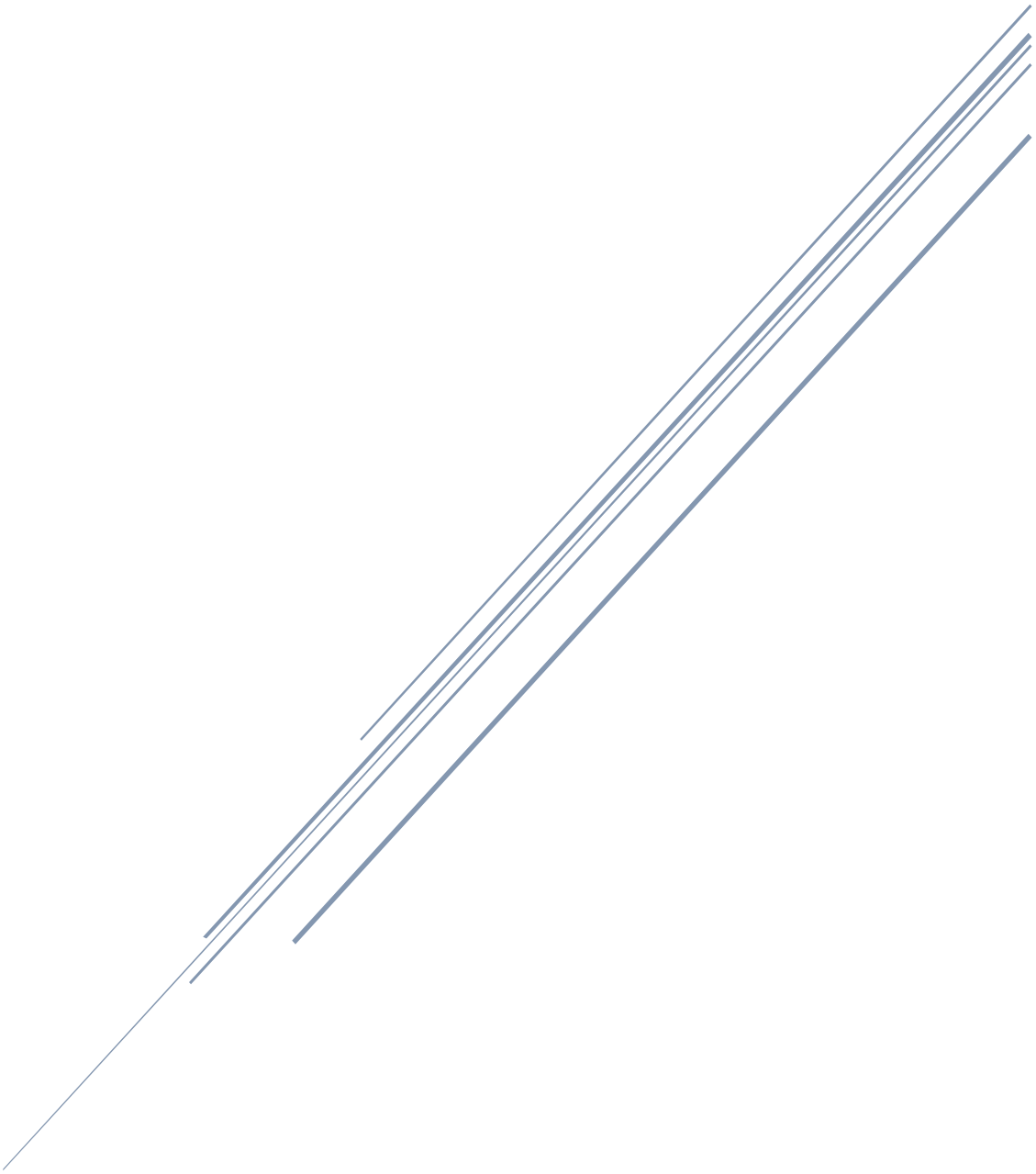
Formation DWWM

Dossier de projet professionnel



DEREDEC Xavier

20/07/2021

Table des matières

Remerciement

Je remercie tout d’abord Mme Landry Séverine pour m’avoir accueilli lors de ma période de stage et m’avoir confié ce projet ainsi que Mme Divo Charlotte pour avoir réalisée la maquette graphique du site vitrine et pour finir Pastore De Vardo Sacha pour m’avoir aidé à solutionner le problème d’upload d’image sur la partie back-office du projet.

1- Présentation

A- Liste des compétences couvertes par le projet

Front-End :

Maquetter une application

Réaliser une interface utilisateur web statique adaptables

Développer une interface utilisateur dynamique

Back-End :

Créer une base de données

Développer les composants d'accès aux données

Développer la partie back-end d'une application web ou web mobile

Elaborer et mettre en œuvre des composant dans une application de gestion de contenues

B- Résumé du projet

Mon projet a été réalisé pour l'entreprise « SASU Auto-Ecole de Pierrefeu » dans l'objectif d'acquérir le label qualité par l'état et afin de se positionner en concurrence des auto-écoles en ligne. L'une des conditions requises pour obtenir ce label est de posséder un site internet contenant les tarifs, la liste des enseignants, les méthodes utilisées ainsi que le déroulement de chaque examen. Les visiteurs doivent aussi pouvoir laisser un avis sur le site internet ou encore contacter l'entreprise afin d'obtenir des renseignements complémentaires sur les taux de réussite et le délai moyen d'obtention du permis. De plus, il m'a également été demandé de développer une interface permettant de gérer les éléments amenés à changer régulièrement comme les tarifs, la liste des forfaits disponibles ou encore les informations relatives aux moniteurs.

Mon projet est donc constitué d'une partie site vitrine et d'une partie back-office permettant la gestion du contenu dynamique.

Pour la partie site vitrine, j'ai décidé d'utiliser le framework Angular couplé au préprocesseur SASS concernant le front-end, et PHP couplé à MySQL pour la partie back-end.

Pour la partie back-office j'ai également utilisé le framework Angular couplé au préprocesseur SASS concernant le front-end, et PHP couplé à MySQL pour la partie back-end, tout en incluant l'API Electron.JS pour le déporter sous forme d'application de bureau.

2- Cahier des charges

A-Expression des besoins

Pour que la « SASU Auto-Ecole de Pierrefeu » soit labellisée par l'État nous avons besoin d'une application web présentant l'établissement et contenant toutes les informations requises dans la charte de labellisation fournie en annexe. La charte graphique devra s'orienter autour du logo de son entreprise afin que l'identité graphique de l'entreprise se reflète sur son site web. Pour la disposition des éléments ainsi que les technologies employées, nous avons carte blanche. Il nous faudra également une interface permettant de pouvoir modifier les éléments dynamiques gérés par la base de données plus simplement.

B-Charte graphique

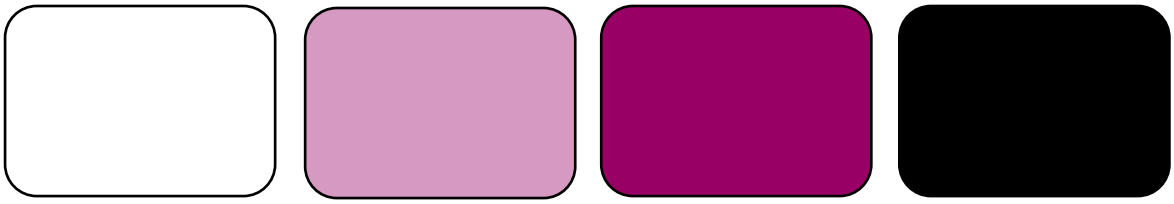
1- Logo

2- Polices d'écriture

Pour les titres, la police choisie est **Advent Pro**

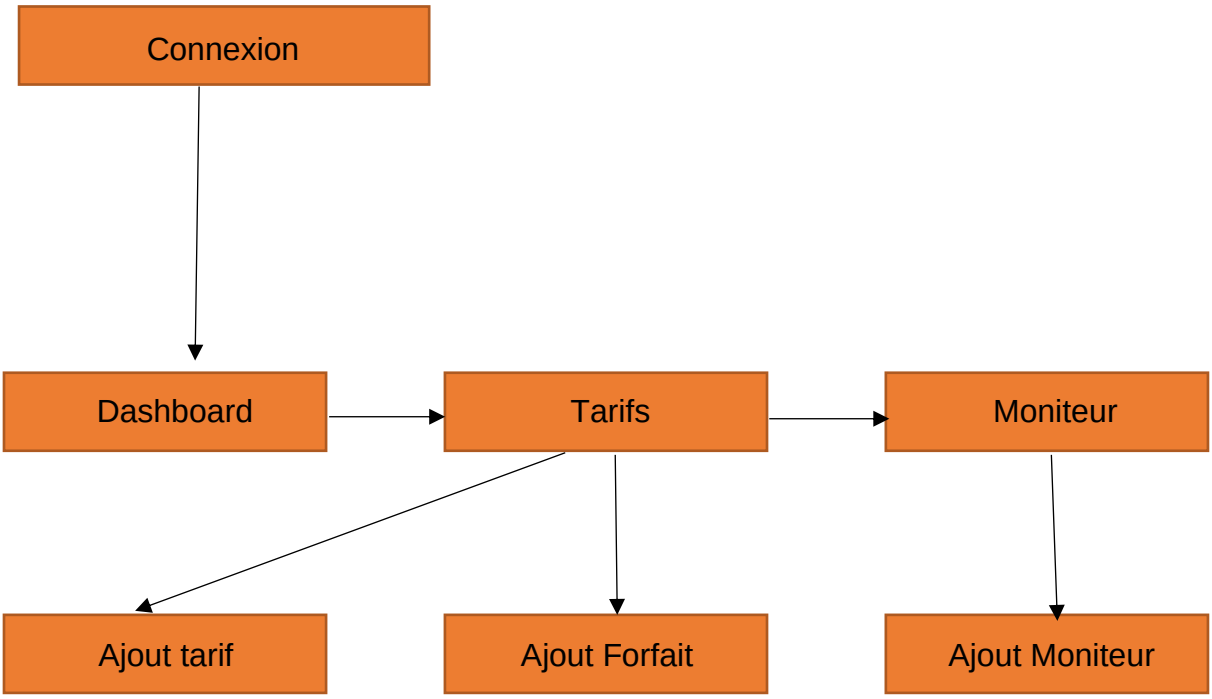
Pour les corps de texte, la police choisie est **Lato**

3- Palette de couleur



#FFFFFF **#D699C2** **#990066** **#000000**

4- Cheminement utilisateur du back-office



3- Spécifications techniques

A- Base de données

Pour la partie base de données j'ai commencé par effectuer une liste des éléments pouvant être amenés à changer régulièrement comme les tarifs, les forfaits ainsi que leurs contenus ou encore la liste des enseignants et de leurs informations.

Suite à cela j'ai réfléchi à quels types de données ils pourraient être associer afin de rédiger le dictionnaire de donnée.

De ce dictionnaire en a découler le modèle conceptuel de donnée avec toutes les table, les colonnes et les contraintes de clef primaire, d'index et de clef étrangère (voir annexe 1).

B- Technologies utilisées

1- Maquettage

J'ai utilisé le logiciel Figma pour toute la partie maquettage de ce projet.

2- Gestion de versions

Afin d'effectuer un suivi des versions de mon j'ai décidé d'utiliser git au travers de l'outil GitHub Desktop.

3- Développement Front-end

Pour la partie front-end de ce projet j'ai décidé d'utiliser Angular couplé avec le préprocesseur SASS ainsi la librairie HammerJS. L'API ElectronJS a aussi été utilisée pour la partie backoffice afin de la déporter sous forme d'application de bureau et d'éviter d'éventuels essais d'intrusion sur le système de gestion du contenu.

4- Développement Back-end

Pour la partie back-end j'ai utilisé le langage PHP couplé à MySQL pour la gestion de la base de données.

C- User Stories

Pour la partie user story de ce projet j'ai commencé par lister toutes les contraintes de fonctionnalité listées dans la charte de labélisation des auto-écoles.

J'ai ensuite ajouté les besoins supplémentaires de l'entreprise et trié chaque fonctionnalité selon leur importance, leurs difficultés et la business value.

Pour finir je les ai consignés dans un tableau trié en 2 volets, Site-Vitrine et Back-Office, afin de pouvoir consulter à chaque étape si toutes les fonctionnalités nécessaires ont bien été ajoutées (voir annexe 2).

D- Maquettage et Wireframe

Pour le site vitrine je n'ai effectué que le wireframe car la partie UI Design a été confiée à une graphiste indépendante nommée **Charlotte Divo.** Pour la partie backoffice je n'ai pas eu à tenir compte des formats mobile et tablette étant donné qu'il va se présenter sous la forme d'une application de bureau, qui ne pourra être exécutée que sur ordinateur (voir annexe 3).

1- Mobile

Accueil

Pour la page d'accueil, j'ai décidé de faire apparaître les éléments les plus significatifs et vendeurs de l'entreprise et de favoriser les carrousels pour l'affichage du contenu dynamique afin d'obtenir une meilleure ergonomie (voir annexe 3).

Permis

Pour la page permis, j'ai décidé d'essayer de limiter la zone d'affichage à la taille d'une vue d'écran afin que les utilisateurs aient le contenu principal dans leur champ de vue initial (voir annexe 3).

Tarifs

Pour la page des tarifs, je suis parti du même postula que pour la page permis tout en me laissant une petite marge pour le téléchargement des tarifs détaillés (voir annexe 3).

Contact

Pour la page de contact j'ai aussi voulu limiter la taille en fonction d'une zone de vue étant donné le peu de contenu qu'elle contient (voir annexe 3).

2- Desktop

Pour le wireframe desktop la seule différence notable pour le site vitrine est la disposition des éléments sur la pages d'accueil (voir annexe 3).

Pour le back office, j'ai décidé de partir sur un système d'interface simple et épurée basé sur des onglets aux noms compréhensibles facilement afin que l'utilisateur puisse gérer son contenu de manière fluide et intuitive.

4- Réalisation du Projet

A- Site vitrine

1- Génération de PDF

Dans le component **prices** de mon site vitrine j'ai inclus un lien permettant de consulter la grille tarifaire détaillée de l'entreprise en version PDF.

Pour ce faire j'ai utilisé les fonctions suivantes :

getPdf()

```
getPdf(){
  this.getService.getAllDetail().subscribe((datas: any[]) => {
    if(datas){
      for(let data of datas) {
        this.detailList.push(
         [data.Nom_tarif, data.Nom_type,
  String(data.Prix_tarif)+"€ TTC"])
      }
      this.pdfService.createPdf(this.detailList);
    }
  });
```

Quand l'utilisateur clique sur le lien, la fonction **getPdf()** s'exécute de la manière suivante :

- Elle souscrit à l'observable retourné par la fonction **getAllDetail()** appartenant au service **get** qui retourne la liste des tarifs détaillés

- Elle stocke chaque ligne du tableau de tarif dans le tableau **detailList** sous la forme :

[**nom_Tarifs, nom_Types, Prix_tarif + € TTC**]

- Elle appelle la fonction **createPdf()** du service **pdfmake** et lui passe en paramètre le tableau **detailList**

getAllDetail()

```
getAllDetail(): Observable<ITarif[]> {
    return this.http.get<ITarif[]>(`http://autoeci.cluster030.hostin
g.ovh.net/script/getDetail.php`);
  }
```

La fonction **getAllDetail()** s'exécute de la manière suivante :

- Elle retourne sous la forme d'un observable le résultat de la requête contenue dans le fichier **getDetail.php**

createPdf()

```
  createPdf(data: any[]) {
    var docDefintition: TDocumentDefinitions = {
      header: {
        columns:[
          {
            text:'Nos Tarifs Detaillées',
            style: 'title',
            alignment: 'center'
          }
        ]
      },
      content: [
        {
          margin: [65,20, 0, 0 ],
          table: {
            headerRows: 1,
            body: data,
          }

        }
      ],



      styles: {
        title: {
          fontSize: 22,
          bold: true,
          color: '#900066'
        },
        header: {
          fontSize: 15,
          bold: true,
```

```
          color: '#000000'
        }
      }
    };
    pdfMake.createPdf(docDefintition).open();
  }
```

La fonction **createPdf()** s'exécute de la manière suivante :

-Elle met en forme le contenu du pdf via la variable **docDefinition** et y inclu le contenu du paramètre **data**

-Elle fait appel à la fonction **createPdf()** de la librairie **PdfMake** en lui donnant comme paramètre la variable **docDefinition**

-Elle exécute de manière asynchrone la fonction **open()** de la librairie **PdfMake** afin d'ouvrir le pdf généré juste au-dessus

2- Envoi du formulaire de contact

Pour transmettre les informations du formulaire de contact par mail j'ai mis en place plusieurs contrôles coté front-end et back end qui sont :

Interface IContact

```
interface IContact {
  nom: string,
  prenom: string,
  sujet: string,
  mail: string,
  message: string
}
```

Afin de faire la passerelle entre mon **reactive formgroup** et mon **PHP,** j'ai décidé d'implémenter une interface pour éviter les problèmes liés au typage.

initform()

```
initForm(){
    this.contactForm = this.formBuilder.group({
      nom: ['', [Validators.required, Validators.pattern(/[0-9a-zA-
Z]/), Validators.maxLength(100)]],
      prenom: ['', [Validators.required, Validators.pattern(/[0-9a-
zA-Z]/), Validators.maxLength(100)]],
      sujet: ['', [Validators.required, Validators.pattern(/[0-9a-
zA-Z]/), Validators.maxLength(100)]],
      mail: ['', [Validators.required, Validators.email, Validators.
maxLength(100)]],
      message: ['', [Validators.required, Validators.pattern(/[0-9a-
zA-Z]/), Validators.maxLength(255)]]
    });
  }
```

Afin de contrôler au mieux les saisies utilisateur coté front-end, j'ai utilisé la technologie **reactive formgroup d'angular** pour effectuer une vérification de premier niveau sur mon formulaire et lever des exceptions ainsi qu'afficher des retours d'erreur à l'utilisateur de manière dynamique.

onSubmit()

```
  onSubmit(){
    if(this.contactForm.valid){
      let infoContact: IContact = {
        nom: this.contactForm.get(['nom'])?.value,
        prenom: this.contactForm.get(['prenom'])?.value,
        sujet: this.contactForm.get(['sujet'])?.value,
        mail: this.contactForm.get(['mail'])?.value,
        message: this.contactForm.get(['message'])?.value
      }
      this.httpClient.post('http://autoeci.cluster030.hosting.ovh.ne
t/script/contact.php', JSON.stringify(infoContact)).subscribe(
        (response: any) => {
          response['success'] ? alert("message envoyé !") : alert("o
ups une erreur est survenue, veuillez reesayer !");
        },
        (error) => console.log(error)
      );
    }
    else{
      alert("veuillez remplir tout les champs correctement !")
    }
  }
```

Afin de contrôler la validité des saisies utilisateur au moment de l'envoi, j'ai implémenté une deuxième vérification qui permet d'envoyer le formulaire uniquement si tous les **validators** sont respectés.

regex_data()

```php
function regex_data($param)
{
    $param = trim($param);
    $param = stripcslashes($param);
    $param = htmlspecialchars($param);
    return $param;
}
```

Afin de contrôler les saisies utilisateurs coté back-end, j'utilise la fonction **regex_data()** qui permet d'échapper les espaces, supprimer les antislash et de convertir les caractères spéciaux en entités HTML pour éviter les injections de code via mon formulaire de contact.

3- Responsive design

Pour la partie responsive du site vitrine j'ai intégré différents breakpoint/media queries au sein de mon **SASS** afin d'adapter la disposition, la taille et la forme de mes différents éléments au format de l'appareil de l'utilisateur.

B- Back Office

1- Ajouter un forfait

Dans mon component **add-pack,** j'ai créé un input de type **submit** qui à l'activation exécute les fonctions suivantes :

publish()

```
publish(){
    const newPack: IPack = {
      nom: this.addPackForm.get('nom')?.value,
      prix: this.addPackForm.get('price')?.value,
      description: this.addPackForm.get('description')?.value,
      type: this.typeId
    }
    this.addService.addPack(newPack);
  }
```

La fonction **publish()** permet d'ajouter un nouveau forfait en effectuant plusieurs actions qui sont :

- Création d'un objet **newPack** de type **IPack** contenant toutes les informations concernant le forfait à ajouter

-Appel de la fonction **addPack()** du service **add** qui prend comme paramètre l'objet créé ci-dessus

addPack()

```
  addPack(newPack: IPack) {
    let packData = JSON.stringify(newPack);
    this.http.post('http://autoeci.cluster030.hosting.ovh.net/script
/addPack.php', packData).subscribe(
      (response: any) => {
        response['success'] ? this.router.navigate(['/dashboard']) :
 console.log('erreur');
      },
      (error) => console.log(error)
    );
```

La fonction **addPack()** permet de transmettre les informations fournies en paramètres à un fichier **PHP** afin de créer un nouveau forfait dans la base de données en suivant ces étapes :

-Création d'une variable **packData** contenant l'objet passé en paramètre converti au format **JSON**

-Appel de la fonction **post()** du module **HTTPClient** d'**Angular**, qui permet de transmettre le contenu de la variable créée ci-dessus au fichier **addPack.php**

-Traitement de la réponse retournée par la fonction citée ci-dessus afin de lever des exceptions en cas d'erreur ou de rediriger sur le component **dashboard** en cas de succès

2- Connexion au back office

Afin de pouvoir sécuriser l'accès aux fonctionnalités du back office mon application fait appel aux fonctions et services suivants :

Service AuthGuard

```
export class AuthGuardService {

  constructor(private authService: AuthService, private router: Rout
er) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnaps
hot): Observable<boolean> | Promise<boolean> | boolean {
    if (this.authService.userGuard != null) {
      return true;
    } else {
      return this.router.navigate(['/auth']);
    }
  }
}
```

Ce service permet de bloquer l'accès aux fonctionnalitées à tout utilisateur non-authentifié en effectuant les actions suivantes :

-Si la valeur de la variable **userGuard** du service **auth** est différente de null elle autorise l'accès

-Sinon elle redirige l'utilisateur sur le component **auth**

Service Auth

```
export class AuthService {

  userGuard: any = sessionStorage.getItem('id');
  userGuardSubject = new Subject<any>();

  constructor(private http: HttpClient, private router: Router) { }

  emituserGuard(){
    this.userGuardSubject.next(this.userGuard);
  }

  signIn(user: IUser):boolean{
    let check: boolean = false;
    let userData = JSON.stringify(user);
    this.http.post('http://autoeci.cluster030.hosting.ovh.net/script
/auth.php', userData).subscribe(
      (response: any) => {
        if(response['success']) {
          sessionStorage.setItem('id', response['id']);
          this.userGuard = sessionStorage.getItem('id');
          this.emituserGuard();
          this.router.navigate(['/dashboard']);
          check = true;
        }
        else{
          check = false;
        }
      },
      (error) => console.log(error)
    )
    return check;
  }

  singOut(){
    sessionStorage.removeItem('id');
    this.userGuard = null;
    this.emituserGuard();
    this.router.navigate(['/auth'])
  }
}
```

Dans mon service **auth,** il y a deux fonctions capitales :

signIn()

-Elle stocke dans une variable **userData** le contenu du paramètre qui lui est passé, converti au format **JSON**

-Elle utilise la fonction **post()** du module **HTTPClient** d'**Angular** afin de transmettre la variable précédemment créée au fichier **auth.php**

-Elle traite la réponse envoyée par le fichier **auth.php** :

-En cas de succès, elle crée un token **JWT** qui est stocké en session storage, met à jour la variable **userGuard** et son **subject** puis redirige l'utilisateur vers le component **dashboard**

-En cas d'échec sur les identifiants elle retourne la valeur false

-En cas d'erreur php elle retourne l'erreur php

SignOut()

-Elle supprime le token **JWT** du session storage

-Elle remet la variable **userGuard** à null et met à jour son **subject**

-Elle redirige l'utilisateur sur le component **auth**

Component auth

Au sein de mon component **auth** les contrôles ont été effectués grâce aux fonctionnalités suivantes :

-Un **reactive formGroup** qui me permet de contrôler les saisies utilisateur de manière immédiate afin de faire un retour graphique directement à l'utilisateur

-Une interface **IUser** qui permet d'éviter les problèmes de typage

auth.php

Du coté **PHP,** plusieurs éléments ont été mis en place afin de sécuriser et contrôler les données d'authentification :

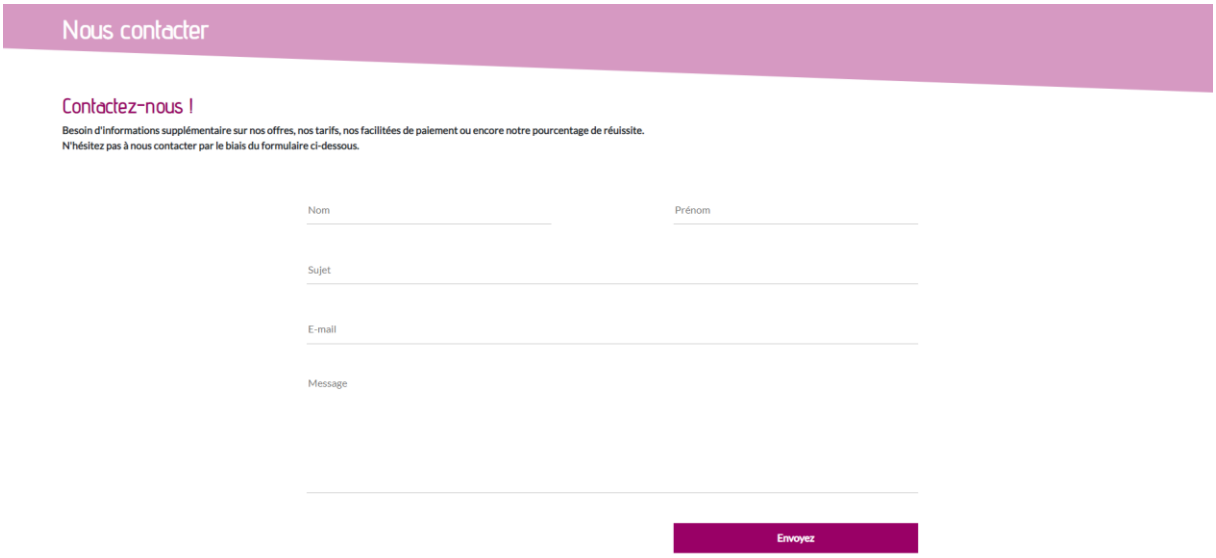
-La fonction **regex_data()** expliquée plus haut pour éviter les injections de code via le formulaire de connexion.

-La fonction **password_verify** de **PHP** afin de vérifier la correspondance entre le mot de passe saisi et le mot de passe hash dans la base de données.

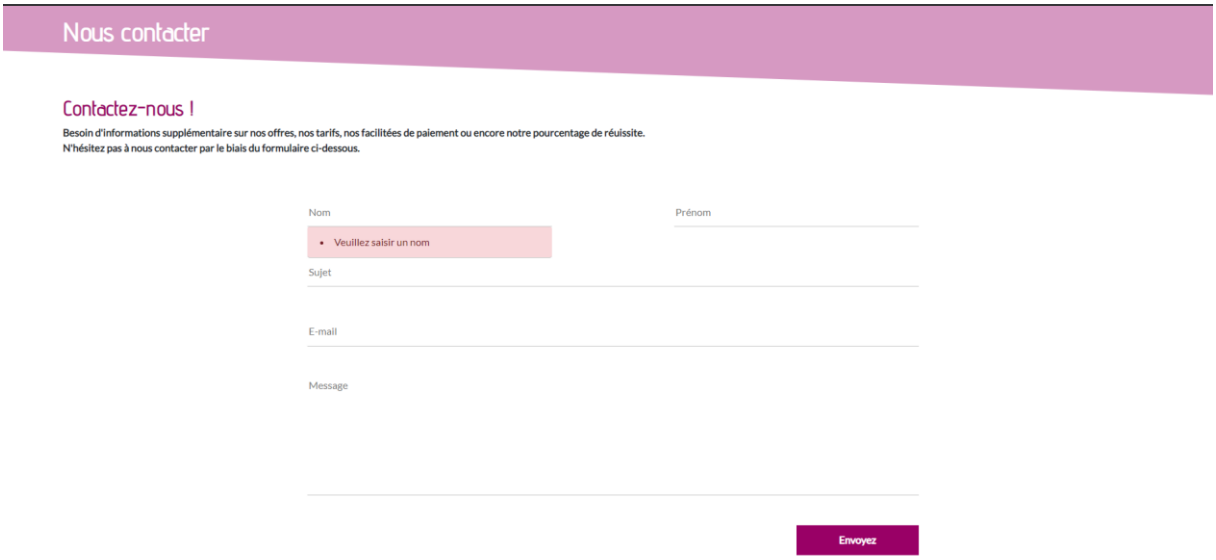
5- Essais end to end

A- Site vitrine

Sur le site vitrine la page contact se présente sous cette forme.



Si une erreur de saisie est constatée, elle est retournée sous cette forme.



Si une erreur s’est produite lors de la requête, la page nous retourne ceci.

B- Back office

1- Formulaire de connexion

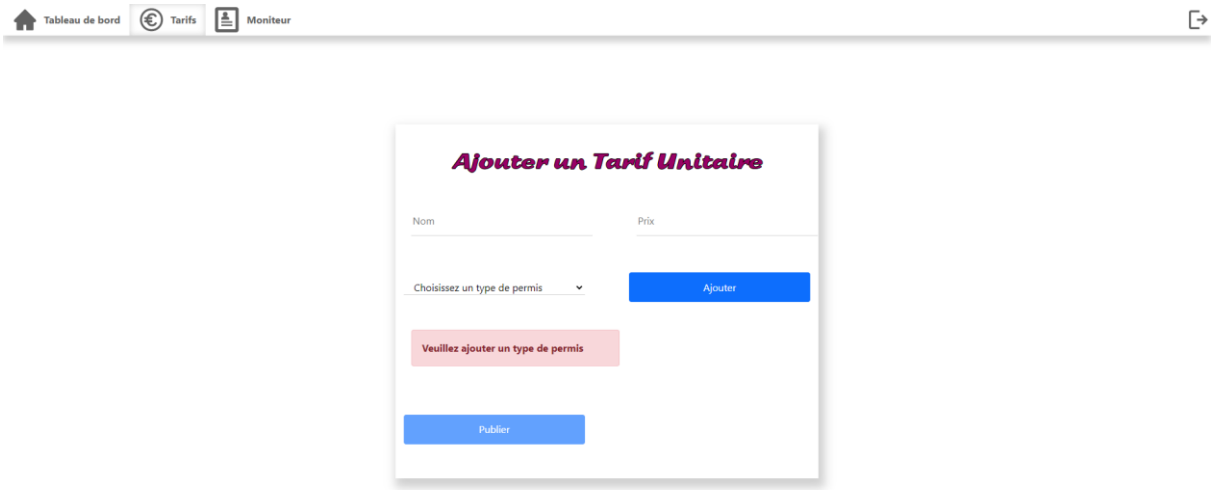
Le formulaire de connexion au back office se présente sous cette forme.

Tant que le formgroup n’a pas le statut **valid** le bouton connexion a l’attribut **disabled.**

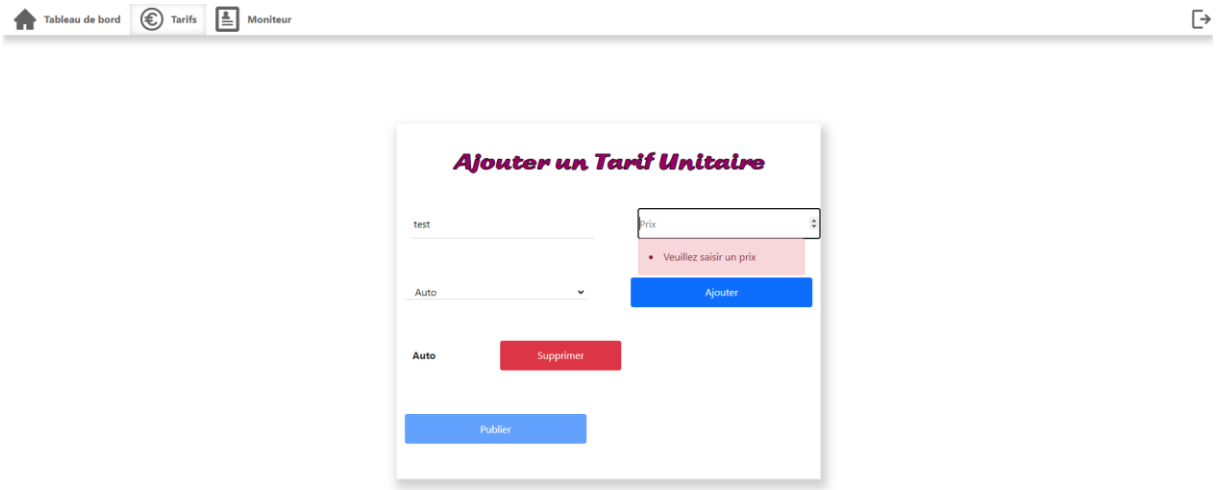
Si l'utilisateur se trompe sur l'identifiant ou le mot de passe la page lui retourne l'erreur ci-dessus.

2- Ajout d'un tarif détaillé

Le formulaire d'ajout d'un tarif détaillé se présente comme ceci.



Tant que le formgroup n'a pas le statut **valid** le bouton connexion a l'attribut **disabled** et à l'arrivée sur la page une erreur s'affiche afin d'informer l'utilisateur qu'il doit obligatoirement ajouter un type de permis.



Si l'utilisateur ne respecte pas l'une des conditions des **validators** d'**Angular** une erreur de ce type et de cette forme s'affiche à l'écran.

6- Veille technologique concernant la sécurité

Au cours de mon projet, j'ai effectué une veille technologique complète sur l'ensemble des failles de sécurités auxquelles mon application pouvait être exposée.

Afin d'optimiser au mieux ma veille je l'ai séparée en 2 partie site vitrine et back office.

A- Site vitrine

Pendant ma veille technologique pour le site vitrine j'ai remarqué un point particulièrement exposé aux failles :

1- Le formulaire de contact

Les failles d'un formulaire sur une application web se situent au niveau de la saisie de données utilisateur qui peut mener à différent cas de figure en matière de sécurité :

Les failles XSS (Cross-site scripting)

Le site wikipedia définit une faille XSS comme un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, provoquant ainsi des actions sur les navigateurs web visitant la page.

Dans le cas de notre formulaire de contact ce type de faille pourrait être exploité en injectant du code **javascript** qui s'exécute en passant un paramètre dans l'URL via la méthode **GET** pouvant mener à une redirection sur un site malveillant afin de collecter des informations concernant l'administrateur ou les utilisateurs ou encore de propager un logiciel malveillant aux utilisateurs, l'affichage de données contenues dans notre **SGBDR** via l'exécution de script comme des fonction **Javascript** ou encore la récupération de données stockées en sessions ou en cookies.

Pour s'en prémunir il faut procéder à des contrôles de saisies, éviter au maximum d'utiliser la méthode **GET,** effectuer une configuration des autorisations d'exécution de script via les **CSP (Content Sécurity Policy)**, retraiter systématiquement le code **HTML** produit par l'application avant de l'envoyer au navigateur.

Sources:

-Wikipedia.fr

-Leblogduhacker.fr

-Root-me.org

-Cnil.fr

Le failles CSRF (Cross Site Request Forgery)

D’après le site **HTTPCS** ce type de faille consiste à faire exécuter à un utilisateur une requête http à son insu afin qu’il exécute des actions en utilisant ses privilèges.

Dans le cas de notre formulaire de contact, ce type de faille pourrait être exploité en injectant du code javascript dans une balise **img** pouvant mener à une redirection sur un site malveillant permettant ainsi de récupérer des informations sensibles contenues dans la base de données ou encore des informations concernant l’administrateur tel que ses identifiants de connexion au back-office ou encore à divers sites ou services tiers.

Pour s’en prémunir il faut procéder à des contrôles de saisies et empêcher la saisie de balise html ou encore de langage comme **javascript,** mettre en place un système d’authentification par jeton tel que le **JWT** et modifier dynamiquement les liens permettant d’effectuer une action qui requière des privilèges de haut niveau.

Sources:

-Httpcs.com

-Leblogduhacker.fr

-Root-me.org

-Cnil.fr

B- Back-Office

Durant ma veille technologique sur la sécurité sur le back-office une question s'est très vite posée :

Comment dissimuler au mieux le formulaire de connexion sur un site ne possédant pas d'espace membre ?

Plusieurs solutions me sont alors venues en tête :

- Créer un sous domaine permettant un accès au back-office grâce à un lien spécifique connu uniquement de l'administrateur.
- Déporter le back-office et son interface de connexion sous forme d'application de bureau grâce à une **webview** embarquée.

Après m'être penché sur les deux solutions une différence notable s'est révélée :

Même en dissimulant le formulaire de connexion dans un sous domaine, l'url peut être trouvée très aisément.

J'ai donc décidé de mettre en œuvre la seconde solution qui me permet donc de ne conserver qu'une et unique faille majeure :

1- Le facteur humain

En termes de sécurité la faille la plus difficile à couvrir reste le facteur humain qui couvre un éventail de problèmes de sécurité assez important.

Voici quelques exemples de problèmes soulevés par cette faille :

- La divulgation des identifiants permettant l'accès à une application avec un haut niveau de privilège par un utilisateur ayant des privilèges élevés.
- Une mauvaise gestion des mots de passe dans la durée pouvant mener à un mot de passe général vieux de plusieurs années.
- Une salle de serveur non sécurisée entrainant alors un risque d'attaque physique des machines hôtes.
- Un pare-feu ou un système d'exploitation non sécurisé permettant de prendre le contrôle d'une machine hôte à distance et ainsi de récupérer ou de modifier plusieurs paramètres ou chemins d'accès à une application.

Les exemples ci-dessous ne couvrent pas la totalité des problèmes liés au facteur humain en termes de sécurité.

Malgré une évolution toujours plus rapide des moyens de sécurisation d'une application, ce facteur reste immuable et pose toujours un énorme problème notamment en termes de confidentialité des données.

Dans le cadre de mon back-office la solution de déporter celui-ci m'a permis à la fois de couvrir une grande partie des failles courantes mais aussi de limiter la capacité d'accès à cette application à un seul poste sous couvert de la non diffusion de l'application.

Seulement un gros problème persiste :

Si une tierce personne arrive à avoir accès à ce logiciel, comment contrer ses tentatives d'attaque ?

Pour répondre à cette question j'ai mis en place des moyens plus conventionnels de sécurisation sur celui-ci comme :

- Un système d'authentification basé sur un token **JWT** stocké en session de la **webview**
- Le contrôle de chaque saisie entrée via l'application afin d'éviter les failles **XSS**
- Un blocage des **devtools** de la **webview** dans la configuration de l'api **ElectronJs** en mode production
- La dissimulation de la barre d'adresse dans la **webview** en mode production

Sources:

-Httpcs.com

-Leblogduhacker.fr

-Root-me.org

-Cnil.fr

-Ssi.gouv.fr

-Wikipedia.fr

-Cisco.com

-Medium.com

-Stackoverflow.com

7- Veille technologique anglophone

Durant la réalisation de mon projet un problème d'envergure est apparu.

Par soucis de confort et de compétences j'avais pris la décision d'utiliser le framework **Angular** concernant la partie **front-end** de mon back-office.

J'ai aussi pris la décision d'utiliser l'api **ElectonJS** afin de déporter cette application web sous forme d'application de bureau pouvant s'exécuter sur le système d'exploitation de l'entreprise qui m'avait missionné.

Cependant une question s'est posée :

Comment coupler le framework Angular à l'api ElectronJS ?

Ma veille pour répondre à cette interrogation a alors commencé et après plusieurs recherches je suis tombé sur un site anglophone du nom de **sitepoint.com** sur lequel un article avait été rédigé par un certain **Ahmed Boucherfa** le 28 mai 2019.

Dans l'introduction de cet article, on pouvait comprendre qu'il s'agissait d'un tutoriel permettant de comprendre comment compiler une application **Angular** avec l'api **ElectronJS**.

A- Extrait d'article anglophone

In this tutorial we'll build a cross-platform desktop application with Electron and web technologies such as TypeScript and Angular.

Electron.js is a popular platform for building cross-platform desktop apps for Windows, Linux and macOS with JavaScript, HTML, and CSS. It's created and maintained by GitHub and it's available under the MIT permissive license. It was initially created for GitHub's Atom editor, but has since been used to create applications by companies like Microsoft (Visual Studio Code), Facebook, Slack, and Docker.

Electron makes use of powerful platforms like Google Chromium and Node.js, but also provides its own set of rich APIs for interacting with the underlying operating system.

Electron provides a native container that wraps web apps so they look and feel like desktop apps with access to operating system features (similar to Cordova for mobile apps). This means we can use any JavaScript library or framework to build our application. In this tutorial, we'll be using Angular.

B- Traduction de l'article

Dans ce tutoriel nous allons compiler une application de bureau multi-plateformes en utilisant Electron et des technologies web comme TypeScript et Angular.

Electron.Js est une api populaire permettant de compiler des applications de bureau multi-plateformes pour Windows,Linux et macOS en utilisant les langages Javascript, HTML et CSS.

Elle a été créée et est maintenue via GitHub. Elle est rendue disponible sous la licence MIT. Elle a été initialement créée pour la plateforme Github Atom editor mais a fini par être utilisée par des entreprise comme Microsoft pour des applications comme Visual Studio Code, Facebook, Slack et Docker.

Electron utilise des librairies performantes comme Google Chromium et Node.js pour fonctionner mais fournis aussi son propre lot d'api afin d'interagir avec les systèmes d'exploitation cités ci-dessus.

Electron fournis un conteneur natif permettant d'embarquer des applications web, elle ressemble donc à des applications de bureau qui peuvent accéder aux fonctionnalités des systèmes d'exploitation comme Cordova sur application mobile.

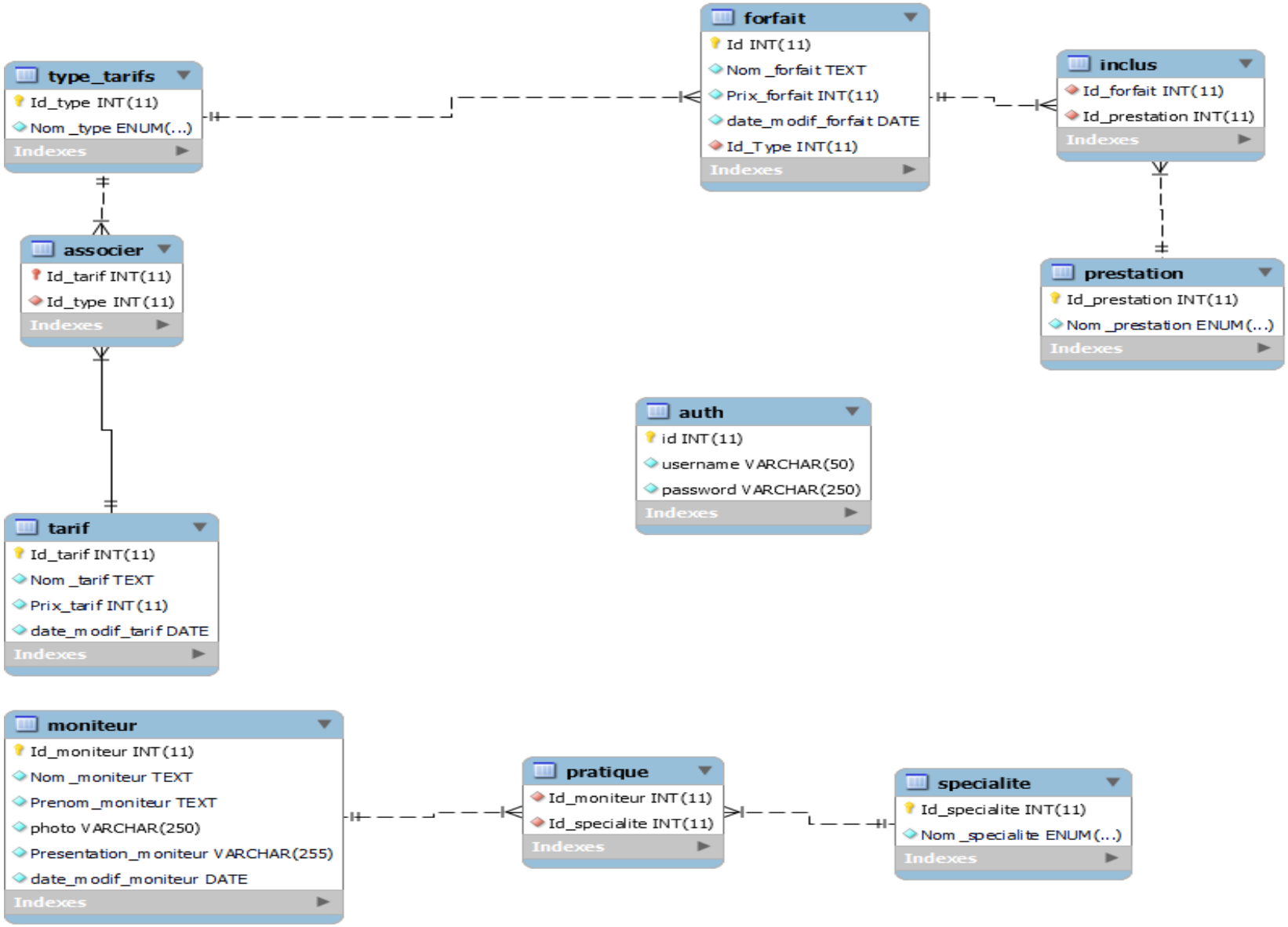
Cela veut dire que l'on peut utiliser n'importe quelle librairie JavaScript ou framework dans notre application. Dans ce tutoriel nous allons utiliser Angular.

8- Axes d'amélioration

Le projet n'étant pas finalisé voici une liste succincte des axes d'amélioration futur que j'apporterai à ce projet :

- Ajout d'un module de pagination sur les différents tableaux présent sur le back-office.
- Ajout d'un component permettant de modifier la liste des prestations et des types de permis.
- Ajout de la fonction de tri des différents tableaux du back-office.
- Ajout d'une barre de navigation secondaire afin de gérer les tarifs des différentes auto-écoles.
- Migration des applications sous VueJS pour une meilleure optimisation des ressources.

Annexe 1


forfait
Id INT(11)
Nom _forfait TEXT
Prix_forfait INT(11)
date_modif_forfait DATE
Id_Type INT(11)
Indexes
type_tarifs
Id_type INT(11)
Nom _type ENUM(...)
Indexes
inclus
Id_forfait INT(11)
Id_prestation INT(11)
Indexes
associer
Id_tarif INT(11)
Id_type INT(11)
Indexes
prestation
Id_prestation INT(11)
Nom _prestation ENUM(...)
Indexes
auth
id INT(11)
username VARCHAR(50)
password VARCHAR(250)
Indexes
tarif
Id_tarif INT(11)
Nom _tarif TEXT
Prix_tarif INT(11)
date_modif_tarif DATE
Indexes
moniteur
Id_moniteur INT(11)
Nom _moniteur TEXT
Prenom_moniteur TEXT
photo VARCHAR(250)
Presentation_moniteur VARCHAR(255)
date_modif_moniteur DATE
Indexes
pratique
Id_moniteur INT(11)
Id_specialite INT(11)
Indexes
specialite
Id_specialite INT(11)
Nom _specialite ENUM(...)
Indexes

Annexe 2

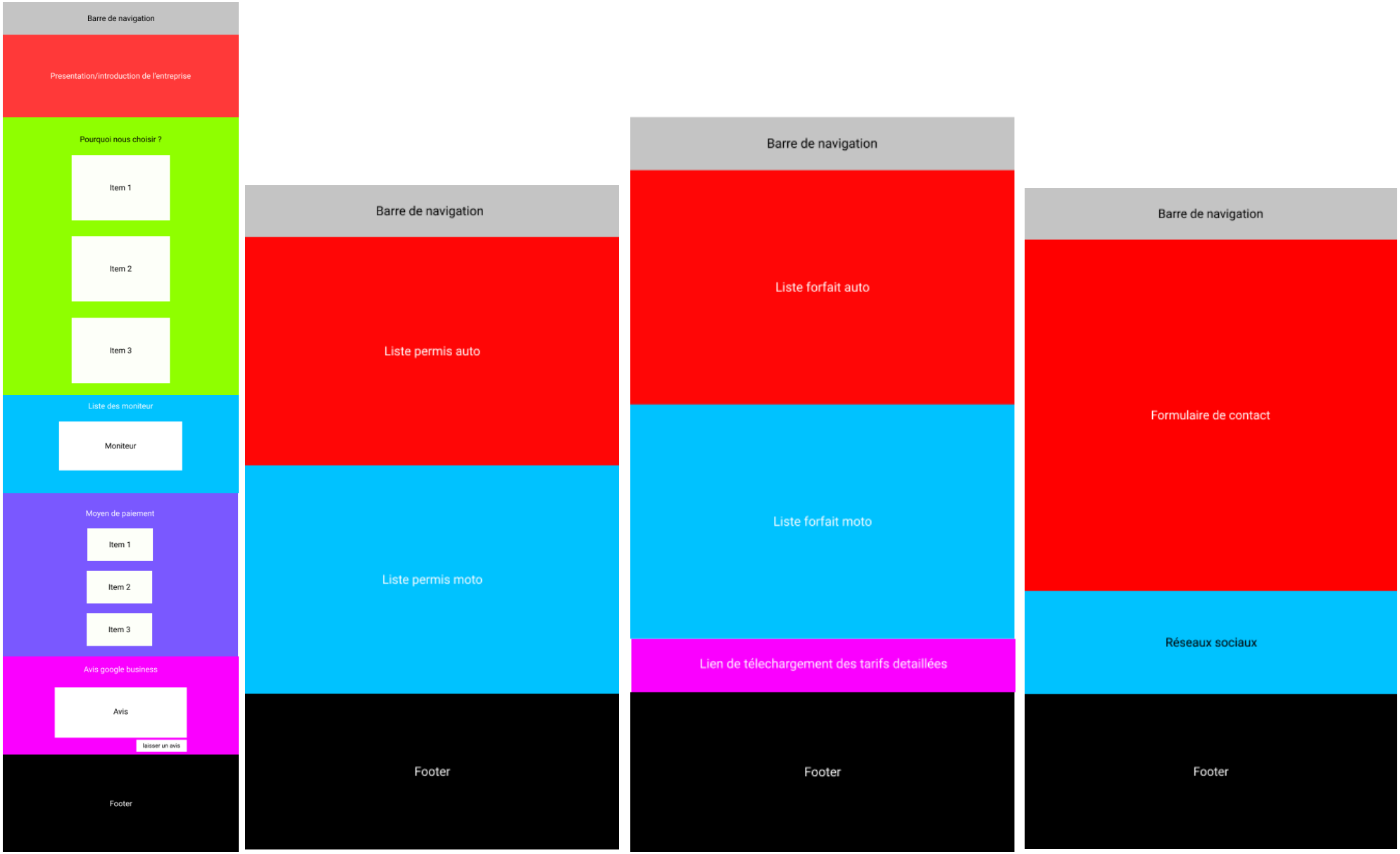
Auto-Ecole de Pierrefeu
Product Backlog Initial

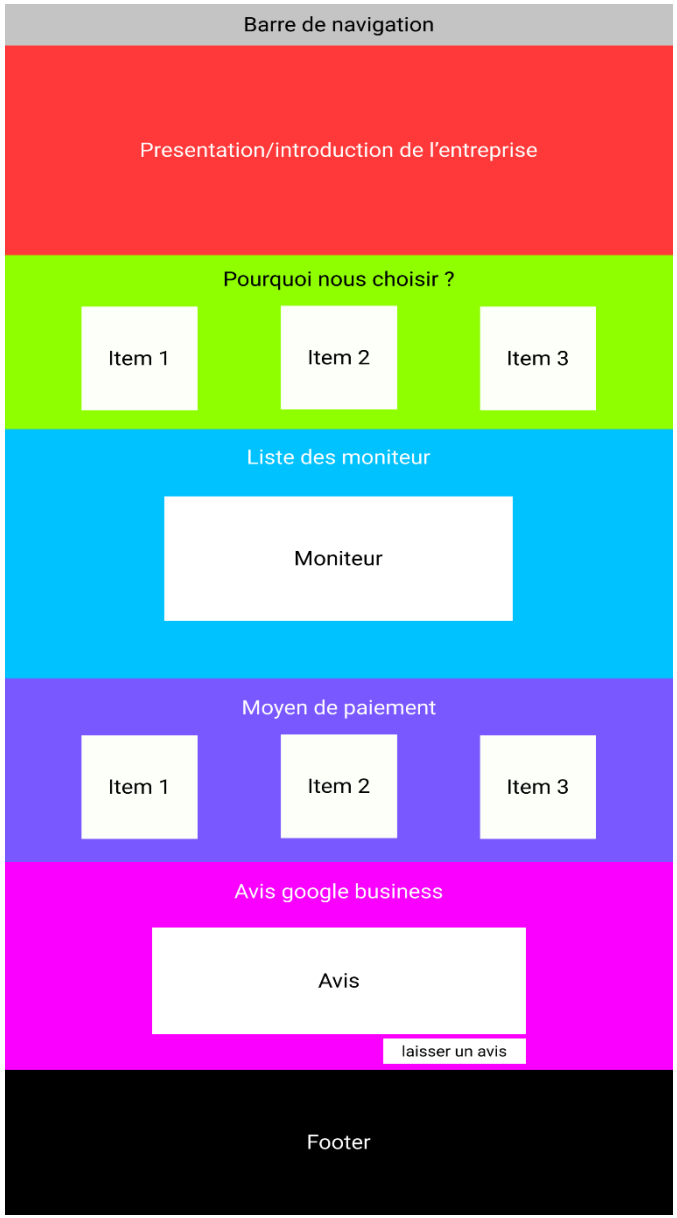
M(ust) S(hould) C(ould) W(on't)

Back Office								
Volet	**ID**	**TITRE**	**EN TANT QUE...**	**JE VEUX/JE PEUX...**	**AFIN DE...**	**MoSCoW**	**BV**	**EFFORT**
BO	1	Lister les forfaits	Administrateur	Afficher la liste de toutes les forfaits	verifier l'existence/consulter les forfaits	M	100	S
BO	2	Lister les tarifs detaillés	Administrateur	Afficher la liste de toutes les tarifs detaillés	verifier l'existence/consulter les tarifs detaillés	M	100	S
BO	3	Lister les moniteurs	Administrateur	Afficher la liste de toutes les moniteurs	verifier l'existence/consulter les moniteurs	M	100	S
BO	4	Se connecter	Administrateur	Me connecter de maniére sécuriser	acceder au fonctionnalitées du backoffice sans crainte	M	50	M
BO	5	Modifier un forfait	Administrateur	Pouvoir modifier les informations concernant un forfait	maintenir à jour les information concernant un forfait	M	100	M
BO	6	Modifier un tarif detaillé	Administrateur	Pouvoir modifier les informations concernant un tarif detaillé	maintenir à jour les information concernant un tarif detaillé	M	100	M
BO	7	Modifier un moniteur	Administrateur	Pouvoir modifier les ionformations concernant un moniteur	maintenir à jour les informations concernant un moniteur	M	100	M
BO	8	Supprimer un forfait	Administrateur	Pouvoir supprimer un forfait	maintenir à jour la liste des forfaits disponnible	M	100	S
BO	9	Supprimer un tarif detaillé	Administrateur	Pouvoir supprimer un tarif detaillé	maintenir à jour la liste des tarifs detaillés disponnible	M	100	S
BO	10	Supprimer un moniteur	Administrateur	Pouvoir supprimer un moniteur	maintenir à jour la liste des moniteurs présent dans l'entreprise	M	100	S
BO	11	Ajouter un forfait	Administrateur	Pouvoir ajouter un forfait	maintenir à jour la liste des forfaits disponnible	M	100	M
BO	12	Ajouter un tarif detaillé	Administrateur	Pouvoir ajouter un tarif detaillé	maintenir à jour la liste des tarifs detaillés disponnible	M	100	M
BO	13	Ajouter un moniteur	Administrateur	Pouvoir ajouter un moniteur	maintenir à jour la liste des moniteurs présent dans l'entreprise	M	100	M
BO	14	Se deconnecter	Administrateur	Pouvoir me deconnecter	eviter les intrusions et modifications des données du site sans autorisation	M	50	S
Site Vitrine								
Volet	**ID**	**TITRE**	**EN TANT QUE...**	**JE VEUX/JE PEUX...**	**AFIN DE...**	**MoSCoW**	**BV**	**EFFORT**
SV	1	Barre de navigation	Utilisateur	Naviguez entre les pages	avoir un accés suimple et complet à tout le contenue du site	M	100	S
SV	2	Responsive design	Utilisateur	Consulter le site de n'importe quel appareil	avoir accés à tout le contenue depuis n'importe quel appareil	M	100	M
SV	3	Carousel des moniteurs	Utilisateur	Consulter la liste de tout les moniteur ainsi que leurs infiormations	connaitre la liste des moniteurs de l'entreprise	M	100	M
SV	4	Carousel Avis	Utilisateur	Consulter les avis google de l'entreprise	connaitre les avis laisser sur google	M	100	S
SV	5	Laisser un avis	Utilisateur	Laisser un avis google	faire connaitre mon avis aux autres visiteurs	M	100	S
SV	6	Formulaire de contact	Utilisateur	Contacter l'entreprise via un formulaire simple	obtenir des renseignement supplémentaire de l'entreprise	M	100	M

((-

Annexe 3


Barre de navigation
Presentation/introduction de l'entreprise
Pourquoi nous choisir ?
Item 1
Item 2
Item 3
Liste des moniteur
Moniteur
Moyen de paiement
Item 1
Item 2
Item 3
Avis google business
Avis
laisser un avis
Footer
Barre de navigation
Liste permis auto
Liste permis moto
Footer
Barre de navigation
Liste forfait auto
Liste forfait moto
Lien de télechargement des tarifs detaillées
Footer
Barre de navigation
Formulaire de contact
Réseaux sociaux
Footer


Barre de navigation
Presentation/introduction de l'entreprise
Pourquoi nous choisir ?
Item 1
Item 2
Item 3
Liste des moniteur
Moniteur
Moyen de paiement
Item 1
Item 2
Item 3
Avis google business
Avis
laisser un avis
Footer