

**Université Paris Nanterre**

Licence MIASHS - Parcours MIAGE

Semestre 4

# Rapport de Projet

## Application de Planification de Voyages

### *"TravelDream"*

**Réalisé par :**

Melissa AKLI

Nikola [Nom de famille]

**Encadré par :**

Thibault Anani

Année universitaire 2024-2025

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Contexte du projet . . . . .	3
1.2	Objectifs et problématique . . . . .	3
1.3	Méthodologie de travail . . . . .	3
<b>2</b>	<b>Analyse des Besoins</b>	<b>4</b>
2.1	Besoins fonctionnels . . . . .	4
2.2	Besoins non fonctionnels . . . . .	5
2.3	Contraintes techniques . . . . .	5
<b>3</b>	<b>Conception de la Base de Données</b>	<b>6</b>
<b>4</b>	<b>Architecture Technique</b>	<b>6</b>
4.1	Technologies utilisées . . . . .	6
4.2	Structure du code . . . . .	7
4.3	Organisation des fichiers . . . . .	8
4.4	Configuration du système . . . . .	8
4.5	Sécurité . . . . .	10
<b>5</b>	<b>Fonctionnalités Implémentées</b>	<b>10</b>
5.1	Module d'Authentification . . . . .	10
5.1.1	Inscription . . . . .	10
5.1.2	Connexion . . . . .	11
5.1.3	Réinitialisation du mot de passe . . . . .	12
5.1.4	Déconnexion . . . . .	13
<b>6</b>	<b>Gestion des Voyages</b>	<b>13</b>
6.1	Visualisation des voyages . . . . .	13
6.2	Carte interactive . . . . .	14
6.3	Assistant de voyage (chatbot) . . . . .	16
6.4	Recherche de destinations (destination.php) . . . . .	18
<b>7</b>	<b>Interface Utilisateur</b>	<b>19</b>
7.1	Charte graphique . . . . .	19
7.1.1	Palette de couleurs . . . . .	20
7.1.2	Typographie . . . . .	20
7.1.3	Iconographie . . . . .	20
7.1.4	Éléments d'interface . . . . .	20
7.2	Maquettes et wireframes . . . . .	21

7.2.1	Page d'accueil et destinations . . . . .	21
7.2.2	Pages d'authentification . . . . .	21
7.2.3	Profil et gestion des voyages . . . . .	22
7.2.4	Carte interactive . . . . .	22
7.2.5	Assistant de voyage (chatbot) . . . . .	23
7.3	Responsive Design . . . . .	23
7.3.1	Approche technique . . . . .	24
7.3.2	Adaptations spécifiques . . . . .	24
<b>8</b>	<b>Tests et Validation</b>	<b>24</b>
8.1	Stratégie de test . . . . .	24
8.1.1	Types de tests implémentés . . . . .	24
8.1.2	Environnement de test . . . . .	25
8.1.3	Outils et méthodes . . . . .	25
8.2	Tests fonctionnels . . . . .	25
8.3	Tests d'intégration . . . . .	25
8.3.1	Flux complets . . . . .	25
8.3.2	Intégrité des données . . . . .	26
<b>9</b>	<b>Difficultés Rencontrées et Solutions</b>	<b>26</b>
9.1	Problèmes techniques . . . . .	26
9.1.1	Compatibilité entre navigateurs . . . . .	26
<b>10</b>	<b>Améliorations Futures</b>	<b>26</b>
10.1	Fonctionnalités à développer . . . . .	26
10.1.1	Application mobile native . . . . .	27
10.1.2	Amélioration des performances . . . . .	27
10.1.3	Renforcement de la sécurité . . . . .	27
10.1.4	Intelligence artificielle et personnalisation . . . . .	27
<b>11</b>	<b>Conclusion</b>	<b>28</b>
11.1	Synthèse du projet . . . . .	28
11.2	Compétences acquises . . . . .	28
11.3	Perspectives . . . . .	29

# 1 Introduction

## 1.1 Contexte du projet

Dans le cadre du projet S4 de la licence MIASHS parcours MIAGE à l'Université Paris Nanterre, nous avons développé une application web nommée "*TravelDream*" permettant aux utilisateurs de planifier leurs voyages de manière complète et organisée. Ce projet a été réalisé en binôme par Melissa et Nikola, et avait pour objectif de mettre en pratique nos compétences en développement web, en conception de bases de données et en modélisation. La planification de voyages est souvent une tâche complexe qui implique la gestion de nombreux éléments : transport, hébergement, activités, budget, etc. Notre application vise à centraliser toutes ces informations en un seul endroit, offrant ainsi aux utilisateurs un outil pratique pour organiser leurs déplacements et profiter pleinement de leurs expériences de voyage.

## 1.2 Objectifs et problématique

L'objectif principal de ce projet était de créer une plateforme interactive permettant aux utilisateurs de :

- Gérer les voyages (dates, destinations)
- Organiser transports, hébergements, activités
- Suivre les dépenses
- Consulter une carte interactive
- Obtenir des recommandations via un assistant virtuel

La problématique centrale que nous avons cherché à résoudre est la suivante : comment simplifier et centraliser le processus de planification de voyage pour les utilisateurs, en leur offrant une interface intuitive et des fonctionnalités complètes ? Pour répondre à cette problématique, nous avons développé une application qui combine :

- Une base de données relationnelle bien structurée
- Une interface utilisateur intuitive et responsive
- Des fonctionnalités adaptées aux besoins des voyageurs
- Des outils de visualisation comme la carte interactive
- Un assistant virtuel pour guider les utilisateurs

## 1.3 Méthodologie de travail

Pour mener à bien ce projet, nous avons adopté une approche méthodique en plusieurs phases :

- **Phase d'analyse :** Définition des besoins fonctionnels et non fonctionnels, création de personas et de cas d'utilisation pour comprendre les attentes des utilisateurs.

- **Phase de conception** : Élaboration du modèle conceptuel de données, conception du schéma relationnel de la base de données, et définition de l'architecture technique du projet.
- **Phase de développement** : Implémentation des fonctionnalités par modules (authentification, gestion des voyages, activités, etc.), développement de l'interface utilisateur, et intégration des différentes composantes.
- **Phase de test** : Vérification du bon fonctionnement des fonctionnalités, tests d'intégration, et correction des bugs identifiés.
- **Phase de finalisation** : Documentation du projet, préparation du rapport, et présentation des résultats.

Tout au long du projet, nous avons maintenu une communication constante entre les membres de l'équipe pour assurer la cohérence du développement et résoudre rapidement les problèmes rencontrés.

Dans les sections suivantes, nous détaillerons les différentes composantes de notre application, en commençant par l'analyse des besoins, puis en abordant la conception de la base de données, l'architecture technique, les fonctionnalités implémentées, et l'interface utilisateur.

## 2 Analyse des Besoins

### 2.1 Besoins fonctionnels

Notre application de planification de voyages *TravelDream* a été conçue pour répondre à un ensemble de besoins fonctionnels précis, identifiés à partir des attentes des utilisateurs potentiels. Ces besoins définissent les capacités et services que notre système doit offrir :

- **Gestion des utilisateurs**
  - Création d'un compte utilisateur avec nom, email et mot de passe
  - Connexion sécurisée
  - Réinitialisation du mot de passe
  - Déconnexion et gestion de session
- **Gestion des voyages**
  - Création d'un nouveau voyage (destination, dates)
  - Consultation de la liste des voyages
  - Modification des informations d'un voyage
  - Suppression d'un voyage
- **Gestion des activités**
- **Gestion des transports**
- **Gestion des logements**
- **Fonctionnalités supplémentaires**

- Carte interactive des destinations
- Assistant virtuel pour recommandations

## 2.2 Besoins non fonctionnels

Au-delà des fonctionnalités, notre application devait également répondre à des exigences non fonctionnelles, qui définissent la qualité et les performances du système :

- **Sécurité**
  - Protection des données personnelles des utilisateurs
  - Hachage des mots de passe pour éviter leur stockage en clair
  - Utilisation de requêtes préparées pour prévenir les injections SQL
  - Validation des entrées utilisateur pour éviter les attaques XSS
- **Utilisabilité**
  - Interface intuitive et facile à prendre en main
  - Design responsive adapté aux différents appareils (ordinateurs, tablettes, smartphones)
  - Temps de chargement rapide des pages
  - Feedback visuel pour les actions utilisateur
- **Performance**
  - Optimisation des requêtes à la base de données
  - Chargement efficace des ressources (images, scripts)
  - Capacité à gérer plusieurs utilisateurs simultanément
- **Maintenabilité**
  - Code modulaire et bien structuré
  - Séparation des préoccupations (modèle, vue, contrôleur)
  - Documentation du code et des fonctionnalités
  - Facilité d'extension pour ajouter de nouvelles fonctionnalités
- **Fiabilité**
  - Gestion appropriée des erreurs
  - Validation des données côté serveur
  - Persistance des données en cas d'erreur

## 2.3 Contraintes techniques

Le développement de notre application a été soumis à plusieurs contraintes techniques, qui ont guidé nos choix technologiques et notre approche tout au long du projet :

- **Environnement de développement**
  - Utilisation de PHP pour le développement backend
  - Base de données MySQL pour le stockage des données
  - HTML, CSS et JavaScript pour le développement frontend

- Utilisation de Bootstrap pour le design responsive
- **Infrastructure**
  - Hébergement sur un serveur local pour le développement
  - Configuration WAMP/MAMP pour l'environnement de développement
- **Compatibilité**
  - Support des navigateurs modernes (Chrome, Firefox, Safari, Edge)
  - Adaptation aux différentes tailles d'écran
- **Intégration**
  - Utilisation de l'API Leaflet.js pour la carte interactive
  - Intégration de l'API Nominatim pour la recherche de lieux

## 3 Conception de la Base de Données

## 4 Architecture Technique

### 4.1 Technologies utilisées

Notre application de planification de voyages "TravelDream" a été développée en utilisant un ensemble de technologies modernes et complémentaires, choisies pour leur fiabilité, leur performance et leur adéquation avec les besoins du projet.

#### Backend

- **PHP 7.4** : Langage de programmation côté serveur utilisé pour développer la logique métier de l'application, gérer les sessions utilisateurs et interagir avec la base de données.
- **MySQL 8.0** : Système de gestion de base de données relationnelle utilisé pour stocker et gérer toutes les données de l'application.
- **PDO** : Extension PHP utilisée pour accéder à la base de données de manière sécurisée, permettant l'utilisation de requêtes préparées pour prévenir les injections SQL.

#### Frontend

- **HTML5** : Langage de balisage utilisé pour structurer le contenu des pages web.
- **CSS3** : Langage de style utilisé pour définir la présentation visuelle des pages web.
- **JavaScript** : Langage de programmation côté client utilisé pour ajouter des fonctionnalités interactives aux pages web.
- **Bootstrap 5.3** : Framework CSS utilisé pour créer un design responsive et moderne, facilitant l'adaptation de l'interface à différentes tailles d'écran.

- **Font Awesome 6.4** : Bibliothèque d'icônes vectorielles utilisée pour améliorer l'interface utilisateur avec des icônes expressives.

## APIs et bibliothèques externes

- **Leaflet.js** : Bibliothèque JavaScript open-source utilisée pour implémenter la carte interactive.
- **Nominatim API** : Service de géocodage utilisé pour la recherche de lieux sur la carte.
- **OpenStreetMap** : Fournisseur de données cartographiques utilisé comme fond de carte dans notre application.

## 4.2 Structure du code

Notre application suit une architecture modulaire qui sépare les différentes préoccupations du système, facilitant ainsi la maintenance et l'évolution du code.

**Organisation générale** Le code est organisé selon une structure proche du modèle MVC (Modèle-Vue-Contrôleur), bien que non strictement implémentée :

- **Modèle** : Les interactions avec la base de données sont gérées principalement dans le fichier config.php qui contient des fonctions pour établir la connexion et effectuer des opérations courantes.
- **Vue** : Les fichiers PHP contenant principalement du HTML constituent la couche de présentation, avec l'aide des fichiers CSS pour le style.
- **Contrôleur** : La logique de traitement est intégrée dans les fichiers PHP principaux, qui gèrent les requêtes utilisateur, traitent les données et préparent les réponses

**Gestion des sessions** La gestion des sessions utilisateur est implémentée de manière centralisée. Le fichier header.php initialise la session avec `session_start()` et vérifie si l'utilisateur est connecté via la fonction `isUserLoggedIn()` définie dans config.php.

**Système de templates** L'application utilise un système simple de templates avec des fichiers d'inclusion :

- `header.php` : Contient l'en-tête HTML commun à toutes les pages, incluant les balises meta, les liens vers les feuilles de style et la barre de navigation.
- `footer.php` : Contient le pied de page commun et les scripts JavaScript.

Ces fichiers sont inclus dans chaque page de l'application, assurant ainsi une cohérence visuelle et fonctionnelle.

## 4.3 Organisation des fichiers

L'organisation des fichiers de notre projet suit une structure logique qui facilite la navigation et la maintenance du code.

### Fichiers PHP principaux

- `config.php` : Contient les paramètres de configuration, les fonctions de connexion à la base de données et diverses fonctions utilitaires.
- `header.php` : Gère l'en-tête des pages et la barre de navigation.
- `footer.php` : Gère le pied de page et l'inclusion des scripts JavaScript.
- `inscription.php` : Gère l'inscription des nouveaux utilisateurs.
- `pageconnexion.php` : Gère la connexion des utilisateurs existants.
- `Réinitialisation.php` : Permet aux utilisateurs de réinitialiser leur mot de passe.
- `logout.php` : Gère la déconnexion des utilisateurs.
- `destination.php` : Affiche les destinations populaires et permet la recherche.
- `mesvoyages.php` : Affiche et gère les voyages de l'utilisateur connecté.
- `map.php` : Implémente la carte interactive avec géolocalisation.
- `chat.php` : Implémente l'assistant virtuel de voyage.

### Fichiers CSS

Les fichiers CSS sont organisés par fonctionnalité ou par page :

- `global.css` : Contient les styles globaux appliqués à l'ensemble de l'application.
- `style.css` : Contient les styles pour les pages d'authentification.
- `accueil.css` : Styles spécifiques à la page d'accueil.
- `destinations.css` : Styles pour la page des destinations.
- `profil_new.css` : Styles pour la page de profil utilisateur.
- `chatbot.css` : Styles pour l'interface de l'assistant virtuel.
- `map.css` : Styles pour la carte interactive.

## 4.4 Configuration du système

**Configuration de la base de données** La configuration de la connexion à la base de données est centralisée dans le fichier `config.php` :

```
$config = [
    'db' => [
        'host' => '127.0.0.1',
        'dbname' => 'PlanVoyages',
        'username' => 'root',
        'password' => 'rootroot',
```

```
        'charset' => 'utf8mb4'
    ],
    'site' => [
        'name' => 'TravelDream',
        'url' => 'http://localhost/projet_voyage',
        'email' => 'contact@traveldream.com'
    ]
];
```

Cette approche permet de modifier facilement les paramètres de connexion sans avoir à modifier chaque fichier qui interagit avec la base de données.

**Fonction de connexion à la base de données** La fonction `getDbConnection()` dans `config.php` établit une connexion PDO à la base de données :

```
function getDbConnection() {
    global $config;

    try {
        $dsn = "mysql:host={$config['db']['host']};dbname={$config['db']['dbname']} charset={$config['db']['charset']}";
        $pdo = new PDO($dsn, $config['db']['username'], $config['db']['password']);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $pdo;
    } catch (PDOException $e) {
        die("Erreur de connexion à la base de données : " . $e->getMessage());
    }
}
```

Cette fonction configure également PDO pour qu'il lance des exceptions en cas d'erreur, facilitant ainsi le débogage.

**Fonctions utilitaires** Le fichier `config.php` contient également plusieurs fonctions utilitaires importantes

- `isUserLoggedIn()` : Vérifie si l'utilisateur est connecté en contrôlant la présence de l'ID utilisateur dans la session.
- `redirect()` : Facilite la redirection vers une autre page.
- `sanitize()` : Nettoie les données d'entrée pour prévenir les attaques XSS.

- `generateAlert()` : Génère des messages d'alerte formatés avec Bootstrap
- `formatDate()` : Formate les dates selon le format spécifié.

## 4.5 Sécurité

Plusieurs mesures de sécurité ont été mises en place :

- Requêtes préparées (PDO) pour éviter les injections SQL.
- Mots de passe hachés avec `password_hash()`.
- Nettoyage des entrées avec `sanitize()`.
- Gestion des sessions pour contrôler l'accès aux pages protégées.

Cette architecture technique robuste et bien organisée a permis de développer une application fonctionnelle, sécurisée et facile à maintenir, tout en offrant une expérience utilisateur fluide et agréable.

# 5 Fonctionnalités Implémentées

## 5.1 Module d'Authentification

Le module d'authentification constitue la porte d'entrée de notre application et assure la sécurité des données utilisateurs. Il comprend quatre fonctionnalités principales : l'inscription, la connexion, la réinitialisation du mot de passe et la déconnexion.

### 5.1.1 Inscription

La page d'inscription (`inscription.php`) permet aux nouveaux utilisateurs de créer un compte sur la plateforme. Cette fonctionnalité a été implémentée avec une attention particulière à la sécurité et à l'expérience utilisateur.

#### Processus d'inscription

1. L'utilisateur accède au formulaire d'inscription via le lien « inscription » dans la barre de navigation.
2. Il remplit les champs requis : nom d'utilisateur, adresse email et mot de passe.
3. À la soumission, le système vérifie si l'email est déjà utilisé.
4. Si l'email est disponible, le mot de passe est haché via `password_hash()` avant d'être stocké.
5. Un message de confirmation s'affiche, invitant l'utilisateur à se connecter.

#### Sécurité

- Utilisation de requêtes préparées pour prévenir les injections SQL.

- Hachage du mot de passe avec bcrypt via password\_hash().
- Validation des données côté serveur.

```
// Vérifier si l'email existe déjà
$stmt = $pdo->prepare("SELECT id_utilisateur FROM Utilisateur WHERE email
= :email");
$stmt->execute(['email' => $email]);

if ($stmt->rowCount() > 0) {
    $alertMessage = "";
    $alertClass = "error";
} else {
    // Insérer l'utilisateur dans la base de données
    $sql = "INSERT INTO Utilisateur (nom, email, mot_de_passe) VALUES
(:nom, :email, :mot_de_passe)";
    $stmt = $pdo->prepare($sql);

    // Exécuter la requête
    if ($stmt->execute(['nom' => $nom_utilisateur, 'email' => $email, 'mot_de_passe'
=> $mot_de_passe])) {
        $alertMessage = " Incription réussie ! <a href='login.php'>Connectez-vous
ici</a>";
        $alertClass = "success";
    }
}
```

### 5.1.2 Connexion

La page de connexion (pageconnexion.php) permet aux utilisateurs existants d'accéder à leur compte et à leurs données personnelles.

#### Processus de connexion

1. L'utilisateur accède au formulaire de connexion via le lien « Connexion » dans la barre de navigation.
2. Il saisit son adresse email et son mot de passe.
3. Le système vérifie l'existence de l'email dans la base de données.
4. Si l'email existe, le mot de passe est vérifié via password\_verify().
5. En cas de succès, les données de l'utilisateur sont stockées en session, puis il est redirigé vers son profil.
6. En cas d'échec, un message d'erreur est affiché.

**Gestion de session** Après une connexion réussie, les informations suivantes sont stockées en session :

- ID de l'utilisateur
- Nom d'utilisateur

- Adresse email

Cela permet d'identifier l'utilisateur et de personnaliser son expérience.

```
// Vérifier si les mots de passe correspondent
if ($nouveau_mdp !== $confirm_mdp) {
    die(" Les mots de passe ne correspondent pas.");
}

// Vérifier si l'email existe
$stmt = $conn->prepare("SELECT id_utilisateur FROM Utilisateur WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    $stmt->bind_result($id_utilisateur);
    $stmt->fetch();

    // Hacher le nouveau mot de passe
    $hashed_password = password_hash($nouveau_mdp, PASSWORD_DEFAULT);

    // Mettre à jour le mot de passe
    $stmt = $conn->prepare("UPDATE Utilisateur SET mot_de_passe = ? WHERE
id_utilisateur = ?");
    $stmt->bind_param("si", $hashed_password, $id_utilisateur);

    if ($stmt->execute()) {
        echo " Mot de passe réinitialisé avec succès ! <a
href='pageconnexion.php'>Connectez-vous ici</a>";
    }
}
```

### 5.1.3 Réinitialisation du mot de passe

La fonctionnalité de réinitialisation du mot de passe (`Reinitialisation.php`) permet aux utilisateurs ayant oublié leur mot de passe de le réinitialiser de manière sécurisée.

#### Processus de réinitialisation

- L'utilisateur accède au formulaire de réinitialisation via le lien « Mot de passe oublié » sur la page de connexion.
- Il saisit son adresse email, un nouveau mot de passe et la confirmation de ce mot de passe.
- Le système vérifie que les deux mots de passe correspondent.
- Si l'email existe dans la base de données, le nouveau mot de passe est haché et mis à jour.
- Un message de confirmation s'affiche et l'utilisateur est invité à se connecter avec son nouveau mot de passe.

#### Sécurité

- Vérification de la correspondance des mots de passe saisis.

- Hachage du nouveau mot de passe avant stockage.
- Utilisation de requêtes préparées pour la mise à jour en base de données.

#### 5.1.4 Déconnexion

La fonctionnalité de déconnexion ( logout.php ) permet aux utilisateurs de se déconnecter en toute sécurité de leur compte.

#### Processus de déconnexion

1. L'utilisateur clique sur le lien « Déconnexion » dans la barre de navigation.
2. Le script détruit la session en cours, effaçant toutes les variables de session.
3. L'utilisateur est redirigé vers la page de connexion.

#### Sécurité

- Destruction complète de la session pour éviter tout accès non autorisé.
- Redirection immédiate pour empêcher toute action après la déconnexion.

```
session_start();
session_destroy();
header("Location: pageconnexion.php");
exit();
```

FIGURE 1 – Interface de la page de connexion

## 6 Gestion des Voyages

### 6.1 Visualisation des voyages

La page `mesvoyages.php` permet aux utilisateurs de consulter l'ensemble de leurs voyages planifiés, en cours ou passés.

**Fonctionnalités :**

- Affichage de la liste des voyages de l'utilisateur, triés par date de début (du plus récent au plus ancien).
- Pour chaque voyage : destination, dates de début et de fin, durée.
- Indication visuelle du statut du voyage : à venir, en cours, terminé.
- Statistiques : nombre total de voyages, destinations uniques, total de jours de voyage.
- Affichage du prochain voyage et du nombre de jours restants.

## 6.2 Carte interactive

### 5.6.1 Carte interactive

La carte interactive (`map.php`) permet aux utilisateurs de visualiser et de rechercher des lieux sur une carte mondiale.

**Fonctionnalités :**

- Affichage d'une carte interactive utilisant Leaflet.js et OpenStreetMap.
- Recherche de lieux via l'API Nominatim.
- Géolocalisation pour afficher la position actuelle de l'utilisateur.
- Ajout de marqueurs sur les lieux recherchés.

**Implémentation technique :**

- Initialisation de la carte :

```
// Initialisation de la carte
var map = L.map('map').setView([20, 0], 2); // Vue globale

// Ajouter les fonds de carte
var streets = L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; OpenStreetMap contributors'
}).addTo(map);
```

La recherche de lieux utilise l'API Nominatim d'OpenStreetMap :

```
// Recherche de lieu avec l'API Nominatim
document.getElementById("searchBtn").addEventListener("click", function() {
    var query = document.getElementById("search").value;
    if (!query) return;

    var url = `https://nominatim.openstreetmap.org/search?format=json&q=${query}`;

    fetch(url)
        .then(response => response.json())
        .then(data => {
            if (data.length > 0) {
                var lat = data[0].lat;
                var lon = data[0].lon;

                // Ajouter un marqueur sur la carte
                L.marker([lat, lon]).addTo(map)
                    .bindPopup(`<b>${data[0].display_name}</b>`)
                    .openPopup();

                map.setView([lat, lon], 10);
            }
        });
});
```

- Géolocalisation avec l'API navigateur :

```
// Fonction de géolocalisation
document.getElementById("geolocateBtn").addEventListener("click", function() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position) {
            var lat = position.coords.latitude;
            var lon = position.coords.longitude;

            if (userMarker) {
                map.removeLayer(userMarker);
            }

            userMarker = L.marker([lat, lon]).addTo(map)
                .bindPopup("Vous êtes ici!")
                .openPopup();

            map.setView([lat, lon], 13);
        });
    }
});
```

### 6.3 Assistant de voyage (chatbot)

L'assistant de voyage (`chat.php`) est un chatbot intelligent qui aide les utilisateurs à trouver des destinations adaptées à leurs préférences.

#### Fonctionnalités

- Interface de chat intuitive.
- Base de données de destinations avec leurs caractéristiques (climat, budget, activités, etc.).
- Algorithme de recommandation basé sur les préférences exprimées par l'utilisateur.
- Présentation des destinations recommandées avec leurs informations principales.

**Implémentation technique** L'assistant est implémenté en JavaScript côté client avec une base de données locale.

```

// Base de données de destinations
const destinations = [
  {
    nom: "Paris",
    pays: "France",
    continent: "Europe",
    climat: "tempéré",
    type: ["ville", "culture", "gastronomie"],
    budget: "moyen",
    saison: ["printemps", "automne"],
    duree_ideale: "3-5 jours",
    activites: ["Tour Eiffel", "Musée du Louvre", "Montmartre", "Croisière sur la Seine", "Shopping"],
    description: "La ville de l'amour avec ses monuments emblématiques, sa gastronomie et son ambiance romantique."
  },
  // Autres destinations...
];

```

L'algorithme de recommandation filtre les destinations selon les préférences de l'utilisateur :

```

// Fonction pour recommander des destinations
function recommanderDestinations(preferences) {
    // Filtrer les destinations selon les préférences
    let resultats = [...destinations];

    // Filtrer par continent si spécifié
    if (preferences.continent && preferences.continent !== "tous") {
        resultats = resultats.filter(dest => dest.continent.toLowerCase() ===
preferences.continent.toLowerCase());
    }
}

```

---

```

// Filtrer par budget si spécifié
if (preferences.budget && preferences.budget !== "tous") {
    resultats = resultats.filter(dest => dest.budget.toLowerCase() ===
preferences.budget.toLowerCase());
}

// Filtrer par type de voyage si spécifié
if (preferences.type && preferences.type !== "tous") {
    resultats = resultats.filter(dest =>
dest.type.includes(preferences.type.toLowerCase()));
}

// Filtrer par climat si spécifié
if (preferences.climat && preferences.climat !== "tous") {
    resultats = resultats.filter(dest => dest.climat.toLowerCase() ===
preferences.climat.toLowerCase());
}

// Limiter à 3 résultats maximum
return resultats.slice(0, 3);
}

```

## 6.4 Recherche de destinations (destination.php)

Cette fonctionnalité permet aux utilisateurs de découvrir des lieux de voyage populaires et d'obtenir des informations détaillées sur ceux-ci.

### Fonctionnalités

- Affichage des destinations populaires avec images et descriptions.
- Barre de recherche pour trouver une destination spécifique.
- Affichage détaillé (meilleure période, budget, activités recommandées).
- Possibilité de planifier directement un voyage vers la destination choisie.

**Implémentation technique** Les informations sont définies dans le code PHP

```
// Informations sur les destinations populaires
$destinations_info = [
    'Paris' => [
        'image' => 'https://images.unsplash.com/
```

```
photo-1499856871958-5b9627545d1a',
    'description' => 'La ville de l'amour avec ses monuments emblématiques, sa
gastronomie et son ambiance romantique.',
    'activites' => ['Tour Eiffel', 'Musée du Louvre', 'Montmartre', 'Croisière sur la
Seine', 'Shopping'],
    'meilleure_periode' => 'Printemps et automne',
    'budget' => 'Moyen à élevé'
],
// Autres destinations...
];
```

La recherche filtre les destinations selon le terme saisi par l'utilisateur :

```
// Si une recherche est effectuée, filtrer les destinations
if (!empty($search_destination)) {
    $filtered_destinations = array_filter($default_destinations, function($dest) use
($search_destination) {
        return stripos($dest, $search_destination) !== false;
});

    if (!empty($filtered_destinations)) {
        $destinations_to_show = $filtered_destinations;
    } else {
        $destinations_to_show = [];
    }
} else {
    $destinations_to_show = $default_destinations;
}
```

## 7 Interface Utilisateur

### 7.1 Charte graphique

L'interface de l'application **TravelDream** a été conçue pour offrir une expérience utilisateur agréable, fluide et cohérente. Elle repose sur une charte graphique harmonieuse appliquée à l'ensemble du site.

#### 7.1.1 Palette de couleurs

- Couleur principale : Bleu ciel (#4A89DC) – Évoque le ciel, l'océan et la liberté.

- **Couleur secondaire** : Vert tendre (#48CFAD) – Symbolise la nature et les paysages.
- **Couleur d'accentuation** : Orange (#FC6E51) – Représente le soleil et l'énergie.
- **Couleurs neutres** :
  - Blanc (#FFFFFF) – Fond principal pour une lecture confortable.
  - Gris clair (#F5F7FA) – Fond secondaire pour distinguer certaines sections.
  - Gris foncé (#434A54) – Texte principal pour un bon contraste.

### 7.1.2 Typographie

- **Titres** : Montserrat – Moderne, sans-serif, excellente lisibilité.
- **Corps de texte** : Open Sans – Clair et fluide pour le contenu principal.
- **Accents décoratifs** : Pacifico – Cursive utilisée avec parcimonie.

### Hiérarchie des tailles

- Titres principaux (h1) : 2.5rem
- Sous-titres (h2) : 2rem
- Titres de section (h3) : 1.5rem
- Corps de texte : 1rem
- Texte secondaire : 0.875rem

### 7.1.3 Iconographie

Nous utilisons la bibliothèque **Font Awesome** pour des icônes vectorielles adaptables à tous les supports.

#### Utilisations principales :

- Illustration des fonctionnalités (ex : avion, lit)
- Navigation (flèches, menu hamburger)
- Actions (ajout, modification, suppression)
- Statuts (succès, erreur, information)

### 7.1.4 Éléments d'interface

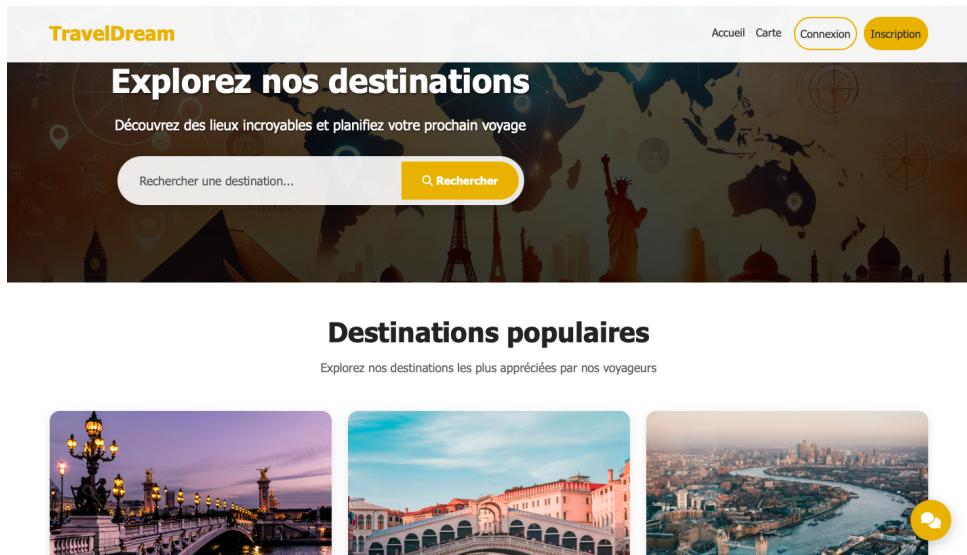
- **Boutons** : Coins arrondis, couleur d'action ( primaire, secondaire, danger), icône si besoin.
- **Cartes** : Coins arrondis, ombre légère, marges internes.
- **Formulaires** : Champs avec bordures fines, labels clairs, retour visuel de validation.
- **Alertes** : Couleurs distinctes selon le type (succès, erreur, info, avertissement).

## 7.2 Maquettes et wireframes

Avant le développement, des maquettes et wireframes ont été réalisés afin de structurer l'interface et garantir une navigation fluide et cohérente.

### 7.2.1 Page d'accueil et destinations

La page d'accueil présente une image de fond évocatrice avec un message d'accueil et une barre de recherche proéminente. En dessous, on trouve une sélection de destinations populaires présentées sous forme de cartes avec images.



### 7.2.2 Pages d'authentification

Les pages de connexion et d'inscription partagent une structure similaire avec un formulaire simple sur un fond d'image de voyage. Cette approche minimaliste permet à l'utilisateur de se concentrer sur l'action à effectuer.



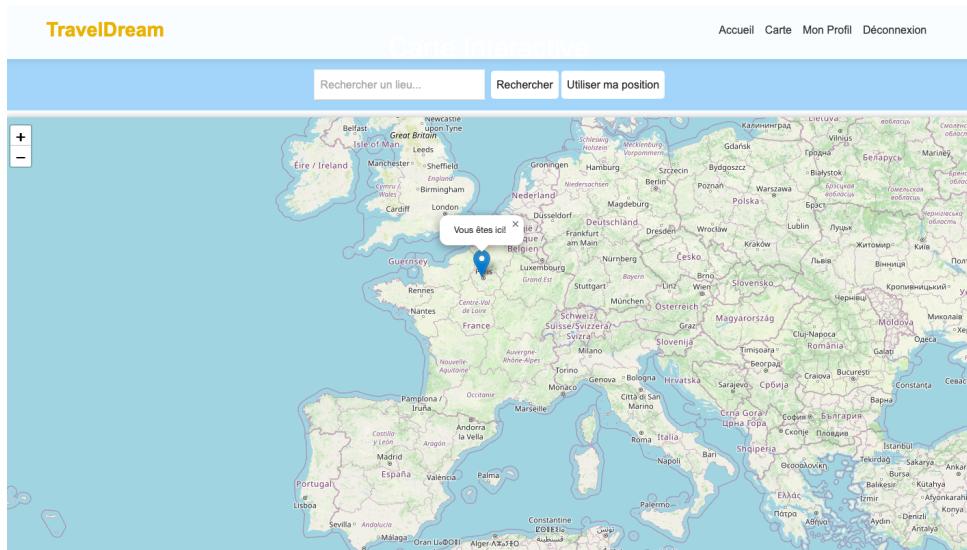
### 7.2.3 Profil et gestion des voyages

La page de profil est divisée en deux sections principales : un résumé du profil avec statistiques à gauche, et la liste des voyages à droite. Chaque voyage est présenté dans une carte contenant les informations essentielles et des boutons d'action.

### 7.2.4 Carte interactive

La carte interactive occupe la majeure partie de l'écran et inclut :

- Une barre de recherche en haut.
- Des boutons de contrôle pour zoom, filtres, etc.

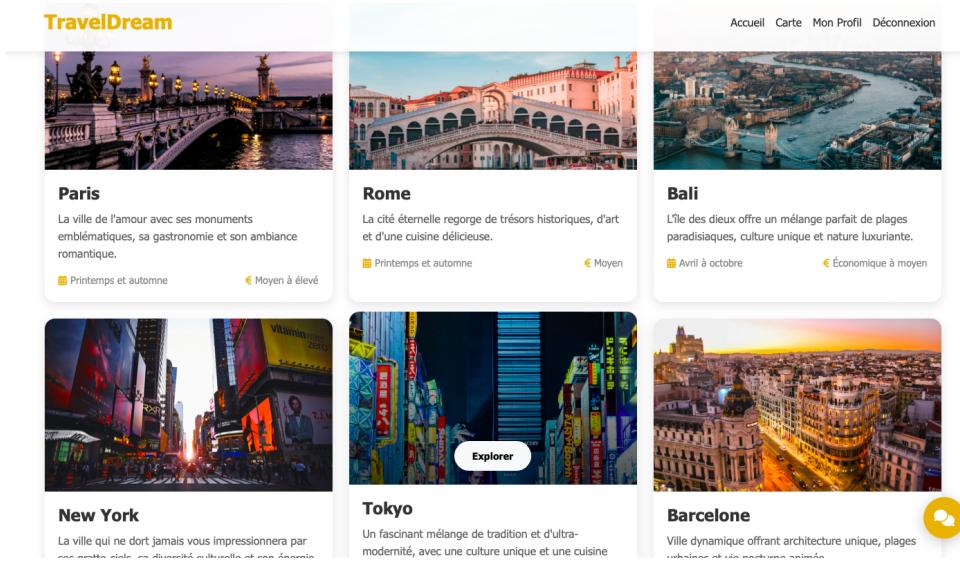


### 7.2.5 Assistant de voyage (chatbot)

L'interface de l'assistant de voyage (chatbot) présente une zone de chat à droite et une zone d'information à gauche, créant un équilibre visuel tout en maintenant la fonctionnalité.

## 7.3 Responsive Design

Notre application a été conçue selon les principes du "mobile-first" et du design responsive, garantissant une expérience utilisateur optimale sur tous les appareils, des smartphones aux grands écrans d'ordinateur.



### 7.3.1 Approche technique

Nous avons utilisé Bootstrap 5 pour garantir une mise en page adaptable :

- Système de grille 12 colonnes.
- Media queries pour adapter l'affichage.
- Classe `img-fluid` pour rendre les images responsives.
- Composants modulables selon la taille d'écran.

### 7.3.2 Adaptations spécifiques

Des ajustements ont été appliqués pour améliorer l'expérience utilisateur :

- Menu hamburger sur petits écrans.
- Cartes de voyage : 3 colonnes (desktop), 2 (tablette), 1 (mobile).
- Réorganisation verticale des sections sur mobile.
- Champs de formulaire adaptés en largeur.

## 8 Tests et Validation

### 8.1 Stratégie de test

Pour garantir la qualité et la fiabilité de notre application *TravelDream*, nous avons mis en place une stratégie de test complète couvrant différents aspects du système. Cette approche méthodique nous a permis d'identifier et de corriger les problèmes avant le déploiement final.

#### 8.1.1 Types de tests implémentés

Notre stratégie a intégré plusieurs types de tests :

- **Tests unitaires** : vérification du bon fonctionnement de chaque composant (fonctions, classes, modules)
- **Tests d'intégration** : validation des interactions entre les différents modules
- **Tests fonctionnels** : contrôle du respect des exigences pour chaque fonctionnalité
- **Tests d'interface utilisateur** : vérification de l'apparence et du comportement sur divers appareils
- **Tests de sécurité** : identification des vulnérabilités, notamment sur l'authentification et la gestion des données

### 8.1.2 Environnement de test

Les tests ont été réalisés dans un environnement dédié, proche de la production :

- Serveur local xamp avec configuration équivalente à celle de production
- Base de données de test avec des jeux de données réalistes
- Navigateurs : Chrome, Firefox, Safari, Edge
- Appareils : ordinateurs, tablettes, smartphones

### 8.1.3 Outils et méthodes

Nous avons utilisé les outils suivants pour le processus de validation :

- **Tests manuels** : scénarios exécutés manuellement avec documentation des résultats
- **Validation W3C** : conformité HTML/CSS aux standards du web
- **Outils développeur navigateur** : inspection DOM, débogage JavaScript, analyse performance
- **Checklists** : vérification systématique des fonctionnalités via des listes de contrôle

## 8.2 Tests fonctionnels

Les tests fonctionnels ont permis de valider que chaque fonctionnalité respecte les besoins utilisateurs et les spécifications définies.

## 8.3 Tests d'intégration

Les tests d'intégration ont permis de vérifier que les différents modules de l'application fonctionnent correctement ensemble.

### 8.3.1 Flux complets

Plusieurs flux d'utilisation ont été testés :

- **Inscription → Connexion → Création d'un voyage → Ajout d'activités**  
→ **Visualisation** : vérification de l'enregistrement correct des données et de leur liaison.
- **Recherche de destination → Planification d'un voyage → Ajout à la liste personnelle** : vérification de la reprise des données de destination et de l'intégration avec la création de voyage.
- **Utilisation de la carte → Enregistrement d'un lieu → Association à un voyage** : vérification de l'enregistrement des coordonnées et de l'intégration avec la gestion des voyages.

### 8.3.2 Intégrité des données

L'intégrité des données a été contrôlée pour :

- Création, modification et suppression de voyages et d'activités
- Vérification des contraintes d'intégrité référentielle (clés étrangères)
- Cohérence des données entre les différentes tables

## 9 Difficultés Rencontrées et Solutions

### 9.1 Problèmes techniques

Durant le développement de l'application *TravelDream*, plusieurs défis techniques ont été rencontrés, nécessitant des solutions adaptées.

#### 9.1.1 Compatibilité entre navigateurs

**Problème** : Affichage incohérent des éléments d'interface (formulaires, flexbox).

**Solution** : Utilisation de Bootstrap, ajout de préfixes CSS et tests sur Chrome, Firefox, Safari et Edge.

## 10 Améliorations Futures

### 10.1 Fonctionnalités à développer

Notre application "TravelDream" offre déjà un ensemble solide de fonctionnalités pour la planification de voyages, mais plusieurs améliorations pourraient être apportées pour enrichir l'expérience utilisateur et étendre les capacités du système. Une fonctionnalité très demandée lors des retours utilisateurs est la possibilité de partager ses voyages avec d'autres personnes. **Description** : Permettre aux utilisateurs de partager leurs plans de voyage avec des amis ou des membres de leur famille, soit en les invitant comme collaborateurs (avec droits d'édition), soit en partageant une version en lecture seule.

**Implémentation envisagée** : - Création d'une table de partage dans la base de données pour gérer les relations entre voyages et utilisateurs invités - Système de gestion des droits (lecture seule, modification) - Génération de liens de partage uniques et sécurisés - Interface de gestion des collaborateurs pour chaque voyage

#### 10.1.1 Application mobile native

Bien que notre site soit responsive, une application mobile native offrirait une meilleure expérience sur smartphone et permettrait d'accéder à des fonctionnalités spécifiques aux appareils mobiles. **Description** : Développer des applications natives pour iOS et Android qui se synchronisent avec la plateforme web. **Avantages** : - Accès hors ligne aux informations de voyage - Utilisation des fonctionnalités natives (appareil photo, GPS, notifications) - Expérience utilisateur optimisée pour mobile - Notifications push pour les rappels (check-in, activités planifiées, etc.)

#### 10.1.2 Amélioration des performances

- **Cache** : Redis ou Memcached
- **Requêtes SQL** : Indexation, optimisation
- **Lazy loading** : Images et contenus lourds
- **Minification** : CSS/JS
- **CDN** : Ressources statiques

#### 10.1.3 Renforcement de la sécurité

**Mesures envisagées** :

- Authentification à deux facteurs (2FA)
- Audit de sécurité et corrections des vulnérabilités
- Chiffrement des données sensibles
- Protection CSRF via tokens
- Rate limiting contre les attaques par force brute

#### 10.1.4 Intelligence artificielle et personnalisation

**Objectif** : Offrir une assistance intelligente via l'IA.

**Applications possibles** :

- Recommandations personnalisées
- Assistant virtuel (NLP)
- Analyse prédictive des meilleures périodes
- Itinéraires adaptés aux préférences

**Conclusion** : Ces évolutions permettraient à *TravelDream* de passer d'un outil de planification à une véritable plateforme immersive et personnalisée dédiée aux voyageurs.

# 11 Conclusion

## 11.1 Synthèse du projet

Le projet "TravelDream" a permis de développer une application web complète dédiée à la planification de voyages, offrant aux utilisateurs un outil centralisé pour organiser tous les aspects de leurs déplacements. Cette application répond à un besoin réel des voyageurs qui doivent souvent jongler entre différentes plateformes et documents pour gérer leurs réservations, activités et informations pratiques. Notre solution propose une approche intégrée qui couvre l'ensemble du processus de planification : - Création et gestion de voyages avec leurs dates et destinations - Organisation des transports et hébergements - Planification d'activités et de visites - Gestion des dépenses et du budget - Exploration des destinations via une carte interactive - Recommandations personnalisées grâce à un assistant virtuel L'architecture technique mise en place repose sur des technologies éprouvées (PHP, MySQL, JavaScript, Bootstrap) et suit une structure modulaire facilitant la maintenance et l'évolution future. La base de données relationnelle a été conçue avec soin pour assurer l'intégrité et la cohérence des données, tout en permettant une gestion efficace des relations entre les différentes entités (utilisateurs, voyages, activités, etc.). L'interface utilisateur, à la fois esthétique et fonctionnelle, offre une expérience intuitive sur tous les appareils grâce à une approche responsive design. Les tests approfondis et les retours utilisateurs ont permis d'affiner l'application et de résoudre les problèmes identifiés, aboutissant à un produit stable et fiable.

## 11.2 Compétences acquises

### Compétences techniques :

- Développement web full-stack (frontend/backend)
- Conception et implémentation de bases de données relationnelles
- Programmation orientée objet en PHP
- Sécurité web (protection contre vulnérabilités)
- Intégration d'APIs tierces (cartes, services)
- Responsive design
- Tests et débogage

### Compétences transversales :

- Gestion de projet
- Travail en équipe
- Résolution de problèmes
- Documentation technique
- UX/UI design
- Veille technologique

### **11.3 Perspectives**

*TravelDream* représente une base solide qui pourrait évoluer dans plusieurs directions prometteuses. Les améliorations futures identifiées (application mobile, intégrations externes, fonctionnalités sociales, etc.) ouvrent la voie à un écosystème complet dédié au voyage. Ce projet pourrait également servir de tremplin pour explorer des domaines connexes comme : - L'intelligence artificielle appliquée aux recommandations de voyage - La réalité augmentée pour l'exploration des destinations - L'analyse de données pour identifier des tendances et préférences de voyage - Les technologies blockchain pour sécuriser les réservations et transactions En conclusion, ce projet S4 de la licence MIASHS parcours MIAGE a été une expérience formatrice et enrichissante, nous permettant d'appliquer concrètement les connaissances acquises durant notre formation et de développer des compétences précieuses pour notre avenir professionnel. "TravelDream" n'est pas seulement une application fonctionnelle, mais aussi le témoignage de notre capacité à concevoir, développer et déployer un projet web complet répondant à des besoins utilisateurs réels.