, git cherry-pick)

Quelques commandes propres au projet Angular IAM-FRONT :

- Npm (node package manager) run start:int (intégration)
- Npm run lint (passe le code en revue à la recherche d'erreur)
- Npm run test (permet d'avoir une première version du code uniforme et propre, retour dans le navigateur avec ‘jasmine’, erreur à corriger pour pouvoir faire une merge request et vérifier les tests pipeline dans gitlab)
- Npm run watch (dans la librairie)

- Npm link dist/common (pour que NPM fasse un lien symbolique de ce répertoire vers sa base de dépendance)
- Npm link @beys-software/common (Permet de relier une au IAM FRONT qui recevra la dépendance)
- Npm run get:assets (Récupère les assets du projet angular)
- Npm install autocompleter (Dépendance d'auto-complétion)

• Réalisations du candidat comportant les extraits de code les plus significatifs et en les argumentant, y compris pour la sécurité et le web mobile

Ici je peux simplement montrer des bouts de code Angular qui sera un peu hors contexte étant donné que je ne suis pas le seul à participer à cette grande application qu'est IAM-FRONT. Je suis aussi disposé à montrer pendant la soutenance mon code et l'expliquer en toute transparence pendant la présentation du power point.

Je peux vous présenter une fonctionnalité à laquelle j'ai participé, le mode sombre d'une librairie de composants qui doit gérer également le dark mode.

Nous sommes dans le projet IAM-FRONT, voyez ici, une balise html qui est un composant au sein de la librairie ''beys-switch'', qui appelle une fonction appelée : onDarkModeClicked() :

profile-preferences.component.html

```
<!-- Switch to enable/disable dark mode -->
<div class="col-md-6 col-xl-7">
    <beys-switch [checked]="darkModeEnabled" (emittedCheck)="onDarkModeClicked()"></beys-switch>
</div>
```

Ici nous sommes dans le « controller » d'Angular, nous étions plus haut dans le template, nous sommes ici dans le .ts associé :

profile-preferences-component.ts

```
/**
 * Toggle dark mode
 */
public onDarkModeClicked(): void {
    this.darkModeEnabled = !this.darkModeEnabled;
    localStorage.setItem('theme', this.darkModeEnabled ? 'dark' : 'light');
    this.documentRef.documentElement.setAttribute( qualifiedName: 'data-theme', value: this.darkModeEnabled ? 'dark' : 'light');
    this.uiService.setDarkMode(this.darkModeEnabled);
}
```

Par exemple, si je clique sur le html, nous avons un « toggler » en ligne numéro un qui inverse le boolean du darkmodeEnabled dont on se sert pour affecter la variable 'theme' du localStorage, via l'expression ternaire le '?' enfin on annonce au reste de l'application le changement de thème via l'uiService et la méthode setdarkMode.

Grace à l'UIService ici présent : (export le rend disponible dans tout le projet)

```ts
export class UIService {

    private _isDarkMode: BehaviorSubject<boolean> =
        new BehaviorSubject<boolean>( _value: undefined);
    public isDarkMode: Observable<boolean> = this._isDarkMode.asObservable();

    private _showSideBarSubject: BehaviorSubject<boolean> = new BehaviorSubject<boolean>( _value: false);
    public showSideBar: Observable<boolean> = this._showSideBarSubject.asObservable();
```

ui.service.ts

```ts
public setDarkMode(isDarkMode: boolean): void {
    this._isDarkMode.next(isDarkMode);
}
```

ui.service.ts

Nous sommes dans le uiService avec la propriété _isDarkMode avec le BehaviorSubject qui sert à annoncer au reste de l'application le thème actuel et les changements de thèmes à venir.

N'oublions pas que ce projet est lié par une librairie commune, appelée beys-software-common-lib, dans le template html de la librairie, nous avons donc ici :

```html
<li *ngIf="darkModeButton">
    <div class="theme-switch-wrapper">
        <label class="switch switch-small theme-switch" for="checkbox">
            <input type="checkbox" id="checkbox" [checked]="darkModeEnable" (change)="onDarkModeClicked()">
            <span class="slider round"></span>
        </label>
        <em class="theme-switch-label">{{ darkModeLabel }}</em>
        <small class="keyboard">M</small>
    </div>
</li>
```

navbar.component.html

Ici nous avons le bouton qui fait appel à la fonction onDarkModeclicked() .

Et dans le « controller » associé, le navbar.component.ts la fonction appelée par le bouton :

```ts
/**
 * Enable or disable dark mode. This method add a data-theme attribute to the html tag
 */
public onDarkModeClicked(): void {
  this.darkModeEnable = !this.darkModeEnable;
  localStorage.setItem('theme', this.darkModeEnable ? 'dark' : 'light');
  this.documentRef.documentElement.setAttribute( qualifiedName: 'data-theme', value: this.darkModeEnable ? 'dark' : 'light');
  this.darkModeActivated.emit(this.darkModeEnable);
}
```

navbar.component.ts

Etant donné que nous voulons changer le thème de l'application de façon générique, depuis les composants de la librairie notamment depuis navbar.component.ts, nous avons également ce composant qui définit la variable thème du localStorage et qui émet la valeur du booléen correspondant au thème. (Similairement au mécanisme de l'application) Un composant dans la librairie, et un dans le projet front.

Vous vous rappelez <beys-switch></beys-switch> ? C'est de l'imbrication Angular. En effet Angular favorise le 'one page application' on appelle dans un template un autre template. Voici le contenu de ce beys-switch :

```
<label class="switch">
    <input type="checkbox" [checked]="hasControlContainer()" (change)="emitChecked($event.target.checked)">
    <span class="slider round"></span>
</label>
```

switch.component.html

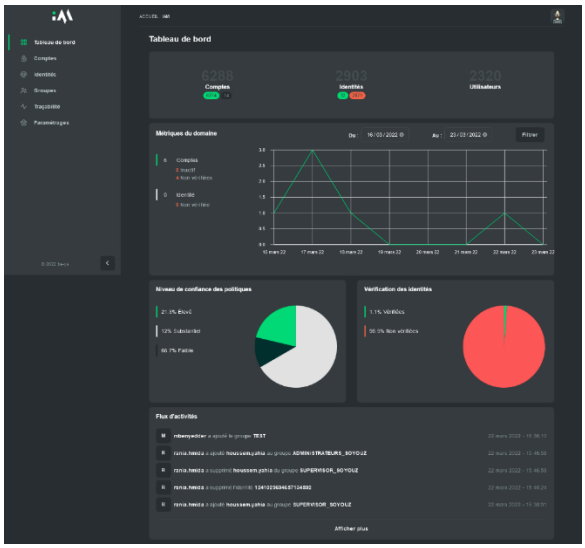
Avec évidemment son « controller » le switch.component.ts :

```
import { Component, EventEmitter, Input, OnInit, Optional, Output } from '@angular/core';
import { ControlContainer, ControlValueAccessor, NG_VALUE_ACCESSOR } from '@angular/forms';

@Component({
    selector: 'beys-switch',
    templateUrl: './switch.component.html',
    styleUrls: ['./switch.component.css'],
```

switch.component.ts

- **Présentation du jeu d'essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues)**

Pour ce point ici, je peux vous montrer l'interface sur laquelle j'ai travaillé, il s'agit d'IAM FRONT. Des petites corrections que j'ai apportées, fonctionnalités, bug corrigé. J'ai aussi créé un composant appelé Stepper dans la librairie commune pour la librairie d'échantillons à Be-Ys Software. Comme précisé plus haut ce fut de la veille Angular principalement, des petits exercices, de la lecture de documentation officielle. Ces deux rubriques seront plus étoffées avec le projet fil rouge. Ci-dessous un exemple de l'interface et dashboard :



Un dashboard, avec des utilisations de librairies, le mode sombre est activé. Ceci est la page d'accueil IAM-Front.

• Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité

Pour le projet IAM FRONT je ne peux pas me prononcer dessus, je ne gère pas sa sécurité ni sa partie BACK. Seulement l'esthétique et la mise en forme. J'ai fait cette partie dans mon projet annexe, en apprenant par exemple à définir les rôles utilisateur, verrouillant les accès si non accrédité, blocage si formulaire non valide, section seulement valide si nous avons un compte crée. J'en parlerai plus en détail dans la description de mon projet, avec des captures d'écran contrant le brute force par exemple. Dans le journal de bord je parle toutefois de veille avec un exemple sur des exercices de reactive forms.

• Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet, à partir de site anglophone

Comme dit un petit peu plus haut dans les conseils, une bonne recherche google avec les mots clé est nécessaire, en commençant par le nom de la technologie utilisée, ensuite nous avons accès souvent à des sites, des forums ou la documentation officielle, stackoverflow, reddit etc. De plus, dans mon journal seront indiqué les problèmes que j'ai pu rencontrer.

• Extrait du site anglophone, utilisé dans le cadre de la recherche décrite précédemment, accompagné de la traduction en français effectuée par le candidat sans traducteur automatique (environ 750 signes)

Question : I'm looking into Angular RxJs patterns and I don't understand the difference between a BehaviorSubject and an Observable.

From my understanding, a BehaviorSubject is a value that can change over time (can be subscribed to and subscribers can receive updated results). This seems to be the exact same purpose of an Observable.

When would you use an Observable vs a BehaviorSubject? Are there benefits to using a BehaviorSubject over an Observable or vice versa?

Answer : BehaviorSubject is a type of subject, a subject is a special type of observable so you can subscribe to messages like any other observable. The unique features of BehaviorSubject are:

It needs an initial value as it must always return a value on subscription even if it hasn't received a next()

Upon subscription, it returns the last value of the subject. A regular observable only triggers when it receives an onnext at any point, you can retrieve the last value of the subject in a non-observable code using the getValue() method.

Unique features of a subject compared to an observable are:

It is an observer in addition to being an observable so you can also send values to a subject in addition to subscribing to it.

In addition, you can get an observable from behavior subject using the asObservable() method on BehaviorSubject.

Observable is a Generic, and BehaviorSubject is technically a sub-type of Observable because BehaviorSubject is an observable with specific qualities.

Traduction française :

Ici nous avons donc un internaute qui se penche sur la différence entre un BehaviorSubject et un Observable en Angular, c'est modèle RXJS. Il ne comprend pas bien la différence et s'interroge sur quand fait-on appel à l'un ou à l'autre.

La réponse explicative nous indique que le BehaviorSubject est un type de sujet, qui lui est un type spécial d'observable. On peut donc « s'abonner » comme avec un observable mais toute fois la différence, c'est qu'il à besoin d'une valeur initiale, car il doit retourner une valeur pendant la souscription à l'abonnement (même sans le next()). Pendant la souscription il renvoie la dernière valeur du sujet. Alors qu'un observable lui aura besoin d'un « onnext() ». Les caracteristiques d'un sujet par rapport à un observable sont : Un observateur en plus d'être un observable, on peut lui envoyer des valeurs, et s'y abonner. Pour faire simple, un observable est générique, et un BehaviorSubject est un Observable avec des qualités spécifiques.

Pour l'anecdote, j'ai même réussi à faire ce genre de recherche pour réparer un fichier qui refusait de se lancer dans un de mes jeux favoris Steam. Il s'agissait du jeu Oblivion (fait par Bethesda) qui plantait systématiquement à la suite d'un événement donné. Il a fallu rechercher la solution sur internet, rechercher le problème et apporter la correction. Maintenant le jeu est jouable. Ce genre de problème arrive dans beaucoup de logiciels, jeux vidéo, applications, mais même sur tout support informatisé, ordinateur, téléphone, console de salons, etc…

• Journal de bord, en entreprise Be-Ys Software

Présentation :

Ici j'ai noté le contenu essentiel et marquant de mes journées. Evidemment certaines journées sont plus mouvementées que d'autres. Parfois j'avais de la veille à faire, de l'apprentissage. J'ai repris à zéro Angular, et me rend compte qu'il est évidemment (?) plus poussé et « costaud » que ce qu'on a pu aborder pendant la formation. Le stage s'est fait en deux parties judicieuses, les premières semaines je faisais des corrections mineures, simples, j'apprenais à me servir de leurs multiples interfaces. J'ai même pu parfois, pendant des temps plus calmes, en fin de sprint, travailler sur mon fil rouge, et effectuer des exercices. De plus les appels aux endpoints d'une API, les collègues de BeYs savent faire, de même que des dashboard, j'ai pu en voir pendant mon stage, ce qui fut pratique pour visualiser le travail à mettre dans le projet annexe. Et puis pour donner suite au point de mi-stage, on a passé la seconde ! Ils m'ont donné un composant Angular entier, à fabriquer, dans un échantillon de librairie et de le relier à leur librairie centrale. Ce fut une tâche ardue, mais un de mes objectifs de stage à faire. J'ai aussi pris des captures d'écrans de certains tickets, évidemment tout est surveillé par l'entreprise qui ne m'autorise pas à sortir du code sensible, par ailleurs, il est hélas impossible de cloner leur projet à la maison, seuls les employés ont ce droit.

Contenu du journal :

Ce projet est majoritairement Front avec TypeScript & Angular. Participation à la refonte de IAM Front V1 vers IAM Front V2. Intégration d'un stepper.

Journal de bord : Stage Be-Ys Software, du 2 Mars au 29 Avril 2022.

Date : Le 2 Mars 2022.

Premier jour de stage, configuration de l'environnement de travail.

Prise de connaissance du projet IAM front.

Résolution d'un premier ticket permettant de naviguer dans l'application Angular (IAM front toujours).

Utilisation de JavaScript dans une page Angular pour changer dynamiquement des informations.

Appel de la librairie BeYs/common au projet IAM FRONT grâce à la commande 'link'.

Le projet a été modifié, corrigé et poussé sur la branche alpha.

À la suite de cela, une merge request (MR) est faite, qui sera vérifiée et acceptée si les tests qualité sont tous cochés. Après, la correction sera définitivement appliquée.

Une capture d'écran a été effectuée.

Le 3 Mars 2022.

Aujourd'hui j'ai pu installer en parallèle le projet fil rouge pendant le temps de pause. Configuration symfony, amélioration de la navbar symfony.

A été assigné un ticket de correction CSS dans la librairie de material design d'Angular. Recherche fastidieuse et solution finalement trouvée.

Cette dernière fut vérifiée, et ajoutée au projet IAM Front.

(Début tranquillement dans le stage pour bien voir la globalité du projet.)

Nom du ticket : Ticket 3076 : Amélioration textes

Une capture d'écran a été effectuée.

Le 4 Mars 2022.

Nouveau ticket obtenu. Correction du problème : Lorsqu'un utilisateur n'a aucun offreur de service dans ses attributs, la page reste blanche.

Recherche du problème dans le projet Angular. Problème identifié ? Utilisation de la solution adéquate.

Apprentissage de la fonction .pipe en Observable, c'est une fonction rxjs qui permet de combiner deux fonctions en une, appelée le pipe.

Une capture d'écran a été effectuée.

Le 7 Mars 2022.

Ticket du 4 Mars terminé.

Exemple de rapport sur Mattermost :

MR FRONT correction du ticket 3098 : Lorsqu'un utilisateur n'a aucun offreur de service dans ses attributs la page reste blanche (fixing blank page when no services providers)

https://sources.devtools.local/gebeeiam/beiam/iam-front/-/merge_requests/139

Correction faite dans le resolver.ts, puis le component.ts adéquat et le template html de ce dit component. (Utilisation de ngIf sur une longueur de tableau)

En clair : Si le rôle est de 'USER' nous verrons la page (avant correction la page ne chargeait pas), mais dans cette page aucun offreur de service ne sera visible, la page ne sera pas pour autant complètement blanche.

Si le rôle est admin, la page nous montrera les domaines. Si le rôle est 'SUPER_ADMIN', la page nous montrera l'intégralité du contenu, domaine et offreurs. Correction soumise et envoyée sur la branche alpha.

Une capture d'écran a été effectuée.

Le 8 Mars 2022.

Une attaque de type DOS (Denial of Service) a été faite contre Almerys, comme l'indique le rapport bruteforce.

Correction de problème de pipeline et de suppression d'indentation. En effet le test de qualité est très strict, tout doit être indenté pareillement, la suppression de certains espaces n'est pas autorisée, le projet doit être uniforme sur toutes ses pages.

Avancée rapide à côté sur le fil rouge avec l'ajout d'un petit footer. Amélioration du site projet fil rouge, étant donné qu'aucun autre ticket ne fut disponible aujourd'hui.

Une capture d'écran a été effectuée.

Le 9 Mars 2022.

Après chaque ticket traité, ne pas oublier de glisser le ticket dans les tâches effectuées dans le tableau de suivi JIRA.

Problème sur Gitlab depuis quelques jours. Avancée sur le business case pendant la résolution du problème interne. (Rien que je puisse faire hélas).

Le problème a été résolu, c'était un problème dans le framework angular de testing 'jasmine', il s'agissait d'un problème d'injection d'un service dans le composant testé, mais n'était pas présent dans un 'testbed'.

Il fallait que les conditions soient remplies pour les éléments testés. Nous avons donc modifié le composant, et donc on a pu modifier les conditions d'affichage.

On s'est assuré par la suite du bon déroulement des tests. Pour pouvoir pousser sur la branche alpha.

Une capture d'écran a été effectuée.

Le 10 Mars 2022.

Je n'ai pas eu de ticket aujourd'hui, j'ai donc pu améliorer mon business case. Des soucis à régler sur le site avec une barre de navigation irrégulière. Aussi, j'ai eu une proposition de démonstration d'Adobe XD avec l'équipe voisine de designer web.

Une capture d'écran a été effectuée.

Le 11 Mars 2022.

Des nouveaux tickets vont arriver pour les journées à venir. Amélioration du business case, page de base spatiale presque totalement terminée.

Faire une page de forum dans la page base terrestre à renommer en 'informations' ou 'communications' terrestres ?

Ticket 3107 obtenu et résolution du problème presque fini, un point et demi sur trois.

Reprise du travail lundi. Faire condition NGIF dans le template approprié selon le mode dans lequel nous sommes ''[required]="true""

Une capture d’écran a été effectuée.

Le 14 Mars 2022.

Poursuite du ticket 3107 Corrections interface front.

Veille avec des exercices sur les Angular reactive forms. Point du stage à faire avec un component lourd Angular à préparer en parallèle.

2eme point du ticket traité et corrigé.

Il s'agissait de trouver dans le projet angular le moyen de rendre le champ pour créer (ajouter/add) un utilisateur obligatoire, mais en mode édition rendre ce champ non obligatoire.Commande git notée avec link de librairie.

Prise de note faite sur les liens link de la librairie Beys au projet.

Une capture d’écran a été effectuée.

Le 15 Mars 2022.

Pas de nouveau ticket. Vérification des pipelines. Le sprint 57 se termine le 18 Mars 2023.

Nouvelles commandes git notées. Branche 3107 propre à présent.

Reactive forms en angular crée. Maintenant, à voir la différence avec un template driven forms.

Lancement des reactives forms, j'ai rassemblé mes anciens cours Angular de Human Booster pour compléter.

Forum symfony à améliorer pour intégrer dans le business case.

Pas besoin de capture d'écran ici. (Journée d’apprentissage)

Le 16 Mars 2022.

Amélioration du forum pour l'intégration dans le fil rouge.

Nouveau ticket obtenu 3123 : [N2] Navigation sur les écrans de sélection de providers/domaines.

Une sidebar qui apparaît, persiste si l'on fait précédent dans le navigateur.

Correction du ticket terminée. Dans le ngOnInit, faire la fonction dans la condition d'affichage sur FALSE.

Une capture d’écran a été effectuée.

Le 17 Mars 2022.

Fin de sprint. Réunion habituelle avec rapport sur le ticket 3123.

Aucune assignation aujourd'hui, a pu réviser et continuer un peu le projet fil rouge.

La partie du forum a été refaite.

Il reste donc à faire : Page d'accueil - Page boutique - Dashboard - Responsive
Pas besoin de capture d'écran ici. (Veille technologique, fil rouge)

Le 18 Mars 2022.

Mes tickets ont été vérifiés et validés. Confirmation sur le mail de l'entreprise. (Outlook pro)

Présentation de la fin du sprint.

Transition sur le front-v2. Démonstration IAM Gestion.

Contenu de la sprint review :

Ticket 2102 : Utilisation de l'API Almerys et Swagger UI. Attribution des rôles avec test sur le rôle admin.

Ticket 3051 : Ajouter ou modifier un attribut via l'API. (Action avec SUPER_ADMIN/ ADMIN) Erreur 202 non désirée au lieu d'une 204. Retour et corrections sur le FRONT.

Ticket 1920 : En tant qu'ADMIN ou DOMAIN_ADMIN, je dois pouvoir supprimer une application. Création d'une application rapide par API 'Swagger' UI. Suppression et retour 204.

Ticket 3106 : En tant qu'utilisateur connecté à IAM Gestion, je veux configurer les préférences du profil. Dark Mode, langages ajoutés.

Ticket 3075 : En tant qu'admin, je veux pouvoir filtrer les flux d'activité (traçabilité). Filtre et correction du flux historique, du plus récent au plus ancien.

Ticket 3070 : [GEBBEID] Update filebeat role for nuta deploiement.

Ticket 3107 : Corrections interface front. Amélioration des textes et interfaces IAM Front v2. Pour rendre l'application polyvalente.

Des rapports de tests ont été effectués. 16 sont en échec (51.61%) et 15 sont réussi (48.39%). Réparations des échecs de tests.

Assignation du ticket 3132 : Améliorations interface, deux points sur trois traités, poursuite lundi 21 Mars.

Une capture d'écran a été effectuée.

Le 21 Mars 2022.

Poursuite du ticket 3132.

Points 1 & 2 terminés.

Pour le point 3 du ticket, peut-être faire une fonction de switch dark mode qui ferait un RELOAD en dehors de l'init avec appel html ?

Des nouveaux points ont été ajouté =

4 La notification d'ajout d'un groupe => doit afficher "ajouter" et non "créé"

5 Contrôle des rôles : Alignement des boutons << < > >> entre les 2 input liste. Les aligner verticalement.

6 Contrôle des rôles : Les ascenseurs ne doivent pas s'afficher si pas scrollable. Dans les blocs de rôles. Chercher du côté de la librairie.

Définition du sprint 58. Les choses sont à corriger selon le retour des testeurs.

Petite correction d'un problème de navbar, étude sur la conversion d'une navbar parfaitement responsive.

Pour ce sprint 58 nous utilisons planitpoker, les votes portent sur la complexité des tâches à effectuer.

Une capture d'écran a été effectuée.

Le 22 Mars 2022.

Poursuite du ticket 3132.

Points 1, 2, 4, 5 & 6 terminés.

Pour corriger les points 5 et 6 de contrôle des rôles, il a fallu donner une hauteur fixe à des div parentes qui après par une fonction de calcul, ont permis de diviser par 4 les hauteurs pour répartir les 4 boutons de contrôle. << < > >>

En modifiant donc la CSS et des div parentes on a pu afficher correctement les contrôleurs. Avec de plus une propriété CSS overflow, désactivation des ascenceurs sur les boîtes de contrôle.

A côté de cela, la barre de navigation du fil rouge a été problématique mais a finalement été corrigée. Tout le business case est donc maintenant responsive.

Le module front est donc validé normalement. Il reste à faire la boutique, l'accueil et le dashboard.

Réunion à 16 heures pour les modifications de la librairie beys software. A chaque sprint les modifications seront faites.

Une capture d'écran a été effectuée.

Le 23 Mars 2022.

Léger souci sur le ticket 3076 qui a été corrigé.

Ticket 3132 avec le DarkMode à poursuivre. Tous les autres points sont corrigés.

Apparition d'un problème avec le ticket 3107. Résoudre le tout et pousser(push) le correctif dans une seule branche.

Ticket 3132 terminé. Il s'agissait d'un problème de DarkMode avec la librairie personnalisée de Be-Ys Software. Cette librairie est un composant enfant de Iam Front. <beys-navbar></beys-navbar> en est l'exemple. Le html est appelé dans un autre template html par ces balises.

Pour solutionner ce problème épineux il a fallu d'abord :

- Définir les couleurs dans une fonction "init", qui seront les couleurs des modes light/dark.

- Faire un 'output' dans le composant (le navbar.component.ts dans la libs beys), ensuite le brancher sur le html. (Navbar.component.html)

- Faire des fonctions, une dans le "ui.service.ts", une dans le "main.component.ts", appel du ui.service et du setColor dans l'initData.

- Raccorder le tout dans le main.component.html.

IAM FRONT (html principal) -> BEYS LIB (librairie html indépendante) -> darkmode - output/emit -> main.ts IAMFRONT enableDarkMode -> uIservice 'isDarkMode' -> Dashboard

'isDarkMode.subscribe'

Ce petit circuit permet de vérifier dans quel état nous sommes, et par un simple boolean 'true false', de changer les couleurs des éléments sur la couleur du reste du site.

Une capture d’écran a été effectuée.

Le 24 Mars 2022.

Aujourd’hui j’ai pu débuter la page d'accueil du business case. Puis aussi faire le début de la page résumé du projet Human Booster.

Le ticket 3076 et le ticket 3132 sont finalement terminés et mis à jour.

La recherche du ticket 3107 et son bug généré montre qu'il est plus complexe qu'il n'y paraît. Peut-être en poursuite le vendredi 25 avec de l'aide.

Problème de Pipeline (Merge Request qui a un mauvais test de qualité), il faudra rétablir ça au plus vite.

Pas besoin de capture d'écran aujourd'hui.

Le 25 Mars 2022.

A l'arrivée, correction directe du Pipeline qui semble fonctionner. Il suffisait de supprimer les fichiers et imports non utilisés. Ensuite faire un ng lint + ng rest, puis le git status, git add -A, git commit et git pull origin 'nom de ticket'.

En attente de nouveau ticket. Si temps libre, page résumée à faire aujourd'hui, et avancer sur la page accueil du Business case. Correction d'un léger souci de traduction pour donner suite à la découverte.

Nouveau ticket 3138 : Changer l'affichage du bouton "retirer" en mode edit.

Une capture d’écran a été effectuée.

Le 28 Mars 2022.

Fiche de résumé complétée et terminée. Je l'ai soumise à Human Booster et le retour fut positif. J'ai donc adopté la correction et retourné le tout.

J'ai mis ma correction de la librairie sur la branche alpha et mon pipeline a été validé.

Maintenant la merge est en attente d'acceptation, les tests sont en cours.

Demain si c'est validé il faudra mettre à jour la dépendance dans le package.json. Et puis faire un git add commit push depuis le front.

Ainsi sera clôturé le ticket 3138.

Le 29 Mars 2022.

Toujours en attente de la MR, elle est bonne sur les tests. Mais elle doit être vérifiée et validée. Une fois cela fait la MR front sera à faire, et puis la petite routine habituelle.

J'ai fait un peu de veille et commencé à faire la page boutique sur le fil rouge, la gestion du panier est délicate cependant.

Le 30 Mars 2022.

Assignation d'un ticket IAM-3156 : [N1] [Ajout d'une identité] Titre erroné.

Ce ticket avait été repéré et réparé il y'a quelque temps. Il sera donc corrigé avec la MR du ticket précédent le 3138.

La MR est validée, ticket 3138 et 3156 corrigés et mis à jour.

Enfin, dans le fil rouge, la première version de la boutique est faite avec son panier, il ne me reste qu'à la peaufiner.

Le 31 Mars 2022.

Création d'un component dashboard angular. Exercice de connexion Angular avec fichier d'exemple pour le business case. Amélioration de la boutique sur le business case. A utiliser avec des librairies JS pour faire des graphiques et données de mois.

Le 1 Avril 2022.

Aujourd'hui fut une journée calme, j'ai refait des exercices Angular pour comprendre les endpoints et modifié le business case.

La page boutique est presque finie à 100%.

La page de contact a été bien modifiée et l'utilisateur peut maintenant modifier ses informations. De nouvelles ressources sont disponibles, il est aussi possible de choisir une autre adresse et un pays.

La page d'admin avec la liste des USER est prête, il faudra la faire avec le dashboard angular.

Le 4 Avril 2022.

Continuer si possible le projet fil rouge. Avec mot de passe hashé à faire. Crud boutique. Gérer la facturation. Recentrer image + favicon. Dashboard.

Aujourd'hui à 14h30 réunion avec Charlotte. J'ai avancé sur le business case et terminé les hashages de mot de passe.

Ce week-end j'ai pu recentrer les images, faire le crud de la boutique. Il me reste la facturation dashboard et favicon. Tout le reste est amélioré, le contenu de certaines choses ou modifications requiert d'être connecté pour pouvoir procéder à l'action.

Le 5 Avril 2022.

Hier soir j'ai pu faire les gestions de rôle symfony. Cela fonctionne et me permet de masquer des zones ou des pages pour le simple utilisateur. Bloquer l'accès à certaines pages et options. Seul l'ADMIN peut faire ce que bon lui semble sur le site.

Aujourd'hui nouveau ticket : intégration d'un 'Stepper'.

Création du stepper qui sera un composant générique dans la libs de beys-software. On exporte des données de la beys-software-lib-sample vers beys-software-common-lib.

Cela permettra d'utiliser <beys-stepper></beys-stepper> qui invoquera ledit stepper.

Découverte du PowerPoint à rendre pour le 25 Avril, sera faite après le dossier projet, entre le 15 et 25.

Le 6 Avril 2022.

Poursuite du projet de faire un composant entier dans une librairie Angular.

Le stepper est finalement affiché en mode DUR. Il faut le rendre dynamique, avec des statuts. C'est du Angular pur. Input/output. Avec des NGIF NGFOR NGCONTAINER.

Poursuite de ce projet le lendemain.

Le 7 Avril 2022.

Aujourd'hui révision Angular. Création de petits projets. Une mini boutique dans un premier temps et un prototypage de dashboard afin de monter en compétences.

Le JIRA a été mis à jour de mes heures effectuées.

Création d'une mini boutique en Angular.

Nouvel exercice Angular qui simule une connexion à une base de données / API, poursuite le 8 Avril.

8 Avril 2022.

Début de réflexion sur le dossier projet d'Human Booster. Une petite moitié déjà effectuée.

Sprint Review 58 aujourd'hui.

Ticket d'amélioration d'interface passée en revue.

Ticket 3107 - Ticket 3138 - Ticket 3028 - Ticket 2964

Poursuite de la montée en compétence d'Angular. Des nouveaux exercices et parcours de documentation pour concevoir le stepper.

11 Avril 2022.

Ce week-end j'ai réussi à intégrer sur le projet fil rouge de nouvelles fonctionnalités. Une barre de recherche fonctionnelle, un bouton précédent, une boutique affichée par ordre alphabétique.

Poursuite de la veille sur Angular avec simulation d'appel à une API grâce à un tutoriel. On a un sprint refinement cet après-midi, pour définir les nouvelles tâches à venir, avec les tickets. Découverte d'un nouveau projet angular à faire en tutoriel, un dashboard qui permet de comprendre l'utilisation de charts, et à appliquer sur le fil rouge plus tard.

12 Avril 2022.

Aujourd'hui j'ai poursuivi ma veille pendant les réunions d'assignation de tâche dans le sprint. J'ai pu effectuer un exercice angular sur le système de dashboard et l'intégrer au projet symfony. Ce petit tableau de bord est accessible par un lien que seul l'ADMIN peux voir dans la page de gestion des utilisateurs. (Impossible d'y accéder autrement)

Ce dashboard simule une connexion à une API en mode « dur » grâce à un service, qui se connecte dans mon cas sur une fausse url de donnée json. (Je comprends la différence avec des endpoints à une api, en se connectant, c'est elle qui renvoie des json, ainsi les json rendu par une API seront forcément changeant et donc dynamique).

Mes données en « dur » montrent une liste de client, d'achat, de statistiques, des courbes, des graphiques, des diagrammes types camembert, c'est une parfaite page de contrôle admin.

Voici les commandes que j'ai utilisé pour ce petit projet angular :

- ng add @angular/material (librairie angular material)
- npm install ng2-charts –save (dépendance ng-2 charts qui a besoin de charts.js ci-dessous)
- npm install chart.js --save
- npm install –save-dev ng2-charts-schematics (autre dépendance à chart.js)
- ng generate @angular/material :dashboard dash (génère un dashboard responsive grâce à angular material)
- ng generate ng2-charts-schematics :radar charts/product-sales-chart
- ng generate ng2-charts-schematics :pie charts/sales-traffic-chart

- ng generate ng2-charts-schematics :line charts/annual-sales-chart
- ng generate ng2-charts-schematics :bar charts/store-sessions-chart

Cela va générer automatiquement, un radar, un camembert, une courbe et des diagrammes en bar pour le dashboard.

- ng generate @angular/material :table orders-table (génère automatiquement comme composant une table angular)

Ce fut intense, mais c'est fonctionnel grâce à de fausses données. Ce panneau est accessible maintenant depuis la page de gestion des utilisateurs.

PARTIE II - Projet fil rouge : La Nimes'Alerie

Dans cette page j'ai établi les tâches à faire pour le site de la Nimes'Alerie.

Ceci est conçu pour gérer le temps et prioriser les tâches à effectuer.

Certaines pages ont été faites de mon propre chef pour construire un site plus étoffé et amusant à parcourir.

J'ai constaté qu'avec ce projet, on a sans cesse de nouvelles idées d'implantation, d'amélioration, je pense que ce ne sera jamais parfait.

On peut parcourir ce site en mode lecture, certaines rubriques ne seront accessibles que par compte, et certaines que si vous êtes l'administrateur de ce site.

Utilisation de technologies : HTML, CSS, JAVASCRIPT, SYMFONY, TWIG, PHP, MYSQL & DQL

Le site de la Nimes'Alerie a été conçu en parallèle, pendant le temps libre, en toute opportunité. Ci-dessous voici les grands points que j'ai pensé à lister et à finir qui sont parfaitement fonctionnels sur mon site.

Je me suis inspiré de ce que j'ai vu en stage pour hiérarchiser les tâches, et les ''qualifier'' avec le « : effectué ».

- Création du projet Symfony de la Nimes'Alerie : effectué

- Fabrication de la base de données répondant aux entités du site : effectué

- Conception d'une top bar restante en haut de l'écran : effectué

- Implantation d'une barre complémentaire à cette top bar qui sera elle aussi fixée en haut : effectué

- Conception d'un footer pour encadrer le contenu du site : effectué

- Importation d'un favicon faisant une identité de site : effectué

- Création d'un formulaire de connexion utilisateur : effectué

- Création d'un formulaire de contact Nimes'Alerie : effectué

- Elaboration d'un CRUD dans les articles de rédaction blog, seul un 'ADMIN' pourra effacer du contenu) /ajout /post /edit /delete : effectué

- Création d'une page bonus cachée dans la barre de navigation : effectué

- Fabrication d'une page space base html/css type classique : effectué

- Création d'un forum grâce à un CRUD : effectué

- Alignement parfait des contenus avec la top bar et le footer : effectué

- Faire en sorte que le site de la Nimes'Alerie soit responsive et ajouter des corrections mineures : effectué

- Développement d'une page d'accueil faisant office de résumée sur le site : effectué

- Conception de la boutique, d'une page détail produit, d'un panier avec un calcul hors taxe (HT) et toute taxe (TTC) : effectué

- Permettre à l'utilisateur de modifier ses informations clients (e-mail, nom/prénom, adresse etc.…) : effectué

- Développement du hashage de mot de passe pour sécuriser la connexion utilisateur (client) : effectué

- Création de vrais produits en BDD (avec la boutique) : effectué

- Gestion de différent rôle propre au site de la Nimes'Alerie (gestionnaire = ADMIN, utilisateur = USER) : effectué

- Résolution d'un problème d'affichage dans la boutique et aussi dans la gestion du panier : effectué

- Possibilité pour le ["ROLE_ADMIN"] d'ajouter, voir, éditer ou supprimer des produits de la boutique : effectué

- Possibilité pour le ["ROLE_ADMIN"] d'accéder par la page 'La Nimes'Team' à la gestion des utilisateurs par un CRUD : effectué

- Une barre de recherche de produit fonctionnelle dans le header : effectué

- Mise en place de la sécurité d'authentification anti brute-force : effectué (A l'aide de composer require symfony/rate-limiter)

- Ajout d'un bouton précédent interne (dans certaines pages) et partout dynamiquement (à côté du bouton ‘’revenir en haut’’) : effectué

- Ajout d’un bouton ‘’mode sombre’’ à côté des deux précédents, qui appelle une fonction de darkMode permettant de changer les couleurs d’affichage selon la lumière : effectué

- Création d’un dashboard avec des utilisations de librairies : effectué

Il me reste à faire au moment où j’écris ce dossier :

- Une facturation qui s’effectue quand je valide un panier.

J’ai encore l’impression que le site peut être peaufiné, amélioré, avec toujours plus de nouvelles implémentations. D’autre part, je souhaite compléter ces deux points avec mon projet annexe, le fil rouge Human Booster. Ce sera dans la description du site de la Nimes’Alerie.

DESCRIPTION DU SITE DE LA NIMES’ALERIE

• Présentation du jeu d’essai élaboré par le candidat de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues)

Nous allons voir ensemble, les différentes pages, ce site est assez complet, reprend beaucoup de choses vues en formation et il est responsive.

Maintenant voici une présentation du site et une description des rubriques.

Présentation du fil rouge : La Nimes’Alerie.

J'ai commencé par étudier la demande du client, un site boutique utilisant deux framework : Angular pour le côté Dashboarding et Symfony pour pratiquement tout le reste.

Angular achemine donc une connexion à une API, pour transformer ces données et en faire un Dashboard. Symfony lui servira à afficher des informations dans des vues (twig) les ‘’controllers’’ respectifs. Le site est découpé en plusieurs sections qui seront reliées par une barre de navigation, et le tout sera fermé par un footer avec des informations génériques.

Le site utilise aussi du JavaScript, pour les petits boutons d'interface, le décompte de la barre du haut, la page bonus. En gros, tout ce que j'ai pu voir en formation, je l'ai repris, rassemblé et compilé dans le site du projet fil rouge !

Nous avons donc :

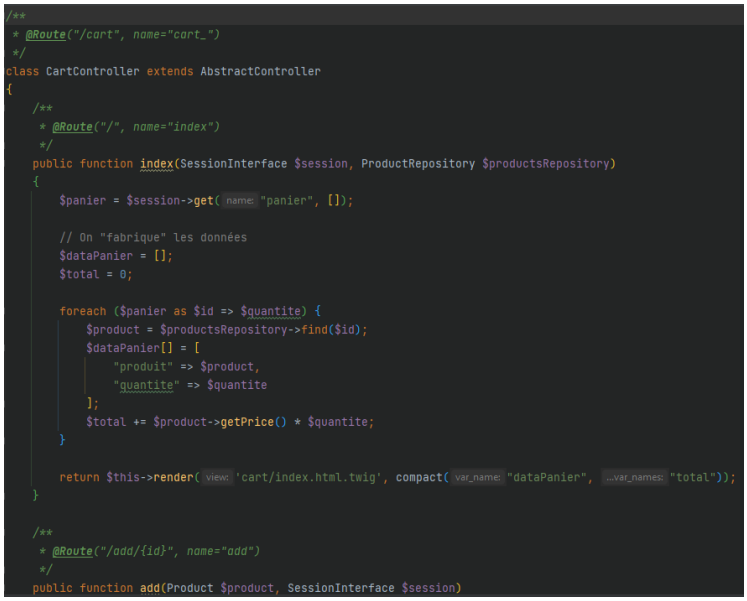
- La barre de navigation avec les différentes rubriques du site, les liens des réseaux sociaux, un compteur de prochain lancement. Une zone de recherche est incluse à l'intérieur.
- Le footer lui vous présentera les moyens de paiement, les coordonnées, la bannière human booster.
- Des boutons magiques en bas à droite pour vous rendre l'expérience plus agréable, mode nuit, précédent et retour en haut de la page.
- Une page de connexion avec un formulaire à remplir, le client renseigne ses coordonnées, son adresse, nom, prénom, email, etc...
- Une page d'accueil qui reprend des éléments récents. Cette page est la page d'arrivée pour l'utilisateur, elle présente brièvement le site, ses fonctions.
- La Nimes'Team, un formulaire de contact, et une section qui permet d'éditer vos informations de compte, et d'aller sur la page ADMIN si vous avez les accréditations.
- Une page de boutique avec les différents articles, détails, photos, et panier. Avec cette page vous pouvez procéder à vos achats. En ajoutant des produits, les supprimant, modifier un contenu du panier, et calcul de la TVA inclus !
- La page Article (de rédaction), ici vous trouverez facilement des sujets de rédaction fait par la communauté, des informations sur la vie du site.
- La base terrestre, qui est un forum communautaire entre utilisateurs de la Nimes'alerie, grâce à cela, vous pouvez parler, débattre, créer des sujets passionnants.
- La pension spatiale permettant d'envoyer vos animaux dans l'espace (en toute sécurité bien sûr).
- Une page de bonus a été faite, elle est cachée dans le header, et permet de jouer à un mini jeu en JavaScript pur, ce ne fut pas simple à intégrer.
- La page admin est protégée par une accréditation. Et si vous tentez d'y accéder par une url, cela ne sera tout simplement pas possible.
- La sécurité anti brute force est aussi présente, ainsi que le hashage des mots de passe. Seuls les admins pourront modifier certaines choses comme l'édition de la boutique.

Pourquoi avoir fait ce site ? Ce site de e-commerce répond à un besoin pertinent dans le monde numérique d'aujourd'hui. Le client souhaitait pouvoir avoir sa propre boutique complète de A à Z. Il est adapté à tous types d'écrans, mobile comme fixe. J'ai amélioré l'expérience en construisant des pages supplémentaires qui font le mélange entre boutique et réseau social de communication. Des accès aux réseaux sociaux traditionnels sont possibles, mais échanger avec les clients, écrire dans le forum, faire vivre la plateforme, c'est cela le site 2.0 ! Vous avez un souci de

commande ? Une question sur un produit ? Notre forum est là pour cela ! Peut-être qu'un autre utilisateur a rencontré ce problème ? Nous mettrons en valeur la solution au problème que vous posterez dans notre site, sachez aussi que vous pouvez écrire cela sous la forme d'un article de blog. Le site est évidemment sécurisé contre le vol de mot de passe, tentative de brute force, et accepte des paiements sécurisés avec carte bleue et Paypal. Vous pouvez tout faire sur la Nimes'Alerie, le monde vous appartient ! Et pas que le monde, l'espace aussi maintenant !

• Réalisations du candidat comportant les extraits de code les plus significatifs et en les argumentant, y compris pour la sécurité et le web mobile

Ici je vous montre une fonction totale d'ajout de panier dans le « cart controller », le panier est un tableau vide avec un total initialisé à zéro. Pour chaque contenu du panier, nous avons la variable $product qui recherche dans le repository un $id. Alors la variable $dataPanier va se remplir de produit et de quantité, et le total, récupérera le produit, son prix, et le multipliera par la quantité, ce qui fera le montant du panier !



```php
/**
 * @Route("/cart", name="cart_")
 */
class CartController extends AbstractController
{
    /**
     * @Route("/", name="index")
     */
    public function index(SessionInterface $session, ProductRepository $productsRepository)
    {
        $panier = $session->get( name: "panier", []);

        // On "fabrique" les données
        $dataPanier = [];
        $total = 0;

        foreach ($panier as $id => $quantite) {
            $product = $productsRepository->find($id);
            $dataPanier[] = [
                "produit" => $product,
                "quantite" => $quantite
            ];
            $total += $product->getPrice() * $quantite;
        }

        return $this->render( view: 'cart/index.html.twig', compact( var_name: "dataPanier", ...var_names: "total"));
    }

    /**
     * @Route("/add/{id}", name="add")
     */
    public function add(Product $product, SessionInterface $session)
```

Les trois captures d'écran représentent le « controller » du panier !

```php
public function add(Product $product, SessionInterface $session)
{
    // On récupère le panier actuel
    $panier = $session->get( name: "panier", []);
    $id = $product->getId();

    if (!empty($panier[$id])) {
        $panier[$id]++;
    } else {
        $panier[$id] = 1;
    }

    // On sauvegarde dans la session
    $session->set("panier", $panier);

    return $this->redirectToRoute( route: "cart_index");
}

/**
 * @Route("/remove/{id}", name="remove")
 */
public function remove(Product $product, SessionInterface $session)
{
    // On récupère le panier actuel
    $panier = $session->get( name: "panier", []);
    $id = $product->getId();

    if (!empty($panier[$id])) {
        if ($panier[$id] > 1) {
            $panier[$id]--;
        } else {
            unset($panier[$id]);
```

```php
    // On sauvegarde dans la session
    $session->set("panier", $panier);

    return $this->redirectToRoute( route: "cart_index");
}

/**
 * @Route("/delete/{id}", name="delete")
 */
public function delete(Product $product, SessionInterface $session)
{
    // On récupère le panier actuel
    $panier = $session->get( name: "panier", []);
    $id = $product->getId();

    if (!empty($panier[$id])) {
        unset($panier[$id]);
    }

    // On sauvegarde dans la session
    $session->set("panier", $panier);

    return $this->redirectToRoute( route: "cart_index");
}

/**
 * @Route("/delete", name="delete_all")
 */
public function deleteAll(SessionInterface $session)
{
    $session->remove( name: "panier");
```

Pour ajouter un panier, on initialise la variable $panier dans une $session, qui récupère le contenu du panier. On peut alors ajouter des produits dans le panier, et le sauvegarder dans la session grâce à $session->set.

Pour la fonction qui supprime du contenu, c'est exactement l'inverse, on décrémente les produits (contre l'incrémentation qui est l'ajout) et pareil on sauvegarde le tout dans une session.

Pour effacer un objet on ne se contente pas de simplement de décrémenter, mais de tout supprimer ce qui est relatif à ce produit (toujours stockage de session). Quant à la fonction de vider le panier, elle récupère la session, « remove » (vide le panier) et recharge la page.

Nous pouvons utiliser des requêtes DQL (Doctrine Query Language), pour faire une fonction de recherche d'un article :

```php
public function findAllNames(string $title): array
{
    // FROM product AS product
    return $this->createQueryBuilder( alias: 'product')
        // SELECT product.title,
        ->select( select: 'product.title')
        // WHERE product.title LIKE '%$title%' => $title est un paramètre
        ->where( predicates: 'product.title LIKE :title')
        ->setParameter( key: 'title', value: '%' . $title . '%')
        ->getQuery()
        ->getResult()
        ;
}
```

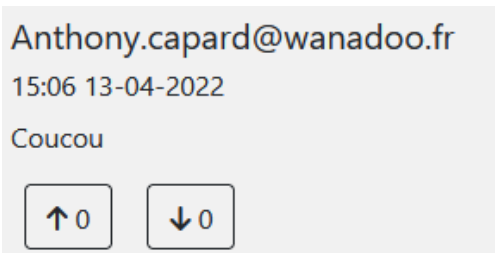
```php
/**
 * @Route("/product/findAllNames/{title}", name="ajax_product_find_all_names")
 */
public function index(string $title): JsonResponse
{
    return new JsonResponse($this->productRepository->findAllNames($title));
}
```

Cette fonction permet de faire une requête sur les produits, sélectionner les produits par leur nom respectif, ou ils sont, et de les trouver « LIKE », obtenir le résultat de la requête.

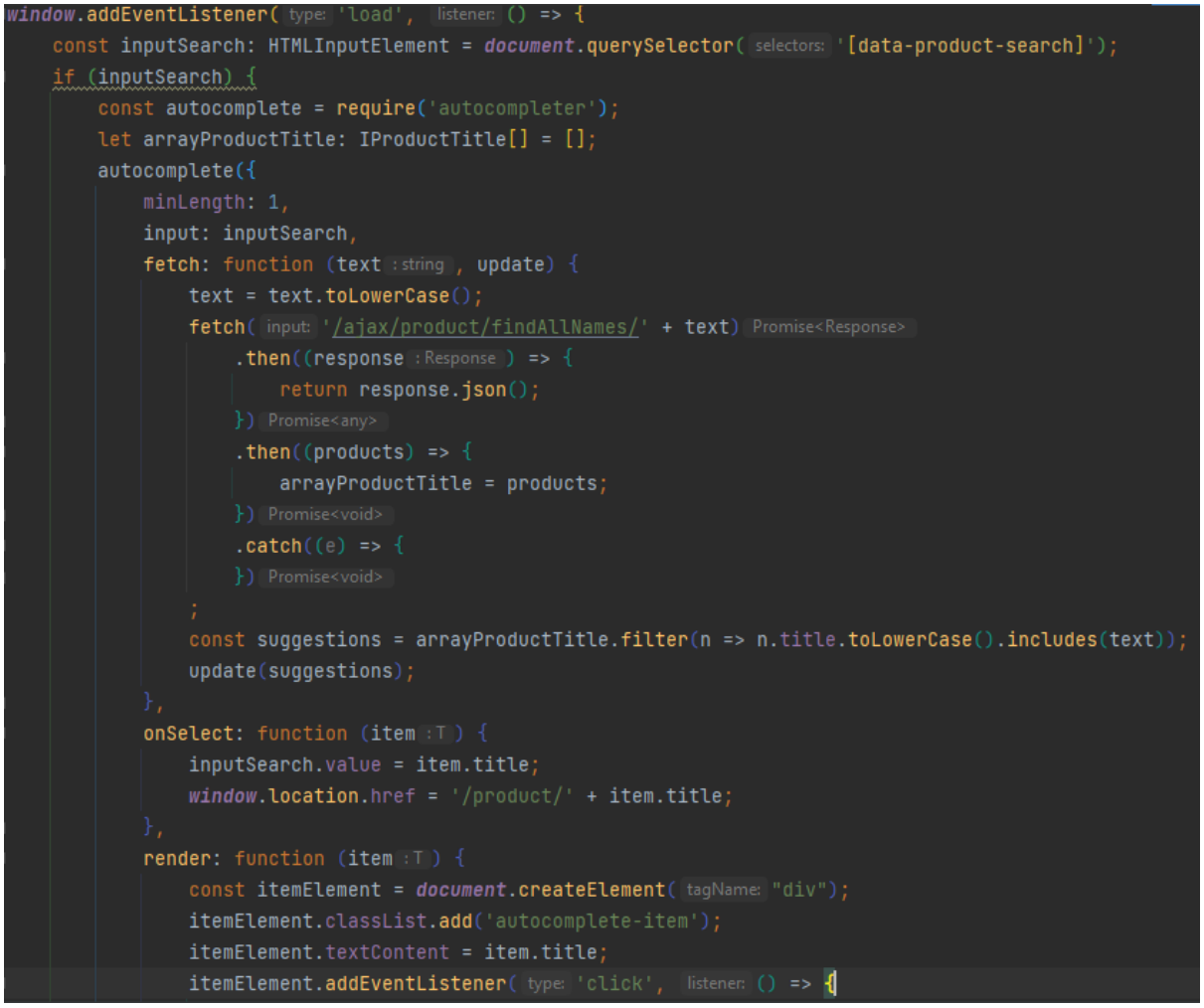
```php
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add( child: 'content', type: TextType::class, [
            'label' => false,
            'attr' => [
                'placeholder' => 'article.index.table.name',
            ]
        ])

        ->add( child: 'thread', type: EntityType::class, [
            'class' => Thread::class,
            'label' => 'Sujet',
            'choice_label' => 'subject',
            'query_builder' => function (EntityRepository $er) {
                return $er->createQueryBuilder( alias: 'thread')
                    ->orderBy( sort: 'thread.subject', order: 'ASC');
            }
        ])
        ->add( child: 'submitButton', type: SubmitType::class, [
            'label' => 'general.button.submit',
        ])
    ;
}
```

Ici c'est un formulaire de création d'un « post », qui sera relié par exemple à une catégorie, ou un sujet, ici c'est donc le Thread ::class à laquelle le « post » sera relié. Ensuite ça sera classé par ordre alphabétique ('ASC'). C'est un petit morceau du forum, qui suit ce schéma : Rubrique -> Sous Rubrique -> Sujet -> Message (« post »).

Toute ces étapes sont imbriquées les unes dans les autres et forment un forum. Nous pouvons créer des rubriques à volonté, des sous rubriques etc… A la toute fin de la chaine nous pouvons voir les messages, et ajouter des « pouces positif » ou « pouces négatif », c'est un système de vote pour voir la pertinence du message ou pas.



Nous voyons aussi la date de création et l'auteur !



```
window.addEventListener( type: 'load', listener: () => {
    const inputSearch: HTMLInputElement = document.querySelector( selectors: '[data-product-search]');
    if (inputSearch) {
        const autocomplete = require('autocompleter');
        let arrayProductTitle: IProductTitle[] = [];
        autocomplete({
            minLength: 1,
            input: inputSearch,
            fetch: function (text :string , update) {
                text = text.toLowerCase();
                fetch( input: '/ajax/product/findAllNames/' + text) Promise<Response>
                    .then((response :Response ) => {
                        return response.json();
                    }) Promise<any>
                    .then((products) => {
                        arrayProductTitle = products;
                    }) Promise<void>
                    .catch((e) => {
                    }) Promise<void>
                ;
                const suggestions = arrayProductTitle.filter(n => n.title.toLowerCase().includes(text));
                update(suggestions);
            },
            onSelect: function (item :T ) {
                inputSearch.value = item.title;
                window.location.href = '/product/' + item.title;
            },
            render: function (item :T ) {
                const itemElement = document.createElement( tagName: "div");
                itemElement.classList.add('autocomplete-item');
                itemElement.textContent = item.title;
                itemElement.addEventListener( type: 'click', listener: () => {
```

Ici nous avons le mécanisme de la barre de recherche de la Nimes'Alerie. Dans ce code nous avons un addEventListener qui est sur un inputHtml, le selecteur dans le cas présent est « data-product-search », la condition est que si vous entrez une donnée dans le champ de texte, une autocompletion sera lancée et vérifiera avec la base de données le champ que vous êtes en train d'entrer. Le texte sera en suggestion, en petit caractères, et son autocomplétion se mettra à jour avec ce que vous écrivez. La fonction recherchera spécifiquement dans les produits (de la boutique) et affichera à partir de cette requête ajax, la page de produit correspondant si vous avez bien renseigné le champ de recherche.

Pour le web mobile, si nous parlons de rendre le tout responsive, le site est fait entièrement avec l'aide de bootstrap 5.

J'ai utilisé des fonctions anti-brute force dans le security.yaml qui empêcheront toutes tentatives de connexion si vous vous trompez trois fois de suite de mot de passe, toutefois vous pourrez retenter au bout de quinze minutes.

Voici une première capture d'écran qui vous permet de voir le paramètre symfony :

```
  # configure the maximum login attempts in a custom period of time
login_throttling:
  max_attempts: 3
  interval: '15 minutes'
```

Certaines sections seront quant à elles inaccessibles si vous n'avez pas les droits d'administrateur requis. Vous ne pourrez pas non plus faire un contournement par l'URL, elle est protégée dans le « controller » de la page reliée. Ci-dessous la capture d'écran vous montrant la commande verrouillant ledit controller entier :

```
#[IsGranted('ROLE_ADMIN', message: 'Vous n\'avez pas les autorisations nécessaires', statusCode: 404)]
#[Route('/user')]
```

D'autre part, vous devrez avoir un compte normal d'utilisateur pour accéder à certaines actions. Bien sûr vous pouvez naviguer librement le site, mais prenons l'exemple du forum ici :

```
{% if app.user %}

    <div class="container mt-2 text-center">
    <div class="row justify-content-center text-center"...>

    <div class="row mt-3"...>
{% else %}
    <div class="container mt-4 text-center"...>
{% endif %}
</div>
```

Ici, nous sommes dans le « template » du forum symfony. Une simple condition disant, si vous êtes connectée vous verrez des boutons pour la suite, sinon, une invitation à créer un compte ou vous identifier avec ce dit compte.

J'ai utilisé une version combinée au rôle admin par le template dans la page de la boutique, voyez plutôt :

```
{% if app.user %}
{% if is_granted('ROLE_ADMIN') %}
    <div class="container text-end">
        <a class="btn btn-danger text-white p-2" href="{{ path('product_new') }}">
            <i class="fa-solid fa-plus"> {{ 'product.index.new'|trans }}</i>
            <i class="fa-solid fa-plus"></i>
        </a>
    </div>
{% endif %}

{% endif %}
```

J’ignore pour l’instant si c’est la meilleure pratique, mais cela fait ce que je souhaite. Si vous avez le rôle admin, vous pourrez voir ces boutons d’actions ! Un petit script vous demandera si vous souhaitez poursuivre cette action :

```
{% if app.user %}
{% if is_granted('ROLE_ADMIN') %}
    <div class="container mb-4">
        <h5 class="fw-bold text-center mb-2 pb-2">Gestion du produit :</h5>
        <a class="logo-link" href="{{ path('product_edit', {'title': product.title}) }}"...>
        <a class="logo-link text-danger"
           href="{{ path('product_delete', {'title': product.title}) }}"
           onclick="return confirm('ATTENTION ! Êtes-vous sûr(e) de vouloir continuer ?')">
            <i class="fas fa-trash-alt"...>
        </a>
    </div>
{% endif %}
{% endif %}
```

Nous sommes ici dans le « user controller », avec une fonction qui nous permet de changer le mot de passe. Nous utilisons ici un générateur de mot de passe hashé, qui est similaire à la création du compte. Quand vous tapez un mot de passe cela génère un « hash » dans la base de données, voici l’exemple avec image :

```
#[Route('/edit/{id}', name: 'user_edit')]
public function edit(Request $request, UserPasswordHasherInterface $userPasswordHasher, UserAuthenticatorInterface $userAuthenticator,
                     AppCustomAuthenticator $authenticator, EntityManagerInterface $entityManager, User $user): Response
{
    $form = $this->createForm( type: UserFormType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // encode the plain password
        $user->setPassword(
            $userPasswordHasher->hashPassword(
                $user,
                $form->get('plainPassword')->getData()
            )
        );

        $entityManager->persist($user);
        $entityManager->flush();
        // do anything else you need here, like send an email

        return $userAuthenticator->authenticateUser(
            $user,
            $authenticator,
            $request
        );
    }
    $this->addFlash( type: 'message',  message: 'Profil mis à jour !');
    return $this->createFormFromEntity($request, $user,  template: 'user/edit.html.twig');
}
```

• Conclusion de la formation à l'apprentissage du métier.

Tout ceci fut dit pendant les remerciements possible grâce à cette formation de développeur web et web mobile. Pour ma part le cursus s’est très bien passé, j’ai eu la chance d’être dans une bonne équipe, bien entouré et bien encadré. Nous avons

rencontré des professionnels du milieu, appris énormément, et ce qui semblait dans mon cas impossible, telle que la création d'un site web de A à Z fut, bien qu'imparfait, réalisable. Mais ce n'est que le début d'une nouvelle étape, d'une nouvelle vie si je puis dire. En effet le stage m'a montré que dans mon cas j'ai encore tellement à voir et à apprendre, entre l'Angular que nous avions fait, et celui pratiqué est d'un niveau plus corsé évidemment. Merci encore à toute cette équipe d'avoir pu rendre tout cela réel, aux formateurs et à l'équipe d'avoir été là parfois à des heures bien plus avancées que celle de la fin de journée de travail pour répondre à des doutes ou des questions.

Par ailleurs, j'ai pris connaissance le 12 Avril d'une poursuite possible de formation en alternance, ce qui est une bonne nouvelle, j'ai postulé évidemment auprès de Human Booster, pour un premier rendez-vous le 10 Mai à 9h30. Je l'ai fait suite aux conseils de l'entreprise ou je suis actuellement en stage qui semble disposée à me prendre en tant qu'alternant. Je suis motivé pour poursuivre dans ces excellentes conditions pour monter toujours plus haut en compétences.

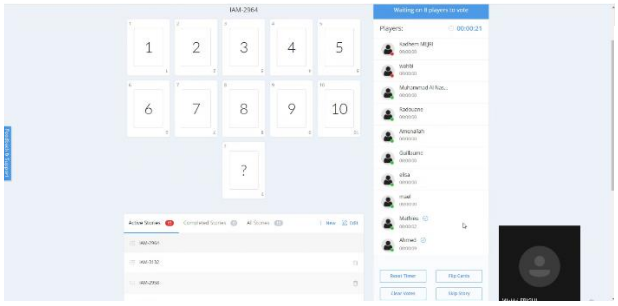
• Les captures d'écran.

Les captures d'écran seront celles de stage et du projet fil rouge ici avec une description explicative jointe.

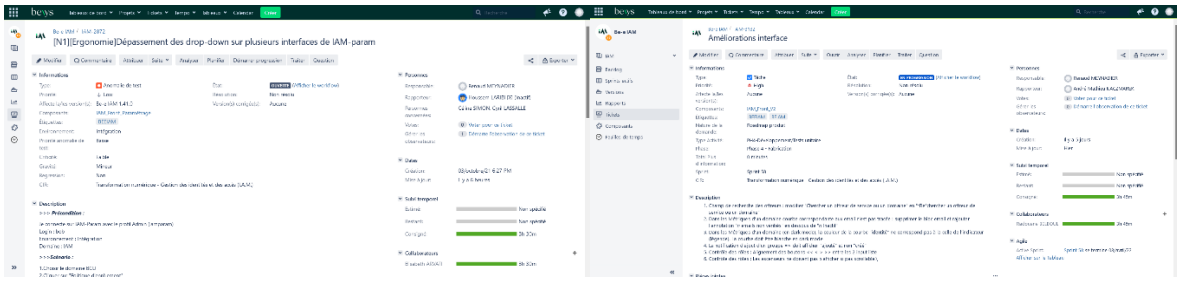
Capture d'écran lié au stage :

Ici je vais vous montrer des captures d'écran relatif à Be-ys software. Cela regroupe ce que j'ai pu voir en stage, planitpoker, les tickets, test qualité, gestions de pipeline jusqu'au déploiement final quand tous les critères sont vérifiés et acceptés.

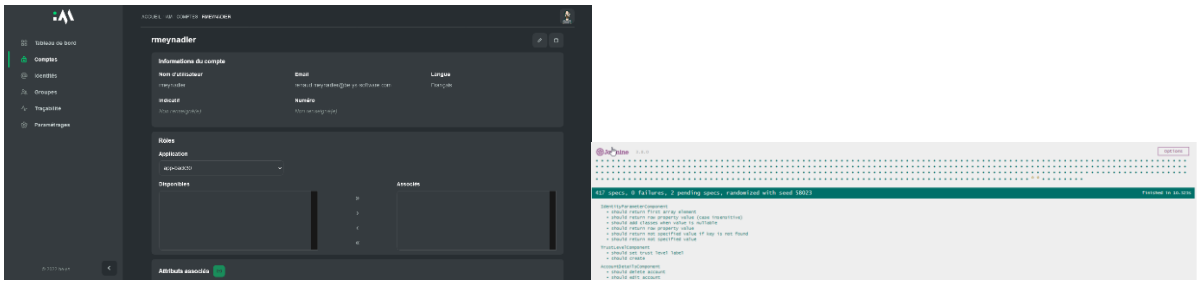
Voici tout d'abord le planit poker qui permet d'estimer la difficulté des tâches par les collaborateurs du projet. Juste à côté vous avez la politique de l'entreprise qui vous rappellera à l'ordre si vous vous aventurez là ou vous ne devriez pas !



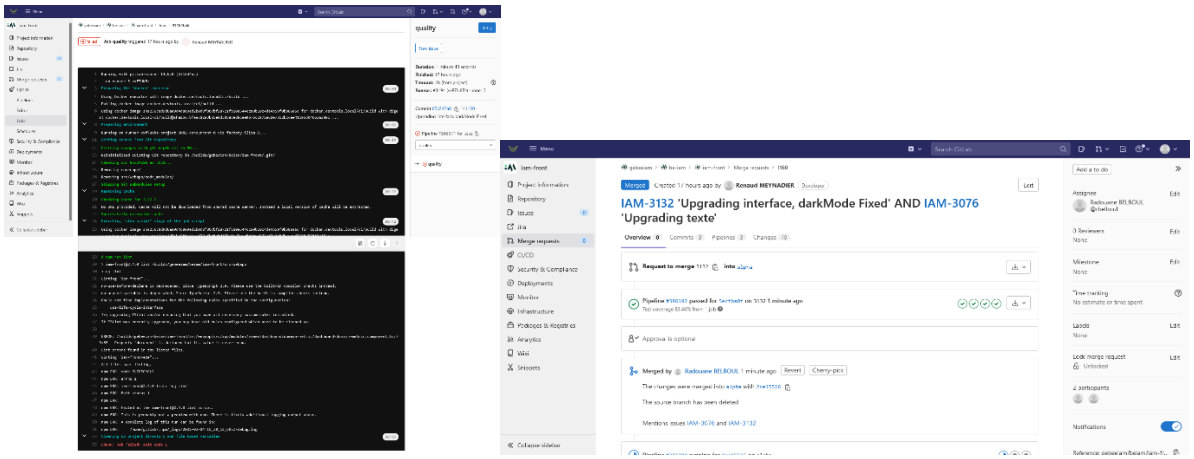
Plus sérieusement, ici nous avons par exemple des tickets traités et terminés. Cette gestion complète est avec Jira.



Dans la capture de gauche j'ai participé au darkMode, et l'alignement parfait des objets entre eux dans la partie « rôles ». A droite, les tests sous « jasmine » qui retournent aucune erreur !



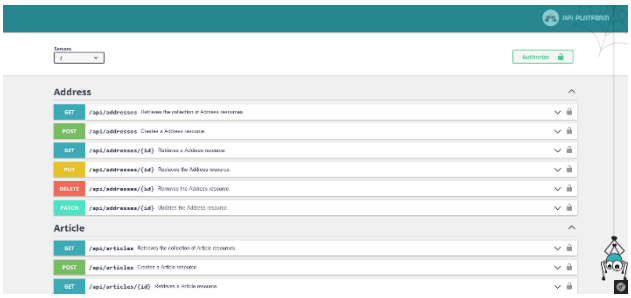
Juste ici je vous montre un exemple d'échec de test avec les pipelines, et les pipelines validées pour donner suite à un code proprement soumis, cela comporte l'indentation, la mise en forme complète, les lignes cassées ou superflues. Tout doit être impeccable ! A gauche, c'est cassé, il a fallu passer le code en revue, et à droite, la totalité est valide, merge request, pipeline, test, prêt à être poussé sur la branche alpha !



Capture d’écran lié au projet fil rouge :

Ci-contre : Le début du mode responsif du site de la Nimes Alerie, avec une barre dépliée permettant de retrouver la barre de navigation en mode mobile.

Ici nous avons l’API, que Be-ys appelle aussi le « swagger ». Cette capture est faite pendant sa génération.



La toute premier navbar de la Nimes’Alerie !

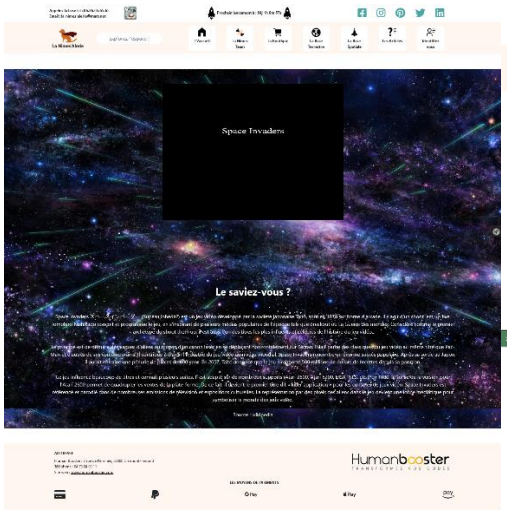
Juste en dessous : la version finale de cette barre de navigation !

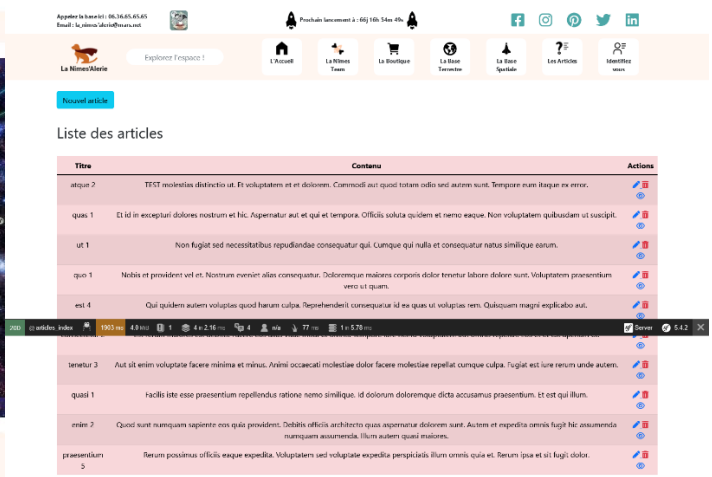
Page d'accueil

Ici à gauche c’est la page bonus qui est cachée dans la barre de navigation, elle représente un bon échantillon de site avec, barre de navigation, contenu, et footer.

A droite, la page d’article qui est mal prise (la barre d’interface symfony la tranche en deux), avec un tableau de contenu que vous pourrez lire, modifier, effacer ou ajouter.

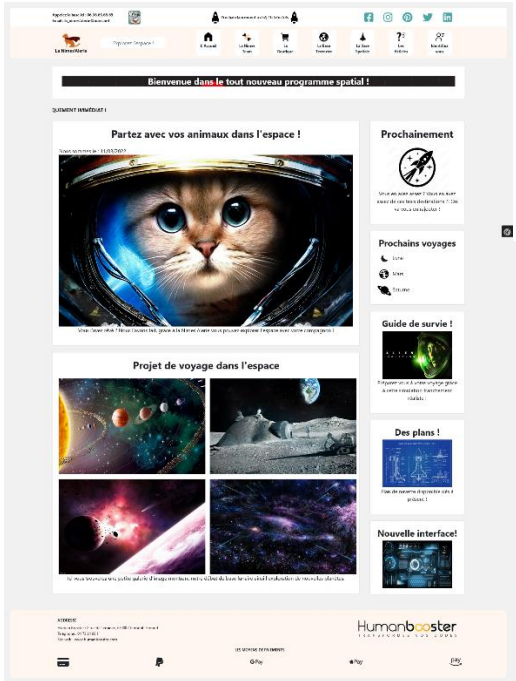
Ces articles ont été généré à l’aides de fixtures.

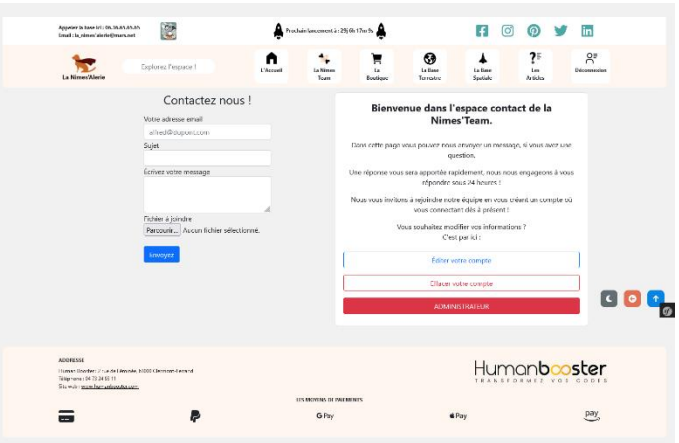




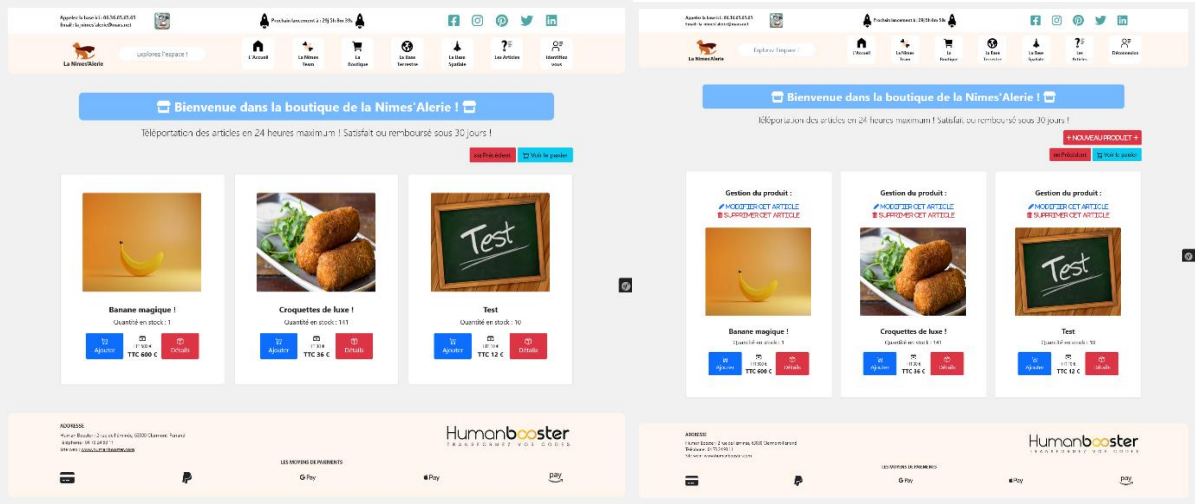
Ci-dessous nous avons à gauche la construction de la page de présentation sur le projet spatial, avec une petite galerie d'image. Cette page est en html statique, très classique.

A droite nous avons la page de contact avec son formulaire. Un petit texte d'explication de la page pour l'utilisateur, et de plus il pourra ici éditer ses informations, supprimer son compte. Notez que le bouton « ADMINISTRATEUR » n'est visible que si vous avez le rôle défini, sinon il n'apparaitra pas pour le simple utilisateur.

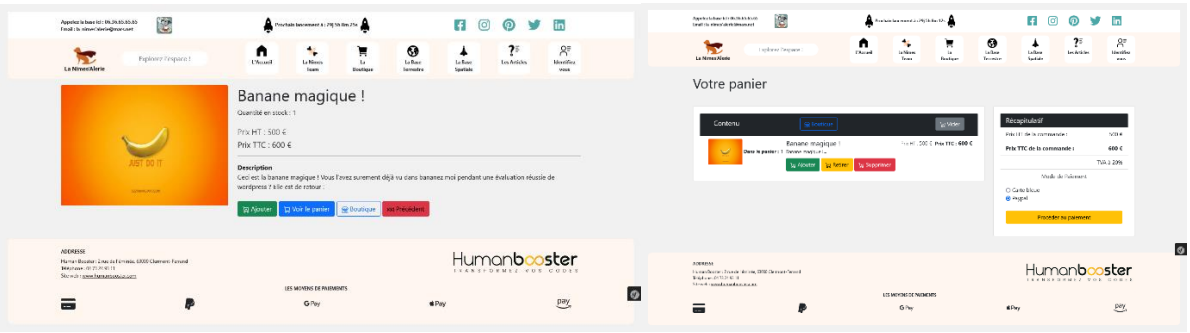




Le plus important évidemment, la site boutique, son panier avec mode de paiement ! Ici à gauche nous avons la page des produits, en mode ‘lecture’ simple, à droite, la même page avec les boutons supplémentaires pour l’administrateur :



Maintenant la page de détail d’un article à gauche, avec le panier à droite avec ses actions associées !



Ces illustrations sont là pour montrer l’évolution et la conception d’un site de e-commerce. En partant de zéro (from scratch) jusqu’à la version finale ! Bien évidemment je n’ai pas tout montré, sinon ça prendrait plus de pages que cela, mais une présentation de site e-commerce sans montrer la boutique, ce n’est pas vraiment une présentation e-commerce ! Comme dit plus haut, la démo live de ce site montrera toutes les possibilités !

Ceci clôture mon dossier projet pour la formation de développeur web & web mobile, en espérant que l’alternance se réalise, merci de m’avoir lu.