

Dossier de projet

HERDA Yanis

Titre : Développeur web et web mobile

Projet réalisé en stage pour la société Mister James



Remerciements

Je tiens à prendre quelques instants pour exprimer mes sincères remerciements à tous les acteurs qui m'ont permis de réaliser cette formation et ce projet.

Tout d'abord je tiens à remercier mes maîtres de stage pour leur accompagnement et leur sérieux à mon égard. Mes deux stages m'ont permis de voir différents aspects du métier et les conseils qui m'ont été offerts ont été précieux.

Je voudrais également remercier les formateurs pour leurs patience et pour la qualité de leurs cours.

Merci aussi à mes camarades de classes avec qui ont a su maintenir une bonne ambiance de classe ainsi qu'une bonne cohésion.

Enfin, je tiens à remercier nos référents de formation pour leur soutien tout au long de notre parcours. Votre présence et votre accompagnement m'ont permis de me concentrer sur mes objectifs.

SOMMAIRE

1 - INTRODUCTION.....	4
1.1 Mon projet de développeur web et web mobile.....	4
2 - Présentation du projet.....	5
2.1 Présentation de l'entreprise.....	5
2.1 Besoin du client et mise en place du cahier des charge.....	5
3 - Environnement humain et technique.....	8
3.1 Environnement humain.....	8
3.2 Environnement technique.....	8
4 - Compétences du référentiel.....	12
4.1 Maquetter une application.....	12
4.2 Réaliser une interface utilisateur web statique et adaptable.....	13
4.3 Développer une interface utilisateur web dynamique.....	18
4.4 Créer une base de données.....	27
4.1 Développer les composants d'accès aux données.....	28
5 - Exemple de veille et traduction	35
6 - Conclusion	36
6 - Annexes.....	37

1 - INTRODUCTION

1.1 Mon projet de développeur web et web mobile

J'ai tout d'abord après un baccalauréat sciences techniques du management et de la gestion intégré la faculté de Caen pour une licence Administration économique et sociale. Ce n'était par vraiment un choix mais une solution par défaut.

Cette licence ne m'a pas vraiment plu mais j'ai tout de même obtenu mon diplôme.

A la fin de cette formation, j'ai postulé à beaucoup de master différents sans avoir une seule réponse positive. J'ai donc enchainé différents petits travaux.

Par la suite, un peu par hasard, j'ai beaucoup entendu sur les réseaux sociaux parler du métier de développeur. L'informatique m'ayant toujours intéressé, j'ai décidé de commencé à apprendre en autodidacte et surtout voir si cela me plaisait.

Ce fût une révélation pour moi, j'adore ça. J'ai donc appris seul tout en travaillant à coté pendant environ 4 mois et par la suite j'ai décidé de rejoindre la formation développeur web et web mobile suite au conseil de mon conseiller pôle emploi afin de continuer à apprendre et à me perfectionner.

A la fin de cette formation, mon objectif principal est de me spécialiser en front-end. Créer de belles interfaces est ce qui me plaît le plus dans le développement.

Pour cela je compte chercher directement un poste de junior développeur ou chercher une alternance pour la rentrée de Septembre 2023 tout en continuant à me former en parallèle.

2 - Présentation du projet

2.1 Présentation de l'entreprise

Mister James est un salon de coiffure et barbier situé à Grentheville près de Caen. Il s'agit d'un tout nouveau salon qui a ouvert très récemment.

Ce salon est constitué d'une seule personne qui est à son compte. Cette personne est Monsieur James Léo, un jeune diplômé en coiffure qui a décidé de se lancer à son compte. Il sera donc mon maître de stage.

2.2 Besoin du client et mise en place du cahier des charges

Les besoins du client étaient divers. On a donc mis en place un cahier des charges pour mettre en place différents points : les objectifs et les fonctionnalités principales, le contenu, la cible et une petite charte graphique.

Le client avait besoin de 2 sites. Le premier site est celui de salon qui sera accessible à tout le monde et le deuxième sera interne et permettra de gérer une base de données clients.

Objectifs et fonctionnalités principales

Pour le **premier site** les objectifs et fonctionnalités principales sont :

- Présenter le salon
- Présenter les services offerts par le salon : coupe et barbe
- Présenter rapidement Léo : son parcours et son expérience
- Permettre la réservation en ligne (objectif principale)
- Fournir des informations sur les prix, les horaires d'ouverture et de fermeture et permettre la prise de contact

Pour le **deuxième site** :

Permettre de gérer une base de données clients grâce à une interface (ajouter, supprimer, modifier, rechercher des clients). Les informations dont il avait besoin pour constituer un profil client sont : nom, prénom et e-mail.

Contenu

Au niveau du contenu, pour le **premier site**, nous avons défini trois pages : une page "Home", une page "Prestations" et une page "Accès/Contact".

Nous avons aussi défini le contenu de la barre de navigation ainsi que du pied de page.

Barre de navigation -> Le logo, lien vers "Home", lien vers "Prestations", lien vers "Accès/Contact".

Pied de page -> Adresse, numéro, Instagram, Copyright, informations légales, conditions générales d'utilisation.

Page "Home" -> En premier lieu devra apparaître les horaires, le lieu, un numéro de téléphone et un "call to action" vers la réservation. Ensuite, une section de présentation du salon, un petit résumé des services avec un bouton de redirection vers la page "Prestations" et un petit résumé de l'emplacement et du contact avec une redirection vers la page "Accès/Contact".

Page "Prestations" -> Tout comme la page "Home", en premier lieu devra apparaître les horaires, le lieu, un numéro de téléphone et un "call to action" vers la réservation. Puis différentes "Card" avec les différents services proposés (Full cut, barbe, cheveux) avec sur chaque "Card" un call to action pour réserver.

Page "Accès/Contact" -> Une carte avec un point où se trouve le salon avec les horaires puis un formulaire de contact.

Pour le **deuxième site**, on a défini seulement une page principale qui va permettre de voir les clients et réaliser les actions nécessaires (ajouter, supprimer, modifier, rechercher des clients).

Cible

Le **premier site** sera pour les clients déjà établi de Mister James ainsi que pour attirer de nouveaux clients.

Le **deuxième site** sera interne à l'entreprise et donc sera utilisé uniquement par Monsieur James.

Charte graphique rapide

Nous avons définis les mêmes couleurs et les mêmes fonts pour les deux sites :

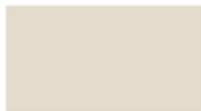
COULEURS



Noir #252525



Jaune moutarde #DCB233



doré/beige #E3DCCB

FONTS

Texte classique : Roboto 16px

NAVBAR : ROBOTO 18px (opacité 60%)

H1 : Lora 32px

COORDONNEES : ROBOTO 24px (opacité 80%)

H2 : Lora 26px

H3 : Lora 22px

3 - Environnement humain et technique

3.1 Environnement humain

J'ai effectué quasiment tout le projet depuis chez moi.

Mes seules interactions ont été avec mon maître de stage afin de faire valider mon travail et pour recueillir ses suggestions. Ces interactions se sont faites soit en présentiel au début pour la mise en place du cahier des charges puis par messages.

3.2 Environnement techniques

Planification et gestion du projet

Etant seul, j'ai décidé de planifier mon projet en instaurant des petits objectifs à court terme. Dès que plusieurs objectifs étaient remplis, je consultais mon maître de stage et réalisais les ajustements nécessaires. (méthodologie agile)

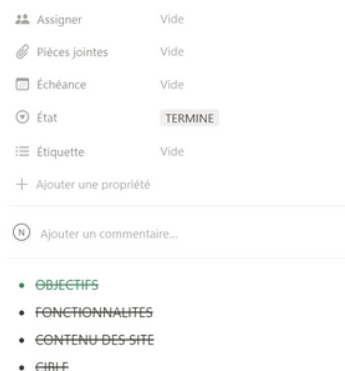
Pour toutes mes notes et rédaction du cahier des charges, j'ai utilisé l'outil Notion qui est un outil puissant de prises de notes et de travail en équipe.

Pour la planification des objectifs, j'ai utilisé l'outil Trello avec les trois colonnes usuelles (à faire, en cours, terminé) que j'ai intégré dans Notion afin d'avoir tout au même endroit.

TRELLO MISTER JAMES



- CAHIER DES CHARGES



GIT

Pour le *versionning* de l'application, j'ai décidé d'utiliser Git ainsi que Github.

Git et github m'offrent la possibilité d'enregistrer mon travail en ligne et de le récupérer sur n'importe quel autre poste de travail, de revenir en arrière si les dernières modifications ne sont pas satisfaisantes et permet surtout de travailler avec différentes branches.

Cela permet de travailler sur plusieurs fonctionnalités ou modifications de code en parallèle, sans affecter le code principal et permet aussi de tester de nouvelles fonctionnalités ou modifications de code sans perturber le code principal.

J'ai personnellement utilisé deux branches : la branche "dev" ainsi que la branche "main".

Je travaillais sur la branche "dev" et une fois qu'une avancée majeure était réalisée, j'utilisais la commande "git checkout main" pour me remettre sur la branche "main" puis j'utilisais la commande "git merge dev" afin d'ajouter toutes les avancées sur la branche "main".

(Les mêmes procédés ont été utilisés pour les 2 sites)

IDE (integrated Development Environment)

J'ai décidé d'utiliser Visual studio code car c'est l'IDE avec lequel j'ai le plus l'habitude de travailler. Il est gratuit, open source et dispose d'une vaste bibliothèque d'extensions facilitant l'écriture du code.

SPÉCIFICITÉS TECHNIQUES

Pour le premier site, j'ai décidé d'utiliser la librairie javascript React.

React permet de réaliser des applications web très performantes grâce au DOM virtuel.

Il permet aussi une expérience utilisateur plus fluide grâce à l'approche SPA (single page application) qui permet de charger une seule fois le site à son lancement. Cela signifie que les utilisateurs peuvent interagir avec l'application sans avoir à attendre que chaque page soit rechargée.

J'ai personnellement utilisé React pour ces raisons et afin d'avoir un code maintenable grâce au découpage du site en différents composants réutilisable dans différentes parties de l'application.

React dispose aussi d'un très gros écosystème ce qui permet d'avoir beaucoup de ressources grâce à la documentation et à différents tutoriels mais aussi cela permet d'étendre ses fonctionnalités de bases grâce à de multiples bibliothèques tierces (ou "packages") qui permettent d'étendre les fonctionnalités de base de React.

J'ai personnellement utilisé pour ce site plusieurs packages :

- React-router (pour avoir différentes pages)
- React-leaflet (pour avoir une carte)
- EmailJS (pour le formulaire de contact)
- Calendly (pour la réservation)

Pour installer ces différents packages j'ai utilisé npm (node packages manager).

Au niveau du CSS, j'ai utilisé la bibliothèque Tailwind. J'ai décidé d'utiliser Tailwind pour maintenir une bonne cohérence visuelle sur le site grâce à la création d'un petit design système.

Je l'ai aussi utilisé car il permet d'adapter facilement et rapidement son interface à tout type d'écran grâce aux classes prédéfinies.

Je trouve l'alliance des composants React avec Tailwind très pratique pour maintenir le code et pour modifier le CSS de composant grâce au "props".

Exemple : On met une "props" sur le background ce qui nous permet de réutiliser le composant avec un background différent en fonction du besoin de façon simple et rapide.

```
<PopupButton  
  className={` ${props.background} px-8 py-2 text-white uppercase font-black rounded drop-shadow-2xl  
  tracking-widest hover:opacity-80 stroke-noir`}  
  url="https://calendly.com/test-calendly/prise-rdv"  
  rootElement={document.getElementById("root")}  
  text='Réserver'  
/>
```

```
<BtnResa1 background='bg-noir' />
```

Pour le deuxième site, il s'agissait d'une simple connexion avec un CRUD (create, read, update, delete). J'ai donc décidé d'utiliser ce que je connaissais c'est à dire Php pour le langage coté serveur et Mysql pour la base de données.

Ce sont des langages populaires dans le monde du développement et il dispose donc de beaucoup de ressources. (documentation et tutoriel)

J'ai également utilisé Wamp pour le serveur local et phpMyAdmin pour la création de la base données.

Pour ce qui est de la maquette j'ai utilisé Figma.

Figma est un outil de conception d'interface utilisateur en ligne qui permet de créer des maquettes, des prototypes et des designs d'interface utilisateur pour les applications web et mobiles.

Je l'ai utilisé car ça prise en main est plutôt simple. C'est un logiciel gratuit et qui ne nécessite aucun téléchargement de logiciel.

Ne disposant pas d'image pour illustrer le premier site, j'ai également utilisé l'intelligence artificielle "MidJourney" pour générer des images temporaires.

(Une maquette a été réalisée seulement pour le premier site).

4 - Compétences du référentiel

Les 3 premiers points (front-end) ont été réalisés sur le premier site et les 2 derniers points (back-end) ont été réalisés sur le deuxième site.

4.1 CP1 - Maquetter une application

Afin de réaliser la maquette avec Figma, je me suis appuyé sur le cahier des charges où toutes les informations dont j'avais besoin figuré déjà. (cf: contenu et charte graphique).

J'ai commencé par la page "Home" au format desktop puis au format mobile. Puis j'ai fait pareil pour les deux autres pages.

HOME DESKTOP



HOME MOBILE



La suite de la maquette est disponible en annexe.

4.2 CP2 - Réaliser une interface utilisateur web statique et adaptable

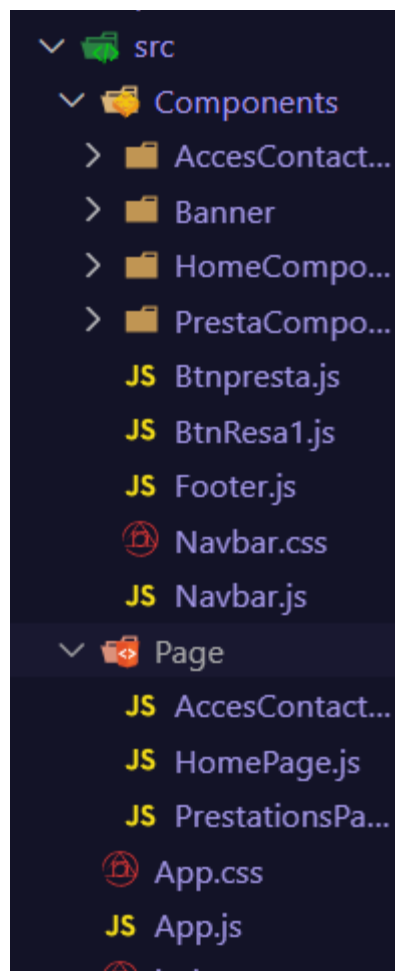
Une fois les maquettes finies et validées par monsieur James, j'ai pu commencer l'intégration du site.

La première étape a été de créer le projet React. Pour cela j'ai créé un dossier MisterJames puis j'ai ouvert un terminal de commande et j'ai tapé la commande suivante :

```
MisterJames>npx create-react-app mister-james
```

Une fois le projet React créé, je l'ai ouvert avec Visual studio code. J'ai commencé par supprimer les fichiers inutiles et j'ai créé l'architecture du projet.

Un fichier "Pages" où sera rendu mes composants qui une fois assemblée dans le bon ordre formeront mes différentes pages et un fichier "Components".



Par la suite afin de naviguer sur le site, j'ai créé les différentes routes grâce à react-router.

INSTALLATION

```
\mister-james> npm i react-router
```

IMPORT DANS REACT

```
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
```

CRÉATION DES DIFFÉRENTES ROUTES

Ce code permet d'assigner le rendu d'un fichier (ici en l'occurrence nos fichiers contenus dans le dossier "pages" ex : <PrestationsPage />) avec un chemin (/prestations). Lorsque l'on tapera l'URL de notre site suivi de "/prestations", il nous sera rendu à l'écran la page "prestations".

```
function App() {
  return (
    <div className=''>
      <Router>
        <Routes>
          <Route path="/" element={<HomePage />}></Route>
          <Route path="/prestations" element={<PrestationsPage />}></Route>
          <Route path="/accès-contact" element={<AccèsContactPage />}></Route>
        </Routes>
      </Router>
    </div>
  );
}
```

Une fois les routes terminées c'est à ce moment que j'ai installé Tailwind en suivant les instructions de la documentation.

Tailwind sera l'élément principal qui me permettra de rendre le site adaptable.

Aujourd'hui la majorité du trafic web se passe sur mobile. Il est donc important que notre site soit parfaitement adapté à ce format.

C'est une des forces de Tailwind car il permet de coder en mobile first. Cela veut dire que l'on doit coder la partie mobile en premier lieu, puis adapter notre code aux autres formats.

La première chose que j'ai faite une fois Tailwind installé a été d'importer les fonts et de mettre en place les différentes couleurs.

Importation des fonts

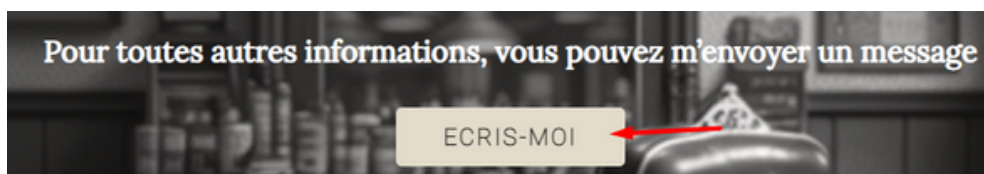
```
index.css x
mister-james > src > index.css
You, il y a 3 semaines | 1 author (You)
1 @import url('https://fonts.googleapis.com/css2?family=Roboto:wght@100&display=swap');
2 @import url('https://fonts.googleapis.com/css2?family=Lora&family=Roboto:wght@100&display=swap');
```

On ajoute les fonts et les différentes couleurs au fichier tailwind.config afin de pouvoir les utiliser avec les classes de Tailwind.

```
tailwind.config.js x
mister-james > tailwind.config.js > ...
You, il y a 3 semaines | 1 author (You)
1 /** @type {import('tailwindcss').Config} */
2 module.exports = {
3   content: [
4     './src/**/*.js,jsx,ts,tsx',
5   ],
6   theme: {
7     extend: {},
8     colors: {
9       moutarde: '#DCB233',
10      card: '#B7A98E',
11      noir: '#252525',
12      dore: '#E3DCCB',
13      white: '#ffffff'
14    },
15    fontFamily: {
16      roboto: ["Roboto", "sans-serif"],
17      lora: ["Lora", "serif"]
18    },
19  },
20 }
```

Exemple d'utilisation d'une classe personnalisée :

```
<a href="/aces-contact"><button className='bg-dore text-noir px-8 py-2 uppercase font-black rounded drop-shadow-2xl tracking-widest hover:opacity-8'>Ecris-moi</button></a>
```



Une fois tout installé, j'ai commencé à coder le site en commençant par le format mobile puis en l'adaptant aux autres formats.

Voici deux exemples qui montre comment j'ai procédé :

Exemple 1 : La barre de navigation

Sur format mobile, on retrouve un menu burger sur la barre de navigation. Ce bouton disparaît sur grand écran grâce à la classe "lg:hidden" (<1024px).

```
className='burger lg:hidden'>
```

Au dessus de de 1024px apparaît un menu avec plusieurs boutons qui correspondent à chaque page. ("Home", "Prestations", "Accès/Contact". Ces boutons ont la classe "hidden" de base et une fois au dessus de 1024px, on passe en "flex" afin qu'ils apparaissent.

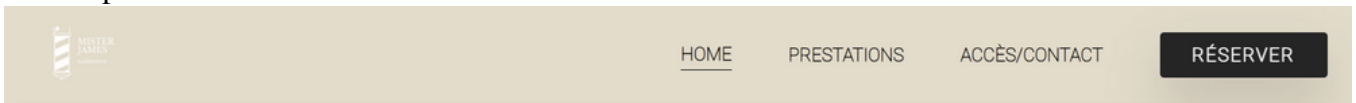
```
<ul className='text-lg uppercase font-semibold leading-7 hidden lg:flex lg:gap-14 lg:items-center w-fit'>
  <li className={`hover:text-white ${location.pathname === '/' ? 'border-b-[1px]' : ''}`}><a href="/">Home</a></li>
  <li className={`hover:text-white ${location.pathname === '/prestations' ? 'border-b-[1px]' : ''}`}><a href="/prestations">prestations</a></li>
  <li className={`hover:text-white ${location.pathname === '/accès-contact' ? 'border-b-[1px]' : ''}`}><a href="/accès-contact">accès/contact</a></li>
  <BtnResal background='bg-noir' />
</ul>
```

Rendu final :

Mobile



Desktop



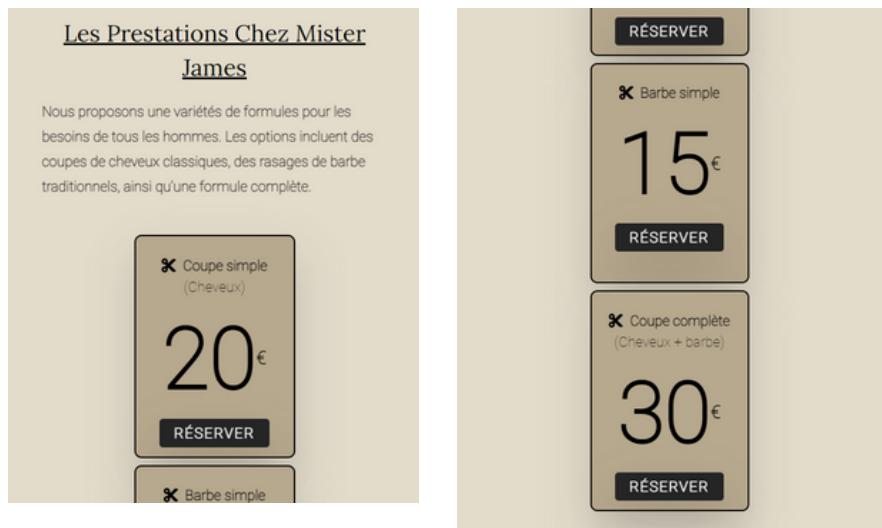
Exemple 2 : Cards des prestations

Sur format mobile, on retrouve des cards avec les différentes prestations qui sont affichées en colonne et en format desktop, ces cards sont affichées sur une ligne. Cette fois la modification s'opère dès 768px grâce à la classe "md".

```
<div className='mt-10 flex flex-col items-center gap-2 md:flex-row md:gap-10 justify-center'>
  <Card titleCard="Coupe simple" desc="(Cheveux)" price="20"/>
  <Card titleCard="Barbe simple" price="15"/>
  <Card titleCard="Coupe complète" desc="(Cheveux + barbe)" price="30"/>
</div>
```

Rendu final :

Mobile



Desktop



Les mêmes procédés ont été utilisés pour chaque partie du site pour le rendre parfaitement adaptable

4.3 CP3 - Développer une interface utilisateur web dynamique

Les principaux outils que j'ai utilisé pour rendre le site dynamique sont Javascript et React ainsi que l'installation de différents packages.

Les tâches à effectuer pour rendre le site dynamique étaient :

- Rendre le menu burger cliquable
- Afficher si le salon est ouvert ou fermé en fonction des horaires
- Permettre à l'utilisateur la prise de rendez-vous
- Afficher une carte dynamique de la position du salon
- Permettre à l'utilisateur de laisser un message

Rendre le menu burger cliquable

Pour rendre le menu cliquable j'ai d'abord importé le "hook" "useState".

Un Hook est une fonction qui permet de « se brancher » sur des fonctionnalités React.

Le Hook useState permet d'ajouter l'état local React à des fonctions composants.

On doit d'abord l'importer de cette façon :

```
import { useState } from 'react'
```

```
const [isOpen, setIsOpen] = useState(false);
```

useState envoie un tableau de taille 2 :

- index 0 = la valeur de l'Etat (donc ici "isOpen" est égale à "false")
- index 1 = une fonction qui permet de changer la valeur de l'état. (ci dessous)

Quand on appuie sur le bouton burger, l'état de isOpen passe de "false" à "true" grâce à la fonction "setIsOpen".

```
<input id="burger" type="checkbox" onClick={() => setIsOpen(!isOpen)} />
```

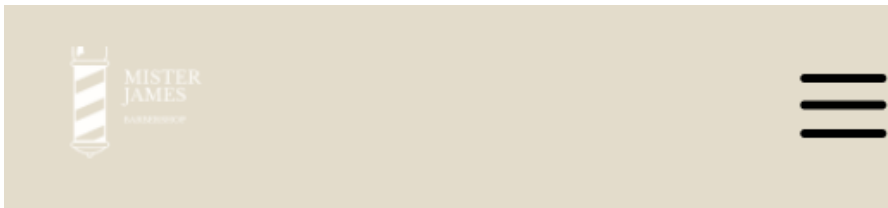
L'étape d'après est de créer une `<div>` qui s'affichera quand on clique sur le menu burger et disparaîtra quand on reclique. Pour faire cela j'ai utilisé une condition (plus précisément un opérateur ternaire) dans mon JSX (JSX est une extension de syntaxe pour JavaScript qui permet d'écrire des éléments similaires à des balises HTML directement dans le code JavaScript).

Donc ici on lui dit que si `isOpen` est égale à `"true"` alors on affiche la `<div>` sinon on affiche rien.

```
{isOpen ? <div>
  <ul className='fixed flex justify-center item-center flex-col lg:hidden gap-14 items-center mx-0 z-30 h-[100vh]
  bg-noir text-white text-2xl uppercase w-[100%]'>
    <li className='hover:opacity-[60%]'><a href="/" className=''>Home</a></li>
    <li className='hover:opacity-[60%]'><a href="/prestations">prestations</a></li>
    <li className='hover:opacity-[60%]'><a href="/accès-contact">accès/contact</a></li>
    <BtnResa1 background='bg-dore' />
  </ul>
</div> : ''}
```

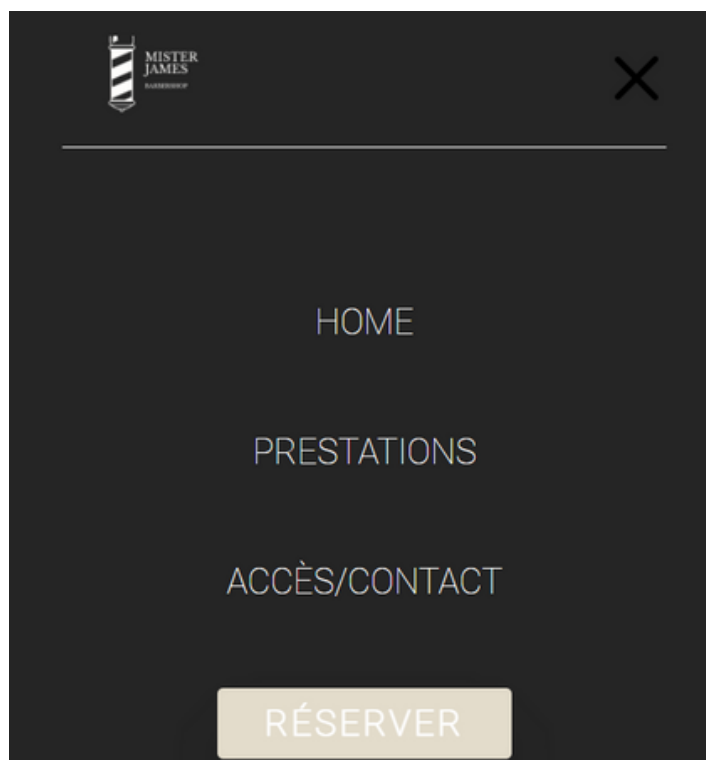
Rendu :

L'état de `isOpen` est `"false"`:



Puis on clique sur le menu l'état passe à `"true"`:

(il repasse à `false` lorsque que l'on clique sur la croix)



Afficher si le salon est ouvert ou fermé en fonction des horaires

Pour afficher si le salon est ouvert ou fermé en fonction des horaires, j'ai utilisé du Javascript avec du JSX.

J'ai d'abord commencé par récupérer la date, l'heure et le jour.

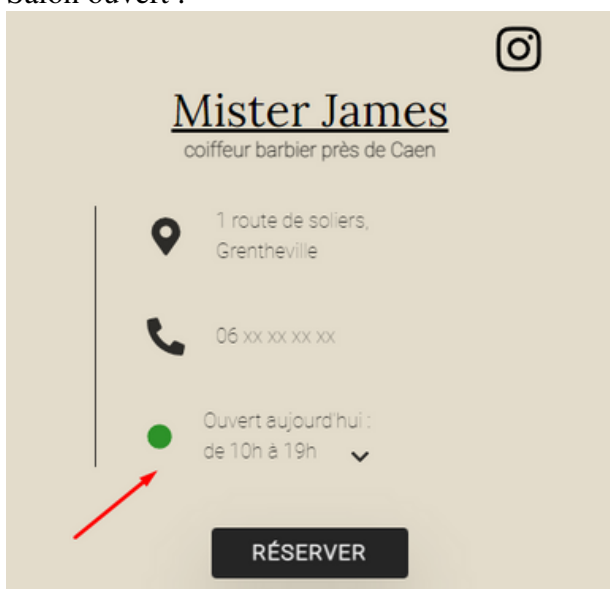
```
const date = new Date();
const hours = date.getHours();
const day = date.getDay();
```

Ensuite j'ai utilisé une condition dans mon JSX (encore un opérateur ternaire) pour lui dire que si on est entre le jour 1 (donc lundi) et le jour 5 (vendredi) et que l'on est entre 9h et 19h alors on affiche que c'est ouvert sinon on affiche que c'est fermé.

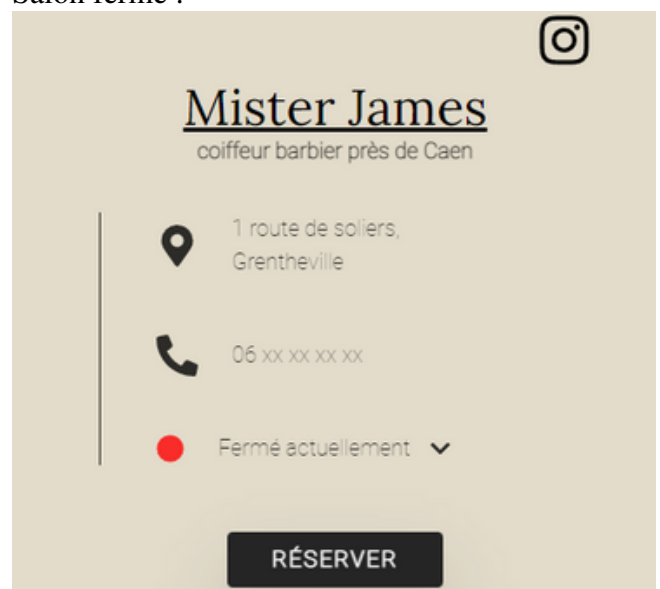
```
{day >= 1 && day <= 5 && hours >= 9 && hours < 19 ?
<
  <div>
    <FaCircle size={20} color='green' />
  </div>
  <span className='whitespace-nowrap block'>Ouvert aujourd'hui :<br /> de 10h à 19h </span>
  <a href='/accès-contact' className='self-end -ml-[40px]'><FaChevronDown /></a>
</> : <
  <div>
    <FaCircle size={20} color='red' />
  </div>
  <span className='block'>Fermé actuellement</span>
  <a href='/accès-contact' className=' -ml-[10px]'><FaChevronDown /></a>
</>
}
```

Rendu :

Salon ouvert :



Salon fermé :



(J'ai conscience que cette fonctionnalité présente une légère faille car elle est basée sur l'heure du client qui peut ne pas être sur le même fuseau horaire que la salon mais j'ai considéré que cette faille n'avait pas grande importance pour un salon de coiffure.)

Permettre à l'utilisateur la prise de rendez-vous

Pour permettre la prise de rendez-vous en ligne, j'ai utilisé Calendly qui est un logiciel gratuit de planification de rendez-vous.

On configure l'outil directement sur le site : <https://calendly.com/>.

Quel est cet événement ?
Prise de rdv, Réunion en privé

Quand les gens peuvent-ils réserver cet événement ?
30 min, indéfiniment

Nous vous recommandons de verrouiller le fuseau horaire de ce type d'événement.

Options supplémentaires

- Questions pour l'invité
Nom, email + 1 question
- Workflows [ESSAYER](#)
Configurez des automatisations concernant vos événements, telles que des notifications par e-mail et par texte, des e-mails de remerciement, etc.
- Notifications et politique d'annulation
Invitations de calendrier, Rappels par email
- Page de confirmation
Page de confirmation Calendly, pas de liens actifs
- Percevoir les paiements
aucun moyen de paiement

Calendar view showing time slots for Monday (LUN) through Friday (VEN). Each day has two time slots: 09:00 - 12:00 and 13:00 - 19:00. There are icons for deleting, adding, and copying slots.

Une fois tout configuré, on peut intégrer Calendly sur notre site de manière très simple.
D'abord on l'installe avec npm.

```
MisterJames@mister-james> npm i react-calendly
```

Une fois installée, il faut créer un bouton qui lancera Calendly au clique. Il s'agit d'un bouton qui sera présent à plusieurs endroits du site. J'ai donc décidé d'en faire un composant "BtnResa1" afin de pouvoir le réutiliser partout dans le code facilement.

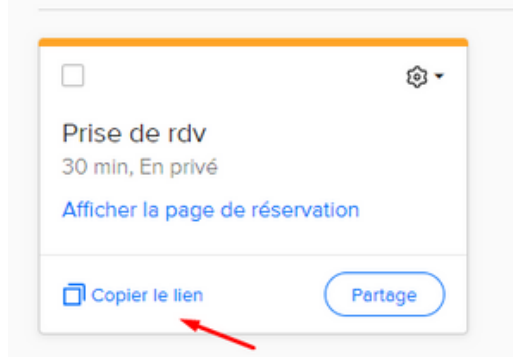
Dans un premier temps, on importe "PopupButton" depuis "react-calendly". Puis on l'utilise dans notre JSX afin de le rendre sur l'écran.

On peut lui passer des classes pour le style, mais surtout une URL qui correspond à ce que l'on a configuré auparavant sur Calendly.

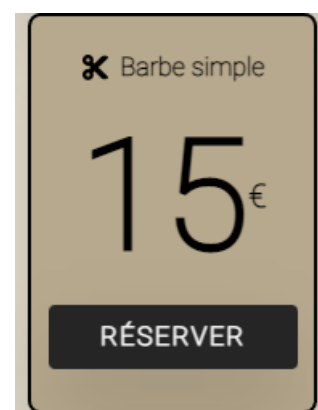
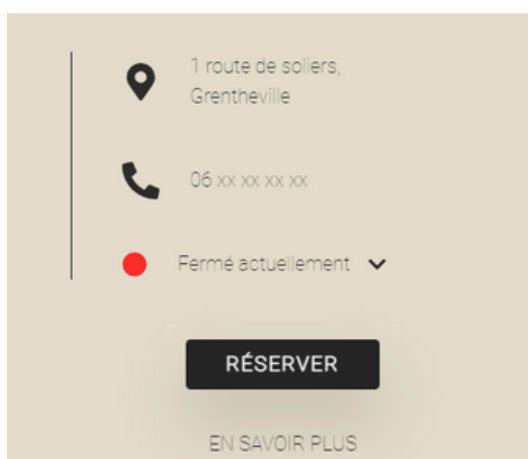
```
import { PopupButton } from "react-calendly";

function BtnResa1(props) {
  return (
    // on utilise les props pour avoir un bg différent dans les 2 menu navbar
    <PopupButton
      className={` ${props.background} px-8 py-2 text-white uppercase font-black rounded drop-shadow-2xl tracking-widest hover:opacity-80 stroke-noir`}
      url="https://calendly.com/test-calendlyy/prise-rdv"
      rootElement={document.getElementById("root")}
      text="Réserver"
    />
  )
}
```

URL Calendly que l'on retrouve sur le site



Rendu : On a un bouton "réserver" disponible à plusieurs endroits du site



Lorsque l'on clique, on a la possibilité de prendre rendez vous en fonction des disponibilités.

The screenshot shows the Calendly booking interface. On the left, the contact information for 'herda yanis' is displayed, including a 30-minute duration and the location '2 route de soliers'. The main section is titled 'Sélectionnez la date et l'heure'. It features a calendar for March 2023 with the 2nd of March selected. To the right of the calendar, a list of available time slots is shown: 13:00, 13:30, 14:00, 14:30, 15:00, and 15:30. Below the calendar, the time zone is set to 'Heure d'Europe centrale (10:54)'. A 'Ajouter au Calendly' button is visible in the top right corner.

On rentre ces coordonnées et la prise de rendez-vous est terminée

The screenshot shows the confirmation step of the Calendly booking process. On the left, the contact information for 'herda yanis' is repeated, along with the selected date and time: '13:00 - 13:30, jeudi, 2 mars 2023'. The main section is titled 'Indiquez vos informations' and contains fields for 'Prénom *', 'Nom *', 'E-mail *', and 'Numéro de téléphone *'. There is also a button labeled 'Ajouter des invités'. A 'Confirmer l'événement' button is at the bottom. A 'Ajouter au Calendly' button is visible in the top right corner.

Confirmé

Vous avez rendez-vous avec herda yanis.

- **Prise de rdv**
- 📅 13:00 - 13:30, jeudi, 2 mars 2023
- 🌐 Heure d'Europe centrale
- 📍 2 route de soliers

Une fois le rendez-vous confirmé, l'utilisateur reçoit un e-mail de confirmation (ou un message en fonction de la configuration de Calendly).

Monsieur James lui aussi recevra un e-mail.

Lorsqu'un rendez-vous est pris, il s'affichera dans l'application calendly à la bonne date ainsi que dans le google calendar de monsieur James. Il pourra à partir de là annuler ou changer la date d'un rendez-vous.

Afficher une carte dynamique de la position du salon

Pour afficher la position du salon sur une carte, j'ai utilisé react-leaflet. Leaflet est une librairie open source qui utilise OpenStreetMap (OpenStreetMap est un projet collaboratif de cartographie en ligne qui vise à constituer une base de données géographiques libre du monde).

J'ai d'abord commencé par l'installer puis je l'ai importé dans un composant Map.js.

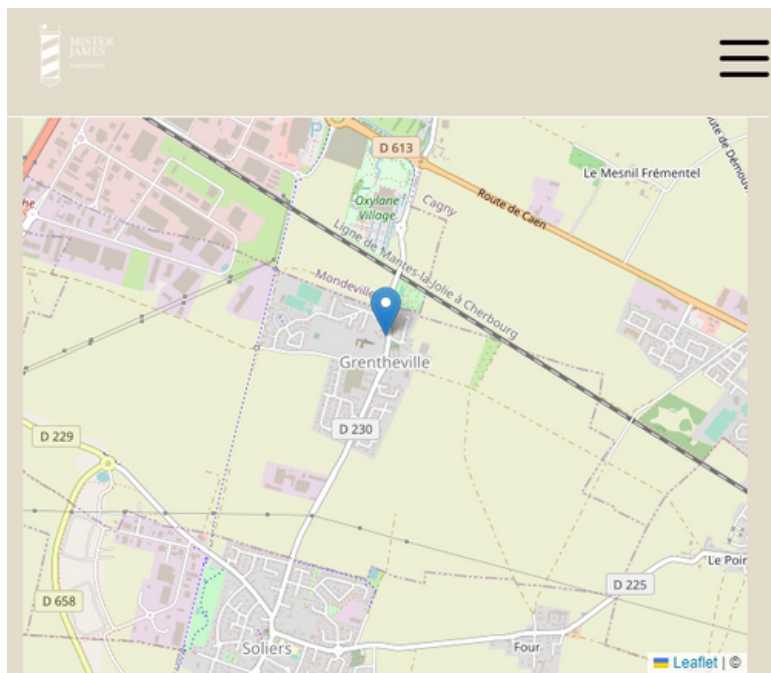
```
MisterJames@mister-james> npm i react-leaflet
```

```
import { MapContainer, TileLayer, Popup, Marker } from 'react-leaflet'
```

Ensuite, en suivant la documentation, j'ai affiché la carte que j'ai centré sur l'endroit où se trouve le salon en affichant un marqueur.

```
function Map() {  
  const position = [49.149053, -0.287389]  
  
  return (  
    <div className='relative z-0'>  
      <MapContainer center={position} zoom={14} scrollWheelZoom={false}>  
        <TileLayer  
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright"></a>  
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"  
        />  
        <Marker position={position}>  
          <Popup minWidth={120} interactive={true}>  
            MISTER JAMES <br /> 2 route de Soliers, <br /> Gretheville  
          </Popup>  
        </Marker>  
      </MapContainer>  
    )  
  )  
}
```

Rendu :



L'utilisation de "useRef()" permet de stocker des références aux champs de formulaire.

Ensuite la fonction "sendEmail" va se connecter à notre template emailJS. Si lorsque l'on appuie sur "envoyer" le message est bien envoyé alors la valeur de "setStatus" passe à "Message envoyé" et les champs du formulaire sont réinitialisés. Sinon un message d'erreur s'affiche.

On affiche "status" à la fin du formulaire :

```
<textarea name="message" className='w-[300px] lg:w-[610px] bg-dore border-2 focus:outline-none border-card rounded py-2 px-3'></textarea>  
</div>  
<button type='submit' className='bg-noir w-fit mx-auto text-white py-2 px-8 rounded'>Envoyer</button>  
{status}
```

Rendu avant l'envoi :



Rendu après l'envoi : Monsieur James recevra le message sur sa boîte mail avec toutes les informations de l'utilisateur.



4.4 CP4 - Créer une base de données

Mysql est le système de gestion de bases de données que j'ai utilisé pour ce projet.

J'ai travaillé uniquement en local grâce à "WampSever" qui me permet d'avoir un serveur local et qui me permet d'accéder à phpMyAdmin pour l'administration de ma base de données.

J'ai donc créé la base de données "James" grâce à phpMyAdmin avec une table clients.

Cette table clients est constituée d'un "id", d'un "nom", d'un "prénom" et d'un "e-mail" et une table "admin" constitué d'un "login" et d'un "pass".

Vision global de la base de données :

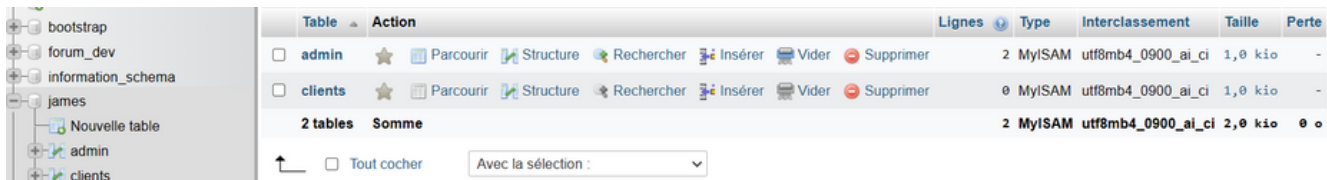
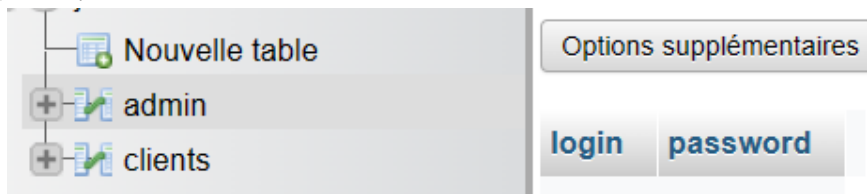


Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> admin	★ Parcourir Structure Rechercher Insérer Vider Supprimer	2	MyISAM	utf8mb4_0900_ai_ci	1,0 kio	-
<input type="checkbox"/> clients	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	MyISAM	utf8mb4_0900_ai_ci	1,0 kio	-
2 tables Somme		2	MyISAM	utf8mb4_0900_ai_ci	2,0 kio	0 o

Table "admin" :

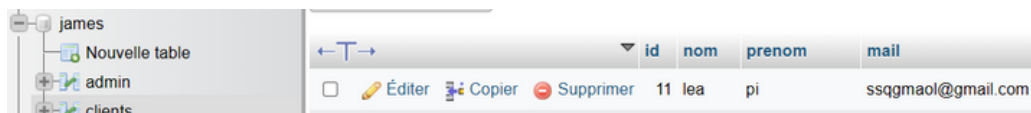


Nouvelle table
admin
clients

Options supplémentaires

login password

Table "client" :



Nouvelle table
admin
clients

Options supplémentaires

id nom prenom mail

11 lea pi ssqgmaol@gmail.com

4.4 CP5 - Développer les composants d'accès aux données

Pour commencer, je me suis connecté avec php à la base de données locale. Pour cela, j'ai utilisé la méthode php "PDO()". Cette méthode permet d'accéder à la base de données et d'exécuter des requêtes sql.

Cette méthode sert aussi à se prémunir contre les injections sql grâce aux requêtes préparées qui permettent de séparer la logique de la requête et les données qui y sont associées, ce qui empêche les données d'être interprétées comme une partie de la requête SQL.

```
<?php

// Paramètres de connexion à la base de données
$host = "localhost";
$dbname = "James";
$username = "root";
$password = "";

// Tentative de connexion à la base de données
try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8mb4", $username, $password);
} catch (PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
    exit();
}

?>
```

Authentification

Une fois connecté à la base de données, j'ai mis en place un système de connexion. Pour réaliser cela, j'ai ajouté une ligne dans ma table "admin (donc un "login" et un "pass").

J'ai ensuite créé mon formulaire de connexion en HTML puis j'ai fait ma requête sql qui va venir vérifier si le "login" et le "pass" sont correctes. Si il sont corrects, alors on est redigéré vers la page index.php et une session est créée.

Sinon un message d'erreur s'affiche.

une session est un moyen de stocker des informations spécifiques à un utilisateur sur le serveur web. Ici l'ouverture de la session va permettre de vérifier si l'utilisateur est bien connecté ou non. Si il n'est pas connecté, il ne pourra pas accéder à une autre page que celle de connexion.

J'ai également utilisé la fonction "htmlspecialchars()" qui sert à convertir certains caractères spéciaux (guillemets, apostrophes...) en entités HTML ce qui sert à se prémunir des vulnérabilités XSS.

```
include "connexion.php";
$error = "";

if (isset($_POST['login']) && isset($_POST['pass'])) {
    $login = htmlspecialchars($_POST['login']) ;
    $password = htmlspecialchars($_POST['pass']);
    //requete qui va verifier si bien dans la BDD
    $query = "SELECT * FROM admin WHERE login = :login AND password = :password";
    $statement = $pdo->prepare($query);
    $statement->execute(['login' => $login, 'password' => $password]);

    $admin = $statement->fetch(PDO::FETCH_ASSOC);

    if ($admin) {
        session_start();
        $_SESSION["autoriser"] = "oui";
        header('Location:../index.php');
        exit;
    } else {
        $error = 'Identifiants invalides.';
    }
}
```

Affichage des clients sur la page index.php

Pour afficher les informations de ma table clients, j'ai d'abord ajouté quelques clients manuellement sur phpMyAdmin.

Puis j'ai fait une requête sql de tous les éléments de ma table client

```
<table>
  <thead>
    <tr>
      <th>Nom</th>
      <th>Prénom</th>
      <th>Email</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    <?php
      $query = $pdo->query("SELECT * FROM clients");
      $clients = $query->fetchAll(PDO::FETCH_ASSOC);
      foreach($clients as $client) {

        ?>
        <tr>
          <td><?php echo $client['nom'] ?></td>
          <td><?php echo $client['prenom'] ?></td>
          <td><?php echo $client['mail'] ;?></td>
```

Rendu :

Nom	Prénom	Email
herda	Yanis	yanis.herda14@gmail.com
james	leo	leo@gmail.com
val	massier	valmas@gmail.com

Ajout d'un client

Pour l'ajout d'un client, j'ai créé un bouton "ajout" sur ma "index.php" qui redirige vers une page "add.php".

Sur cette page "add.php" j'ai créé un formulaire avec "nom", "prenom", et "mail".

Ensuite on crée notre requête sql. On récupère nos champs de formulaire grâce à "POST" que l'on stock dans des variables (on sécurise avec htmlspecialchars()).

Puis on prépare notre requête et on lie nos variables avec les paramètres de notre requête grâce "bindParam" et on l'exécute. Puis on redirige vers la page "index.php".

```
if(isset($_POST['submit'])){
    $nom = htmlspecialchars($_POST['nom']);
    $prenom = htmlspecialchars($_POST['prenom']);
    $email = htmlspecialchars($_POST['mail']);

    $query = $pdo->prepare("INSERT INTO clients (nom, prenom, mail) VALUES (:nom, :prenom, :mail)");
    $query->bindParam(':nom', $nom);
    $query->bindParam(':prenom', $prenom);
    $query->bindParam(':mail', $email);
    $query->execute();

    // Redirection vers la page index après l'ajout du client
    header("Location:../index.php");
    exit();
}
```

```
<div class="header">
    <h1>Ajout d'un nouveau clients</h1>
</div>
<form method="post">
    <div>
        <label>Nom</label>
        <input type="text" name="nom">
    </div>
    <div>
        <label>Prénom</label>
        <input type="text" name="prenom">
    </div>
    <div>
        <label>E-mail</label>
        <input type="email" name="mail">
    </div>
    <div class="ajout-retour">
        <button type="submit" name="submit">Ajout</button>
        <a href=".." index.php" class="retour">Retour</a>
    </div>
</form>
```

Rendu :

Index.php

Ajout d'un nouveau client

Nom

Prénom

E-mail

[Retour](#)

Nom	Prénom	Email
JE SUIS UN TEST	TEST	TEST@GMAIL.COM

Modification d'un client

Pour la modification d'un client, j'ai ajouté sur ma page "index.php" un bouton "modifier" qui redirige vers une page "edit.php" en prenant en paramètre "ID".

Nom	Prénom	Email	Action
JE SUIS UN TEST	TEST	TEST@GMAIL.COM	<u>Modifier</u>

```
<a href="fonctions/edit.php?id=php echo $client['id']?">Modifier</a>
```

J'ai commencé par récupérer l'id correspondant au bon client grâce à la variable super global "\$_GET".

```
$id = $_GET['id'];
```

Ensuite j'ai utilisé le même procédé que sur ma page "index.php" pour récupérer les informations de ma base de données mais cette fois-ci je récupère seulement les informations du client avec l'id correspondant.

Cela me permet d'avoir un formulaire de modification avec les bonnes informations préremplies.

```
<h1>Modifier</h1>
</div>
<?php
// Récupération des informations du client dans la table "clients"
$query = $pdo->prepare("SELECT * FROM clients WHERE id = :id");
$query->bindParam(':id', $id);
$query->execute();
$client = $query->fetch();
?>
<form method="post">
  <div>
    <label>Nom</label>
    <input type="text" name="nom" value="php echo $client["nom"] ?&gt;" &gt;
  &lt;/div&gt;
  &lt;div&gt;
    &lt;label&gt;Prénom&lt;/label&gt;
    &lt;input type="text" name="prenom" value="<?php echo $client["prenom"]; ?&gt;" &gt;
  &lt;/div&gt;
  &lt;div&gt;
    &lt;label&gt;E-mail&lt;/label&gt;
    &lt;input type="email" name="mail" value="<?php echo $client["mail"]; ?&gt;" &gt;
  &lt;/div&gt;
  &lt;div class="ajout-retour"&gt;
    &lt;button type="submit" name="submit" class="btn-1"&gt;Update&lt;/button&gt;
    &lt;a href="../index.php" class="retour"&gt;Retour&lt;/a&gt;
  &lt;/div&gt;
&lt;/form&gt;</pre
```

Rendu lorsque j'appuie sur modifier depuis ma page "index.php" :

Modifier

Nom

JE SUIS UN TEST

Prénom

TEST

E-mail

TEST@GMAIL.COM

Update

[Retour](#)

Ensuite, il faut une nouvelle requête sql afin de modifier les informations du client. On récupère les informations du nouveau formulaire puis met à jour les informations en reliant chaque champs du formulaire à la bonne colonne de la table client. On prépare la requête puis on l'exécute et on redirige vers la page "index.php".

```

if(isset($_POST['submit'])){
    $nom = htmlspecialchars($_POST["nom"]);
    $prenom = htmlspecialchars($_POST["prenom"]);
    $mail = htmlspecialchars($_POST["mail"]);

    // Mise à jour des informations du client dans la table "clients"
    $query = $pdo->prepare("UPDATE clients SET nom = :nom, prenom = :prenom, mail = :mail WHERE id = :id");
    $query->bindParam(':id', $id);
    $query->bindParam(':nom', $nom);
    $query->bindParam(':prenom', $prenom);
    $query->bindParam(':mail', $mail);
    $query->execute();

    if($query){
        header('location:../index.php');
    } else {
        echo "erreur";
    }
}

```

Rendu :

On appuie sur modifier depuis index.php:

Modifier

Nom

JE SUIS UN TEST

Prénom

TEST

E-mail

TEST@GMAIL.COM

Update

Retour

On modifie et on appuie sur update :

Modifier

Nom

MODIF

Prénom

TEST

E-mail

MODIF@GMAIL.COM

Update

Retour

index.php :

Nom	Prénom	Email
MODIF	TEST	MODIF@GMAIL.COM

Suppression d'un client

Même principe que pour la modification. Je crée un bouton supprimer "X" qui redirige vers une page "deconnexion.php" qui prend un id en paramètre.

Action
Modifier X

```

<a href="fonctions/delete.php?id=php echo $client['id'] ?">X</a>

```

Puis je prépare et exécute une requête sql qui "DELETE" en fonction de l'id et redirige vers "index.php".

```

<?php
include 'connexion.php';
$id = $_GET['id'];

$query = $pdo->prepare("DELETE FROM clients WHERE id = :id");
$query->bindParam(':id', $id);
$query->execute();
header("Location:../index.php");

```

Recherche d'un client

Je crée un bouton recherche qui redirige vers la page search.php.

Sur cette page je crée un formulaire d'un seul champ "nomR".

```
<div class="header">
    <h1>Rechercher un client (par son nom)</h1>
</div>
<form method="post">
    <div>
        <label>Nom du client à recherché</label>
        <input type="text" name="nomR">
    </div>
    <div class="ajout-retour">
        <button type="submit" class="btn-1">Rechercher</button>
        <a href="../index.php" class="retour">Retour</a>
    </div>
</form>
```

Ensuite, je prépare et exécute une requête sql qui va chercher dans ma base de données tous les champs qui contiennent la valeur de l'input "nomR" puis j'affiche tous les résultats.

```
if (isset($_POST['nomR'])) {
    $nomR = $_POST['nomR'];

    $recherche = htmlspecialchars($_POST["nomR"]);

    // Requête pour rechercher les clients correspondant au nom recherché
    $query = $pdo->prepare("SELECT * FROM clients WHERE nom LIKE :nomR");
    $query->bindValue(':nomR', "%$nomR%");
    $query->execute();
    $clients = $query->fetchAll(PDO::FETCH_ASSOC);

    // Affichage des résultats de recherche
    echo "<h2>Résultats pour '$nomR' :</h2>";
    echo "<ul>";
    foreach ($clients as $client) {
        echo "<li>" . $client["nom"] . " " . $client["prenom"] . " (" . $client["mail"] . ")</li>";
        echo "<a class='modif-search' href=edit.php?id=". $client['id'] . " >Modifier</a>";
        echo "<a class='modif-search' href=delete.php?id=". $client['id'] . " >X</a>";
    }
    echo "</ul>";
}
```

Rendu :

Rechercher un client (par son nom)

Nom du client à rechercher

[Retour](#)

Résultats pour 'l' :

- lea pi (ssqgmaol@gmail.com)
Modifier X
- leoo james (leo@gmail.com)
Modifier X

5 - Exemple de veille et traduction

Lors du développement du premier site, j'ai utilisé plusieurs technologies différentes ce qui m'a obligé à effectuer une veille permanente.

La documentation que j'ai le plus utilisé et celle de Tailwind car il existe énormément de classes et il est difficile de tout retenir.

EXTRAIT DE LA DOCUMENTATION TAILWIND :

Overview

Every utility class in Tailwind can be applied conditionally at different breakpoints, which makes it a piece of cake to build complex responsive interfaces without ever leaving your HTML.

There are five breakpoints by default, inspired by common device resolutions:

- sm 640px,
- md 768px,
- lg 1024px,
- xl 1280px,
- 2xl 1536px

To add a utility but only have it take effect at a certain breakpoint, all you need to do is prefix the utility with the breakpoint name, followed by the `:` character:

This works for every utility class in the framework, which means you can change literally anything at a given breakpoint — even things like letter spacing or cursor styles.

By default, Tailwind uses a mobile-first breakpoint system, similar to what you might be used to in other frameworks like Bootstrap.

TRADUCTION :

Aperçu ;

Chaque classe utilitaires de Tailwind peut être appliqué à différents points d'arrêts, ce qui en fait un jeu d'enfant pour construire des interfaces responsifs complexe sans quitter son HTML.

Il y a cinq points d'arrêts par défaut qui sont inspirés par les résolutions des appareils communément utilisés :

- sm 640px,
- md 768px,
- lg 1024px,
- xl 1280px,
- 2xl 1536px

Pour ajouter une classe utilitaire mais qu'elles fassent seulement effet à un certain point d'arrêt, tout ce que vous avez à faire est d'ajouter le préfixe de la classe en question avec les caractères `:`.

Cela marche pour toutes les classes utilitaires du Framework, ce qui veut dire que vous pouvez littéralement tout changer en donnant des points d'arrêts -- même les choses comme l'espace entre les lettres ou le styles des curseurs.

Par défaut, Tailwind utilise un système de point d'arrêts mobile-first, similaire à ce que qui peut être utilisé dans d'autres Framewrok comme Bootstrap.

6 - Conclusion

Ce projet m'a permis de beaucoup progresser sur énormément de points.

J'ai d'abord pu me mettre dans une situation réelle d'un développeur freelance en réalisant un projet complet du début à la fin. (maquette, cahier des charges, site)

Etre totalement livré à soi même sur la partie technique n'est pas toujours simple mais nous oblige à effectuer des travaux de veilles et de recherches permanentes.

Je me suis aperçu qu'avec peut être quelques années d'expérience c'est un mode de travail qui me plairait.

Ce projet m'a aussi permis de m'améliorer sur les différentes technologies utilisées dans le projet.

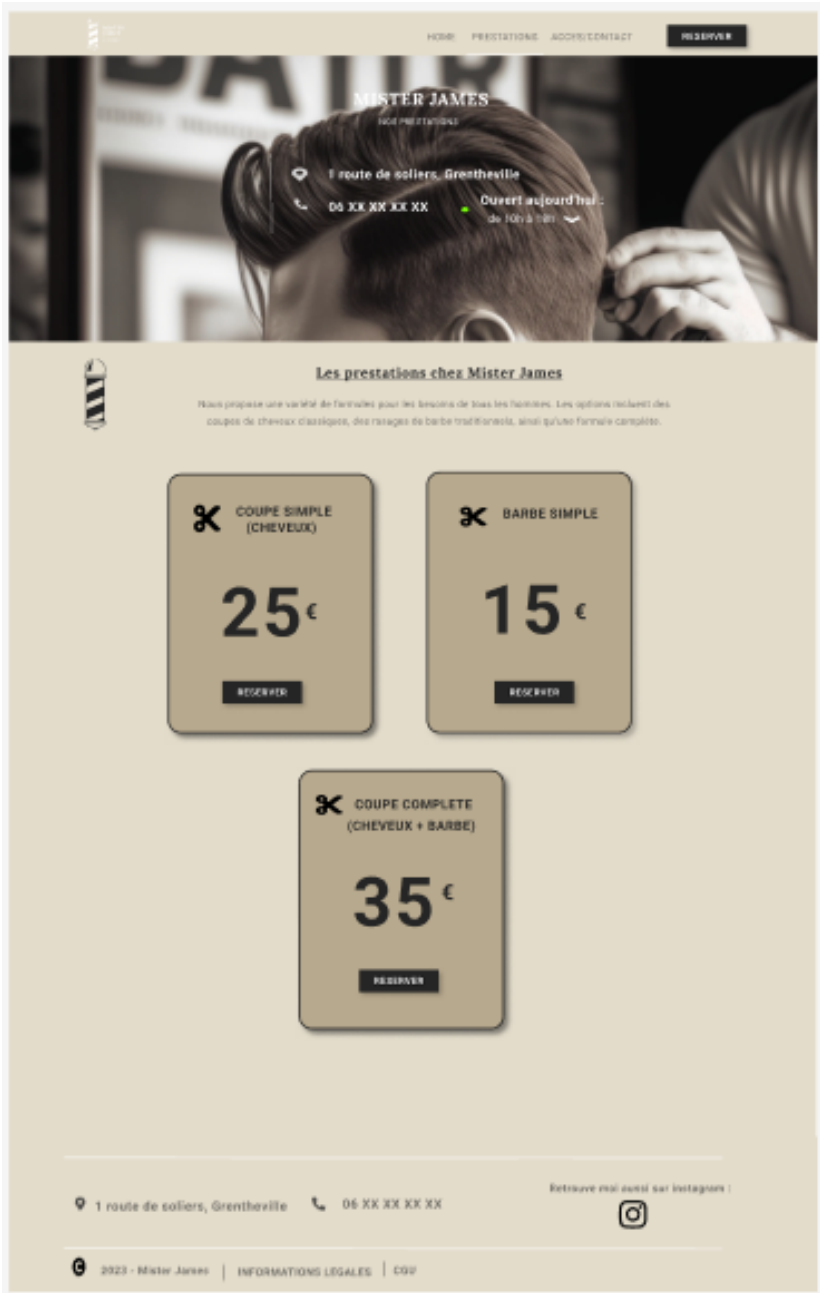
Je pense tout particulièrement à React et Tailwind qui sont des outils modernes et qui ont été les outils que j'ai le plus utilisés.

J'ai pu aussi m'améliorer sur des langages plus traditionnels tels que Php et Mysql.

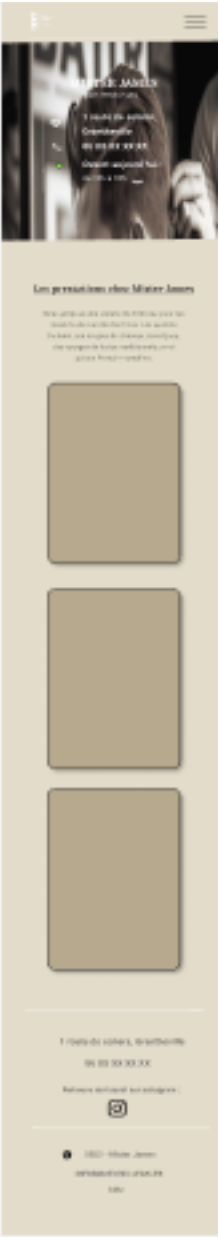
Enfin, j'ai pu développer des compétences secondaires avec l'utilisation de "Figma" pour le maquettage, de "git" pour le *versionning*, du travail en mode projet grâce à des outils comme Trello et grâce à la mise en place du cahier des charges.

7 - Annexes

PRESTATIONS DESKTOP



PRESTATIONS MOBILE



ACCES/CONTACT DESKTOP

HOME PRESTATIONS ACCES/CONTACT RESERVER

Normatrans Skatepark Stade de Gretheville Gretheville Pean Christophe Rte de Cormelles D230 Rte de Four

CONTACT

HORAIRES

LUNDI : FERME
MARDI : 9h-18h
MERCREDI : 9h-18h
JEUDI : 9h-18h
VENDREDI : 9h-18h
SAMEDI : 9h-18h
DIMANCHE : FERME

Ecris-moi

E-mail

Prenom Nom

MESSAGE

Envoyer

1 route de soliers, Gretheville 06 XX XX XX XX

Retrouve moi aussi sur Instagram :

2023 - Mister James | INFORMATIONS LEGALES | CGU

PRESTATIONS MOBILE

HOME PRESTATIONS ACCES/CONTACT RESERVER

Normatrans Skatepark Stade de Gretheville Gretheville Pean Christophe Rte de Cormelles D230 Rte de Four

HORAIRES

LUNDI : FERME
MARDI : 9h-18h
MERCREDI : 9h-18h
JEUDI : 9h-18h
VENDREDI : 9h-18h
SAMEDI : 9h-18h
DIMANCHE : FERME

Ecris-moi

Prenom Nom

Email

MESSAGE

Envoyer

1 route de soliers, Gretheville 06 XX XX XX XX

Retrouve moi aussi sur Instagram :

2023 - Mister James | INFORMATIONS LEGALES | CGU

Lien de la maquette global : <https://www.figma.com/file/ZzdvvNZcAGGHhFctk4QZJt/LEO-MAQUETTE?node-id=0%3A1&t=PVOImXT5EGLFx9IS-1>