15/03/2023

Dossier de projet

Titre Développeur web et web mobile 2023

Présenté par :
James Gnagne

Référent de formation :
Arièle Callay

Centre de formation :
VIA formation

Sommaire

Mon projet de développeur Web et Web Mobile

Mon intérêt pour le code a commencé grâce à mon DUT Génie Électrique et Informatique Industrielle, où j'ai découvert la programmation et ses applications dans le monde de l'industrie. Puis, après avoir effectué deux années en licence d'informatique, j'ai décidé de me lancer dans une formation de développeur web pour commencer à coder directement. J'ai rapidement été attiré par le développement web en raison de son aspect concret et de ses nombreuses applications dans le monde moderne.

Je suis convaincu que le développement web est un métier d'avenir, car l'informatique est de plus en plus omniprésente et de nombreuses entreprises ont besoin de sites web et d'applications pour leurs activités. En outre, les conditions de travail sont flexibles, ce qui me permettra de travailler à distance si nécessaire.

J'ai choisi de suivre une formation de développeur web et web mobile afin de renforcer mes compétences et de pouvoir travailler sur des projets concrets. Je suis confiant dans mes capacités à travailler en autonomie, à être curieux et persévérant, et à collaborer efficacement avec les autres membres d'une équipe.

À l'avenir, je suis impatient de travailler sur des projets passionnants et de contribuer à la création de sites et d'applications web innovants.

Présentation du projet en globalité

Mon projet s'est déroulé dans une école de conduite à Cagny. Cette entreprise dispense des cours de conduite et propose des formations pour obtenir les permis moto, voiture et remorque. L'objectif de mon projet était de créer un site web pour cette école de conduite.

La partie front-end permet aux visiteurs du site d'identifier l'entreprise, de consulter les horaires, de contacter l'entreprise, de la localiser et de découvrir les différentes formations proposées. Ce site vitrine permet aux personnes intéressées de trouver les informations dont elles ont besoin pour passer leur permis dans cette entreprise, tout en améliorant la visibilité de l'entreprise.

La partie back-end permet aux personnes ayant déjà passé leur permis dans cette école de conduite de créer un compte, de se connecter et de laisser un commentaire sur le site pour donner leur avis. Les utilisateurs peuvent également envoyer un e-mail à l'entreprise via un formulaire de contact. Enfin, le site permet aux utilisateurs de choisir des horaires de réservation directement en ligne via un formulaire. Ce formulaire envoie un message à l'entreprise par e-mail pour indiquer la date, l'heure et la raison de la réservation. L'entreprise répondra ensuite favorablement ou non à la demande.

Pour réaliser ce projet, j'ai d'abord rencontré ma tutrice de stage à plusieurs reprises pour déterminer la maquette et le cahier des charges. Ensuite, j'ai développé le site en utilisant les langages suivants : HTML, CSS, JS et PHP.

Cahier des charges

Lors de ma rencontre avec ma tutrice de stage, nous avons discuté du projet afin de savoir à l'avance ce que je devais réaliser. Nous avons donc tout d'abord élaboré un cahier des charges. Le voici :

Objectif du projet :

L'objectif de ce projet est de créer un site web pour l'école de conduite à Cagny afin de fournir aux visiteurs des informations sur l'entreprise, ses services et ses formations, et de permettre aux utilisateurs de laisser des commentaires, de contacter l'entreprise et de réserver des horaires en ligne.

Fonctionnalités du site :

-Page d'accueil et page de présentation avec présentation de l'entreprise et ses services et affichage des horaires d'ouverture de l'entreprise.

-Page Formations pour décrire les différents types de formations proposées.

-Page Agence avec un formulaire pour envoyer un e-mail à l'entreprise et un formulaire de réservation pour permettre aux utilisateurs de réserver des horaires de cours en ligne et une carte pour localiser l'entreprise.

-Page livre d'or pour permettre aux utilisateurs de se connecter et de laisser un commentaire sur leur expérience avec l'entreprise.

Contraintes techniques :

Le site doit être développé en utilisant les langages HTML, CSS, JS et PHP. Le site doit être compatible avec les principaux navigateurs web tels que Google Chrome et Safari, Mozilla Firefox, Opera.

Cahier des charges fonctionnel :

Le site doit permettre aux visiteurs de consulter les informations de l'entreprise, les horaires, les formations proposées et de les contacter via un formulaire. Les utilisateurs doivent pouvoir créer un compte, se connecter et laisser un commentaire sur le site. Le formulaire de réservation doit permettre

aux utilisateurs de choisir une date, une heure et la raison de la réservation. Une fois le formulaire rempli, un message doit être envoyé par l'entreprise par e-mail pour valider la demande.

Cahier des charges technique :

Le site doit être développé en utilisant les langages HTML, CSS, JS et PHP. La base de données doit être créée pour stocker les informations des utilisateurs et les commentaires. Les pages doivent être responsive et s'adapter à différents types d'appareils (ordinateurs, smartphones, tablettes).

Planning :

-Rencontre avec la tutrice de stage pour définir la maquette et le cahier des charges.

-Développement du site en utilisant les langages HTML, CSS, JS et PHP.

-Test, validation des fonctionnalités du site et vérifier la compatibilité du site avec les navigateurs.

-Correction des bugs et des erreurs.

-Mise en ligne du site.

Livrables :

-Code source du site web

-Explication technique pour réutilisation du site par l’entreprise

Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité : CP1 - Maquetter une application

Pour mon projet, j'ai utilisé Figma pour réaliser la maquette de la page d'accueil, de la présentation et de la page agence. Cette approche a permis de montrer à l'auto-école pour laquelle j'ai effectué mon stage à quoi ressemblerait le site et quelle serait sa couleur. Il n'y avait pas de contraintes strictes imposées pour la réalisation de ce projet.

J'ai décidé de ne pas inclure les pages de formation dans la maquette car elles étaient similaires aux pages principales du site et n'étaient pas nécessaires pour la présentation du projet à la tutrice. En revanche, j'ai réalisé les pages Accueil, Présentation et Agence en maquette pour les versions desktop, mobile et tablette.

Pour le site final, j'ai ajouté deux formulaires sur la page Agence qui seront utilisés ultérieurement pour la partie back-end. Comme il n'y avait pas de contraintes particulières concernant leur design, je n'ai pas inclus ces formulaires dans la maquette.

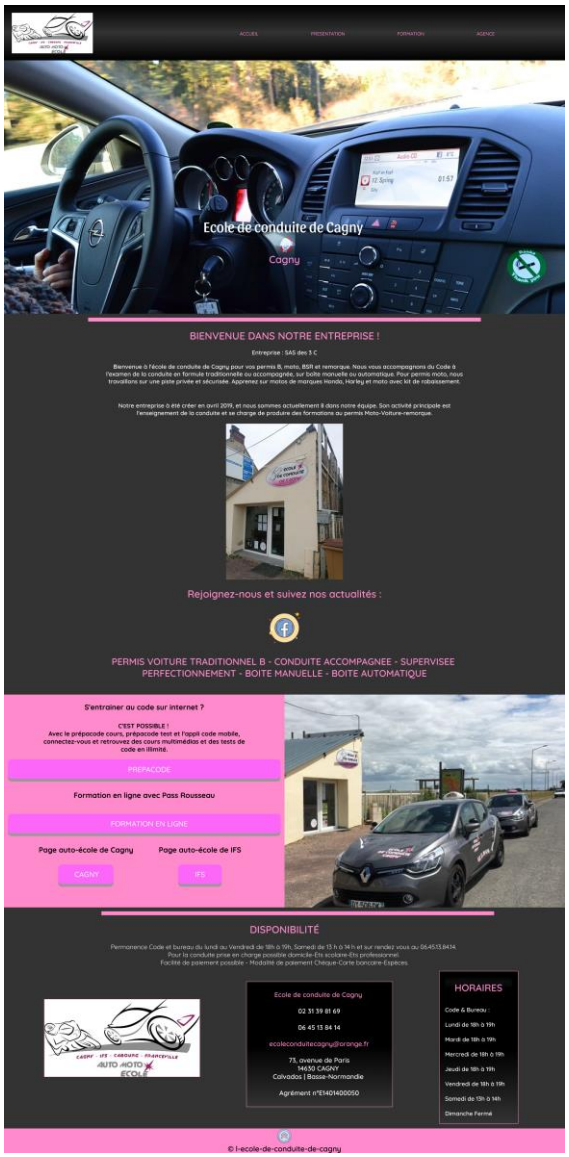
J'ai également ajouté une page Livre d'or qui n'était pas présente dans la maquette initiale. J'ai jugé qu'il n'était pas nécessaire de réaliser une maquette pour cette page car le design du site était déjà établi et la tutrice et moi-même étions d'accord sur celui-ci. De plus, le site était presque fini, donc faire des maquettes pour ces pages et ces fonctionnalités n'aurait pas permis de revenir en arrière. La page Livre d'or contient des boutons et des modales permettant aux utilisateurs de se connecter, de s'inscrire, d'envoyer des messages, etc.

L'utilisation de Figma est relativement simple : il suffit de créer des blocs et de les assembler afin de créer des composants. Il faut ensuite définir les tailles qui correspondent au site que vous voulez réaliser, choisir la bonne police et la bonne charte graphique. Il faut ensuite reproduire ces étapes pour les 3 tailles d'écran de base : j'ai réalisé 3 types d'écran principalement utilisés, à savoir mobile, tablette et desktop.

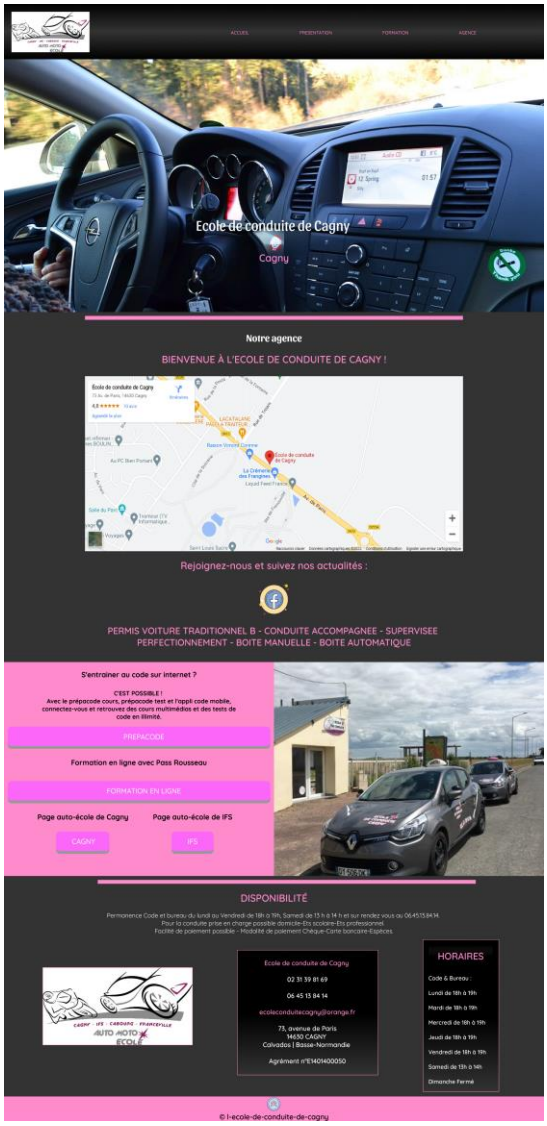
Je vais à présent montrer les différentes maquettes pour les 3 types d’écran :

Maquettes écran desktop (width : 1440), tablette (width : 768) et mobile (width : 450) pour la page Accueil :

Maquettes écran desktop (width : 1440), tablette (width : 768) et mobile (width : 450) pour la page Présentation :



Maquettes écran mobile (width : 1440), tablette (width : 768) et mobile (width : 450) pour la page Agence :



CP2 - Réaliser une interface utilisateur web statique et adaptable

L'objectif de cette partie est de mettre en avant la compétence en matière d'adaptabilité d'un site web sur mobile et tablette. Nous allons nous concentrer sur le code le plus significatif de la page d'accueil qui démontre cette compétence : le menu de navigation. Le code HTML et CSS montre comment le site s'adapte en fonction des différents supports, en particulier sur les petits écrans. Le contenu du menu de navigation est organisé de manière à ce qu'il soit facilement consultable et accessible sur tous les types d'appareils. Par exemple, la version mobile du menu de navigation est optimisée pour une utilisation à une main et utilise des icônes pour économiser de l'espace.

Pour montrer comment l'adaptabilité a été prise en compte lors de la conception de cette page web, nous allons explorer en détail le code. De plus, pour assurer la compatibilité avec les navigateurs, j'ai utilisé des autoprefixer CSS qui correspondent aux navigateurs.

J'ai utilisé le site https://autoprefixer.github.io/ pour ajouter les préfixes correspondants au navigateur que le site doit supporter. Ce site est très utile pour rendre le site web que l'on veut créer adaptable avec le plus de navigateurs possible.

Le code CSS utilise les préfixes CSS générés par Autoprefixer pour assurer la compatibilité avec les différents navigateurs. Le préfixe -ms- est utilisé pour les anciennes versions d'Internet Explorer, -webkit- pour Safari et Chrome, -moz- pour Firefox, et -o- pour Opera.

```
<header id="menu">
    <div class="logo">
        <a href="Accueil.html" target="_blank"><img src="../img/logo.jpg" alt="Accueil"></a>
    </div>
    <ul class="menu_li">
        <li class="lien"><a href="Accueil.html">accueil</a></li>
        <li class="lien"><a href="Présentation.html">présentation</a></li>
        <li class="lien formation">
            <a href="#">formation</a>
            <div class="dropdown">
                <a href="PermisB.html">Permis B</a>
                <a href="Accompagnee.html">Conduite accompagnée</a>
                <a href="FormationA2.html">Formation A2</a>
                <a href="FormationAM.html">Formation AM</a>
                <a href="Formation125.html">Formation 125</a>
                <a href="PasserelleA2_A.html">Passerelle A2 => A</a>
                <a href="Horsforfait.html">prestation Hors forfait</a>
            </div>
        </li>
        <li class="lien"><a href="Agence.php">agence</a></li>
        <li class="lien"><a href="Livre_d'or.php">Livre d'or</a></li>
    </ul>
    <div class="icon">
        <i class="fa fa-bars"></i>
    </div>
    <div class="menu_mobile">
        <a href="Accueil.html">accueil</a>
        <a href="Présentation.html">présentation</a>
        <a href="#" class="dropdown_formation">formation</a>
        <div class="dropdown_mobile">
            <a href="PermisB.html">
                <i class='fas fa-angle-right'></i>
                Permis B
            </a>
            <a href="Accompagnee.html">
                <i class='fas fa-angle-right'></i>
                Conduite accompagnée
            </a>
            <a href="FormationA2.html">
                <i class='fas fa-angle-right'></i>
                Formation A2
            </a>
            <a href="FormationAM.html">
                <i class='fas fa-angle-right'></i>
                Formation AM
            </a>
            <a href="Formation125.html">
                <i class='fas fa-angle-right'></i>
                Formation 125
            </a>
            <a href="PasserelleA2_A.html">
                <i class='fas fa-angle-right'></i>
                Passerelle A2 => A
            </a>
            <a href="Horsforfait.html">
                <i class='fas fa-angle-right'></i>
                prestation Hors forfait
            </a>
        </div>
        <a href="Agence.php">agence</a>
        <a href="Livre_d'or.php">Livre d'or</a>
    </div>
</header>
```

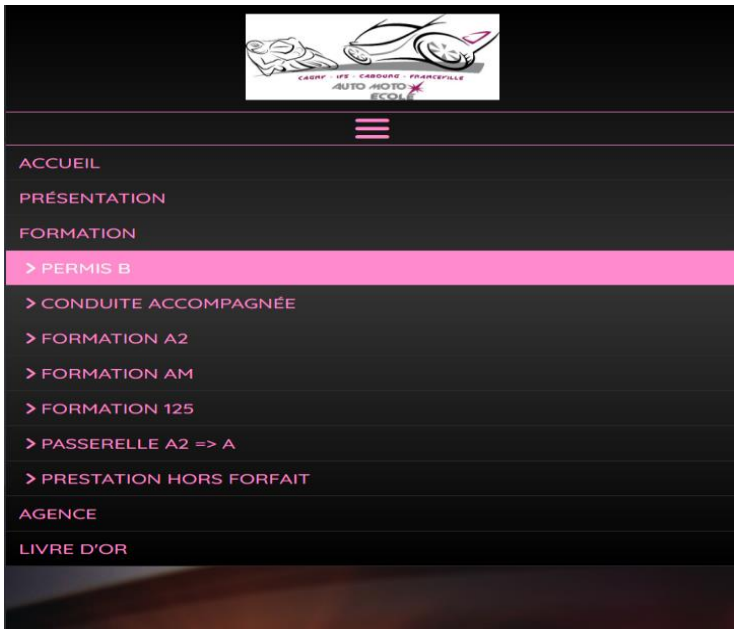
Ce code HTML correspond au menu de navigation du site web. Il est constitué d'un header avec un logo, une liste de liens et un bouton "hamburger" pour la version mobile.

Le menu de navigation utilise des classes CSS pour gérer l'affichage en fonction de la taille de l'écran. Par exemple, la classe "menu_li" est utilisée pour la version desktop et tablette ; la classe "menu_mobile" pour la version mobile.

Menu tablette :

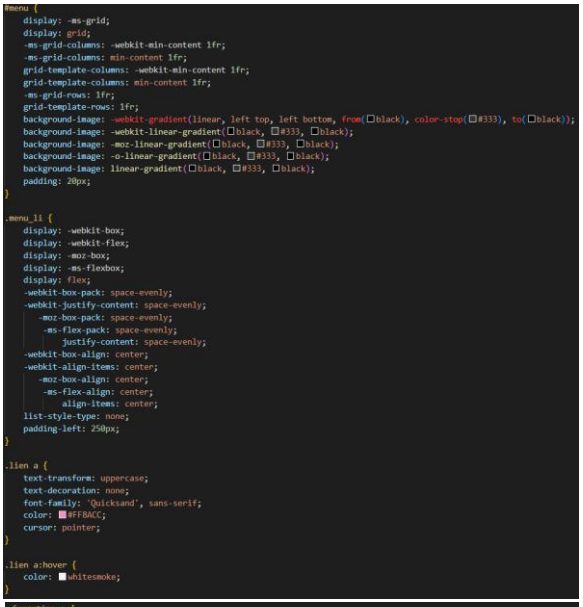
Pour la version mobile, les liens du menu sont affichés sous forme de liste verticale et le sous-menu "formation" est remplacé par un bouton qui ouvre un sous-menu déroulant. Les icônes "angle-right" sont utilisées pour indiquer les liens de sous-menu.

Menu mobile :



Pour créer un site web statique et adaptatif, il est nécessaire de coder la partie CSS en suivant une approche mobile first ou desktop first (c'est-à-dire qu'il faut coder d'abord le CSS pour la partie desktop, puis gérer les autres tailles d'écran avec les media queries). Pour ce projet, j'ai opté pour une approche desktop first. Une fois la partie desktop codée, il faut créer des media queries en CSS pour gérer les autres tailles d'écran. Pour cela, il suffit de définir une règle de media queries dans laquelle on place le code correspondant à la partie adaptée. Je vais d'abord expliquer comment j'ai réalisé la partie desktop, puis je détaillerai les autres parties réalisées avec les media queries.

Voici le code version desktop :



```
#menu {
    display: -ms-grid;
    display: grid;
    -ms-grid-columns: -webkit-min-content 1fr;
    -ms-grid-columns: min-content 1fr;
    grid-template-columns: -webkit-min-content 1fr;
    grid-template-columns: min-content 1fr;
    -ms-grid-rows: 1fr;
    grid-template-rows: 1fr;
    background-image: -webkit-gradient(linear, left top, left bottom, from(□black), color-stop(□#333), to(□black));
    background-image: -webkit-linear-gradient(□black, □#333, □black);
    background-image: -moz-linear-gradient(□black, □#333, □black);
    background-image: -o-linear-gradient(□black, □#333, □black);
    background-image: linear-gradient(□black, □#333, □black);
    padding: 20px;
}

.menu_li {
    display: -webkit-box;
    display: -webkit-flex;
    display: -moz-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-pack: space-evenly;
    -webkit-justify-content: space-evenly;
       -moz-box-pack: space-evenly;
        -ms-flex-pack: space-evenly;
            justify-content: space-evenly;
    -webkit-box-align: center;
    -webkit-align-items: center;
       -moz-box-align: center;
        -ms-flex-align: center;
            align-items: center;
    list-style-type: none;
    padding-left: 250px;
}

.lien a {
    text-transform: uppercase;
    text-decoration: none;
    font-family: 'Quicksand', sans-serif;
    color: ■#FFBACC;
    cursor: pointer;
}

.lien a:hover {
    color: ■whitesmoke;
}
```

```
.formation a {
    padding-bottom: 20px;

}

.formation:hover .dropdown {
    top: 100px;
    width: 250px;
    opacity: 1;
    visibility: visible;
    -webkit-transition: all .6s;
    -o-transition: all .6s;
    -moz-transition: all .6s;
    transition: all .6s;

}

.dropdown {
    position: absolute;
    top: 120px;
    width: 200px;
    opacity: 0;
    visibility: hidden;
    border-top: 3px solid ■#808080;
    background: -webkit-gradient(linear, left top, left bottom, from(□#000), color-stop(50%, □#000), to(□#333));
    background: -webkit-linear-gradient(top, □#000 0%, □#000 50%, □#333 100%);
    background: -moz-linear-gradient(top, □#000 0%, □#000 50%, □#333 100%);
    background: -o-linear-gradient(top, □#000 0%, □#000 50%, □#333 100%);
    background: linear-gradient(to bottom, □#000 0%, □#000 50%, □#333 100%);
    z-index: 1;

}

.dropdown a {
    color: ■#FFBACC;
    padding: 10px 0 10px 10px;
    text-decoration: none;
    display: block;
}

.dropdown a:hover {
    color: ■whitesmoke;
    background-color: ■#FFBACC;
    -webkit-transition: background-color 0.3s;
    -o-transition: background-color 0.3s;
    -moz-transition: background-color 0.3s;
    transition: background-color 0.3s;
}
```

```
.icon,
.menu_mobile,
.dropdown_mobile,
.menu_mobile a {
    display: none;
}
```

La première partie de code avec l'ID **#menu** définit la mise en page du menu principal. Elle utilise des propriétés telles que **display**, **grid-template-columns**, **grid-template-rows** et **background-image** pour positionner les éléments du menu et lui donner un style visuel.

La classe. menu_**li** définit les styles pour les éléments de la liste du menu. Elle utilise les propriétés **display**, **justify-content**, **align-items**, **list-style-type** et **padding-left** pour placer les éléments du menu dans une liste horizontale avec un espace uniforme entre eux.

Les classes. lien **a** et **.formation a** définissent les styles pour les liens dans le menu. Elles utilisent les propriétés **text-transform**, **text-decoration**, **font-family**, **color** et **cursor** pour donner un style visuel aux liens et pour définir la façon dont ils se comportent lorsque l'utilisateur interagit avec eux (par exemple, en changeant de couleur au survol).

La classe **.dropdown** définit les styles pour le menu déroulant qui s'affiche lorsque l'utilisateur survole l'élément "Formation" du menu principal. Elle utilise les propriétés **position**, **top**, **width**, **opacity**, **visibility**, **border-top** et **background** pour positionner et styliser le menu déroulant. Les propriétés **transition** sont également utilisées pour animer la transition entre l'état caché et l'état affiché du menu déroulant.

La dernière partie du code, avec les sélecteurs : **.icon**, **.menu_mobile**, **.dropdown_mobile** et **.menu_mobile a**, utilise la propriété display pour cacher les éléments du menu hamburger, de sorte que seul le menu principal soit visible.

Voici le code version tablette :

```css
@media screen and (max-width:920px) {
    #menu {
        display: -webkit-box;
        display: -webkit-flex;
        display: -moz-box;
        display: -ms-flexbox;
        display: flex;
        -webkit-box-orient: vertical;
        -webkit-box-direction: normal;
        -webkit-flex-direction: column;
           -moz-box-orient: vertical;
           -moz-box-direction: normal;
            -ms-flex-direction: column;
                flex-direction: column;
        padding: 0;
    }

    .logo {
        -webkit-align-self: center;
            -ms-flex-item-align: center;
                -ms-grid-row-align: center;
            align-self: center;
        padding: 10px 0;
    }

    .menu_li {
        margin: 0;
        padding: 10px 0;
        border-top: 1px #FF8ACC solid;
        border-bottom: 1px #FF8ACC solid;
    }

    .dropdown {
        top: 185px;
    }

    .formation:hover .dropdown {
        top: 165px;
    }
}
```

Les propriétés CSS utilisées permettent de changer la disposition des éléments du menu et de les adapter à une tablette.

Le conteneur #menu est défini comme une boîte flexible avec une orientation verticale. Cela permet de placer les éléments du menu les uns au-dessus des autres. La classe **.logo** est centrée horizontalement dans le conteneur et les éléments de la liste du menu ont une marge et un rembourrage pour les espacer.

La classe **.dropdown** est utilisée pour afficher les sous-menus lorsque l'utilisateur survole les éléments du menu. En règle générale, les sous-menus sont affichés en dessous de l'élément parent, mais ici la propriété **top** est utilisée pour ajuster leur positionnement en fonction de l'espace disponible sur l'écran.

La classe **.formation:hover .dropdown** est utilisée pour ajuster la position des sous-menus lorsque l'utilisateur survole l'élément "Formation" du menu. Cette règle permet de compenser la hauteur de l'élément parent et d'afficher le sous-menu de manière appropriée.

Voici le code version mobile :

```css
@media screen and (max-width:600px) {
    .menu_li {
        display: none;
    }

    .icon {
        display: block;
        border-top: 1px #FF8ACC solid;
        border-bottom: 1px #FF8ACC solid;
        font-size: 30px;
        color: #FF8ACC;
        text-align: center;
        cursor: pointer;
    }

    .menu_mobile a {
        display: block;
        text-decoration: none;
        text-transform: uppercase;
        font-family: 'Quicksand', sans-serif;
        color: #FF8ACC;
        padding: 10px 0 10px 10px;
        border-bottom: 1px solid #303030;
    }

    .menu_mobile .dropdown_mobile a {
        padding-left: 15px;
    }

    .menu_mobile a:hover {
        color: whitesmoke;
        background-color: #FF8ACC;
        -webkit-transition: background-color 0.3s;
        -o-transition: background-color 0.3s;
        -moz-transition: background-color 0.3s;
        transition: background-color 0.3s;
    }
}
```

Ce code correspond à une media query pour les écrans ayant une largeur maximale de 600 pixels. Il modifie l'apparence du menu pour qu'il soit plus adapté aux petits écrans.

Les éléments de la liste du menu sont cachés avec la propriété **display: none** afin qu'ils n'apparaissent pas dans le menu mobile. Le bouton hamburger est affiché en utilisant la propriété **display: block**, avec une bordure supérieure et inférieure pour séparer visuellement le bouton du reste de la page.

Les liens dans le menu mobile sont affichés en bloc et sans liste, avec une police de caractères en majuscules, une couleur de texte rose et un padding de 10 pixels en haut et en bas et de 10 pixels à gauche. La propriété "**border-bottom**" est utilisée pour créer une bordure en dessous de chaque lien.

Le code ajoute également une marge à gauche aux liens des sous-menus classe **dropdown_mobile**, en modifiant la propriété **padding-left**.

Lorsqu'un lien du menu mobile est survolé, la couleur de fond change en rose plus clair avec une transition de 0,3 secondes.

CP3 - Développer une interface utilisateur web dynamique

Je vais démontrer dans cette partie en quoi mon site web est dynamique en énonçant toutes les fonctionnalités qui le rendent dynamique.

```
let slideIndex = 0;
let slideIndex2 = 0;
showSlides();
showSlides2();

function showSlides() {
  let i;
  let slides = document.getElementsByClassName("slide");
  for (i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
  }
  slideIndex++;
  if (slideIndex > slides.length) { slideIndex = 1 }
  slides[slideIndex - 1].style.display = "block";
  setTimeout(showSlides, 3000);
}

function showSlides2() {
  let i;
  let slides2 = document.getElementsByClassName("slide2");
  for (i = 0; i < slides2.length; i++) {
    slides2[i].style.display = "none";
  }
  slideIndex2++;
  if (slideIndex2 > slides2.length) { slideIndex2 = 1 }
  slides2[slideIndex2 - 1].style.display = "block";
  setTimeout(showSlides2, 3000);
}
```

Le code JavaScript fourni définit deux variables, slideIndex et slideIndex2, à 0, puis appelle deux fonctions, showSlides() et showSlides2().

La fonction showSlides() commence par récupérer tous les éléments ayant la classe "slide" en utilisant la méthode getElementsByClassName(). Ensuite, elle boucle à travers chaque élément et leur applique la propriété CSS "display: none", ce qui les cache tous. La variable slideIndex est ensuite incrémentée de 1, puis une vérification est effectuée pour s'assurer qu'elle n'est pas supérieure à la longueur de la liste d'éléments récupérés. Si c'est le cas, la variable slideIndex est réinitialisée à 1. Enfin, la diapositive actuelle est affichée en définissant le style CSS "display: block", tandis que la fonction setTimeout() est utilisée pour rappeler la fonction showSlides() après 3 secondes, afin de passer à la diapositive suivante.

La fonction showSlides2() est similaire à showSlides(), mais elle récupère les éléments ayant la classe "slide2", et incrémente la variable slideIndex2 à la place. Elle utilise également setTimeout() pour rappeler la fonction après 3 secondes.

En résumé, ces fonctions permettent de faire défiler automatiquement les diapositives de deux carrousels différents.

```
function affiche(x, y) {
  x.addEventListener('click', () => {

    if (y.style.display === "block") {
      y.style.display = "none";
    } else {
      y.style.display = "block";
    }

  })
}

let formation = document.querySelector(".formation");
let dropdown = document.querySelector(".dropdown");

let burger = document.querySelector(".icon");
let menu = document.querySelector(".menu_mobile");

let formation_mobile = document.querySelector(".dropdown_formation");
let dropdown_mobile = document.querySelector(".dropdown_mobile");
// clique lien formation : affiche sous menu dropdown
affiche(formation, dropdown);
// clique burger : affiche menu-mobile
affiche(burger, menu);
// clique lien menu : affiche sous menu dropdown_mobile
affiche(formation_mobile, dropdown_mobile);
```

Ce code définit une fonction "affiche" qui prend deux arguments "x" et "y", qui représentent deux éléments HTML. La fonction ajoute un écouteur d'événement sur l'élément "x" pour écouter les clics. Lorsque l'élément est cliqué, la fonction vérifie si l'élément "y" est actuellement affiché en regardant sa propriété "display". Si "y" est actuellement affiché, la fonction le cache en définissant son style d'affichage sur "none", sinon, elle l'affiche en définissant son style d'affichage sur "block".

Le reste du code récupère des éléments HTML spécifiques en utilisant la méthode querySelector, puis utilise la fonction "affiche" pour ajouter des écouteurs d'événements pour afficher ou masquer les éléments correspondants. Plus précisément, lorsque l'élément "formation" est cliqué, la fonction affiche l'élément "dropdown". Lorsque l'icône "burger" est cliquée, la fonction affiche l'élément "menu_mobile". Enfin, lorsque l'élément "formation_mobile" est cliqué, la fonction affiche l'élément "dropdown_mobile".

```
var acc = document.getElementsByClassName("accordion");
var i;

for (i = 0; i < acc.length; i++) {
  acc[i].addEventListener("click", function () {
    this.classList.toggle("active");
    var panel = this.nextElementSibling;
    if (panel.style.maxHeight) {
      panel.style.maxHeight = null;
    } else {
      panel.style.maxHeight = panel.scrollHeight + "px";
    }
  });
}
```

Cette partie de code définit un accordéon (accordion) qui permet d'ouvrir et de fermer des sections dans une page web. Elle commence par récupérer tous les éléments ayant la classe "accordion" en utilisant la méthode getElementsByClassName. Ensuite, pour chaque élément "accordion", la fonction addEventListener est appelée avec le type d'événement "click" et une fonction de rappel qui s'exécute lorsqu'un clic se produit sur l'élément.

La fonction de rappel commence par utiliser la méthode toggle pour ajouter ou supprimer la classe "active" de l'élément "accordion" cliqué. Ensuite, elle récupère l'élément suivant immédiat (nextElementSibling), qui doit être la section à ouvrir ou fermer, et vérifie si la propriété maxHeight de cette section est définie. Si c'est le cas, cela signifie que la section est actuellement ouverte, donc la fonction la ferme en définissant maxHeight sur null. Sinon, la section est fermée, donc la fonction l'ouvre en définissant maxHeight sur la hauteur réelle de la section (en utilisant la propriété scrollHeight) et en ajoutant "px" à la fin. Cela permet de s'assurer que la section s'ouvre complètement, quel que soit le contenu qu'elle contient.

```
const chevron = document.querySelector(".chevron-up");
chevron.addEventListener('click', () => {
  window.scrollTo({
    top: 0,
    left: 0,
    behavior: "smooth"
  })
})
```

Ce code JS permet de faire défiler la page vers le haut lorsqu'un utilisateur clique sur un élément ayant la classe "chevron-up".

Tout d'abord, la constante "chevron" est définie en utilisant la méthode querySelector pour sélectionner le premier élément ayant la classe "chevron-up".

Ensuite, un écouteur d'événement est attaché à l'élément "chevron" en utilisant la méthode addEventListener avec le type d'événement "click" et une fonction de rappel anonyme qui s'exécute lorsqu'un clic se produit sur l'élément.

La fonction de rappel utilise la méthode scrollTo pour faire défiler la page vers le haut. Les propriétés top et left sont définies à 0 pour se positionner tout en haut de la page, et la propriété behavior est définie sur "smooth" pour permettre un défilement en douceur.

En résumé, ce code permet de créer un bouton de retour en haut de page pour faciliter la navigation sur une page web. Lorsque l'utilisateur clique sur ce bouton, la page défile de manière fluide vers le haut de la page.

La page livre d'or.php présente plusieurs fonctionnalités dynamiques. On peut y ajouter, supprimer ou modifier des messages en cliquant sur des boutons, dont certains ouvrent des modals qui disparaissent lorsqu'on clique sur la croix. De plus, les données des messages sont récupérées depuis la base de données en utilisant du JSON et AJAX, ce qui permet une lecture dynamique de ces données sans avoir à recharger la page.

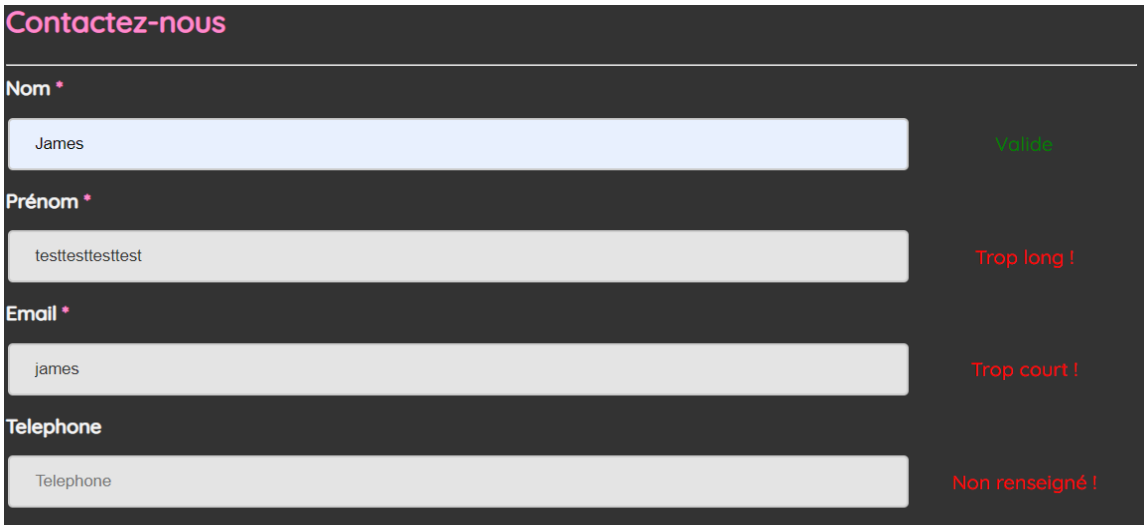

```
function openCreateMessage() {
  document.getElementById("modal8").style.display = "flex";
}
function closeCreateMessage() {
  document.getElementById("modal8").style.display = "none";
}
```

La page agence.php présente aussi plusieurs fonctionnalités dynamiques lors de la complétion de formulaire et de l'envoi.

```
function verifChaine(type) {
    // On vérifier la longueur de la chaine de caractère renseignée
    let chaine = document.getElementById(type).value;

    if (chaine == "") { // Si le champs n'est pas renseigné, on affiche un message d'erreur en rouge
        document.getElementById('span_' + type).innerHTML = "Non renseigné !";
        document.getElementById('span_' + type).style.color = "red";
    } else if (chaine.length < 3) { // Si la taille est inférieure à 3 caractères, on affiche un message d'erreur en rouge
        document.getElementById('span_' + type).innerHTML = "Trop court !";
        document.getElementById('span_' + type).style.color = "red";
        return false;   // on retourne "false" pour dire que le formulaire n'est pas valide
    } else if (chaine.length > 12) { // Si la taille est supérieure à 12 caractères, on affiche un message d'erreur en rouge
        document.getElementById('span_' + type).innerHTML = "Trop long !";
        document.getElementById('span_' + type).style.color = "red";
        return false;   // on retourne "false" pour dire que le formulaire n'est pas valide
    } else { // La taille du champ est entre 3 et 12 caractères, on affiche un message en vert
        document.getElementById('span_' + type).innerHTML = "Valide";
        document.getElementById('span_' + type).style.color = "green";
        return true;    // on retourne "true" pour dire que le formulaire est valide
    }
}
document.getElementById('name').addEventListener('keyup', function () {
    verifChaine('name');
});
```

Cette fonction permet de tester la longueur des chaines taper en temps réel.

Exemple de saisie :



Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité
CP5 - Créer une base de données

J'ai créé une base de données nommée "auto-ecole" qui avait pour objectif de gérer les données des utilisateurs pour la création de compte, la suppression, la modification et permettre à l'utilisateur connecté d'envoyer, supprimer, modifier et lire les messages. Tout d'abord, j'ai créé la base de données, puis les tables correspondant aux utilisateurs et aux messages (ces deux tables sont liées entre elles).

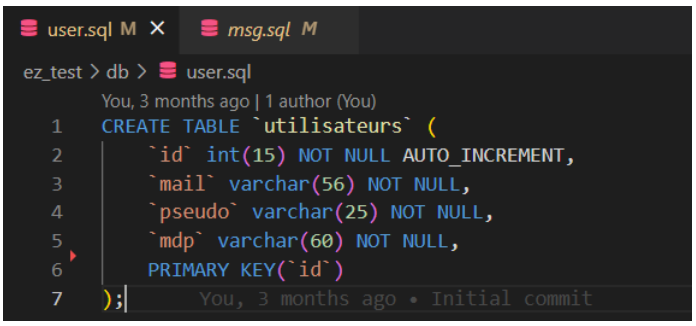
Pour créer la base de données, j'ai lancé mon serveur local (dans mon cas, WAMP) puis j'ai ouvert mon navigateur et saisi l'URL suivante dans la barre d'adresse : http://localhost/phpmyadmin/. La page d'accueil de PHPMyAdmin est alors apparue :

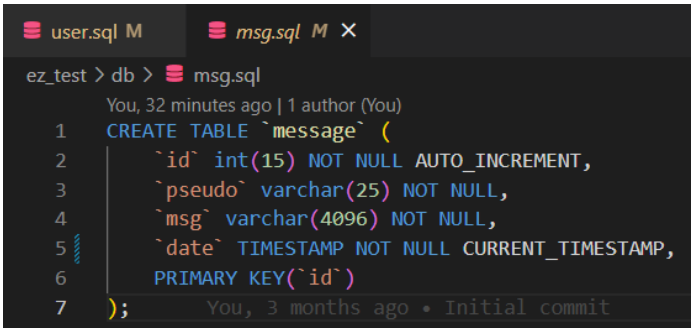
Une fois la connexion établie, je suis redirigé vers la page d'accueil de PHPMyAdmin.

Je clique sur le bouton "Nouvelle base de données" dans le menu de gauche de PhpMyAdmin, puis dans le champ "Nom de la base de données", j'entre le nom de ma nouvelle base de données.

Ensuite, je clique sur le bouton "Créer" pour créer la base de données. Maintenant, je peux commencer à créer des tables et à ajouter des données à ma base de données.

J'ai créé ensuite les tables "utilisateur" et "message" dans un fichier SQL. Ces deux tables sont liées entre elles : dans la table "message", il y a une colonne "pseudo" qui correspond au pseudo de la personne qui a écrit le message. Lorsqu'un utilisateur voudra écrire un message, son pseudo sera entré dans la table, ce qui créera un lien entre les deux tables. Ce lien pourra ensuite être utilisé, par exemple, pour supprimer ou modifier le message.



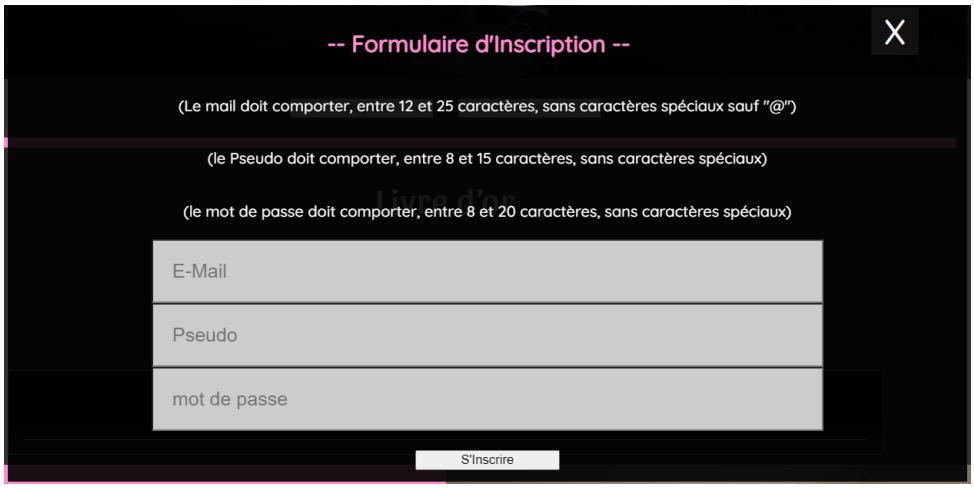


Une fois que le code des tables est prêt, il faut importer les fichiers dans la base de données.

CP6 - Développer les composants d'accès aux données
Et
CP7 - Développer la partie back-end d'une application web ou web mobile

La première chose à faire lorsque l'on se retrouve sur la page livre d'or est de s'inscrire en appuyant sur le bouton inscription.

Lorsque cette action est faite, un formulaire d'inscription s'affiche (c'est un modal)




```
<!-- Modal d'inscription -->
    <div class="modal" id="modal1">
        <div class="close"><span class="closeBtn" onclick="closeInscription()">X</span></div>
        <span class="h2bis text-center">-- Formulaire d'Inscription --</span>
        <p class="text-center">(Le mail doit comporter, entre 12 et 25 caractères, sans caractères spéciaux sauf "@")</p>
        <p class="text-center">(le Pseudo doit comporter, entre 8 et 15 caractères, sans caractères spéciaux)</p>
        <p class="text-center">(le mot de passe doit comporter, entre 8 et 20 caractères, sans caractères spéciaux)</p>
        <form class="form" action="../../utilisateur/create.php" method="post">
            <input class="pseudoTxtarea" type="text" placeholder="E-Mail" name="mail" id="mail">
            <input class="pseudoTxtarea" type="text" placeholder="Pseudo" name="pseudo" id="pseudo">
            <input class="pseudoTxtarea" type="password" placeholder="mot de passe" name="mdp" id="mdp">
            <div class="text-center">
                <input type="submit" value="S'Inscrire" class="pseudoBtn">
            </div>
        </form>
    </div>
```

Ce code correspond à un formulaire d'inscription qui est affiché dans une fenêtre modale (id="modal1"). Le formulaire contient des champs pour que

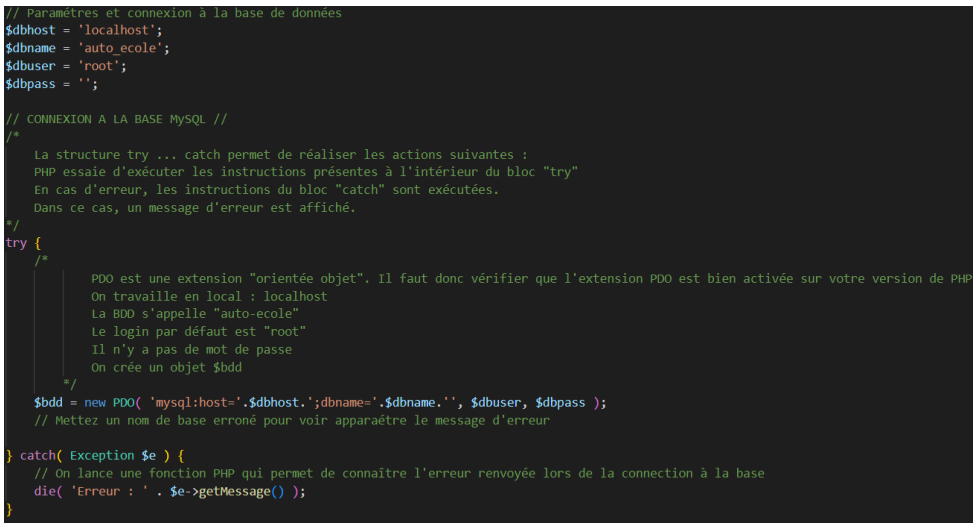
l'utilisateur entre son adresse e-mail (name="mail"), son pseudo (name="pseudo"), et son mot de passe (name="mdp").

Il y a également un bouton de soumission (type="submit") qui envoie les données du formulaire à un fichier de traitement (action="../../utilisateur/create.php") lorsque l'utilisateur clique dessus.

Le formulaire est conçu pour être utilisé avec la méthode POST (method="post"). Lorsque l'utilisateur soumet le formulaire, les données entrées dans les champs du formulaire sont envoyées sous forme de variables HTTP POST au fichier de traitement "create.php" situé dans le répertoire "utilisateur" à l'intérieur du répertoire parent "....".

Dans la partie back-end, pour récupérer la valeur de l'input correspondant au name "mail", il faut utiliser la superglobale $_POST['mail']. Cela permet de récupérer les données envoyées par le formulaire après la soumission. Les données sont envoyées au format clé-valeur, où la clé correspond au name de l'input et la valeur correspond à la valeur saisie par l'utilisateur dans l'input.

La première chose à faire pour créer un utilisateur dans la partie back-end est de se connecter à la base de données.



```php
// Paramètres et connexion à la base de données
$dbhost = 'localhost';
$dbname = 'auto_ecole';
$dbuser = 'root';
$dbpass = '';

// CONNEXION A LA BASE MySQL //
/*
    La structure try ... catch permet de réaliser les actions suivantes :
    PHP essaie d'exécuter les instructions présentes à l'intérieur du bloc "try"
    En cas d'erreur, les instructions du bloc "catch" sont exécutées.
    Dans ce cas, un message d'erreur est affiché.
*/
try {
    /*
            PDO est une extension "orientée objet". Il faut donc vérifier que l'extension PDO est bien activée sur votre version de PHP
            On travaille en local : localhost
            La BDD s'appelle "auto-ecole"
            Le login par défaut est "root"
            Il n'y a pas de mot de passe
            On crée un objet $bdd
        */
    $bdd = new PDO( 'mysql:host='.$dbhost.';dbname='.$dbname.'', $dbuser, $dbpass );
    // Mettez un nom de base erroné pour voir apparaétre le message d'erreur

} catch( Exception $e ) {
    // On lance une fonction PHP qui permet de connaître l'erreur renvoyée lors de la connection à la base
    die( 'Erreur : ' . $e->getMessage() );
}
```

Ce code permet de se connecter à la base de données "auto_ecole" en utilisant l'extension PDO de PHP.

Tout d'abord, les paramètres de connexion sont définis : $dbhost correspond à l'hôte (localhost), $dbname correspond au nom de la base de données (auto_ecole), $dbuser correspond à l'utilisateur (root) et $dbpass correspond au mot de passe (vide dans ce cas).

Ensuite, la connexion à la base de données est effectuée en créant un objet PDO et en passant les paramètres de connexion en argument. Si une erreur survient lors de la connexion, elle est capturée et affichée à l'utilisateur grâce à la fonction die().

Une fois la connexion établie, on peut effectuer des requêtes SQL sur la base de données en utilisant l'objet $bdd.

Avant d'exécuter des requêtes SQL sur la base de données, il est crucial de sécuriser les données entrées par l'utilisateur dans les champs de formulaire pour éviter toute injection de code malveillant. Voici un exemple de code pour vérifier la valeur saisie dans l'input qui correspond au "pseudo".

```php
// On initialise une variable dans laquelle nous allons stocker des retours PHP
$retour = '';
// On initialise une variable booléenne qui nous permettra de savoir si une erreur a été rencontrée
$erreur = false;
```

```php
if(isset($_POST['pseudo']) && $_POST['pseudo'] != ''){
    $longueur_chaine = strlen($_POST['pseudo']);
    if($longueur_chaine < 8 || $longueur_chaine > 15){
        $erreur = true;
        $retour .= 'Le pseudo doit être composé de 8 caractères minimum et ne doit pas dépasser 15 caractères.<br>';
    }
    $exp = "/[a-zA-Z0-9]/";
    if(!preg_match($exp, $_POST['pseudo'])){
        $erreur = true;
        $retour .= 'Le pseudo saisie n\'est pas valide : il doit être composé que de chiffres et de lettre.<br>';
    }
    if (!checkConsecutiveChars($_POST['pseudo'])) {
        $erreur = true;
        $retour .= 'Le pseudo contient des caractères consécutifs.<br>';
    }
    if (!checkAlternateChars($_POST['pseudo'])) {
        $erreur = true;
        $retour .= 'Le pseudo contient des caractères alternatifs.<br>';
    }
} else {
    $erreur = true;
    $retour .= "Veuillez renseigner le champ 'pseudo'.<br />";
}
```

(Un code similaire existe pour les autres valeurs saisies : mail et mdp).

Pour sécuriser les données, je test :

-L'existence du champ concerné.

-la longueur de la chaîne avec le premier test de longueur (entre 8 et 15 caractères).

-les caractères utilisés avec la fonction preg_match() qui vérifie la composition de la valeur saisie.

-empêche la répétition de caractères comme, par exemple : « zzzz » ou « ezez » avec la fonction « checkConsecutiveChars() » et « checkAlternateChars() ».

Si l'une des conditions n’est pas remplie, la variable $erreur est définie à « true ».

```php
if(!$erreur){
    $pseudo = htmlentities(secure_donnee(trim($_POST['pseudo'])));
    //mdp
    $mdp = htmlentities(secure_donnee(trim($_POST['mdp'])));
    // hash
    $hash = password_hash($mdp, PASSWORD_DEFAULT);
    $mail = htmlentities(secure_donnee(trim($_POST['mail'])));
    $query = $bdd->prepare( 'SELECT * FROM utilisateurs WHERE pseudo = :pseudo' );
    $query->bindParam( ':pseudo', $pseudo );
    $query->execute();
    foreach ( $query->fetchAll( PDO::FETCH_ASSOC ) as $row ) {
        $retour .= 'La référence saisie est déjà utilisée !<br>';
        $erreur = true;
        break;
    }
    $query->closeCursor();

    if ( !$erreur ) {
        $sql = "INSERT INTO utilisateurs VALUES(NULL,:mail,:pseudo,:mdp)";
        $requete = $bdd->prepare( $sql );
        $requete->bindParam( ':mail',  $mail );      You, 3 months ago • Initial commit …
        $requete->bindParam( ':pseudo', $pseudo );
        $requete->bindParam( ':mdp', $hash );
        if ( $requete->execute() ) {
            $retour .= 'L\'utilisateur a été ajouté avec succès.<br>';
        } else {
            $retour .= 'Une erreur est apparue lors de l\'ajout de l\'utilisateur.<br>';
        }
        $requete->closeCursor();
    }
}
```

La fonction htmlentities() permet d'échapper les caractères spéciaux en les transformant en entités HTML, pour éviter les attaques de type XSS (injection de code).

La fonction secure_donnee vise à sécuriser les données en ajoutant des caractères d'échappement aux apostrophes, guillemets, antislashs et autres caractères spéciaux dans une chaîne de caractères. Cela peut aider à prévenir les attaques d'injection SQL ou d'injection de code, en évitant que des données malveillantes soient interprétées par le serveur comme du code à exécuter.

La fonction trim() permet de supprimer les espaces en début et en fin de chaîne pour éviter les erreurs liées aux saisies accidentelles d'espaces.

Enfin, une fois que tout ceci est fait, il est possible d'exécuter des requêtes SQL sur la base de données. On test d’abord si l’utilisateur n’existe pas avant de l’insérer dans la base de données.

Ce code prépare une requête SQL pour insérer un nouvel utilisateur dans une table "utilisateurs" en utilisant des paramètres nommés (:mail, :pseudo, :mdp) pour éviter les injections SQL.

Ensuite, la fonction bindParam() est utilisée pour associer les valeurs des paramètres nommés avec des variables PHP. Cette étape est importante pour s'assurer que les valeurs entrées par l'utilisateur sont correctement sécurisées et encodées.

La fonction execute() est appelée pour exécuter la requête préparée. Si l'exécution de la requête est réussie, un message de confirmation est renvoyé. Sinon, un message d'erreur est renvoyé.

Enfin, la méthode closeCursor() permet de libérer la connexion de la requête préparée. Cela permet notamment de libérer des ressources sur le serveur de base de données en mettant fin à la requête préparée, ce qui peut être utile lorsque plusieurs requêtes sont exécutées de manière répétitive.

En effet, après avoir exécuté une requête préparée, il est recommandé de fermer le curseur pour libérer les ressources utilisées par cette requête, afin de prévenir d'éventuelles erreurs.

```php
if ( $retour != '' ) {
    session_start();
    $_SESSION['retour'] = $retour;
}
```

Si la variable de retour n'est pas vide on renvoi la valeur de la variable sur la page livre d'or grâce à la déclaration d'une variable de session.

```php
<?php if (isset($_SESSION) && isset($_SESSION['retour'])): ?>
    <div class="alert alert-retour">
        <?= $_SESSION['retour'];?>
    </div>
<?php unset($_SESSION['retour']); endif;?>
```

Pour se connecter, la logique est la même que précédemment : l'utilisateur appuie sur le bouton de connection, remplit le formulaire, puis envoie celui-ci. À ce moment-là, il y a une connexion à la base de données suivie d'une sécurisation des données, d'une définition de toutes les fonctions qui seront utilisées, et enfin d'une interrogation de la base de données pour déterminer si l'utilisateur existe et peut se connecter.

Cette page s'appelle "fonction.php" et utilise la fonction suivante pour conserver les variables de session d'une page à l'autre.

```php
//on démarre une session ou la récupére si elle est déjà démarré
session_start();
```

On teste s'il existe un utilisateur qui correspond au pseudo et au mot de passe que l'on a saisis.

```php
// Si les données reçues sont valides, on va les sécuriser en s'aidant de notre fonction créée au début
if(!$erreur && isset($_POST['pseudo1'],$_POST['mdp1'])){
    //on enlève les espaces avec trim et on interprête pas les balises html s'il y en a avec htmlentities
    $pseudo = htmlentities(secure_donnee(trim($_POST[ 'pseudo1' ])));
    $mdp = htmlentities(secure_donnee(trim($_POST[ 'mdp1' ])));

    // On vérifie que le pseudo existe en base de données
    $query = $bdd->prepare( 'SELECT * FROM utilisateurs WHERE pseudo=:pseudo1' );
    $query->bindParam( ':pseudo1', $pseudo);
    $query->execute();
    $row = $query->fetchAll(PDO::FETCH_ASSOC);
    if(!count($row)){
        // S'il n'y a pas de résultat...
        $retour .= 'L\'utilisateur avec ce pseudo n\'existe pas.<br>';
    }else{
        //on récupère le hash correspondant au besoin pseudo associé en base de données
        $hash = $row[0]['mdp'];
        //on effectue un test de verification de mot de passe : celui que l'on à rentrer avec le hash enregistré en base de données
        if (password_verify($mdp, $hash)) {
            //on appel la fonction setConnected qui permet de connecter l'utilisateur avec en paramètres les valeurs des inputs
            //pseudo et mdp du form de connexion stocké pour l'instant grâce à la varible globale $_POST
            //puis stocké dans les variables $pseudo et $mdp
            setConnected($pseudo, $hash);
        } else {
            $retour .= 'L\'utilisateur avec ce mot de passe n\'existe pas.<br>';
        }
    }
}
```

La fonction setConnected() permet de créer et définir les variables de session une fois que toutes les vérifications sont effectuées.

```php
//fonction permettant de connecter l'utilisateur en passant par la session
function setConnected($loginPostForm, $passwordPostForm) {
    //si la variable  session existe et est non nul
    if (isset($_SESSION)) {
        //on stocke dans les variables de sessions login et mdp les valeurs des variables en paramètre login et mot de passe
        //(qui seront celles renseignés par l'utilisateur dans le form envoyé avec la méthode POST d'où le nom des variables)
        $_SESSION['loginPostForm'] = $loginPostForm;
        $_SESSION['passwordPostForm'] = $passwordPostForm;
    }
}
```

Le code de cette page (fonction.php) est inclus dans la page livre d'or avec l'instruction "include 'fonction.php';" pour pouvoir utiliser les fonctions qui s'y trouvent dans la page livre d'or.

```php
<?php
include_once("function.php")
?>
```

La fonction isConnect() vérifie si les variables de session existent et renvoie "true" si l'utilisateur est connecté.

```php
//fonction permettant de vérifier si un utilisateur a une session active, s'il est connecté donc
function isConnect() {
    //si une session est déclarer et non nul et si les variables de sessions login et mdp sont déclaré et non nul également
    if (isset($_SESSION) && isset($_SESSION['loginPostForm']) && isset($_SESSION['passwordPostForm'])) {
        //on retourne vrai
        return true;
    }else {
        //sinon on retourne faux
        return false;
    }
}
```

Si l'utilisateur est connecté, il est alors possible d'afficher le bouton de déconnexion et le bouton du profil car ces fonctionnalités ne sont pertinentes que pour un utilisateur connecté.

```php
<?php if(isConnect()) { ?>
    <button class="pseudoBtn"><a id="btnDeco" href="deco.php">Se Deconnecter</a></button>
    <button class="pseudoBtn" id="profil" onclick="openProfil()">Profil</button>
<?php }else{ ?>
    <button class="pseudoBtn" id="inscription" onclick="openInscription()">inscription</button>
    <button class="pseudoBtn" id="connection" onclick="openConnect()">Connection</button>
<?php } ?>
```

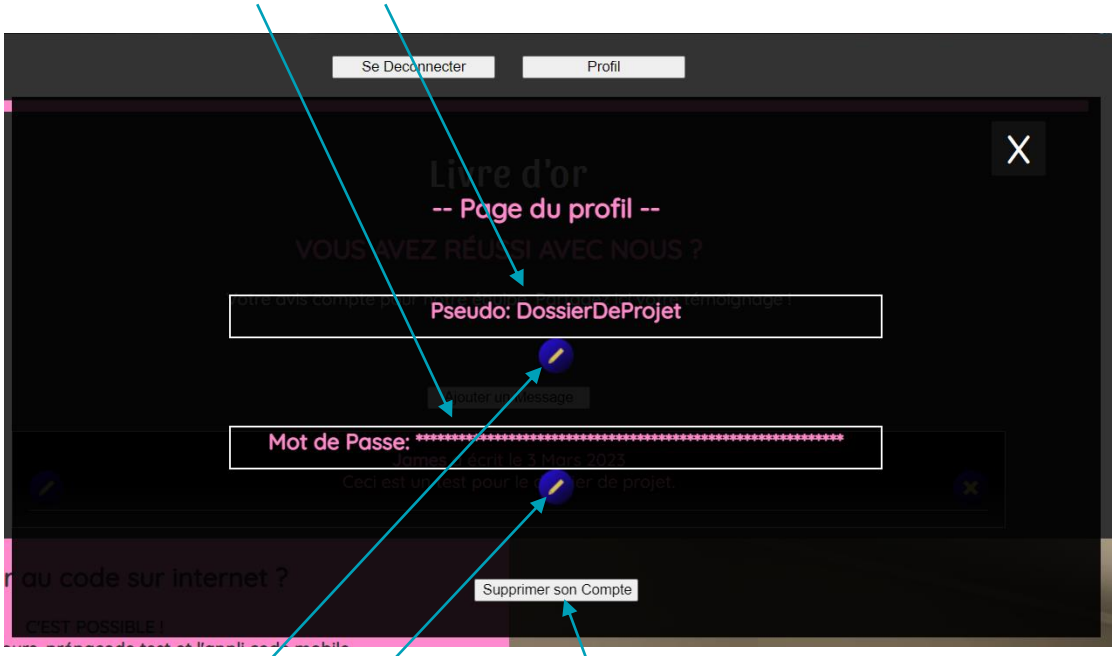
Le bouton de déconnexion appelle une page qui détruit les variables de session, de sorte que l'utilisateur actuel est déconnecté.

```php
<?php
//on démarre une session ou la récupére si elle est déjà démarré
session_start();


//fonction permettant de déconnecter l'utilisateur
function deconnect() {
    //on detruit la session donc toutes les données associées à cette session
    session_destroy();
    //on détruit la variable globale session
    unset($_SESSION);
    //on redirige l'utilisateur vers le Livre_d'or
    header("Location:Livre_d'or.php");
    //et on ajoute exit pour que le serveur ne continue pas à travailler pour rien après la redirection
    exit();
}

//et on appel la fonction deconnect qui va permettre de déconnecter l'utilisateur...
deconnect();

?>
```

Dans cette section, j'ai précédemment expliqué comment un utilisateur peut créer un compte, se connecter et se déconnecter. Maintenant, je vais vous montrer comment un utilisateur peut modifier et supprimer son compte. De plus, l'utilisateur peut également voir son pseudo grâce aux variables de sessions.

Ce modal s'affiche lorsque l'on clique sur le bouton profil.

Voici les variables de sessions.



Le bouton de modification de pseudo ou de mot de passe fonctionne selon la même logique que précédemment (l'inscription). Lorsque le formulaire est soumis, les valeurs saisies sont envoyées sous forme de variables HTTP POST au fichier de traitement "update.php". En revanche, le bouton qui appelle la page "destroy.php" permet de supprimer définitivement le compte de l'utilisateur.

La page "update.php" est similaire aux pages "create.php" et "fonction.php" : elle commence par une connexion à la base de données suivie d'une sécurisation des données (ici « newPseudo » et « newMdp ») et d'une définition de toutes les fonctions qui seront utilisées.

Il y a ensuite la modification du pseudo et du mot de passe de l'utilisateur. La modification du pseudo de l'utilisateur modifie aussi la colonne pseudo qui est présente dans la table message.

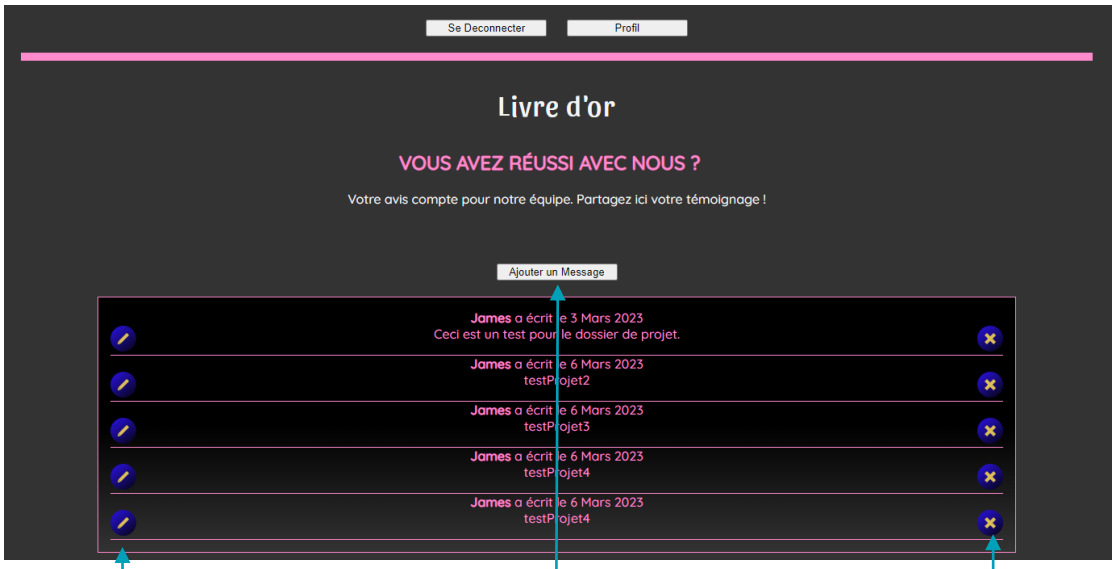
```php
if(!$erreurnewPseudo && isset($_POST['newPseudo'])){
    $newPseudo = htmlspecialchars(secure_donnee(trim($_POST[ 'newPseudo' ])));
    $requete = $bdd->prepare( 'UPDATE utilisateurs SET pseudo = :newPseudo WHERE pseudo = :pseudo AND  mdp = :mdp' );
    $requete->bindParam( ':newPseudo', $newPseudo );
    $requete->bindParam( ':pseudo', $_SESSION['loginPostForm'] );
    $requete->bindParam( ':mdp', $_SESSION['passwordPostForm']);
    $requete->execute();
    $requete->closeCursor();
    $requete = $bdd->prepare( 'UPDATE message SET pseudo = :newPseudo WHERE pseudo = :pseudo' );
    $requete->bindParam( ':newPseudo', $newPseudo );
    $requete->bindParam( ':pseudo', $_SESSION['loginPostForm'] );
    $requete->execute();
    $requete->closeCursor();
    $_SESSION['loginPostForm'] = $newPseudo;
}
```

La page "destroy.php" contient une connexion à la base de données, suivie d'une suppression du compte correspondant à l'utilisateur connecté. Tous les messages liés à cet utilisateur sont également supprimés.

```php
$requete=$bdd->prepare('DELETE FROM utilisateurs WHERE id = :id') or die (print_r($bdd->errorInfo()));
$requete->bindParam(':id',$id);
$requete->execute();
$requete->closeCursor();
$ez=$bdd->prepare('DELETE FROM message WHERE pseudo=:pseudo') or die (print_r($bdd->errorInfo()));
$ez->bindParam(':pseudo',$_SESSION['loginPostForm']);
$ez->execute();
$ez->closeCursor();
$retour .= 'Les informations concernant le nouveau membre ont bien été supprimées de la base.<br>';
deconnect();
```

Je viens de détailler le CRUD (Create, Read, Update, Delete) pour le modèle utilisateur.

Concernant les données des messages, ils existent aussi un CRUD permettant de créer des messages, de les voir, de les modifier et de les supprimer. Voici le rendu final avec un exemple :



Modification de message. Création de message. Suppression de message.

Création de message :

```php
if(!$erreur){
    $createMessage = htmlentities(secure_donnee($_POST['createMessage']));
    $requete = $bdd->prepare("INSERT INTO `message` VALUES(NULL,:pseudo,:msg,NOW())");
    $requete->bindParam( ':pseudo',  $_SESSION['loginPostForm']);
    $requete->bindParam( ':msg',  $createMessage);
    $requete->execute();
    $requete->closeCursor();
}      You, 3 months ago • Initial commit
```

Lecture de message :

```php
$sql = 'SELECT * FROM `message`';
$requete = $bdd->prepare($sql);
if($requete->execute()){
$all = $requete->fetchAll( PDO::FETCH_ASSOC );
echo json_encode( $all, JSON_PRETTY_PRINT );
```

Modification de message :

```php
if ( !$erreur ) {
    $msg = htmlentities( secure_donnee( $_POST[ 'modifMessage' ] ) );
    $sql = 'UPDATE message SET msg = :msg WHERE pseudo = :pseudo AND id= :idMessage';
    $requete = $bdd->prepare( $sql );
    $requete->bindParam( ':msg',  $msg );
    $requete->bindParam( ':pseudo',  $_SESSION[ 'loginPostForm' ]);
    $requete->bindParam( ':idMessage',  $_POST[ 'idMessage' ]);
    if ( $requete->execute() ) {
        $retour .= 'Le message a été modifier avec succès.<br />';
    } else {
        $retour .= 'Une erreur est apparue lors de la modification du message.<br />';
    }
    $requete->closeCursor();
}
```

Suppression de message :

```php
$requete=$bdd->prepare('DELETE FROM message WHERE id = :id') or die (print_r($bdd->errorInfo()));
$requete->bindParam(':id',$id);
$requete->execute();
$retour .= 'Les informations concernant le message de ce membre ont bien été supprimées de la base.<br>';
$requete->closeCursor();
```

C'est avec ce code que tous les messages peuvent s'afficher :

```javascript
/********************* affiche les messages *************************/
document.addEventListener("DOMContentLoaded", function () {
  function EzMessageLigne(ezJson) {
    let date = new Date(ezJson.date);
    let jourMois = date.getDate();
    let mois = date.getMonth();
    let annee = date.getFullYear();
    return '<div id="'+ezJson.id+'" style="border-bottom:1px #FF8ACC solid;">'+
                '<div><strong>'+ezJson.pseudo+ '</strong> a écrit le ' +jourMois+' '+numberToMonth(mois)+' '+annee+'</div>'+
                '<div style="display:flex;justify-content:space-between;">'+
                '<div class="Icons"><img src="../img/edit.png" alt="editMessage" onclick=\'modifMessage('+ezJson.id+')\'></div>'+
                '<div class="msg">'+ezJson.msg+'</div>'+
                '<div class="croix" ><img src="../img/delete.png" alt="deleteMessage" onclick="deleteMessage('+ezJson.id+')"></div>'+
                '</div>'+
              '</div>';
  }

  function EzMessage(ezJson){
      var html="";
      for (var i=0;i<ezJson.length;i++) {
          html+=EzMessageLigne(ezJson[i]);
      }
      return html;
  }

  var xhr;
  xhr = getRequest();
  var reponse;
  var json;
  if (xhr != false) {
      xhr.open("POST", "../../message/show.php", true);
      xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
      xhr.onreadystatechange = function () {
          if (xhr.readyState == 4 && xhr.status == 200) {
              reponse = xhr.responseText;
              json = JSON.parse(reponse);
              let html5tr=EzMessage(json);
              document.getElementById("message").innerHTML=html5tr;
          } else {
              reponse = "Problème lors de l'appel AJAX";
          }
      };
      xhr.send();
  }
});
/********************* Fin d'affichage des messages *************************/
```

Le code est destiné à créer un template HTML en JavaScript pour afficher des messages, il effectue une requête AJAX POST pour récupérer des données JSON à partir du fichier show.php et les afficher dans une page HTML. Le JSON est ensuite converti en chaîne de caractères HTML à l'aide des fonctions EzMessage() et EzMessageLigne(). L'utilisation d'un événement DOMContentLoaded garantit que le code s'exécutera une fois que le DOM sera prêt.

La fonction EzMessageLigne prend un objet JSON en entrée et renvoie une chaîne de caractères HTML contenant un modèle de message. Le modèle contient des événements onclick pour la modification et la suppression de messages, qui récupèrent l'ID du message à partir des données JSON. Cette approche permet d'identifier facilement l'ID du message à supprimer ou à modifier. La fonction EzMessage(ezJson) prend un tableau d'objets JSON ezJson en entrée et renvoie une chaîne de caractères HTML contenant toutes les lignes de message formatées.

Le code crée une instance d'objet XMLHttpRequest xhr et utilise la méthode getRequest() pour créer une instance d'objet XMLHttpRequest, puis ouvre une connexion POST avec le fichier show.php. Il déclare également l'en-tête "Content-type" pour spécifier que les données envoyées sont de type formulaire.

La fonction onreadystatechange de xhr est définie pour traiter la réponse de la requête AJAX. Si l'état de la requête est 4 (terminé) et le statut est 200 (OK), la réponse est stockée dans la variable reponse et convertie en objet JSON avec la méthode JSON.parse(). Ensuite, la fonction EzMessage() est appelée avec le tableau d'objets JSON json comme argument pour créer une chaîne de caractères HTML contenant les messages. Cette chaîne est ensuite insérée dans la page HTML avec la méthode innerHTML pour afficher les messages. Si la requête échoue, la variable reponse est définie pour indiquer le problème.

C'est avec ce code qu'il est possible de supprimer des messages :

```
/********************* delete message ***************************/
function deleteMessage(id){
  var xhr;
  let data = "&id="+id;
  xhr = getRequest();
  var reponse;
  if (xhr != false) {
      xhr.open("POST", "../../message/destroy.php", true);
      xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
      xhr.onreadystatechange = function () {
          if (xhr.readyState == 4 && xhr.status == 200) {
              reponse = xhr.responseText;
              let child = document.getElementById(id);
              child.parentNode.removeChild(child);
          } else {
              reponse = "Problème lors de l'appel AJAX";
          }
      };
      xhr.send(data);
  }
}
/********************* Fin delete message ***************************/
```

Cette fonction utilise une requête HTTP asynchrone pour supprimer le message de la base de données côté serveur à partir du fichier destroy.php, puis une fois que la suppression est réussie, elle supprime l'élément HTML correspondant à l'id du message de la page en utilisant la méthode removeChild(). Cela permet de mettre à jour la page en temps réel sans avoir à la recharger.

Sur la page agence.php, les formulaires de contact et de réservation sont traités de la même manière. Une fois que l'utilisateur a rempli et soumis le formulaire, un événement est déclenché, qui appelle une fonction de test d'erreur dans la partie front-end en JavaScript. Cette fonction teste les erreurs en vérifiant différents critères pour chaque champ du formulaire.

```
let name = document.getElementById('name').value;
// test de l'existence du nom
if (name !== '') {
    // Verification du name
    if (!verifChaine('name')) {
        textRetour += "Le nom doit être composé de 3 caractères minimum et ne doit pas dépasser 12 caractères.<br>";
    }
    // On vérifie à l'aide d'expression régulière que le nom respecte bien la forme ABCD1234
    if (!name.match(/[a-zA-Z0-9]/)) {
        textRetour += 'Le nom saisie n\'est pas valide : il doit être composé que de chiffres et de lettre.<br>';
    }
    //check si la chaine a des caractères consecutifs
    if (!checkConsecutiveChars(name)) {
        textRetour += 'Le nom contient des caractères consécutifs.<br>';
    }
} else {
    // le nom n'existe pas
    textRetour += 'Vous n\'avez pas tapez votre nom !<br>';
}
```

La fonction SendFormulaire() est appelée lorsque toutes les vérifications sont passées avec succès. Cette fonction utilise la technologie AJAX pour envoyer les données du formulaire à la page contact.php sans recharger la page actuelle. Cette opération est effectuée en arrière-plan, ce qui permet une expérience utilisateur plus fluide.

```
//le retour n'est pas vide lorsqu'il y a une erreur : il faut donc l'afficher
if (textRetour != "") {
    document.getElementById('textEchec').style.display = "block";
    document.getElementById('textEchec').innerHTML = textRetour;
} else {
    // Si tout est bon, on transmet à la fonction qui va créer une requête AJAX afin de transmettre les parametres à la page cible PHP
    SendFormulaire();
}
```

Sur la page contact.php, une fois les données du formulaire reçues via la requête AJAX, la partie back-end en PHP effectue une deuxième vérification des données. Voici un exemple de code :

```php
if ( isset( $_POST[ 'name' ] ) && $_POST[ 'name' ] != '' ) {
    $longueur_chaine = strlen($_POST['name']);
    if($longueur_chaine < 3 || $longueur_chaine > 12){
        $erreur = true;
        $echec['name'] = 'Le nom doit être composé de 3 caractères minimum et ne doit pas dépasser 12 caractères.';
    }
    $exp = "/[a-zA-Z0-9]/";
    if(!preg_match($exp, $_POST['name'])){
        $erreur = true;
        $echec['name'] = 'Le nom saisie n\'est pas valide : il doit être composé que de chiffres et de lettre.';
    }
    if (!checkConsecutiveChars($_POST['name'])) {
        $erreur = true;
        $echec['name'] = 'Le nom contient des caractères consécutifs.';
    }
    $name = htmlentities(secure_donnee(trim($_POST[ 'name' ])));
} else {
    $erreur = true;
    $echec['name']= 'Vous n\'avez pas tapez votre nom !';
}
```

En effet, une fois que les données ont été validées côté serveur, la fonction mail est appelée pour envoyer le courrier électronique. Cette fonction prend en argument le destinataire, l'objet du message, le corps du message et l'expéditeur.

```php
if ( !$erreur ) {
    mail( 'jamesgnagne3@gmail.com', $Objet, $message, $headers );
    $succes = 'Votre message a bien été envoyé. Merci de m\'avoir contacté :)';
    session_start();
    $_SESSION[ 'succes' ] = $succes;
}
```

Conclusion

Grâce à un cahier des charges bien défini, j'ai réussi à développer toutes les fonctionnalités demandées, telles que la présentation de l'entreprise, les horaires, les formations, la réservation en ligne, les commentaires et le formulaire de contact. Le développement du site a nécessité l'utilisation de plusieurs langages tels que HTML, CSS, JS et PHP, ainsi que la création d'une base de données pour stocker les informations des utilisateurs et les commentaires. Le site est responsive et s'adapte à différents types d'appareils, et il est compatible avec les principaux navigateurs web tels que Google Chrome, Safari, Mozilla Firefox et Opera.

Le site répond aux attentes de l'école de conduite de Cagny en offrant une expérience utilisateur fluide et efficace. En fin de compte, j'ai réussi à livrer le code source du site web ainsi qu'une explication pour permettre à l'entreprise de réutiliser le site pour toute modification future. Ils peuvent également me contacter si nécessaire. Néanmoins, l'auto-école n'a pas encore décidé de mettre le site en ligne.

Ce projet a été une occasion pour moi de mettre en pratique mes compétences en développement web et de relever des défis techniques.

Recherche basée sur un site anglophone

Etant donné que je n'avais pas encore mis le site en ligne mais que je devais quand même tester si mes formulaires étaient fonctionnels, j'ai dû trouver une solution pour envoyer des e-mails en local avec WAMP. Pour cela, j'ai tapé ceci sur Google : "send mail in local with wamp". J'ai sélectionné l'un des premiers résultats qui se trouvait en anglais, le site est : https://applerinquest.com/how-to-send-an-email-using-smtp-in-localhost-with-wamp/. J'ai testé la solution proposée sur la page du site : voici l'extrait de la page (en anglais).

PAGE D'ORIGINE

DOWNLOAD SENDMAIL.ZIP

Sendmail.exe is a simple windows console application that emulates **sendmail's "-t" option** to deliver emails piped via stdin. It is intended to ease running Unix code that has **<code>/usr/lib/sendmail</code>** hardcode as an email delivery means. It doesn't support deferred delivery and requires an SMTP server to perform the actual delivery of the messages. You can download the **Sendmail package** from the line below.
download sendmail.zip

INSTALL SENDMAIL

- Extract the **sendmail.zip** then you will see the **sendmail** folder
- Copy and paste the **sendmail folder** at **"C:\wamp64\"** location where your Wampserver is installed. In my case, I installed Wampserver in C:\wamp64.
- Inside the sendmail folder, find the **sendmail.ini** file and edit it with the following setting. As I mentioned earlier, I will use the **Gmail account** for the mail server setting.

sendmail.ini

```
smtp_server=smtp.gmail.com
smtp_port=587
auth_username=your_username
auth_password=your_password
```

- Save and close the file

2-factor authentication on Google Account issue (Solved)

Important note, Google recently forces you to enable **"2-factor authentication"** on your Google account. If your Google account is already enabled **"2-factor authentication"**, you **can not use the regular password of your account** for setting in the **sendmail.ini**. Instead, you need to generate **an app-specific password** (16 characters) from your Google account and use the new app passwords you just generate in **auth_password** in **sendmail.ini**. Below is how to generate the app-specific password from your Google account.

How to generate an app-specific password from your Google account

Note that, the Google account interface is changed from time to time. You may not see the exact steps I show below. But you still want to look for the app-specific password section.

- Login to your **Google account**
- Navigate to the **Security menu**
- On the **Security page**, look for "**App passwords**" under the "**Signing in to Google**" section.
- Click on the "**App passwords**", and the page will redirect to the verify page. Just enter your regular password for your Google account.
- Then you will see the **App passwords** page.
- On the **App passwords** page, select the app and device you want to generate the app password for. If none of the choices match your need, you will choose "**Other (Custom name)**". In my case, I choose "**Other (Custom name)**". We will continue with the "**Other (Custom name)**" choice.
- After that, you need to type the **label name of your new app password**. Then you will click on the "**GENERATE**" button.
- Next, the new popup with the 16 characters will show up. You want to copy those 16 characters and paste them into the config file you want. In this post, we will paste the 16 characters into **auth_password** in **sendmail.ini**.

CONFIGURE PHP.INI

- At the **wamp icon** (green color), click on that **wamp icon** then navigate to **PHP** and then look for the **php.ini** menu.
- Click on the **php.ini** menu then follows the setting below.

```
[mail function]
SMTP=smtp.gmail.com
smtp_port=587
sendmail_path ="C:\wamp64\sendmail\sendmail.exe"
sendmail_from ="yourmail@gmail.com"
```

Notice that the **SMTP** and **smtp_port** settings are the same as we set in **sendmail.ini** above.

- Next, save the **php.ini** and **restart** all services of Wampserver
- Lastly, after restarting all services, you should see the wamp icon turns green.

TRADUCTION

POUR TELECHARGER SENDMAIL.ZIP :

Sendmail.exe est une application console Windows simple qui émule l'option "-t" de sendmail pour livrer des e-mails transmis via stdin. Elle est destinée à faciliter l'exécution de code Unix qui a <code>/usr/lib/sendmail</code> codé en dur comme moyen de livraison d'e-mails. Elle ne prend pas en charge la livraison différée et nécessite un serveur SMTP pour effectuer la livraison réelle des messages. Vous pouvez télécharger le package Sendmail à partir du lien ci-dessous. Télécharger sendmail.zip.

INSTALLER SENDMAIL :

• Extrayez le fichier sendmail.zip, puis vous verrez le dossier sendmail.

• Copiez et collez le dossier sendmail à l'emplacement "C:\wamp64" où est installé votre serveur Wamp. Dans mon cas, j'ai installé Wampserver dans C:\wamp64.

• Dans le dossier sendmail, recherchez le fichier sendmail.ini et éditez-le avec les paramètres suivants. Comme je l'ai mentionné précédemment, j'utiliserai le compte Gmail pour le serveur de messagerie.

• Enregistrez et fermez le fichier.

PROBLEME DE L'AUTHENTIFICATION A DEUX FACTEURS SUR LE COMPTE GOOGLE (RESOLU) :

Note importante : Google vous oblige récemment à activer l'authentification à deux facteurs sur votre compte Google. Si votre compte Google a déjà activé l'authentification à deux facteurs, vous ne pouvez pas utiliser le mot de passe habituel de votre compte pour le configurer dans sendmail.ini. Au lieu de cela, vous devez générer un mot de passe d'application spécifique (16 caractères) à partir de votre compte Google et utiliser les nouveaux mots de passe d'application que vous venez de générer dans auth_password dans sendmail.ini. Voici comment générer le mot de passe d'application spécifique à partir de votre compte Google.

COMMENT GENERER UN MOT DE PASSE SPECIFIQUE A UNE APPLICATION A PARTIR DE VOTRE COMPTE GOOGLE :

Notez que l'interface du compte Google change de temps en temps. Vous pouvez ne pas voir les étapes exactes que je montre ci-dessous. Mais vous devez toujours chercher la section de mots de passe spécifiques à l'application.

• Connectez-vous à votre compte Google

• Accédez au menu Sécurité

• Sur la page Sécurité, cherchez «Mots de passe d'application» dans la section «Connexion à Google».

• Cliquez sur «Mots de passe d'application», et la page sera redirigée vers la page de vérification. Entrez simplement votre mot de passe habituel pour votre compte Google.

• Ensuite, vous verrez la page Mots de passe d'application.

• Sur la page Mots de passe d'application, sélectionnez l'application et l'appareil pour lesquels vous souhaitez générer le mot de passe de l'application. Si aucun des choix ne correspond à vos besoins, choisissez «Autre (nom personnalisé)». Dans mon cas, j'ai choisi «Autre (nom personnalisé)». Nous continuerons avec le choix «Autre (nom personnalisé)».

• Après cela, vous devez taper le nom d'étiquette de votre nouveau mot de passe d'application. Ensuite, cliquez sur le bouton «GÉNÉRER».

• Ensuite, la nouvelle fenêtre contextuelle avec les 16 caractères apparaîtra. Vous voulez copier ces 16 caractères et les coller dans le fichier de configuration que vous voulez. Dans ce post, nous collerons les 16 caractères dans auth_password dans sendmail.ini.

CONFIGURER PHP.INI :

• Cliquez sur l'icône Wamp (de couleur verte), puis naviguez jusqu'à PHP et recherchez le menu php.ini.

• Cliquez sur le menu php.ini, puis suivez les paramètres ci-dessous.

Notez que les paramètres SMTP et smtp_port sont les mêmes que ceux que nous avons définis dans sendmail.ini ci-dessus.

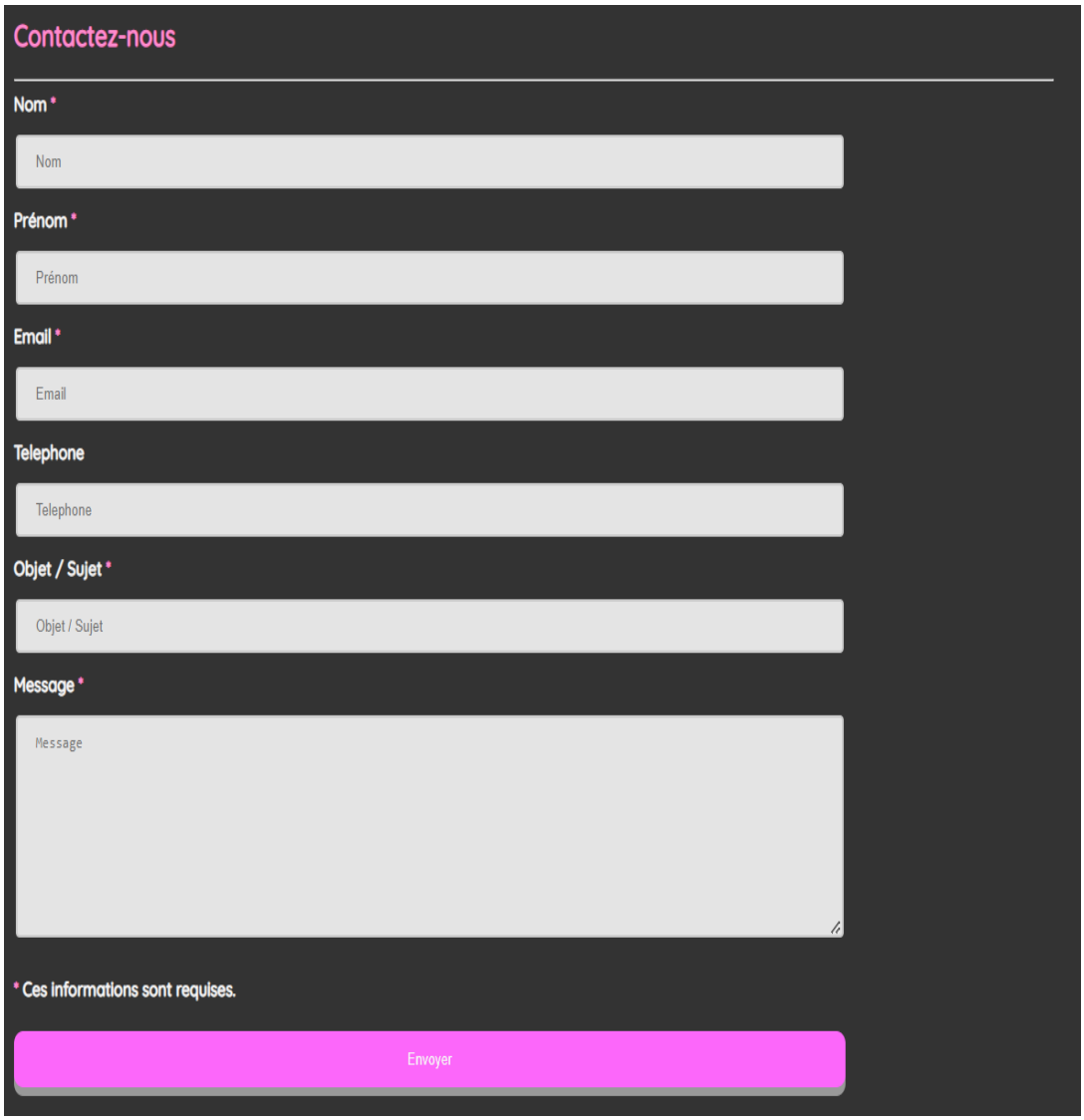
• Ensuite, enregistrez le fichier php.ini et redémarrez tous les services de Wampserver.

• Enfin, après avoir redémarré tous les services, vous devriez voir l'icône Wamp devenir verte.

Après avoir suivi ces étapes, j'ai testé la solution et j'ai pu envoyer des emails en local avec succès.

Annexe

Formulaire de contact :



Formulaire de réservation :