

# DOSSIER DE PROJET

Romaric Brice NGANAWARA

Développeur Web Full Stack

Studi

**Application Web / Web mobile**

*“ Afrique Centrale Découverte ”*

## Remerciements

Je voudrai remercier toute l'équipe **studi** pour ses compétences, ses expériences, surtout sa disponibilité lors de cette formation.

Je remercie particulièrement mes **formateurs** et mon **encadrant** pour leur disponibilité, réactivité et surtout pour le savoir transmis.

Mes remerciements vont notamment à l'endroit de Monsieur **Schadrac WANI**, qui par ses nombreuses années d'expériences de développeur m'a beaucoup apporté, surtout à chaque fois que j'avais besoin de conseils.

Je remercie toutes celles et ceux, ayant contribués de près, d'une manière ou d'une autre, à l'aboutissement de ce travail.

## Sommaire

Remerciements.....	2
Sommaire.....	3
Acronyme.....	4
Introduction.....	5
Compétences du référentiel couvertes par le projet .....	5
Développer la partie frontend d'une application web / web mobile en intégrant les recommandations de sécurité .....	5
Développer la partie backend d'une application web / web mobile en intégrant les recommandations de sécurité .....	5
Résumé du projet .....	6
Le cahier des charges .....	7
Spécifications techniques .....	8
Spécifications fonctionnelles .....	9
La réalisation du projet .....	11
Maquettage .....	11
GitHub .....	13
Première partie : FRONTEND .....	14
Description	
Illustrations	
Deuxième partie : BACKEND.....	26
Description	
Illustrations	
Description de la veille effectuée pendant le projet .....	44
Reference ou URL du site utilisé lors de la précédente recherche .....	46
Traduction .....	46
Conclusion .....	48

## **QUELQUES ABBREVIATIONS ET ACCRONYMES ESSENTIELS**

**HTTP** : HyperText Transfer Protocol

**FTP** : File Transfer Protocol

**PHP**: Hypertext Preprocessor / Préprocesseur hypertexte

**HTML** (Anglais) : HyperText Markup Language / Langage de balise hypertexte

**CSS** (Anglais) : Cascading Style Sheets / Feuille de style en cascade **PAQ** : Plan d'assurance qualité

**UML** (Anglais) : Unified Modelling Language / Langage de Modelisation Unifié

**MCD** : Modèle Conceptuel de Données **MLD** : Modèle Logique de Données  
**MPD** : Modèle Physique de Données

**DOM** (Anglais) : Document Object Model / Modèle, Structure d'objet du document **W3C** (Anglais) :

**World Wide Web** Consortium **FEVAD** : Fédération de l'E-commerce et de la Vente À Distance

**API** : Interface de Programmation Applicative

## **I - INTRODUCTION**

### **COMPETENCES DU REFERENTIEL COUVERTES PAR LE PROJET**

#### **1 - Réaliser une interface utilisateur web statique et adaptable**

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

L'application web de la société est adaptable sur différents écrans et intègre la technologie mobile first, ce qui permet notamment de faire usage du Framework Bootstrap, du préprocesseur SASS, ainsi que des médias queries pour la mise en page statique. La bibliothèque JQuery/javascript est utilisée en complément pour permettre l'interactivité, le contrôle des saisies, le traitement des formulaires conforme avec les formats des données dans la base de données.

#### **2 - Développer une interface utilisateur web dynamique**

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

**Symfony** reste évidemment un ensemble de composants POO, un Framework MVC tout à fait adapté à la situation. Mais connu pour un apprentissage pouvant prendre un peu plus de temps, mon choix d'assurer la partie dynamique de l'application était orienté vers le langage PHP et son objet PDO.

L'usage du PHP grâce à son objet PDO, permet d'établir une connexion à la base de données (POSTGRES, réalisée en amont), et permet par la suite d'implémenter les fonctionnalités de l'applications, à travers les différentes méthodes de communication ou d'échange (requêtes).

Le backend reçoit et traite les requêtes provenant du frontend. Il détermine comment répondre à ces requêtes. Il assure la communication avec la base de données à partir de la méthodologie CRUD), du système de gestion de base de données.) et renvoie les données nécessaires au frontend pour affichage

## **II – RESUME DU PROJET**

L'agence de voyages "Afrique Centrale Découverte" est un projet en réflexion depuis quelques années et bientôt dans sa phase opérationnelle. La situation socio-politique des pays visés par cet objectif où existe de réels problèmes sécuritaires, a retardé la mise en place de son application web.

Des circuits de voyages seront bientôt proposés aux différents touristes, souhaitant évidemment découvrir de belles natures (faune, parcs etc....) de l'Afrique centrale. Ces lieux sont repartis principalement entre Le CAMEROUN, Le GABON, la CENTRAFRIQUE et le CONGO.

L'objectif consiste à proposer en ventes, des offres avantageuses, attractives à ses clients, et potentiels clients, afin de maximiser son chiffre d'affaire.

En outre, la société percevra des commissions en fonction du pourcentage de ventes de ses circuits contenant : Des vols, hôtels/motels ou hébergement privé.

Mandaté par son gérant, je me suis projeté dans la réalisation de son interface web dynamique. Ce dernier souhaite une application Web adaptable à tous les écrans, à partir des éléments graphiques fournis en amont symbolisant l'image de la société.

L'attractivité, la sécurité et surtout l'interactivité demeurent les éléments essentiels de L'application selon le gérant. Par exemple : Afin procéder à une réservation, lorsqu'un visiteur accède au site, et s'intéresse à un service, il doit alors créer son compte client des formulaires sont disponibles à cet effet, puis procéder à des manipulations jusqu'à la validation de sa réservation. Sachant qu'une réservation aboutie, peut être régler de différentes manières. Et des liens vers d'autres applications, ainsi que des pages publicitaires pourrions être possible.

Plusieurs compétences doivent être mutualisées en frontend et backend, pour la partie statique (HTML/CSS) : l'usage du Framework Bootstrap, sa bibliothèque et ses composants, des média queries), du préprocesseur SASS, ainsi que la maîtrise de l'IDE Visual studio code et ses dépendances.

Des pages interactives et dynamiques à prévoir (usage des langages de programmation dynamique (Javascript / PHP). La conception d'un système de gestion des bases de données par exemple : POSTGRES), l'usage éventuel d'API pour faciliter et garantir la liaison des données.

## **III – CAHIER DES CHARGES**

Application web de : Afrique Centrale Découverte Agence de voyages  
Représentée par Monsieur Lewis MONGAI son initiateur gérant.

Est une jeune entreprise qui intervient dans le domaine de voyages touristiques, dans 4 pays principaux pays de l'Afrique centrale que sont : La Centrafrique, le Congo, le Cameroun et le Gabon.

Son but consiste à mettre à la disposition de sa clientèle ainsi que ses visiteurs des informations sur les villes visées notamment par les destinations proposées. Elle joue un rôle d'intermédiaire (prestataire de service), entre ses clients et les entreprises opérant dans ce secteur : compagnies aériennes, entreprise d'hébergement (hôtel/motel) etc.

Elle négocie ses commissions avec ses clients (grandes entreprises) en fonction du pourcentage de ses ventes.

L'application web de l'agence contiendra une dizaine de pages, entre autres, des pages pouvant informer les clients sur les lieux de visites, les photos, le coût de voyages, de l'hébergement, ainsi que les attractions possibles dans ces villes avec la possibilité d'être en contact avec des bureaux ou guides touristiques locaux etc.

L'agence proposera des ventes de circuits à ses clients en fonction de leurs budgets et intérêts. Elle permet aux visiteurs de rechercher des circuits, une réservation est possible que lorsque l'utilisateur est inscrit et dispose d'un espace personnel. Il peut alors modifier ou annuler une réservation voir mettre à jour ses coordonnées. Les circuits sont composés obligatoirement d'un vol (aller et retour), et d'une réservation d'hôtel/motel facultative. Les clients sont informés sur les moyens locaux de transport (essentiellement routier).

Les clients de l'agence ont la possibilité de donner leur avis à travers une page dédiée, tout comme entrer en contact avec la société par téléphone ou grâce à un formulaire, ou alors se rendre à l'agence.

L'application affichera les avis clients, indique les heures d'ouverture de l'agence. Des maquettes doivent être fournies au gérant (client) à sa demande avant la phase de conception. **L'administrateur** de l'application à la possibilité d'ajouter des clients ou potentiels clients directement en base de données sinon depuis un formulaire d'inscription disponible sur la page web.

#### **IV - SPEFICIATIONS TECHNIQUES ET FONCTIONNELLES**

L'analyse des besoins, à partir du cahier des charges m'a permis de dégager les étapes et méthodes essentielles à la réalisation des différentes fonctionnalités que pourra composer l'application.

S'agissant d'un projet personnel, il m'a été possible de par ma formation de réaliser certaines tâches, notamment celles du **Product Owner, Chef de Projet, L'UI, UX Designer etc.** Je tiens tout de même à rappeler qu'elles sont de nature à être attribuées à d'autres membres d'une équipe de développement, bien évidemment, si ces derniers ont une existence physique au sein du projet ou de l'entreprise.

## **1 - Spécifications Techniques :**

Méthode utilisée : **UML** : (Unified Modeling Language), qui est un langage de modélisation graphique orienté objet, basé sur des pictogrammes symbolisant les démarches à suivre (que va-t'en faire ? Comment va-t'en le faire ? Etc.).

L'élaboration des besoins sur la base des diagrammes de comportements se décompose de façon suivante.

Les services attendus :

Accéder à l'application pour consulter les offres (rendre l'application disponible sur le web), procéder à des recherches de circuits, s'inscrire pour disposer d'un espace client, se connecter, modifier ses coordonnées, procéder à une réservation, donner son avis et contacter l'agence.

User Story 1 : **Accéder à l'application.**

Utilisateurs concernés : **Tous les acteurs**, à partir d'une adresse web (<http>)

User Story 2 : **procéder à des recherches de circuits.**

Utilisateurs concernés : **Tous les acteurs.**

User Story 3 : **S'inscrire.**

Utilisateurs concernés : **Visiteurs**

User Story 4 : **se connecter.**

Utilisateurs concernés : **Administrateur, Clients.**

User Story 5 : **Modifier ses coordonnées.**

Utilisateurs concernés : **Clients.**



User Story 6 : **Procéder à une réservation.**

Utilisateurs concernés : **Clients.**

User Story 7 : **Laisser des avis**

Utilisateurs concernés : **Clients.**

User Story 8 : **Contacter l'agence.**

Utilisateurs concernés : **Clients, visiteurs.**

En effet, seul l'**administrateur** gère le backoffice de l'application. Des formulaires, et actions seront mis à disposition pour implémenter ces différentes fonctionnalités.

Les **Diagramme de cas d'utilisation, diagramme de séquence et diagramme de classe** figurent bien en annexe.

## **2 - Spécifications fonctionnelles :**

L'élaboration des besoins sur la base des diagrammes de structure (**UML**) se décompose de façon suivante.

Considérant que :

- ☐ **Une réservation** correspond à un voyage qui correspond aussi à un circuit. Elle comporte, un nombre (seul, en couple ou en groupe, une liste de choix est mise à disposition), un prix et un type de paiement)
- ☐ **Un circuit**, est le type de voyage principal. Il comporte obligatoirement une destination, un type soit (vol aller/retour) simple ou vol (aller/retour) + réservation d'hôtel). Il comporte également, une date Aller et une date retour, un prix et une photo du lieu).

**Table et Données attendues :**

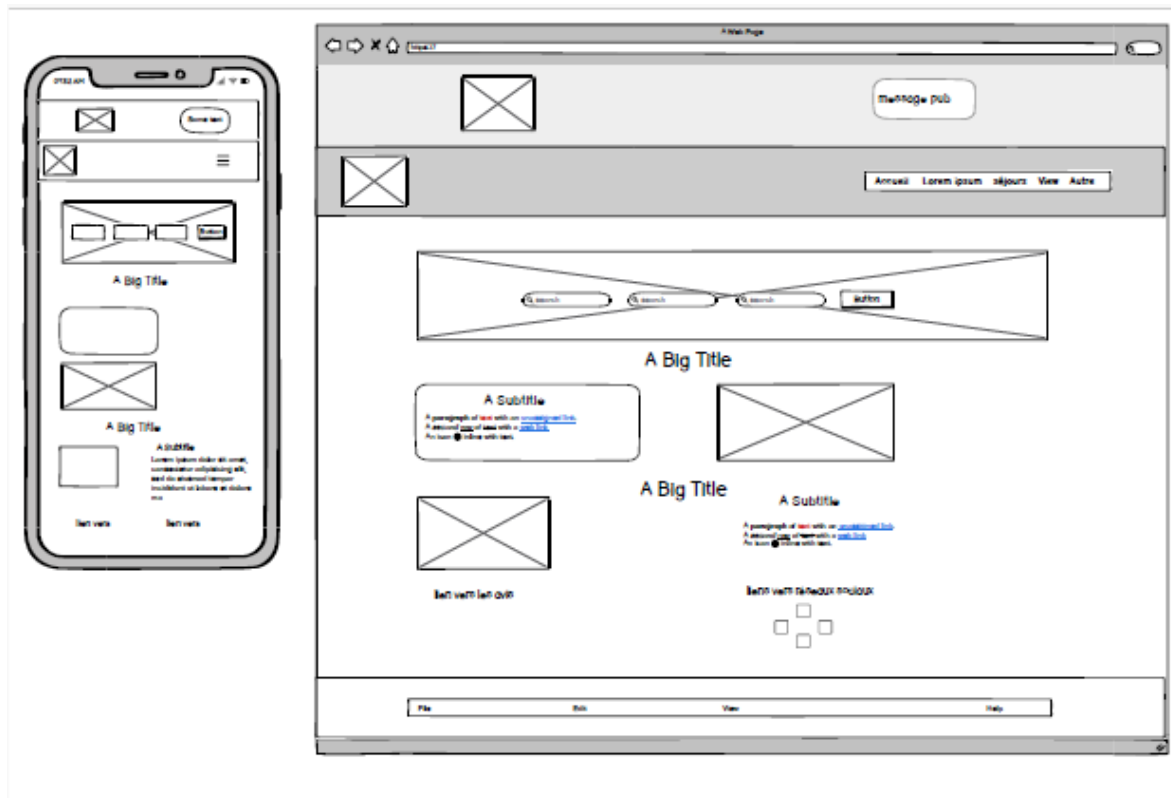
- ✓ **clients** (**id client**, civilité, nom, prénom, âge, nationalité, téléphone, email, mot de pass, date\_création, statut\_compte, **id réservation**)
- ✓ **réservations** (**id réservation**, nombre personne (le client voyage seul, en couple ou un groupe de personnes), date réservation, prix **id circuit**)
- ✓ **circuits** (**id circuit**, destination, date départ, date retour, prix, photo, type circuit (vol aller/retour simple ou alors vol aller/retour + hôtel).
- ✓ **avis** (**id avis**, pseudo, note, commentaire date\_avis).
- ✓ **contacts** (**id contact**, type demande, nom, email, téléphone, message date contact).
- ✓ **users** (**id**, libellé).

## IV – REALISATIONS

### MAQUETTAGE

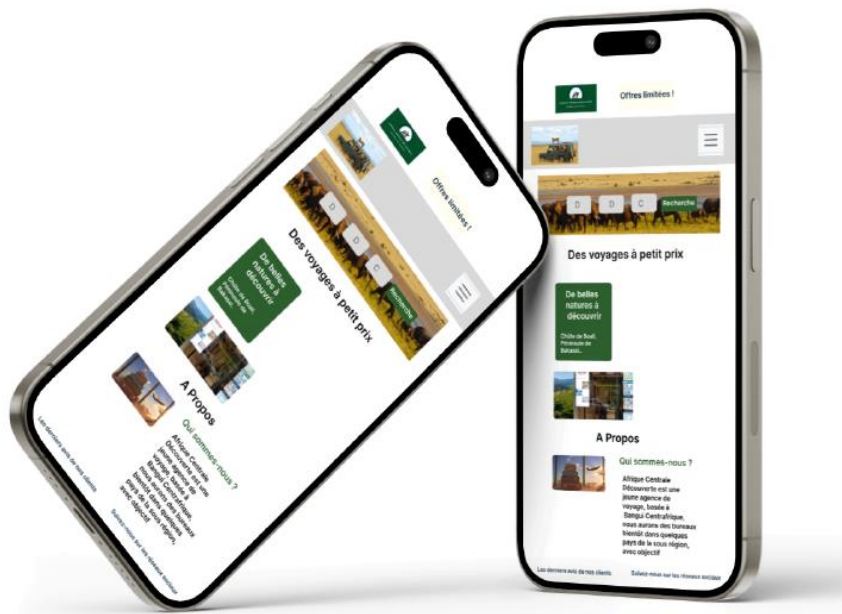
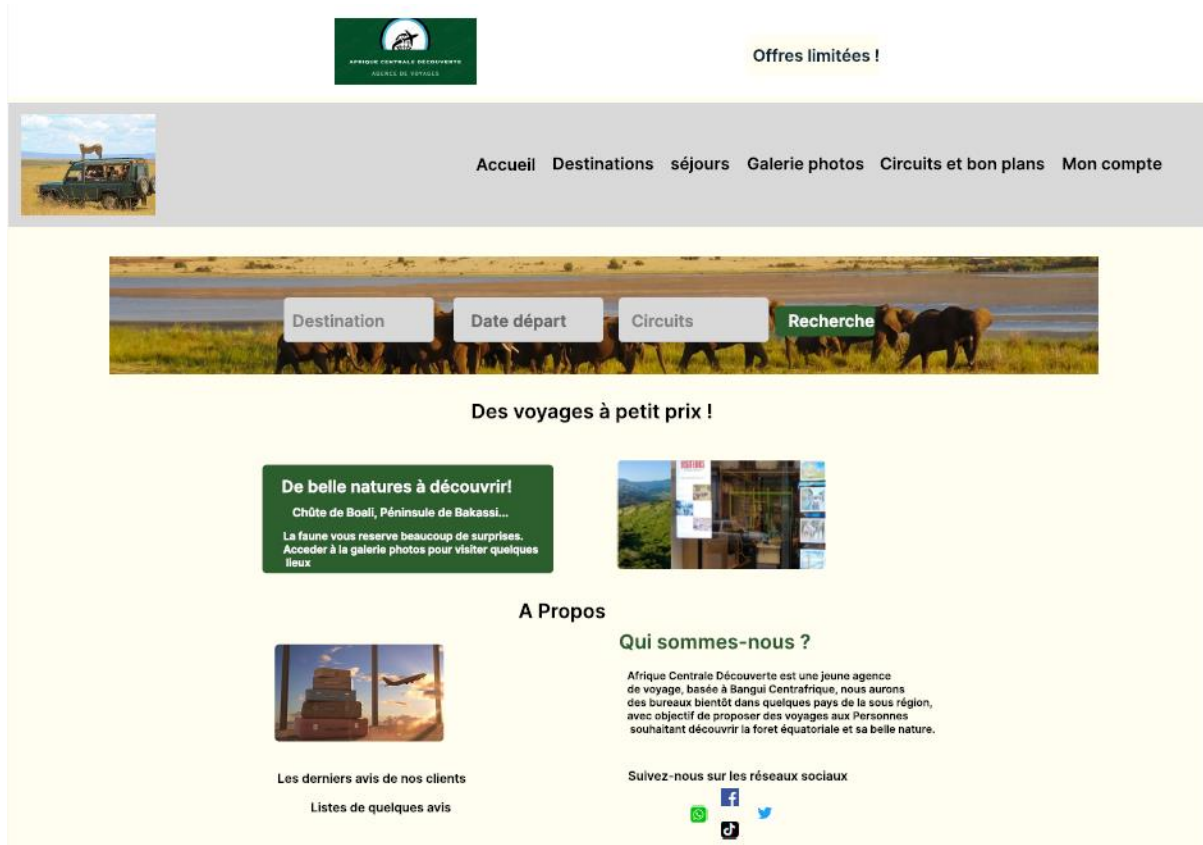
Après une phase d'analyse et de recherche, suivi d'une série de phase de tests, j'ai pu établir chronologiquement un zoning, des wireframes en brouillon ainsi qu'un mockup suivant la technologie Design Thinking.

Cette étape, évidemment, l'une des plus importantes du projet, m'a notamment permis d'échanger de façon régulière sur la nature des besoins de l'application, et également sur l'expérience utilisateurs avec toutes les parties prenantes.



Le zoning ayant permis de matérialiser les idées de base, demeure cependant une maquette de basse fidélité, servant juste à positionner les éléments essentiels de l'application à savoir : Logo, entête, barre de recherche, Navigation, pied de page, présents sur l'interface graphique communément appelée, la page d'accueil ou page principale du site.

**Mockups** : Interface graphique haute fidélité, est réalisée à partir des wireframes après validation de toutes les parties prenantes, et en adéquation avec le cahier des charges elle est la représentation visuelle du contenu de l'application.

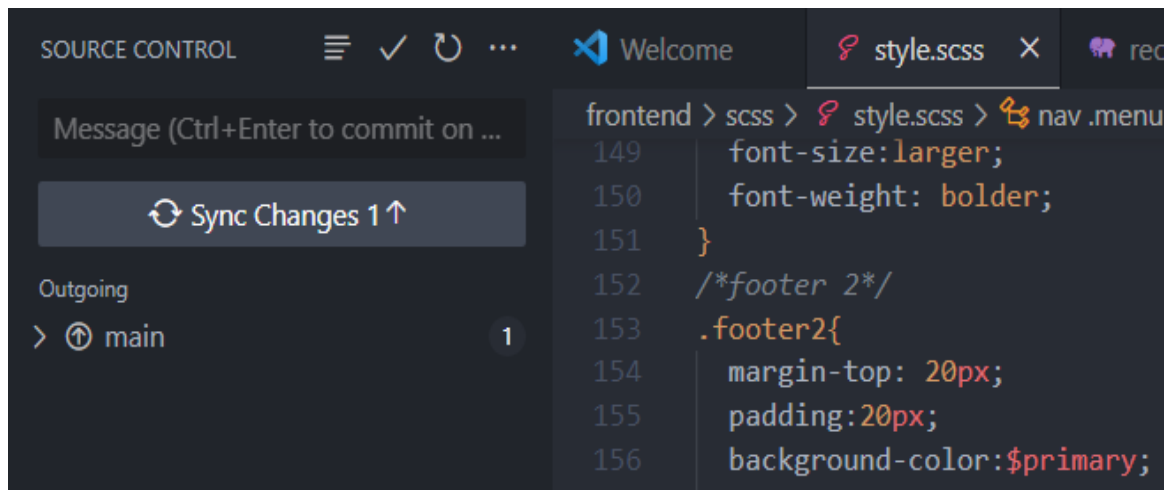


## GITHUB :

Reconnu comme étant un système de Gestion de version, Git permet à la fois de travailler aussi bien en local qu'à distance, facilite un accès au contenu de son travail partout à travers ses commandes.

Il a été d'une grande utilité dans le développement de cette application.

J'ai pu initialiser un dépôt local github depuis mon éditeur VS code. Ce qui m'a évidemment permis de procéder à des sauvegardes à chaque étape importante du projet. Par exemple lors d'une implémentation de fonctionnalité.



## PREMIERE PARTIE : FRONTEND

Afin de réaliser l'application adaptable aux différentes tailles d'écran, comme indiqué précédemment, j'ai choisi l'usage du Framework Bootstrap,

particulièrement de ses composants et classes (Containers Flexbox , Grid, Button, table etc.).

J'ai d'abord créé un répertoire englobant tout mon projet et qui porte le nom du projet.

Ensuite, j'ai créé 2 dossiers, un pour le frontend et un second pour la partie backend ainsi qu'un fichier readme.md pour la sauvegarde de version.

Contenu du dossier **Frontend** :

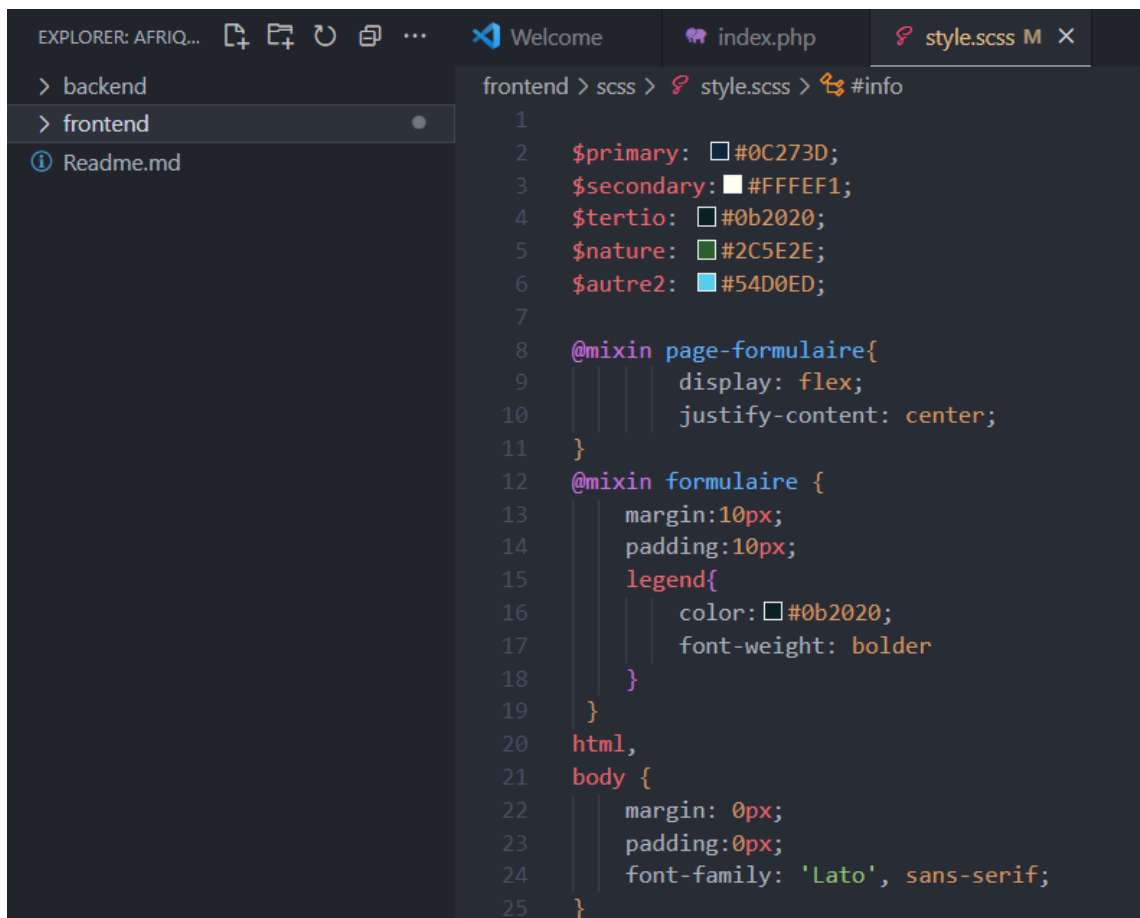
Ce dossier est composé des sous dossiers, comme

- ✓ Un dossier images (regroupant toutes les images et logo faisant partie de la charte graphique de l'application).
- ✓ Un dossier SCSS dans lequel se trouve mes fichiers style.scss, style.css et style.css.map.
- ✓ Un dossier JS dans lequel se trouve mes fichiers Javascript et bibliothèque JQuery
- ✓ Un dossier ASSET dans lequel se trouve mes feuilles de style Bootstrap qui sont liées aux contenus pour permettre la réalisation des styles.
- ✓ Un dossier ICON englobant les émoticônes fournis par la bibliothèque Bootstrap
- ✓ Un dossier INCLUDE, contenant mes fichiers susceptibles d'être appelé souvent dans les autres pages.
- ✓ Un dossier page contenant le : Dashboard et les pages destinées au compte client.

**VISUAL STUDIO CODE** : Est L'IDE avec lequel j'ai réalisé ce travail.

Après avoir établie la liaison entre mes fichiers et les différentes pages, j'ai pu intégrer SASS pour le customiser avec Bootstrap, et procéder par la suite à ma première sauvegarde de version GIT.

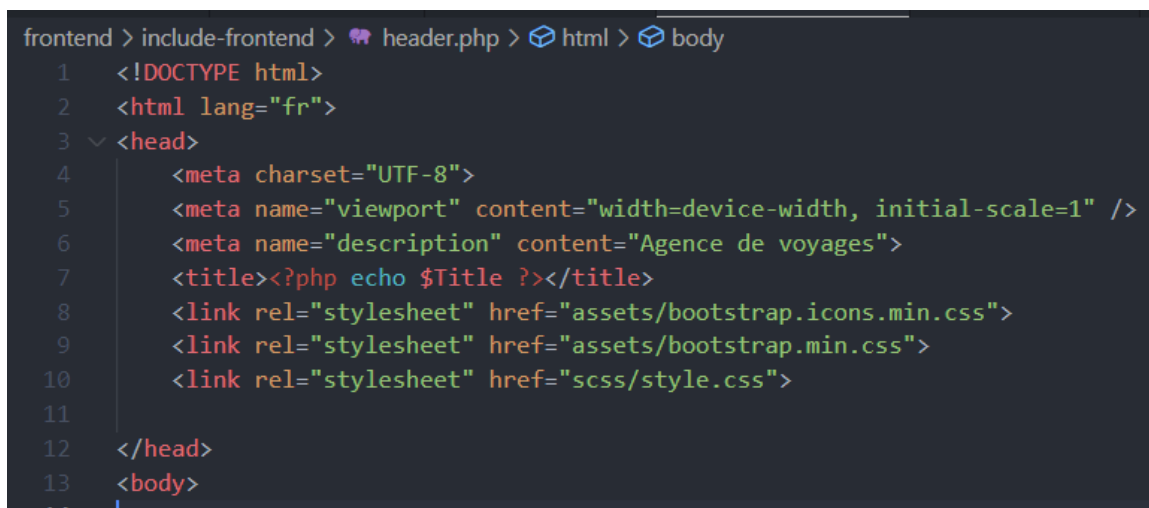
Extrait :



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with 'backend' and 'frontend' folders. The 'frontend' folder is expanded, showing 'Readme.md'. The code editor displays the 'style.scss' file with the following content:

```
1
2 $primary: #0C273D;
3 $secondary: #FFFEF1;
4 $tertio: #0b2020;
5 $nature: #2C5E2E;
6 $autre2: #54D0ED;
7
8 @mixin page-formulaire{
9     display: flex;
10    justify-content: center;
11 }
12 @mixin formulaire {
13     margin:10px;
14     padding:10px;
15     legend{
16         color:#0b2020;
17         font-weight: bolder
18     }
19 }
20 html,
21 body {
22     margin: 0px;
23     padding:0px;
24     font-family: 'Lato', sans-serif;
25 }
```

Liaison entre fichiers



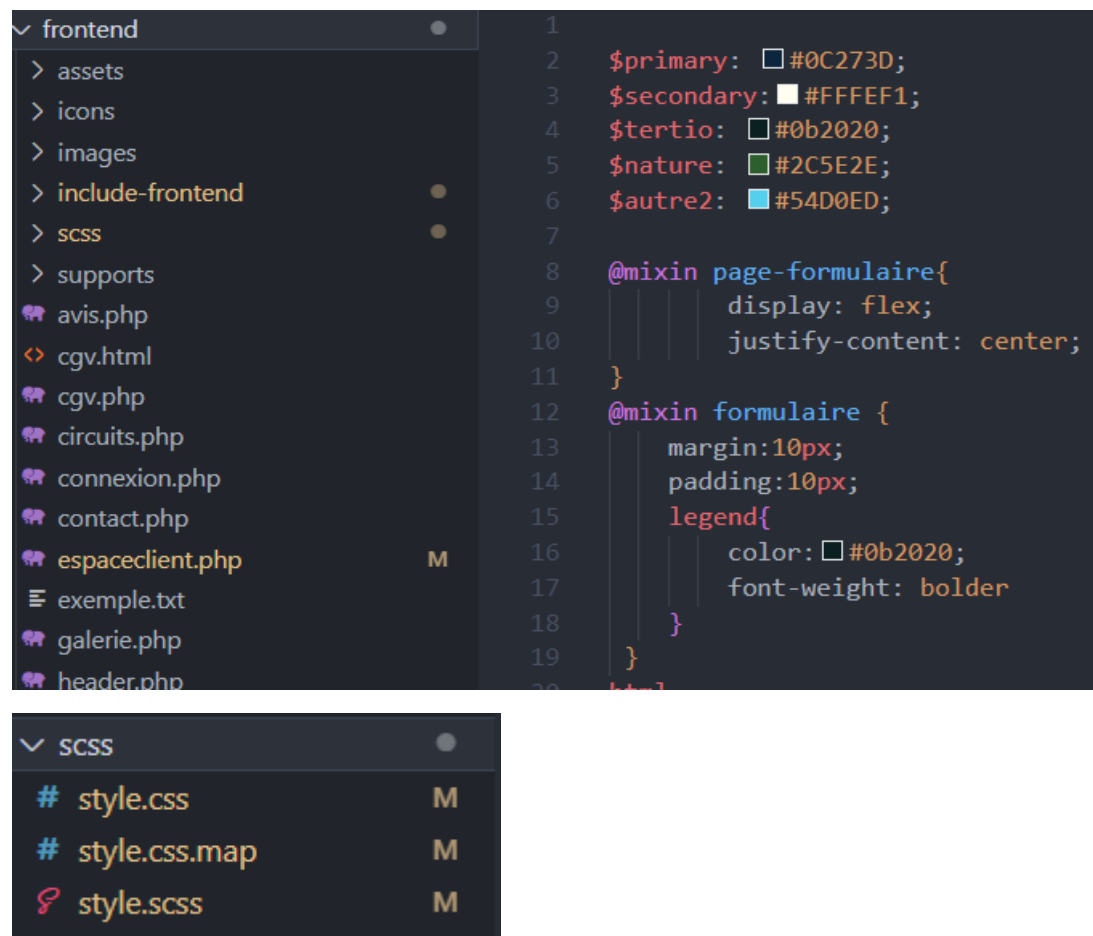
The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with 'frontend' and 'include-frontend' folders. The 'include-frontend' folder is expanded, showing 'header.php'. The code editor displays the 'header.php' file with the following content:

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1" />
6     <meta name="description" content="Agence de voyages">
7     <title><?php echo $Title ?></title>
8     <link rel="stylesheet" href="assets/bootstrap.icons.min.css">
9     <link rel="stylesheet" href="assets/bootstrap.min.css">
10    <link rel="stylesheet" href="scss/style.css">
11
12 </head>
13 <body>
14
```

## DOSSIER FRONTEND : arborescence des sous dossiers et fichiers

J'ai procédé à l'installation du Framework Bootstrap localement en téléchargeant le fichier depuis son site officiel, en l'enregistrant localement. J'aurais pu le faire à travers un gestionnaire de paquets (NPM), sauf qu'il y aurait eu autant de fichier dont je n'en avais forcément pas besoin pour ce type d'application.

Et je n'ai pas choisi non plus de travailler à partir du CDN en ligne pour éviter toute que le fichier soit indisponible à un moment à raison d'un bug, même si nous savons qu'un tel incident es évidemment rare.



L'image ci-dessus, illustre évidemment l'organisation du répertoire et la customisation de ma feuille de style.

A Gauche, l'arborescence de mon répertoire.

A droite, un extrait de ma feuille de style après customisation.

## RESPONSIVE

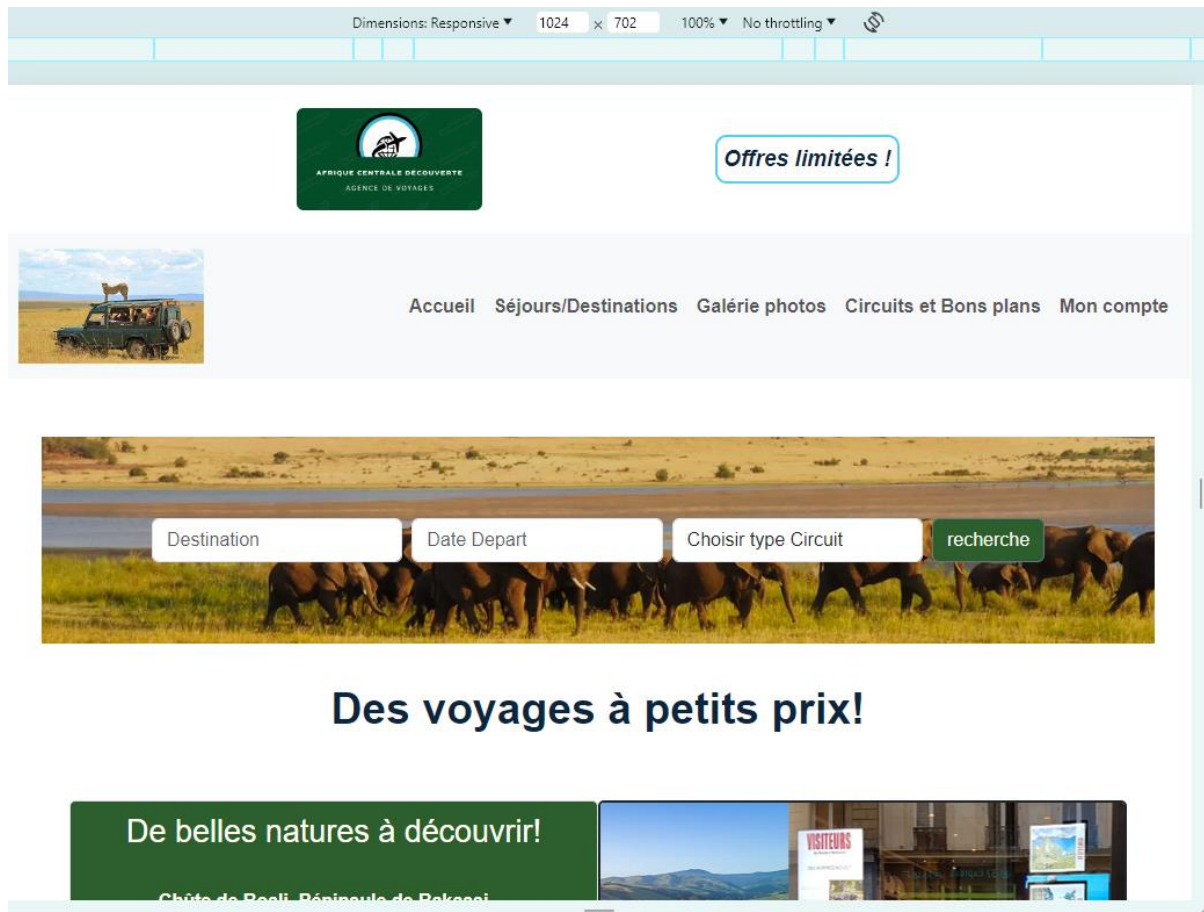
Il n'était quasiment pas nécessaire de faire usage des médias queries pour rendre responsive la quasi-totalité des pages de l'application.

Un bon usage des class containers de Bootstrap et ses breackpoints a permis à l'application de s'adapter à tous les écrans.

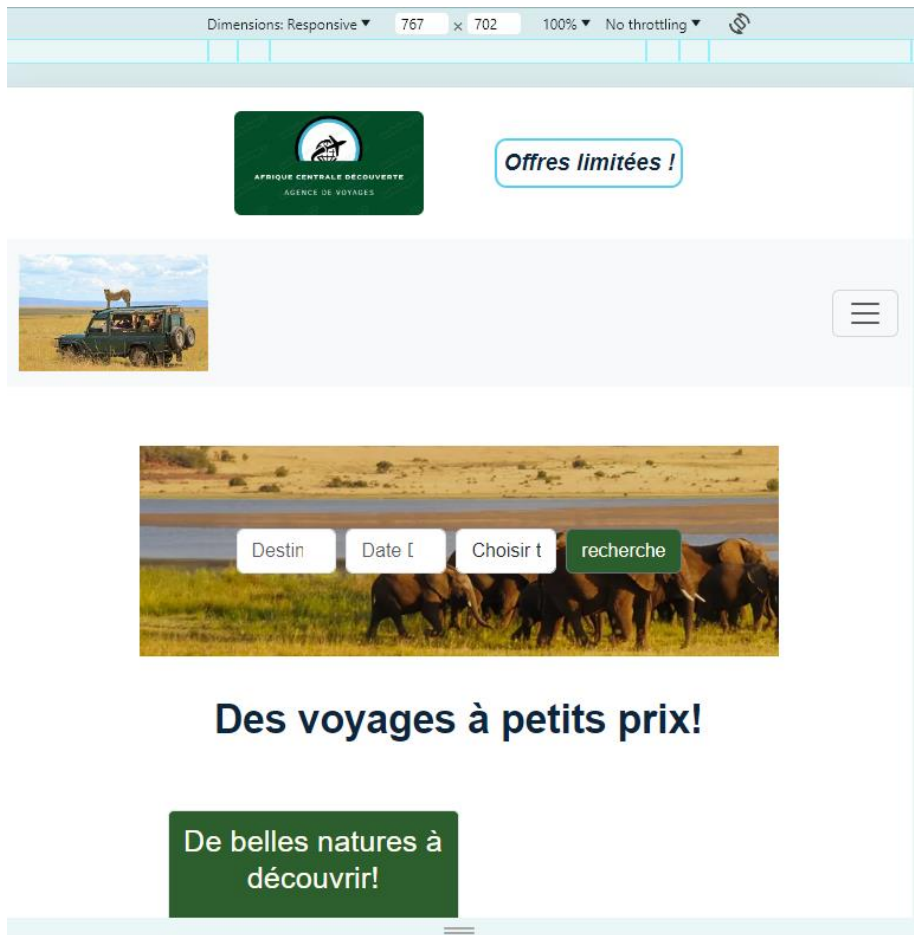


Même si, finalement ces derniers m'ont permis d'adapter davantage la disposition de certaines pages, notamment sur des plus petits écrans de tailles inférieures à 430px.

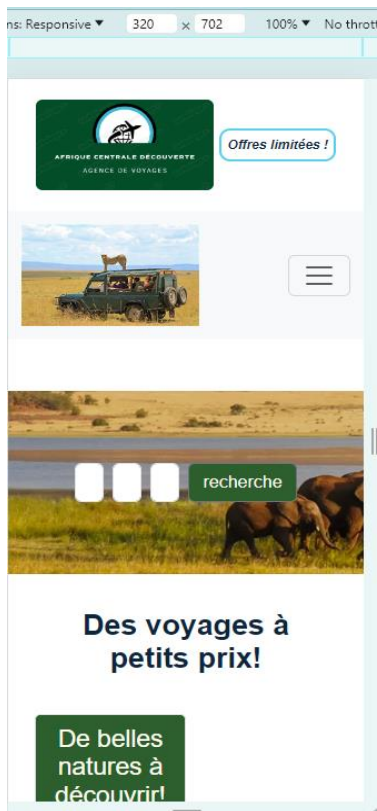
Extrait de l'écran en version desktop :



Le menu devient un menu burger une fois que la taille de l'écran est inférieure à 768 px et le reste du contenu s'adaptent.



Affichage version mobile taille écran 320 px



Extrait de la réalisation du carrousel en CSS.

Ce dernier se retrouve sur la page principale de l'application et s'adapte aux différents Breakpoints.

```
frontend > scss > style.scss > #info
92  /*carouselle*/
93  #caroussel{
94      width: 375px;
95      height: 230px;
96      padding:0px;
97      overflow: hidden;
98      border: 2px solid;
99      border-radius: 5px;
100 }
101 .images{
102     display: flex;
103     animation-duration:30s;
104     animation-direction: alternate;
105     animation-name: imagesAnimee;
106     animation-iteration-count: infinite;
107 }
108 @keyframes imagesAnimee{
109     0%{
110         transform: translateX(0);
111     }
112     20%{
113         transform:translateX(-350px);
114     }
115     40%{
116         transform:translateX(-700px);
117     }
118     60%{
119         transform:translateX(-1050px);
120 }
```

Rendu

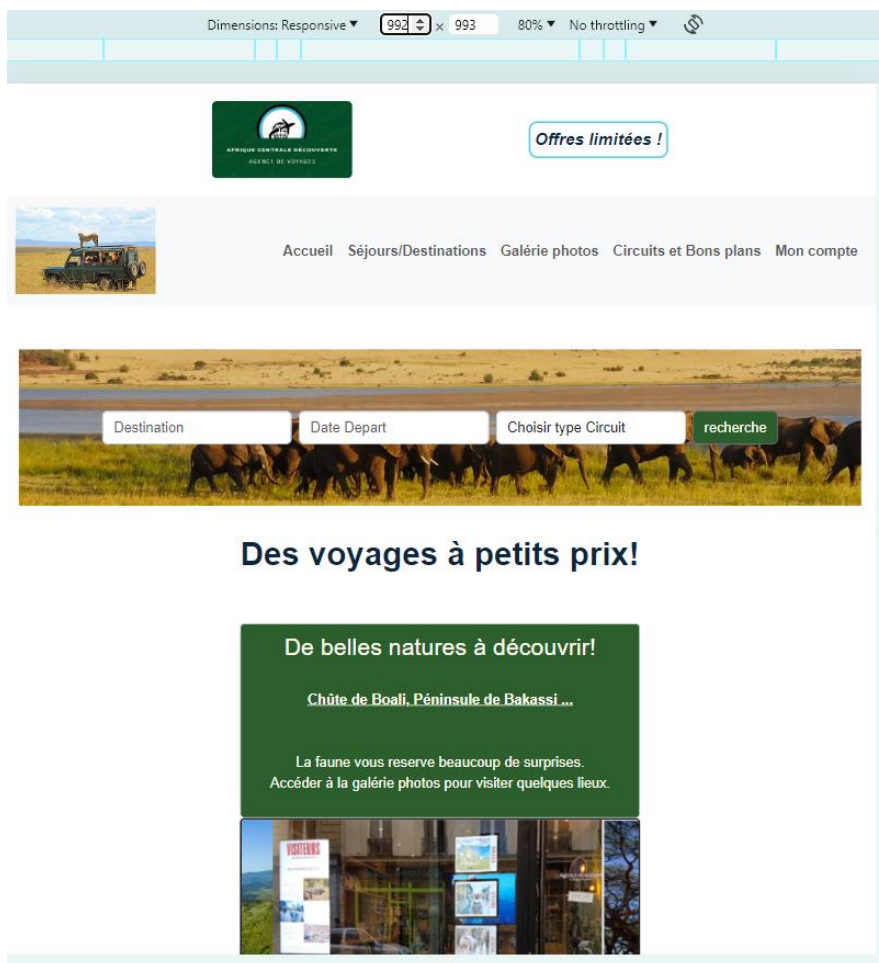


**MEDIA QUERIES**

Extrait du code permettant de réadapter l’affichage, à partir d’un écran inférieur à 992px. On remarque que le contenu principal est centralisé verticalement.

```
/* Média queries */
@media screen and (max-width: 992px){
  .apropos {
    flex-direction: column;
    justify-content: center;
    align-items: center;
  }
}
@media screen and (max-width: 768px){
  #carroussel{
    width: 375px;
    height: 230px;
  }
  #info{
    width: 375px;
    height: 230px;
    margin-bottom: 5px;
  }
}
```

## Résultat obtenu



**JAVASCRIPT :** Grâce à javascript et JQuery, j'ai pu rendre dynamique mes interfaces utilisateurs, évidemment la partie frontend de l'application.

Afin d'être plus organiser sur les données attendues. Ci-dessous un extrait des formulaires (interface utilisateurs) ainsi que les scripts js dynamique.

```
<script>
let Myform = document.getElementById('myform');

Myform.addEventListener('submit',function(e){

    let inputNom = document.getElementById('name')
    let inputNote = document.getElementById('note')
    let Commentaire = document.getElementById('commentaire')
    let myRegex = /^[a-zA-Z]+$/;

    if (inputNom.value == ""){
        let Erreur = document.getElementById('erreur');
        Erreur.innerHTML = "Veuillez saisir votre nom";
        Erreur.style.color = 'red';
        e.preventDefault();
    }else if (myRegex.test(inputNom.value) == false){
        let Erreur = document.getElementById('erreur');
        Erreur.innerHTML = "Le nom doit composer que de lettres sans espace ni tiret";
        Erreur.style.color = 'red';
        e.preventDefault();
    }
    if (inputNote.value == ""){
        let Erreur2 = document.getElementById('erreur2');
        Erreur2.innerHTML = "Veuillez choisir une note";
        Erreur2.style.color = 'red';
        e.preventDefault();
    }
    if (Commentaire.value == ""){
        let Erreur3 = document.getElementById('erreur3');
```

Contrôle de saisie des caractères en javascript, formulaire avis client.

## Avis

### Laissez-nous votre avis

Pseudo / Nom :\*

Le nom doit composer que de lettres sans espace ni tiret

Note :\*

Commentaire :\*

## Réalisation de calcul de données en JQuery

```
/* Récupération du prix initial du circuit */
var prixCircuit = $('#prix').val();
var nombrePassager = $('#nombre');
/* Détection du changement sur le champ Formule */
nombrePassager.on('change', function() {
    /* Récupération de la valeur du champ */
    var valeurNombrePassager = parseInt($(this).val());
    /* Les différents cas de figure */
    switch(valeurNombrePassager) {
        case 1:
        case 2:
        case 3:
            /* Calcul du prix total */
            var prixTotal = prixCircuit * valeurNombrePassager;
            /* Assignment de la valeur dans le champ dédié */
            $('#prix_total').val(prixTotal);
            break;
        default:
            /* Affichage d'un message d'erreur en cas d'une mauvaise sélection de l'utilisateur */
            $('#erreur-nombre').html("Veuillez choisir une formule !");
            break;
    }
});
})(jQuery);
```

Rendu visuel : Le prix du circuit multiplié par le nombre de personnes, affichage du résultat dans le champ prix total de la réservation.

## Reservations

### Choix formule et type de règlement

**Prix Circuit :**

85000 F.CFA

85000

**Nombre de personnes : \***

Couple

**Prix total de la réservation :**

170000

**Type de règlement : \***

--Choisir--

Valider ma reservation

**Extrait :** Formulaire de contact et contrôle des données en Javascript.

## Contactez l'agence ?

**Pour toutes vos questions, joignez-nous!**

Type demande : \*

-- Veuillez renseigner le type de votre demande --

Nom : \*

Dupont

E-mail: \*

monadresse@.....

Telephone:

+236....

Message:\*

Ecrivez ici votre message...

Envoyer

```
let Form = document.getElementById('form');
Form.addEventListener('submit', (e) =>{

    let Motif = document.getElementById('motif');
    let inputNom = document.getElementById('nom');
    let inputEmail = document.getElementById('email');
    let inputTel = document.getElementById('tel');
    let Message = document.getElementById('message');

    let nomformat = /^[a-zA-Z-\s]+$/;
    let mailformat = /^[\\w-\\.]+@[\\w-]+\\.+([\\w-]{2,4})$/;
    let telformat = /^[0-9]*$/;

    //controle saisie champ obligatoire motif//
    if (Motif.value.trim() == ""){
        let Erreurtype = document.getElementById('erreurtype');
        Erreurtype.innerHTML = "Vous devez choisir un type!";
        Erreurtype.style.color = 'red';
        e.preventDefault();
    }
});
```

```
document.getElementById('email').addEventListener('input', function (e){
let mailformat = /^[\\w-\\.]+@[\\w-]+\\.+([\\w-]{2,3})$/

if (mailformat.test(email.value) == false){
let Erreurmail = document.querySelector('#erreurmail');

Erreurmail.innerHTML = "Format email invalide !";
Erreurmail.style.color = 'red';
e.preventDefault();
}else {
    let Erreurmail = document.querySelector('#erreurmail');
    Erreurmail.innerHTML = "email valide !";
    Erreurmail.style.color = 'green';
}
});
```



Extrait : interface utilisateur.

# Formulaire d'inscription

Je m'inscris pour profiter des offres !

Civilité : \*

--Indiquer votre civilité--

Nom : \*

Prenom : \*

Isabelle

Age : \*

--Votre age--

Nationalité: \*

Telephone: \*

--Votre Nationalité--

E-mail: \*

Mot de passe: \*

adresse@.

mot de pass

Envoyer

Déjà Inscrit(e)? [Connectez-vous](#)

```
<main class="container inscription">
  <section>
    <form id="form_inscription" method="POST" action="">
      <fieldset>
        <legend>Je m'inscris pour profiter des offres !</legend>
        <div class="mb-3">
          <label class="form-label"><b> Civilité : *</b></label>
          <select class="form-control" name="civilité" id="civil">
            <option value="">--Indiquer votre civilité--</option>
            <option value="Monsieur">Monsieur</option>
            <option value="Madame">Madame</option>
          </select>
          <span id="erreurcivilité"></span>
          <?php
            if(isset($_GET['civilité']) && ($_GET['civilité']==1)){
              echo '<span class="red"> Veuillez indiquer votre civilité </span>';
            }
          <?>
        </div>
        <div class="row mb-3">
          <div class="col">
            <label class="form-label"><b>Nom : *</b></label>
            <input class="form-control" type="text" name="nom" id="nom" maxlength="15">
            <span id="erreurnom"></span>
            <?php
              if(isset($_GET['nom']) && ($_GET['nom']==1)){
                echo '<span class="red"> Veuillez saisir votre nom </span>';
              }
            <?>
          </div>
        </div>
      </fieldset>
    </form>
  </section>
</main>
```



Extrait : contrôle de données interactif en javascript.

```
document.getElementById('email').addEventListener('input', function (e){
let emailRegex = /^[\\w-\\.]+@([\\w-]+\\.){2,3}$/;

if (emailRegex.test(email.value) == false){
let Erreurmail = document.querySelector('#erreuremail');

Erreurmail.innerHTML = "Format email invalide !";
Erreurmail.style.color = 'red';
e.preventDefault();
}else {
let Erreurmail = document.querySelector('#erreuremail');
Erreurmail.innerHTML = "Champ valide !";
Erreurmail.style.color = 'green';
}
});

document.getElementById('password').addEventListener('input', function (e){
let passRegex = /^[a-zA-Z+[0-9]+[#?!@$$%^&*~]+$/;

if (passRegex.test(password.value) == false){
let Erreurpassword = document.querySelector('#erreurpassword');

Erreurpassword.innerHTML = "Champ invalide !";
Erreurpassword.style.color = 'red';
e.preventDefault();
}else {
let Erreurpassword = document.querySelector('#erreurpassword');
Erreurpassword.innerHTML = "Champ valide !";
Erreurpassword.style.color = 'green';
}
});
```

```
let Form_inscription = document.getElementById('form_inscription');

Form_inscription.addEventListener('submit',function(e){

let Civilete = document.getElementById('civil')
let Name = document.getElementById('name')
let Prenom = document.getElementById('prenom')
let Tel = document.getElementById('telephone')
let Age = document.getElementById('age')
let Nationalite = document.getElementById('nationalite')
let Email = document.getElementById('email')
let Password = document.getElementById('password')

let inputRegex = /^[a-zA-Z]+$/;
let telRegex = /^[0-9]+$/;
let emailRegex = /^[\\w-\\.]+@([\\w-]+\\.){2,4}$/
let passwordRegex = /^[a-z]+[A-Z{1},0-9]$/;

if (Civilete.value.trim() == ""){
let Erreurecivilete = document.getElementById('erreurecivilete');
Erreurecivilete.innerHTML = "Le champ Civilité est obligatoire !";
Erreurecivilete.style.color = 'red';
e.preventDefault();
}
if (Name.value.trim() == ""){
let Erreurnom = document.getElementById('erreurnom');
Erreurnom.innerHTML = "Le champ Nom est obligatoire!";
Erreurnom.style.color = 'red';
e.preventDefault();
}else if (inputRegex.test(Name.value) == false){
```

## **DEUXIEME PARTIE : Backend**

### **Contenu du dossier Backend :**

- ✓ Un dossier asset contenant les feuilles de style CSS et Bootstrap pour une mise en page de l'espace administrateur et client au besoin.
- ✓ Un dossier include, contenant les différents fichiers à rappeler.
- ✓ Un dossier pages, contenant les différentes fonctionnalités implémentées
- ✓ Un dossier Uploads et image éventuellement.
- ✓ Fichier.sql, contenant les scripts de la création des tables de la base de données.
- ✓ Index.php qui représente la page principale backend (Dashboard).

Pour la création et administration de ma base de données, j'ai choisi la méthode **UML**, pour mes analyses et études de cas.

### **Tâches à réaliser :**

- ☐ La création d'une base de données.
- ☐ Etablir la communication entre le frontend et le backend d'une application à travers les requêtes via le protocole HTTP. Nos formulaires déjà conçus précédemment nous serviront à cet effet (formulaire contact, espace client, formulaire d'inscription, boutons liés aux différents types de services proposés.
- ☐ Manipulation des données (CRUD)
- ☐ Gestion de rôles des utilisateurs
- ☐ La sécurité de l'application.

### **Rappel des Spécifications techniques :**

L'élaboration des besoins sur la base des diagrammes de structure (**UML**) se décompose de façon suivante.

Considérant que :

- ❑ **Une réservation** correspond à un voyage = un circuit (seul, en couple ou en groupe), une case à cocher ou liste de choix est proposé à cet effet. Elle comporte un libellé, une date, un prix, une image)
- ❑ **Un circuit**, égale à un voyage et comporte un type (obligatoirement un vol (aller/retour) simple ou alors un vol (aller/retour) + réservation d'hôtel. Il comporte une destination, des dates aller et retour, un prix et une photo du lieu).

### Table et Données attendues :

- ✓ **Client** (**id client**, civilité, nom, prénom, âge, nationalité, téléphone, email, mot de pass, date\_création, statut\_compte)
- ✓ **Réservation** (**id réservation**, nombre de personnes, (le client voyage seul, en couple ou un groupe de personnes), prix, type\_réglelement, statut **id circuit**)
- ✓ **Circuits** (**id circuit**, destination, date départ, date retour, prix, image, type circuit (vol aller/retour simple ou alors vol aller/retour + hôtel).
- ✓ **Avis** (**id avis**, pseudo, note, commentaire date\_avis).
- ✓ **Contact** (**id contact**, type demande, nom, email, téléphone, message date contact).
- ✓ **Admin** (**id**, email, password).

## Conception de la Base de données

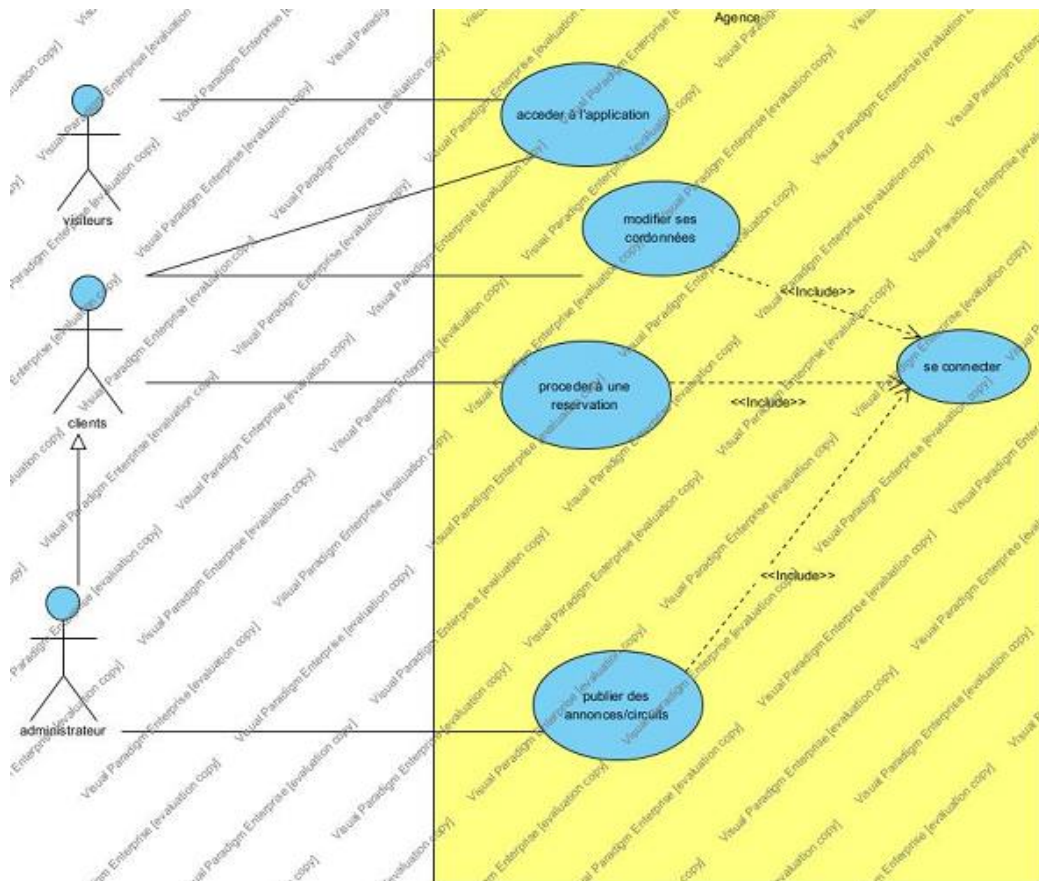
Méthode utilisée : **UML**

L'UML grâce à ses diagrammes de structure et comportement, m'a permis de modéliser la conception mon application.

Grâce à ce dernier, j'ai établi des interactions entre mes entités/acteurs, et implémenté les fonctionnalités.

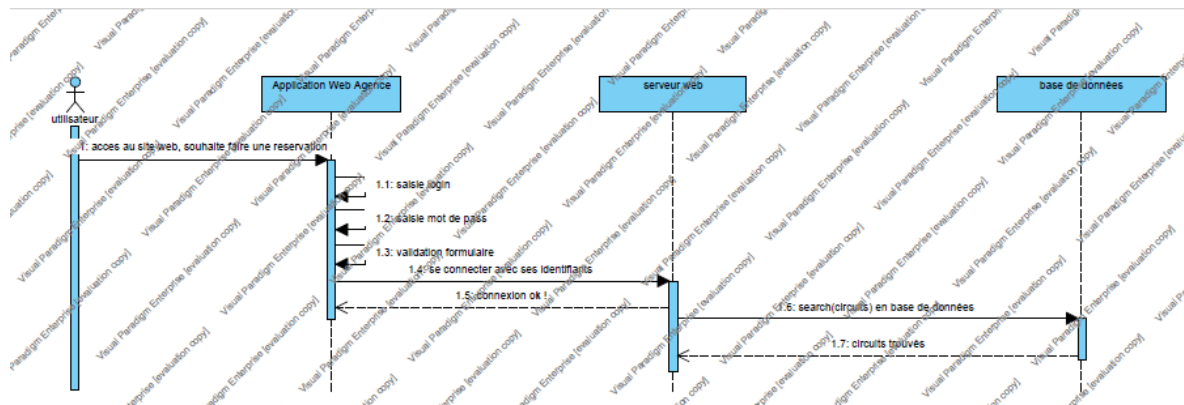
Ci-dessous un extrait des diagrammes utilisés :

### Diagramme de cas d'utilisation

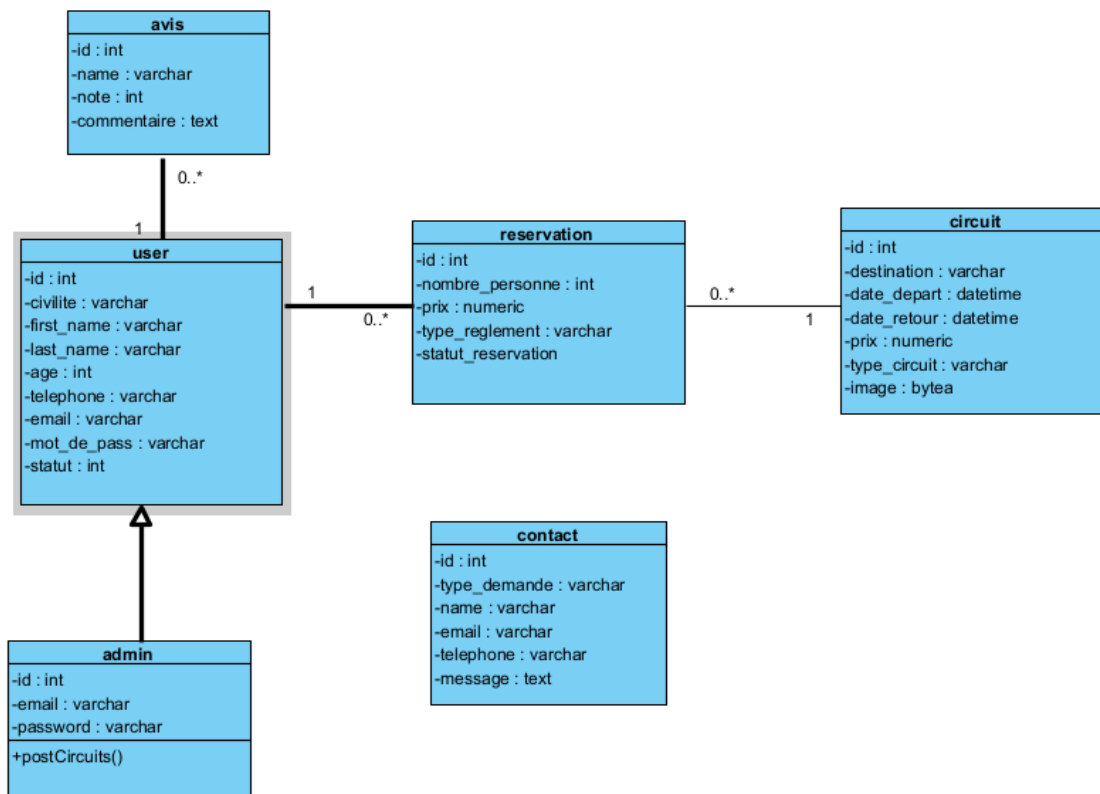


Ce diagramme illustre les cas d'utilisations principales de l'application.

## Diagramme de séquence



## Diagramme de classes



Ci-dessus l'extrait du diagramme de classes indiquant toutes les entités de l'application.

## Description des tables de données et relations

Un utilisateur (client) peut faire plusieurs réservations sur différentes dates. Une réservation concerne un seul client.

Un circuit peut faire l'objet de plusieurs réservations. Une réservation correspond et concerne un seul circuit.

Ainsi, la table réservation possède 2 clés étrangères des tables clients et circuits qui feront référence aux entités associées, c'est à dire à leurs clés primaires respectives.

La relation entre les clients et la tables avis si elle existe, indique qu'un client peut émettre plusieurs avis et qu'un avis peut être posté par un seul client.

## Structure physique des données

Grace aux classes de mon diagramme de classe, j'ai pu élaborer les différentes tables que compose ma base de données.

Les propriétés de mes classes deviennent notamment des champs dans la base de données, ci-dessous un extrait des tables principales.

```
CREATE TABLE IF NOT EXISTS agence.client
(
    id bigint NOT NULL DEFAULT nextval('agence.client_id_seq'::regclass),
    civilite character varying COLLATE pg_catalog."default" NOT NULL,
    nom character varying COLLATE pg_catalog."default" NOT NULL,
    prenom character varying COLLATE pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    nationalite character varying COLLATE pg_catalog."default" NOT NULL,
    telephone character varying COLLATE pg_catalog."default" NOT NULL,
    email character varying COLLATE pg_catalog."default" NOT NULL,
    mot_de_pass character varying COLLATE pg_catalog."default" NOT NULL,
    date_inscription timestamp with time zone NOT NULL,
    statut_client integer,
    CONSTRAINT client_pkey PRIMARY KEY (id),
    CONSTRAINT mail UNIQUE (email)
)
```

```

CREATE TABLE IF NOT EXISTS agence.reservations
(
    id bigint NOT NULL DEFAULT nextval('agence.reservations_id_seq'::regclass),
    nombre_personne integer NOT NULL,
    prix numeric NOT NULL,
    type_reglement character varying COLLATE pg_catalog."default" NOT NULL,
    id_circuit integer NOT NULL,
    date_reservation timestamp with time zone NOT NULL,
    statut character varying COLLATE pg_catalog."default",
    id_client integer,
    CONSTRAINT reservations_pkey PRIMARY KEY (id),
    CONSTRAINT id_circuit FOREIGN KEY (id_circuit)
        REFERENCES agence.circuits (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT id_client FOREIGN KEY (id_client)
        REFERENCES agence.client (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

```

```

CREATE TABLE IF NOT EXISTS agence.circuits
(
    id bigint NOT NULL DEFAULT nextval('agence.circuits_id_seq'::regclass),
    destination character varying COLLATE pg_catalog."default" NOT NULL,
    date_depart date NOT NULL,
    date_retour date NOT NULL,
    prix numeric NOT NULL,
    type_circuit character varying COLLATE pg_catalog."default" NOT NULL,
    date timestamp with time zone,
    image character varying COLLATE pg_catalog."default",
    CONSTRAINT circuits_pkey PRIMARY KEY (id)
)

```

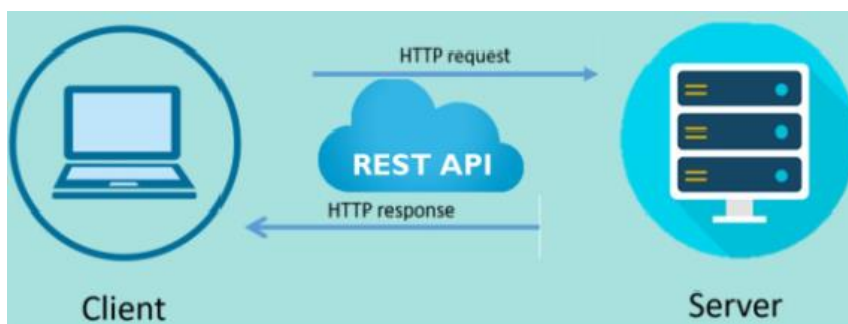
Nous pouvons remarqué l’usage de **CRUD** à travers les requêtes de créations des tables de données . **CREATE**

## Communication entre le frontend et backend.

### Mode de communication : HTTP

Le projet est développé en local, notamment sous le nom du serveur localhost. Des ordres envoyés par notre application frontend au backend se font à travers un navigateur web, via l'URL et sous format alphanumérique (HTTP Request / requête-réponse entre un navigateur-client et un serveur).

Le Backend dans sa réponse communique avec le frontend dans un format Jason : {"nom de la variable" : "valeur variable"}



**Méthode d'envoi :** Get , Post, Put Delete , Update,

Ci-dessous un exemple, d'envois de requêtes.

Ces envois sont effectués en POST, préféré ici à la Méthode GET faisant transiter les informations de façon visible et non sécurisée.

Réponse Backend est composée de différents code de retour ainsi que de messages indiquant les différents statuts si la requête a abouti ou pas, tout en spécifiant les raisons ou difficultés liés aux échecs ou incidents.

Exemple : Une requête pour enregistrer un utilisateur en base de données échouée. Car il existe une contrainte email unique qui rend impossible d'enregistrer une même adresse email plusieurs fois

HTTP Request (POST) : URL avant l'envoi.

Exemple d'une réponse du serveur

PDOException: SQLSTATE[23505]: Unique violation: 7 ERREUR: la valeur d'une clé dupliquée rompt la contrainte unique « email » DETAIL: La clé « (email)=(emiliedup@yahoo.fr) » existe déjà.

**Gestion et manipulation des données :** PHP / Objet PDO



Le langage PHP à travers son extension Data Object (PDO) propose une interface pour accéder aux différentes bases de données, selon les pilotes.

Celui utilisé pour cette application est bien le PDO\_PGSQL.

Grâce à ce dernier j'ai pu établir la connexion à la base de données, l'interroger et communiquer de façon permanente avec elle.

Extrait code établissant la connexion à la base de données avec la possibilité de traiter une exception en cas d'incident (Méthode pouvant garantir une forme de sécurité ne pas laisser afficher les informations sensibles de la base de données).

```
backend > connexion.php > ...
1  <?php
2
3  $host = 'localhost';
4  $dbname = 'applications';
5  $username = 'postgres';
6  $password = 'Zidane1010.';
7
8  $dsn="pgsql:host=$host;port=5433;dbname=$dbname;";
9
10 try{
11     $conn = new PDO ($dsn, $username, $password);
12
13     if($conn){
14         // echo 'Vous êtes bien connecté à la base de données';
15     }
16 }catch (PDOException $e){
17     echo $e->getMessage();
18 }
19
20 ?>
```

La connexion a pu être établie à partir d'un **DNS**, composé du nom du serveur (**localhost**), du nom de la base de données (**applications**) et du nom de son utilisateur ou propriétaire (**Postgres**).

Ce DNS est repris par l'objet PDO en premier paramètre avec l'identifiant de connexion en seconde lieu pour enfin permettre la connexion.

**CRUD** : Create, Read, Update, Delete, considérés comme des verbes, ils demeurent le principal mécanisme de communication, utilisé pour le traitement des données de l'application (requête, interrogation de la base de données).

**INSERT** : Insertion des données en Base de données depuis un formulaire et grâce à l'objet PDO.

## Formulaire d'inscription

**Je m'inscris pour profiter des offres !**

Civilité : \*

--Indiquer votre civilité--

Nom : \*

Prenom : \*

Isabelle

Age : \*

--Votre age--

Nationalité: \*

--Votre Nationalité--

Telephone: \*

E-mail: \*

adresse@.

Mot de passe: \*

mot de pass

Envoyer

Déjà Inscrit(e)? [Connectez-vous](#)

Exemple : Une requête pour enregistrer un utilisateur en base de données échouée, suite à une contrainte email unique rendant impossible le stockage d'une même adresse email plusieurs fois.

HTTP Request (POST) : URL avant l'envoi.

localhost/afrique-centrale-decouverte/backend/inscription.php

### Exemple d'une réponse serveur

PDOException: SQLSTATE[23505]: Unique violation: 7 ERREUR: la valeur d'une clé dupliquée rompt la contrainte unique « email » DETAIL: La clé « (email)=(emilledup@yahoo.fr) » existe déjà.

```
$verifEmail = 'SELECT * FROM agence.client WHERE email = :email';
$stmt = $conn -> prepare($verifEmail);
$stmt -> bindValue(':email', $email);
$result = $stmt -> execute ();
if($result == true){
    header("location:?existe=1");
}

$insertClient = 'INSERT INTO agence.client(civilite, nom, prenom, age, nationalite, telephone, email, mot_de
values (:civilite, :nom, :prenom, :age, :nationalite, :tel, :email, :pwd, :date)';

$reqInsertion = $conn -> prepare ($insertClient);
$save = $reqInsertion->execute([
```

**SELECT** : sélectionner des données (Interrogation de la base de données pour afficher des circuits à partir d'un certain critère, la pagination de la rubrique circuits).

Extrait de la requête permettant d'afficher une liste de circuits.

```
<?php
/* Nombre de circuit à afficher par page */
$pagelimit = 12;

/* Nombre total des circuits en base de données */
$reqTotalCircuit = "SELECT COUNT(id) AS nb FROM agence.circuits";
$tdrTotalCircuit = $conn -> query($reqTotalCircuit);
$tdrTotalCircuit -> execute();
$resultatTotalCircuit = $tdrTotalCircuit -> fetch();
$nbTotalCircuit = $resultatTotalCircuit['nb'];

$totalPage = ceil($nbTotalCircuit / $pagelimit);

/* Récupération du numéro de la page sélectionnée par l'utilisateur */
$numeroPage = (isset($_GET['page'])) ? $_GET['page'] : 1;

$start = ($numeroPage - 1) * $pagelimit;
$end = $pagelimit;

//echo "Nombre circuit par page {$pagelimit}<br/>Total circuit en base : {$nbTotalCircuit}";
$reqaffich = "SELECT * FROM agence.circuits ORDER BY id LIMIT $end OFFSET $start";

$tdr = $conn -> query($reqaffich);
$result = $tdr -> fetchAll();

foreach($result as $key => $value) {
?>
```

Résultat permettant d'afficher une liste spécifique de circuits.



**Destination :** Bakassi  
**Type Circuit :** aller/retour simple  
**Date de départ :** 2024-04-09  
**Date de retour :** 2024-04-16  
**Référence Annonce :** 49A24  
**Prix Circuit:** 165000  
[Je réserve](#)



**Destination :** Bakassi  
**Type Circuit :** aller/retour simple + hotel  
**Date de départ :** 2024-04-09  
**Date de retour :** 2024-04-16  
**Référence Annonce :** 50A24  
**Prix Circuit:** 235000  
[Je réserve](#)



**Destination :** Pongara  
**Type Circuit :** aller/retour simple  
**Date de départ :** 2024-04-09  
**Date de retour :** 2024-04-16  
**Référence Annonce :** 51A24  
**Prix Circuit:** 170000  
[Je réserve](#)

**INSERTION** : insérer des données (Page contact).

Extrait du script consistant à modifier insérer des données depuis un formulaire

```
$req='INSERT INTO agence.contacts(type_demande, nom, email, telephone, message, date_contact)
values (:motif, :nom, :email, :telephone, :message, :date)';
$reqInsertion = $conn -> prepare ($req);
$enregister = $reqInsertion->execute([
```

l' url avant :   localhost/afrique-centrale-decouverte/frontend/contact.php

## Contactez l'agence ?

**Pour toutes vos questions, joignez-nous!**

Type demande : \*

-- Veuillez renseigner le type de votre demande --

Nom : \*

Dupont

E-mail: \*

monadresse@.....


Telephone:

+236....

Message:\*

Ecrivez ici votre message...

Envoyer

Url : Rédirection client:  localhost/afrique-centrale-decouverte/frontend/succes-validation.php

## Formulaire envoyé avec succès !

Merci de votre visite à bientôt !

Confirmation de la démarche si tout se passe dans les conditions établies dans le code.

**UPDATE :** Modification

Extrait du script consistant à modifier les informations personnelles du client.

```
function validation_donnees($donnees){
    $donnees = trim($donnees);
    $donnees = stripslashes($donnees);
    $donnees = htmlspecialchars($donnees);
    return $donnees;
}

$id = ($_SESSION["donnees_client"]['id']);
$nom = validation_donnees($_POST['nom']);
$telephone = validation_donnees($_POST['tel']);
$nationalite = validation_donnees($_POST['nat']);

$reqModif = 'UPDATE agence.client SET nom =:nom, telephone =:tel, nationalite =:nat WHERE id =:id';
$stmt = $conn -> prepare($reqModif);
$stmt -> bindValue(':id',$id);
$stmt -> bindValue(':nom',$nom);
$stmt -> bindValue(':tel',$telephone);
$stmt -> bindValue(':nat',$nationalite);
$valider = $stmt-> execute();

if($valider){
    header("location:?pages=gestion-compteclient.php&probleme=1");
}
```

Statut Compte

Mes réservations

Informations personnelles

Bonjour, Alain [Me déconnecter](#) 

## Modifier mes informations personnelles

### Mise à jour des données

Nom :

MATONGO

Telephone:

00243548778778

Nationalité :

Congolaise

Valider mes modifications

**DELETE** : Suppression des données

Extrait du script consistant à supprimer des circuits depuis le Backend.

```
//suppression de circuits //
if(array_key_exists('valider',$_POST)) {

    $id = $_POST['id'];
    $reqsupp = 'DELETE FROM agence.circuits WHERE id = :id';
    $statement = $conn -> prepare($reqsupp);
    $statement -> bindValue(':id',$id);
    $supp = $statement -> execute();
    if($supp) {
        header('location:?gestion-circuit.php&suppression=1');
        exit();
    }
}
```

TABLEAU DE BORD

GÉRER LES CLIENTS

GÉRER LES CIRCUITS

GÉRER LES RÉSERVATIONS

GÉRER LES AVIS

## Gestion des circuits

	Date de retour	Prix	Type_circuit	Date	Actions
	2024-04-26			2-14 08:02:01+01	<div>Modifier</div> <div>supprimer</div>
	2024-04-19			2-14 08:02:03+01	<div>Modifier</div> <div>supprimer</div>

localhost

Vous confirmez cette suppression 60 ?

OK

Annuler

```
<form method="POST" action="">
    <input type="hidden" name="id" value="<?php echo $value['id']; ?>" readonly="true">
    <input type="text" name="prix" placeholder="Indiquer le nouveaau prix">
    <!--<a class="btn btn-warning" href="modif-circuit.php">Modifier</a-->

    <button class="btn btn-warning btn-sm" type="submit" name="modifier" id="modifier"
    onclick="return confirm('Confirmez-vous la modification <?php echo $value['id']; ?> ?')">
        Modifier</button>
    <button class="btn btn-danger btn-sm" type="submit" name="valider"
    onclick="return confirm('Confirmez-vous cette suppression <?php echo $value['id']; ?> ?')">
        supprimer</button>
</form>
```

**La pagination : traitement Backend**




```
<nav aria-label="navigation">
  <ul class="pagination justify-content-center">
    <!-- Lien page précédente désactivé si on se trouve sur la page de début-->
    <li class="page-item <?= ($numeroPage == 1) ? "disabled" : "" ?>">
      <a href="circuits.php?<?= $numeroPage - 1 ?>" class="page-link text-primary2">Précédant</a>
    </li>
    <?php for($i = 1; $i <= $totalPage; $i++)
    {
      ?>
      <li class="page-item"></li>
      <a href="circuits.php?page=<?php echo $i; ?>" class="page-link text-primary2"><?php echo $i; ?></a>
    </li>
    <?php
    }
    ?>
    <!-- Lien page suivante désactivé si on se trouve sur la page de fin-->
    <li class="page-item <?= ($numeroPage == $totalPage) ? "disabled" : "" ?>">
      <a href="circuits.php? <?= $numeroPage + 1 ?>" class="page-link text-primary2">Suivant</a>
    </li>
  </ul>
</nav>
```



Destination : Pongara  
Type Circuit : aller/retour simple + hotel  
Date de départ : 2024-04-09  
Date de retour : 2024-04-16  
Référence Annonce : 52A24  
Prix Circuit: 240000



Destination : Dzanga-Sangha  
Type Circuit : aller/retour simple  
Date de départ : 2024-04-12  
Date de retour : 2024-04-19  
Référence Annonce : 53A24  
Prix Circuit: 95000



Destination : Dzanga-Sangha  
Type Circuit : aller/retour simple + hotel  
Date de départ : 2024-04-12  
Date de retour : 2024-04-19  
Référence Annonce : 54A24  
Prix Circuit: 130000

Je m'identifie pour réserver

```
<?php
/* Nombre de circuit à afficher par page */
$pagelimit = 12;
/* Nombre total des circuits en base de données */
$reqTotalCircuit = "SELECT COUNT(id) AS nb FROM agence.circuits";
$tdrTotalCircuit = $conn -> query($reqTotalCircuit);
$tdrTotalCircuit -> execute();
$resultatTotalCircuit = $tdrTotalCircuit -> fetch();
$nbTotalCircuit = $resultatTotalCircuit['nb'];

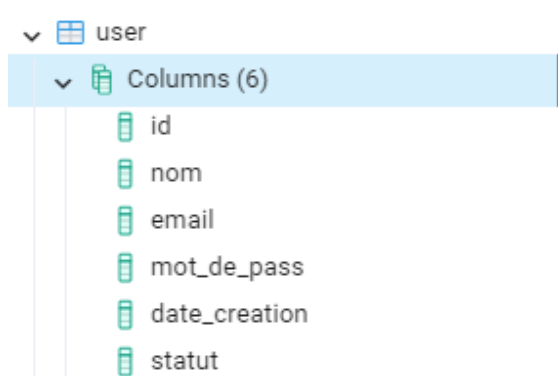
$totalPage = ceil($nbTotalCircuit / $pagelimit);
/* Récupération du numéro de la page sélectionnée par l'utilisateur */
$numeroPage = (isset($_GET['page'])) ? $_GET['page'] : 1;
$start = ($numeroPage - 1) * $pagelimit;
$end = $pagelimit;
```

GESTION DES UTILISATEURS

Cette application dispose d'un espace administration, gérer cependant par un seul membre. On peut le remarquer à travers la table user, composée d'un id, un nom, une adresse email, un mot de pass et un statut pouvant permettre d'établir des rôles quand il s'avère nécessaire.

Ces rôles pourront être attribués à travers des requêtes et méthodes en base de données (CREATE, ALTER, GRANT, UPDATE, DELETE, REVOKE etc.) proposées par notre SGBD de POSTGRES :

A titre d'exemple d'ailleurs il existe une colonne statut de type Integer dans la table user servant à définir le rôle. Un champ statut = 1 indique le rôle admin puis statut = 2, indique un rôle employé. (Mieux prévoir table rôle séparées)



user	
Columns (6)	
id	
nom	
email	
mot_de_pass	
date_creation	
statut	

Un Extrait du code définissant droits ou accès aux données.

```
backend > include > menu-nav.php > nav.nav-bar.menu
1
2 <nav class="nav-bar menu">
3   <b><a href="tableau-de-bord.php">Tableau de bord</a></b>
4   <?php
5     /* statut 1 = Admin*/
6     //if(in_array([2], $_SESSION['donnees_user']['statut'])) {
7     if ($_SESSION['donnees_user']['statut']==1){
8     ?>
9     <b><a href="gestion-client.php">Gérer les Clients</a></b>
10    <?php
11    }
12    ?>
13    <?php
14    ?>
15    <b><a href="gestion-circuit.php">Gérer les circuits</a></b>
16    <?php
17    if ($_SESSION['donnees_user']['statut']==1){
18    ?>
19    <b><a href="gestion-reservation.php">Gérer les Réservations</a></b>
20    <?php
21    }
22    ?>
23    <b><a href="gestion-avis.php">Gérer les avis</a></b>
24  </nav>
```

Script de connexion au compte client.



```
$email = $_POST['email'];
$password = $_POST['mot_de_pass'];
//verifier si email existe en base de données //
$reqSelect = 'SELECT * FROM agence.client WHERE email = :email';
$reqExec = $conn -> prepare($reqSelect);
$reqExec -> bindvalue('email', $email);

$reqExec-> execute ([
    ":email" => $_POST['email'],
]);
$emailExist = $reqExec -> fetch(PDO::FETCH_ASSOC);
// verification email base données //
//client trouvé email correct //
if(!empty($emailExist)){
    //client trouvé//
    $passwordhash = $emailExist['mot_de_pass'];

    if(password_verify($password, $passwordhash)){
        $_SESSION['donnees_client'] = $emailExist;
        $emailExist = $clientConnecte;
        header("location:pages/tableau-de-bord-client.php");
        exit();
    }
}
```

Formulaire de connexion

## Accéder à mon Compte

Merci de vous Identifier, pour profiter de nos services!

Email : \*

Dupont

Mot de passe : \*

Mot de pass

Connexion

Nouveau? [Inscrivez-vous d'abord!](#)

Mot de passe oublié? [Réinitialiser mot de pass.](#)

Dashboard client

Statut Compte

Mes réservations

Informations personnelles

Bonjour, Alain [Me déconnecter](#) 

### Mon Compte Client

puis 2024-02-14 12:02:03+01

[Je laisse mon avis](#)

## GESTION DE LA SECURITE DE L'APPLICATION

La sécurité de l'application est un élément fondamental dans le domaine du développement web. Elle consiste à établir des règles, méthodes et traitements à caractère sécuritaire contre les pirates, contre des espions/fraudeurs ou encore contre différents types d'attaques malveillantes.

Lors de ma formation, j'ai appris aussi bien la plupart des méthodes utilisées par les attaquants et des mesures de sécurité à mettre en place en amont du développement.

- ✓ Sécurisé le serveur : avec des certificats serveur **SSL / TLS (HTTPS)** en l'activant ou achetant éventuellement.
- ✓ Assurer une **veille technologique** : Par exemple se renseigner de façon permanente, rester en contact des informations via des sites web professionnellement réputés.
- ✓ **Les failles d'injection SQL** également appelées SQLi. Ce genre de faille engendre de risque d'exposition et la divulgation de données sensibles, elles sont liées essentiellement à la base de données.

Pour y faire face, l'utilisation de requêtes préparées ou paramétrées doit être systématiquement envisagée.

La quasi-totalité des requêtes intègre bien cette stratégie. ci-dessous un extrait de requêtes permettant de séparer les données des instructions SQL, ce qui élimine le risque d'injection de code malveillant.

```
$reqInsert = 'INSERT INTO agence.avis (nom, note, message, date_avis, statut)
values (:nom, :note, :message, :date, :statut)';

$insert = $conn -> prepare ($reqInsert);
$save = $insert-> execute([
```

Cross-Site Scripting (XSS) : un des types de vulnérabilité la plus courante, qui consiste pour les attaquants à faire des injections dans les navigateurs, des champs de formulaires pour exécuter de code malveillant.

Afin de lutter contre, j'ai dû procéder au contrôle de tous les formulaires, d'abord de façon basique en html, puis en javascript coté client dans le cadre de l'amélioration de l'expérience utilisateur et surtout en PHP pour serveur.

Ci-dessous, aperçu de fonctions et méthodes pour filtrer et transformer les données.

```
function validation_donnees($donnees){
    $donnees = trim($donnees);
    $donnees = stripslashes($donnees);
    $donnees = htmlspecialchars($donnees);
    return $donnees;
}

$civilite = validation_donnees($_POST['civilite']);
$nom = validation_donnees($_POST['nom']);
$prenom = validation_donnees($_POST['prenom']);
$age = validation_donnees($_POST['age']);
$nationalite = validation_donnees($_POST['nationalite']);
$tel = validation_donnees($_POST['tel']);
$email = validation_donnees($_POST['email']);
$pwd = validation_donnees($_POST['pwd']);

// hachage password //
$pwd = password_hash($_POST['pwd'], PASSWORD_BCRYPT);
```

### ✓ Sécurisation du mot de pass :

Afin de garantir la fiabilité et sécurité des mots de pass , j'ai dû utiliser la fonction native de PHP password\_hash () prenant en paramètre la valeur du champ à remplir par le client ainsi que la fonction en argument.

Cette technique permet à la fois de lutter efficacement contre les Attaques par force brute à partir d'un hasch.

```
// hachage password //
$pwd = password_hash($_POST['pwd'], PASSWORD_BCRYPT);
```

### Résultat obtenu

	nom character varying	email character varying	mot_de_pass character varying	date_creation timestamp with time zone
1	NGANAWARA	romaric.nganas@gmail.com	\$2y\$10\$F0GU3uABTIIRJkBmtZ4jQ.Bt3zbpEyyqiA3NDp/m11/220HveN...	2024-02-15 11:02:12+01
2	Sandrine	sandrine@gmail.com	\$2y\$10\$Bn0PgL96jY7UJaMXvepbjOCXqt.P4V16IY3L04ncP0c6h7At89G...	2024-02-15 11:02:06+01

### ✓ Inclusion de fichier :

La méthode couramment utilisée est l'inclusion de fichiers locaux. Ils utilisent une variable non filtrée pour inclure un fichier local. L'attaquant peut remplacer le nom du fichier .php par un chemin de fichier absolu sur le serveur. Il existe cependant plusieurs autres types n'ayant nécessairement pas de lien avec mon application et donc non abordés ici.

- ✓ Mettre à jour de façon régulière des correctifs de sécurité suivant les recommandations. Appliquer une meilleure gestion des droits d'accès aux ressources par utilisateur.

## VI - Description de la veille effectuée par le candidat durant le projet

Pendant le développement de mon application, j'ai dû créer un cookie dans le but d'enregistrer l'id des clients connectés afin de faciliter toute identification.

En plus des enjeux que nous savons sur l'usage des cookies grâce à la formation étudiée, Je n'ai pas hésité à aller chercher de nouvelles informations sur la façon d'implémenter avec sécurité cette fonctionnalité.

Mes recherches m'ont conduit à un site Anglophone dénommé [debugpointer.com](http://debugpointer.com). Ce site m'a apporté des informations aussi bien sur l'importance et surtout les conséquences de sa mauvaise implémentation.

La recherche fait ressortir tous les éléments déjà vu en formation.

Erreur à ne pas faire :

- ☐ Ne pas définir de dates d'expiration pour les cookies.
- ☐ Ne pas chiffrer les cookies sensibles.
- ☐ Ne pas filtrer ses champs de formulaire
- ☐ Ne pas sécuriser son protocole HTTP

Lors d'une attaque de détournement de cookies, l'attaquant vole les cookies HTTP en écoutant la communication entre un utilisateur et une application Web, en accédant à l'ordinateur ou aux données du navigateur Web de l'utilisateur, ou en accédant à la mémoire du serveur Web.

Les attaquants peuvent pirater des cookies à l'aide de techniques telles que l'exploitation des vulnérabilités de script inter-site (XSS), l'exécution d'attaques de l'homme du milieu (MITM), l'exploitation des vulnérabilités de débordement de tampon sur les serveurs Web ou l'implantation de logiciels malveillants tels que des chevaux de Troie.

Raisons pour lesquelles, tout le contenu doit nécessairement être sécurisé et obéir à un principe de base : la mise en place du **certificat SSL/Protocole TLS**. Limiter l'accès aux outils et interfaces d'administration aux seules personnes habilitées.

Recueillir le consentement de l'internaute avant le dépôt d'un **cookie**, en fixant des durées de validité aux cookies dans les navigateurs client s'il s'avère important pour éviter les attaques imprévisibles.

La mise à jour régulière des package et les **bibliothèques** tierces utilisés afin d'éviter les vulnérabilités connues et obtenir les dernières fonctionnalités et améliorations de **sécurité**.

Mise en place de fonctions de validation intégrées dans PHP. Pour le control strict des entrées utilisateur pour éviter les injections de code malveillant.

Stocker les informations de session dans un emplacement sécurisé tel qu'une base de données, plutôt que dans des fichiers sur le serveur quand c'est possible. Utiliser des identifiants de session robustes et renouveler les sessions après une période bien, définie.

Telles ont été les recommandations retenues lors de cette recherche

**VIII - Extrait du site anglophone, utilisé dans le cadre de la recherche.**

L'extrait est disponible à cette adresse :

<https://debugpointer.com/security/pass-the-cookie>

## **Pass the Cookie Attack – Types, Examples & Preventing it**

Pass the Cookie Attack What is a Cookie?

Before we dive deep, let's familiarize ourselves with the foundation stone – the cookie. In the digital realm, a cookie is a teensy piece of data sent from a Website, stored on the user's computer by their web browser.

The attacker takes your cookie and “passes” or presents it to a website, duping it into thinking they're you. It's like getting your friend's membership card and using it to get into an exclusive club. You're not the member, but the club thinks you are ...

.

## **Traduction en français**

C'est quoi un cookie ?

Avant d'approfondir le sujet commençons par le définir.

Dans le domaine numérique, un cookie est un petit élément de données envoyé depuis un site Web, stocké sur l'ordinateur d'un utilisateur par son navigateur Web. Ils servent à mémoriser des préférences et mémorise les sessions.

Crée en 1994, ils avaient pour but de rendre de traiter facilement les paniers d'achat en ligne, chose pénible à réaliser avant.

Les cookies ont évolué, et retrouve au nombre de 3 : Cookies de session, les cookies persistants.

Exemple sur les cookies tiers : Créés pour suivre les habitudes de navigation d'un utilisateur sur différents sites Web, ils permettent aux annonceurs de proposer des services aux utilisateurs.

Comment se réalise le piratage ? Il suffit qu'un attaquant récupère le cookie d'un utilisateur, qui contient d'habitude son identifiant de session. Avec ce cookie, ils peuvent usurper l'identité de l'utilisateur, obtenant ainsi un accès non autorisé aux zones restreintes des sites Web, aux données personnelles etc.

Comment récupérer le cookie ?

Cross-Site Scripting (XSS) : Les attaquants peuvent injecter des scripts malveillants dans des sites Web qui, une fois exécutés, leur envoient directement les cookies de l'utilisateur.

Des mesures à prendre :

**Coté utilisateur final :**

- Vérifiez toujours la présence de la petite icône de cadenas dans votre barre d'adresse.
- Ne pas se contenter de fermer son navigateur mais se déconnecter du site d'abord.
- Mettre à jour régulièrement son navigateur.
- Suppressions régulières des cookies

**Pour garantir la sécurité en tant que développeur.**

- Utilisation de l'attribut HttpOnly : Cela garantit que le cookie n'est pas accessible via JavaScript. C'est une défense solide contre les attaques XSS.
- Authentification à deux facteurs : même si un cookie est volé, l'accès nécessitera une étape de vérification supplémentaire.
- Définir les dates d'expiration des cookies.
- Assurer le filtrage de toutes les entrées (https, formulaires etc) en contrôlant systématiquement les données saisies par les utilisateurs.
- Être transparent avec le client sur les enjeux de ce mécanisme en obtenant obligatoirement son consentement.
- Sensibiliser ses collaborateurs sur de bonnes conduites à tenir.

## CONCLUSION

Ce projet qui regorge notamment des compétences fondamentales recommandées du métier de développeur Web Full stack, peut encore faire l'objet d'un travail supplémentaire.

Différentes fonctionnalités peuvent être implémentées en dehors de cette région, moins développée avec des moyens très limités.

Dans l'optique d'exercer dès que possible à l'issue de cette formation, je pourrai encore améliorer l'application, non seulement pour approfondir mes connaissances dans le domaine, mais également la rendre plus complète, en intégrant par exemple certaines fonctionnalités frontend par exemple un système de routage ou encore un système de paiement carte bancaire coté Backend.