

Université Paris Nanterre

Licence MIASHS - Parcours MIAGE

Semestre 4

Rapport de Projet

Application de Planification de Voyages

"TravelDream"

Réalisé par :

Melissa AKLI

Nikola KALETKA

Encadré par :

Thibault Anani

Table des matières

1	Introduction	3
1.1	Contexte du projet	3
1.2	Objectifs et problématique	3
1.3	Méthodologie de travail	3
2	Analyse des Besoins	3
2.1	Besoins fonctionnels	3
2.2	Besoins non fonctionnels	4
2.3	Spécificités techniques	4
3	Conception de la Base de Données	5
4	Architecture Technique	5
4.1	Structure du code	5
4.2	Organisation des fichiers	6
4.3	Configuration du projet	7
4.4	Sécurité	8
5	Fonctionnalités Implémentées	9
5.1	Module d'Authentification	9
5.1.1	Inscription	9
5.1.2	Connexion	10
5.1.3	Réinitialisation du mot de passe	11
5.1.4	Déconnexion	12
6	Gestion des Voyages	12
6.1	Création et Modification d'un Voyage	12
6.2	Visualisation des voyages de l'utilisateur (<code>mesvoyages.php</code>)	13
6.3	Visualisation Détaillée d'un Voyage (<code>voyage_detail.php</code>)	14
6.4	Partage d'un Voyage	14
6.5	Carte interactive	15
6.6	Assistant de voyage (chatbot)	16
6.7	Recherche de destinations (<code>destination.php</code>)	18
7	Interface Utilisateur	20
7.1	Charte graphique	20
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)	22
7.3	Responsive Design	23
7.3.1	Approche technique	23
7.3.2	Adaptations spécifiques	24
8	Tests et Validation	24
8.1	Stratégie de test	24
8.1.1	Types de tests implémentés	24
8.1.2	Environnement de test	24
8.1.3	Outils et méthodes	24
8.2	Tests fonctionnels	25
8.3	Tests d'intégration	25
8.3.1	Flux complets	25
8.3.2	Intégrité des données	25
9	Difficultés Rencontrées et Solutions	26
9.1	Problèmes techniques	26
9.1.1	Compatibilité entre navigateurs	26
9.1.2	Gestion dynamique des formulaires complexes	26
9.1.3	Intégration des API externes (Leaflet/Nominatim)	26
10	Améliorations Futures	27
10.1	Fonctionnalités à développer	27
10.1.1	Application mobile native	27
10.1.2	Gestion collaborative des voyages	27
10.1.3	Gestion des dépenses et budget	27
10.1.4	Amélioration des performances	28
10.1.5	Renforcement de la sécurité	28
10.1.6	Intelligence artificielle et personnalisation avancée	28
11	Conclusion	28
11.1	Synthèse du projet	28
11.2	Compétences acquises	29
11.3	Perspectives	29

1 Introduction

1.1 Contexte du projet

Dans le cadre de ce projet, réalisé en binôme par Melissa et Nikola, nous avons développé une application web nommée "*TravelDream*" permettant aux utilisateurs de planifier leurs voyages de manière complète et organisée. Ce projet a aussi pour objectif d'augmenter nos compétences en développement web, en conception de bases de données et en modélisation.

1.2 Objectifs et problématique

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.

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 ce faire notre application va combiner une base de données bien structurée, une interface utilisateur intuitive et responsive, des fonctionnalités adaptées aux besoins des voyageurs (création, modification et partage de voyages, carte interactive) et un chatbot d'aide à la décision

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
- **Phase de conception** : 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 parties.
- **Phase de test** : Vérification manuelles du bon fonctionnement des fonctionnalités et correction des bugs identifiés.
- **Phase de finalisation** : Création d'un rapport

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.

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
 - Partage d'un voyage avec d'autres utilisateurs
 - Vue détaillée d'un voyage
- **Gestion des activités** (associées à un voyage)
- **Gestion des transports** (associés à un voyage)
- **Gestion des logements** (associés à un voyage)
- **Gestion des restaurants** (associés à un voyage)
- **Gestion d'une checklist avant le départ** (associée à un voyage)
- **Gestion des transports urbains** (associés à un voyage)
- **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
- **Utilisation**
 - Interface facile à prendre en main
 - Design responsive adapté aux différents appareils (ordinateurs, smartphones)
 - Temps de chargement rapide des pages
 - Feedback visuel pour les actions utilisateur (par exemple le curseur qui change quand c'est un bouton)

2.3 Spécificités techniques

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

- **Technologies utilisées**
 - PHP 8.4 pour le développement backend avec l'extension PDO pour accéder à la base de données de manière sécurisée et permettant de se prémunir des injections SQL grâce à l'utilisation de requêtes préparées

- Base de données MySQL 8.0 pour le stockage des données
- HTML5, CSS et JavaScript pour le développement frontend
- Bootstrap 5.3 pour le design responsive ainsi que Font Awesome 6.4 comme bibliothèque d'icônes
- **Infrastructure**
- Hébergement sur un serveur PHP local configuré pour le développement
- **API tierces utilisées**
 - Leaflet.js pour la carte interactive avec OpenStreetMap pour le fond de carte
 - Nominatim pour la recherche de lieux

3 Conception de la Base de Données

La base de données, nommée `planvoyages`, est le cœur de notre application. Elle est conçue pour stocker de manière structurée toutes les informations relatives aux utilisateurs, à leurs voyages, et aux détails de chaque voyage. Le schéma relationnel est implémenté en MySQL.

Les tables principales sont :

- `utilisateur` : stocke les informations des utilisateurs (identifiants, nom, email, mot de passe haché).
- `voyage` : contient les informations générales sur un voyage (destination, dates, référence à l'utilisateur propriétaire).
- `voyage_partage` : table de liaison pour gérer le partage des voyages entre utilisateurs. Elle lie un voyage à un utilisateur avec qui il est partagé.
- `activite`, `logement`, `transport`, `depense`, `restaurant`, `transport_ville`, `ticket_activite`, `item_checklist_avant_depart` : ces tables stockent les détails spécifiques associés à chaque voyage (activités prévues, réservations de logement, moyens de transport, etc.). Chacune de ces tables est liée à la table `voyage` par une clé étrangère.

Différentes contraines sont définies (clés primaires, clés étrangères, vérifications de dates, ...). Par exemple, la date de début d'un voyage doit être avant sa date de fin.

4 Architecture Technique

4.1 Structure du code

Organisation générale Le code est organisé en s'inspirant du modèle MVC (Modèle-Vue-Contrôleur), que nous avons essayé d'implémenter au mieux. Pour l'implémenter complètement, un framework PHP comme Symfony aurait pu être utilisé. Voici comment nous avons fait :

- **Modèle** : Les interactions avec la base de données sont gérées par un ensemble de fonctions utilitaires dans `utils/database_utils.php`. Ce fichier contient des fonctions pour établir la connexion (`getDbConnection()`) et exécuter des requêtes (`executeQuery()`, `fetchAll()`, `fetchOne()`, `insert()`, `update()`). Les paramètres de connexion à la base de données sont définis dans `config/config.php`.
- **Vue** : Les fichiers PHP contenant principalement du HTML (par exemple, `creation_voyage.php`, `mesvoyages.php`, `voyage_detail.php`) et avec l'aide des fichiers CSS pour le style.

- **Contrôleur** : La logique de traitement est intégrée dans les fichiers PHP principaux (par exemple, `creation_voyage_envoyer.php`, `share_voyage.php`, `pageconnexion.php`), qui gèrent les requêtes utilisateur, traitent les données, interagissent avec le modèle et préparent les réponses pour la vue.

Gestion des sessions La gestion des sessions utilisateur est implémentée de manière centralisée. Le fichier `header.php` initialise la session avec `session_start()` (si pas déjà démarrée) et des fonctions utilitaires comme `isUserLoggedIn()` (définie dans `utils/user_utils.php`) sont utilisées pour vérifier l'état de connexion de l'utilisateur. Le fichier `utils/login_check_utils.php` est inclus par `utils/utils.php` et gère la redirection des utilisateurs non connectés pour les pages protégées. En effet, certaines de nos pages (liste, modification de voyage, création, partage) ne sont pas accessibles aux utilisateurs non connectés.

Système de templates L'application utilise un système simple de templates en incluant des fichiers PHP dans d'autres fichiers :

- `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, ainsi que le bouton et le conteneur du chatbot flottant.

Ces fichiers sont inclus dans chaque page de l'application, assurant ainsi une cohérence visuelle et fonctionnelle. Le fichier `utils/utils.php` sert de point d'entrée pour inclure tous les fichiers utilitaires nécessaires (configuration, base de données, utilisateur, général, vérification de connexion).

4.2 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 (à la racine)

- `header.php`, `footer.php` : Composants de l'interface utilisateur.
- `inscription.php`, `pageconnexion.php`, `Réinitialisation.php`, `logout.php` : Gestion de l'authentification.
- `destination.php` : Affiche les destinations populaires et permet la recherche.
- `mesvoyages.php` : Affiche et gère les voyages de l'utilisateur connecté (liste, création, partage).
- `creation_voyage.php` : Formulaire pour créer ou modifier un voyage.
- `creation_voyage_envoyer.php` : Traitement du formulaire de création/modification de voyage.
- `voyage_detail.php` : Affichage détaillé d'un voyage.
- `share_voyage.php` : Traitement du partage d'un voyage.
- `get_users.php` : Récupération de la liste des utilisateurs pour le partage.
- `map.php` : Implémente la carte interactive avec géolocalisation.
- `chat.php` : Implémente l'assistant virtuel de voyage.

Dossier config

- `config.php` : Contient les paramètres de configuration, notamment pour la base de données et le site. Le détail de la configuration est donnée dans la partie 4.3 de ce rapport

Dossier utils

- `utils.php` : Fichier central important tous les autres utilitaires.
 - `database_utils.php` : Fonctions d'interaction avec la base de données (connexion, exécution de requêtes).
 - `user_utils.php` : Fonctions relatives à la gestion des utilisateurs (ex : `isUserLoggedIn()`).
 - `general_utils.php` : Fonctions utilitaires générales (redirection, sanitization, alertes, formatage de dates).
 - `login_check_utils.php` : Logique de vérification de connexion pour les pages protégées.
- ORGANISATION CSS //TODO //TBD

Dossier static (pour les fichiers CSS, images, etc.)

4.3 Configuration du projet

Configuration de la base de données La configuration de la connexion à la base de données est centralisée dans le fichier `config/config.php` pour n'avoir à mettre ses identifiants de connexion à la base de données une seule fois.

```

1 $config = [
2     'db' => [
3         'host' => '127.0.0.1',
4         'dbname' => 'PlanVoyages',
5         'username' => 'root',
6         'password' => '',
7         'charset' => 'utf8mb4',
8     ],
9     'site' => [
10        'name' => 'TravelDream',
11        'url' => 'http://localhost/projet_voyage',
12        'email' => 'contact@traveldream.com',
13    ],
14];

```

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 `utils/database_utils.php` établit une connexion PDO à la base de données en utilisant les paramètres de `config/config.php`.

```

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 Les fichiers du dossier `utils` contiennent plusieurs fonctions utilitaires importantes :

- `isUserLoggedIn()` (dans `user_utils.php`) : Vérifie si l'utilisateur est connecté en contrôlant la présence de l'ID utilisateur dans la session.
- `redirect()` (dans `general_utils.php`) : Facilite la redirection vers une autre page.
- `sanitize()` (dans `general_utils.php`) : Nettoie les données d'entrée pour prévenir les attaques XSS.
- `generateAlert()` (dans `general_utils.php`) : Génère des messages d'alerte formatés avec Bootstrap.
- `formatDate()` (dans `general_utils.php`) : Formate les dates selon le format spécifié.
- Les fonctions d'accès à la base de données (`executeQuery()`, `fetchAll()`, etc. dans `database_utils.php`)

4.4 Sécurité

Plusieurs mesures de sécurité ont été mises en place :

- Requêtes préparées (PDO) pour éviter les injections SQL, utilisées via les fonctions de `utils/database_u...`
- Mots de passe hachés avec `password_hash()` lors de l'inscription et vérifiés avec `password_verify()` lors de la connexion. Cela évite
- Nettoyage des entrées avec `htmlspecialchars()` via la fonction `sanitize()` (dans `general_utils.php`)
- Gestion des sessions pour contrôler l'accès aux pages protégées (via `utils/login_check_utils.php`).

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 (dans `inscription.php`) vérifie si l'email est déjà utilisé en interrogeant la table `Utilisateur`.
4. Si l'email est disponible, le mot de passe est haché via `password_hash()` avant d'être stocké dans la table `Utilisateur`.
5. Un message de confirmation s'affiche, invitant l'utilisateur à se connecter.

Sécurité

- Utilisation de requêtes préparées (via `fetchValue` et `insert` de `utils/database_utils.php`) pour prévenir les injections SQL.
- Hachage du mot de passe avec `bcrypt` via `password_hash()`.
- Validation des données côté serveur (vérification de l'unicité de l'email).

```

// 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 (dans pageconnexion.php) vérifie l'existence de l'email dans la table Utilisateur.
4. Si l'email existe, le mot de passe fourni est vérifié par rapport au mot de passe haché stocké, via `password_verify()`.
5. En cas de succès, les données de l'utilisateur (ID, nom, email) sont stockées en session (`$_SESSION`), puis il est redirigé vers sa page de profil (mesvoyages.php).
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 (`$_SESSION['id_utilisateur']`)
- Nom d'utilisateur (`$_SESSION['nom_utilisateur']`)
- Adresse email (`$_SESSION['email']`)

Cela permet d'identifier l'utilisateur et de personnaliser son expérience sur les pages nécessitant une authentification.

```

// 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 (`Réinitialisation.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 (table `Utilisateur`), le nouveau mot de passe est haché avec `password_hash()` 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 (via `fetchOne` et `update` de `utils/database_utils.php`) 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 `logout.php` appelle `session_start()` puis `session_destroy()`, effaçant toutes les variables de session.
3. L'utilisateur est redirigé vers la page de connexion (`pageconnexion.php`).

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 Gestions des Voyages

La gestion des voyages est une fonctionnalité centrale de *TravelDream*. Elle permet aux utilisateurs de créer, modifier, visualiser et partager leurs plans de voyage.

6.1 Création et Modification d'un Voyage

La création et la modification d'un voyage sont gérées via la page `creation_voyage.php` pour l'interface utilisateur et `creation_voyage_envoyer.php` pour le traitement des données.

Interface de création/modification (`creation_voyage.php`) L'utilisateur accède à cette page soit via un bouton "Nouveau voyage" sur `mesvoyages.php`, soit en cliquant sur "Modifier" pour un voyage existant.

- Si un `id` de voyage est passé en GET, le formulaire est pré-rempli avec les informations du voyage existant et de tous ses éléments associés (checklist, transports, logements, activités, restaurants, transports urbains), récupérés depuis la base de données.
- L'utilisateur peut saisir ou modifier le titre du voyage, les dates de début et de fin.
- Des sections dynamiques permettent d'ajouter, modifier ou supprimer des éléments pour :
 - La checklist avant départ (description, état fait/non fait).
 - Les transports (type, villes départ/arrivée, dates, horaires, bagages, terminal).
 - Les hébergements (nom, dates, adresse, horaires check-in/out, numéro de réservation, petit déjeuner).

- Les transports urbains (type de transport, type de ticket, prix, informations, lieu d'achat).
- Les activités (nom, date, horaire, adresse, informations, gestion des tickets associés avec nom, prix et lien).
- Les restaurants (nom, adresse, type, date, horaire).
- Du code JavaScript est utilisé pour gérer l'ajout et la suppression dynamiques de ces éléments dans le formulaire, ainsi que pour permettre de renommer les sections.

Traitement du formulaire (`creation_voyage_envoyer.php`) Lors de la soumission du formulaire :

1. Le script récupère l'ID de l'utilisateur depuis la session.
2. Si un `voyage_id` est présent (modification), les informations de base du voyage (destination, dates) dans la table `Voyage` sont mises à jour.
3. Si aucun `voyage_id` n'est présent (création), un nouveau voyage est inséré dans la table `Voyage`, et le nouvel ID est récupéré.
4. Pour chaque catégorie d'éléments (checklist, transport, logement, etc.) :
 - Toutes les entrées existantes associées à ce `voyage_id` dans la table correspondante (par exemple, `item_checklist_avant_depart`, `Transport`) sont d'abord supprimées.
 - Ensuite, les nouvelles données soumises via le formulaire pour cette catégorie sont insérées dans la table. Pour les activités avec ticket, une insertion est également faite dans `ticket_activite`. Cette stratégie de "supprimer puis insérer" simplifie la gestion des modifications, ajouts et suppressions d'éléments multiples.
5. Après traitement, l'utilisateur est redirigé vers `mesvoyages.php` avec un message de succès.

Toutes les opérations de base de données sont effectuées en utilisant les fonctions utilitaires de `utils/database_*` (comme `insert`, `executeQuery`).

6.2 Visualisation des voyages de l'utilisateur (`mesvoyages.php`)

- La page `mesvoyages.php` est le tableau de bord principal de l'utilisateur connecté. Elle affiche :
- Un résumé du profil utilisateur (nom, email, statistiques sur le nombre de voyages et de destinations).
 - Un bouton pour initier la création d'un nouveau voyage (redirige vers `creation_voyage.php`).
 - Des statistiques globales (nombre total de jours de voyage, prochain voyage prévu avec décompte).
 - La liste des voyages de l'utilisateur. Sont affichés les voyages créés par l'utilisateur ainsi que ceux qui ont été partagés avec lui (via la table `voyage_partage`).
 - Les voyages sont séparés en deux sections : "Voyages en cours" et "Voyages passés / à venir", déterminés en comparant les dates du voyage avec la date actuelle.
 - Pour chaque voyage, sont affichés : la destination, les dates de début et de fin, la durée, et un statut (en cours, à venir, terminé). Une icône indique si le voyage est partagé.
 - Des boutons d'action pour chaque voyage :
 - "Modifier" : redirige vers `creation_voyage.php?id={id_voyage}` pour éditer le voyage.
 - "Partager" (uniquement si l'utilisateur est propriétaire du voyage) : ouvre un modal pour gérer le partage.
 - "Détails" : redirige vers `voyage_detail.php?id={id_voyage}` pour une vue détaillée.

6.3 Visualisation Détailée d'un Voyage (`voyage_detail.php`)

Cette page permet à l'utilisateur de consulter toutes les informations relatives à un voyage spécifique.

- L'accès se fait via le bouton "Détails" sur la page `mesvoyages.php`, en passant l'ID du voyage en paramètre GET (`voyage_detail.php?id={id_voyage}`).
- Le script vérifie que l'utilisateur connecté est soit le propriétaire du voyage, soit un utilisateur avec qui le voyage a été partagé (via la table `voyage_partage`).
- Toutes les informations du voyage sont récupérées depuis la base de données : informations générales (destination, dates), checklist, transports, logements, transports urbains, activités (y compris les détails des tickets), et restaurants.
- Les informations sont présentées de manière structurée et lisible, section par section. Les dates et heures sont formatées pour une meilleure lisibilité.
- Des boutons d'action sont disponibles : "Modifier" (lien vers `creation_voyage.php`), "Partager" (si l'utilisateur est propriétaire, ouvre le modal de partage), et "Imprimer" (utilise la fonctionnalité d'impression du navigateur).

6.4 Partage d'un Voyage

La fonctionnalité de partage permet au propriétaire d'un voyage d'autoriser d'autres utilisateurs de *TravelDream* à consulter les détails de ce voyage.

Interface de partage

- Sur la page `mesvoyages.php` (et `voyage_detail.php`), un bouton "Partager" est disponible pour chaque voyage dont l'utilisateur connecté est le propriétaire.
- Cliquer sur ce bouton ouvre une fenêtre modale (Bootstrap Modal) '#shareModal'.
- Au chargement du modal, une requête AJAX est envoyée au script `get_users.php`.
- `get_users.php` récupère la liste de tous les utilisateurs (sauf l'utilisateur actuel) depuis la table `Utilisateur`, ainsi que la liste des utilisateurs avec qui le voyage spécifique est déjà partagé (depuis la table `voyage_partage`).
- Le modal est ensuite peuplé avec une liste d'utilisateurs sous forme de cases à cocher. Les utilisateurs avec qui le voyage est déjà partagé ont leur case pré-cochée.

Traitement du partage (`share_voyage.php`)

- Lorsque l'utilisateur soumet le formulaire du modal de partage, les données (ID du voyage et liste des ID des utilisateurs sélectionnés) sont envoyées en POST à `share_voyage.php`.
- Le script vérifie d'abord que l'utilisateur connecté est bien le propriétaire du voyage.
- Il procède ensuite à une opération de "supprimer puis insérer" sur la table `voyage_partage` :
 1. Toutes les entrées existantes pour cet `id_voyage` dans `voyage_partage` sont supprimées.
 2. De nouvelles entrées sont insérées pour chaque utilisateur coché dans le formulaire du modal.
- Une réponse JSON indiquant le succès ou l'échec de l'opération est retournée au client, qui peut alors afficher une notification et fermer le modal.

Actuellement, le partage permet la consultation. Une évolution pourrait être d'ajouter des niveaux de permission (lecture seule, édition collaborative).

6.5 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);
        });
    }
});
```

6.6 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. Il est accessible via un bouton flottant sur toutes les pages (`footer.php`).

Fonctionnalités

- Interface de chat intuitive intégrée dans une iframe.
- Base de données de destinations (définie en JavaScript dans `chat.php`) avec leurs caractéristiques (climat, budget, activités, etc.).
- Algorithme de recommandation basé sur les préférences exprimées par l'utilisateur (continent, type de voyage, budget, climat, durée).
- Présentation des destinations recommandées avec leurs informations principales et un lien vers la page `destination.php` pour plus de détails.

Implémentation technique L'assistant est implémenté en JavaScript côté client dans le fichier `chat.php`. Il maintient un état de la conversation (`chatbotState`) et pose une série de questions pour affiner les préférences de l'utilisateur.

```

// 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 (stockées dans le tableau JavaScript `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);
}

```

6.7 Recherche de destinations (destination.php)

Cette fonctionnalité, accessible via la page `destination.php`, permet aux utilisateurs de découvrir des lieux de voyage populaires et d'obtenir des informations détaillées sur ceux-ci.

Fonctionnalités

- Affichage d'une sélection de destinations populaires (définies dans un tableau PHP `$destinations_info`) avec images et descriptions.
- Barre de recherche pour trouver une destination spécifique parmi celles prédéfinies.
- Si une destination est recherchée ou sélectionnée, affichage détaillé de ses informations (description, meilleure période, budget, activités recommandées).

- Possibilité pour un utilisateur connecté de planifier directement un voyage vers la destination choisie (lien vers `creation_voyage.php` avec la destination pré-remplie).

Implémentation technique Les informations sur les destinations populaires sont codées en dur dans un tableau PHP au sein du fichier `destination.php`. La recherche filtre ce tableau.

```
// 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 Palettes 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 (`destination.php`) 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.

The wireframe shows the TravelDream homepage. At the top, there's a navigation bar with 'TravelDream' logo, 'Accueil', 'Carte', 'Connexion' (which is highlighted with a yellow oval), and 'Inscription'. Below the navigation is a large banner with the heading 'Explorez nos destinations' and a subtext 'Découvrez des lieux incroyables et planifiez votre prochain voyage'. It features a search bar with 'Rechercher une destination...' and a 'Rechercher' button. The background of the banner is a world map with various travel-related icons like compasses and planes. Below the banner is a section titled 'Destinations populaires' with the subtext 'Explorez nos destinations les plus appréciées par nos voyageurs'. This section displays three images of famous landmarks: the Pont Alexandre III in Paris, the Rialto Bridge in Venice, and a view of London with the River Thames and Tower Bridge. A small yellow speech bubble icon is in the top right corner of this section.

7.2.2 Pages d'authentification

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

The wireframe shows the TravelDream login page. It features a 'TravelDream' logo at the top left and a navigation bar with 'Accueil', 'Carte', 'Connexion' (highlighted with a yellow oval), and 'Inscription'. The main area has a background image of a world map with travel icons like planes, compasses, and landmarks. Overlaid on this background is a central login form with the heading 'Accéder à mon voyage'. It contains fields for 'Email' and 'Mot de passe', and a 'Connecter' button. Below the form are links for 'Mot de passe oublié?' and 'S'inscrire'. A small yellow speech bubble icon is in the bottom right corner of the form area.

7.2.3 Profil et gestion des voyages

La page de profil (`mesvoyages.php`) est divisée en deux sections principales : un résumé du profil avec statistiques à gauche, et la liste des voyages (en cours, passés/à venir) à droite. Chaque voyage est présenté dans une carte contenant les informations essentielles et des boutons d'action (modifier, partager, détails).

The screenshot shows the TravelDream user interface for managing trips. At the top, there's a navigation bar with links for Accueil, Carte, Mon Profil, and Déconnexion. On the left, a sidebar for 'Melissa' shows her profile picture, name, email, and stats: 2 Voyages and 2 Destinations. It also has a '+ Nouveau voyage' button. Below this is a 'Statistiques' section showing 26 days of travel and a next trip to Roma in 34 days. The main area is titled 'Mes Voyages' and lists two trips: one to Roma from June 20 to June 30 (11 days away), and another to Londre from May 10 to May 24 (15 days away). Each trip card includes a 'Modifier' button, a 'Détails' button, and a small info icon. A floating yellow speech bubble icon is visible on the right.

7.2.4 Carte interactive

La carte interactive (`map.php`) occupe la majeure partie de l'écran et inclut :

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

The screenshot shows the TravelDream interactive map page. At the top, there's a search bar with placeholder 'Rechercher un lieu...', a 'Rechercher' button, and a 'Utiliser ma position' button. The main area is a detailed map of Europe and parts of Africa and Asia. A blue marker indicates the user's current location in Paris, France. The map shows country borders, city names, and major rivers. In the bottom right corner, there's a small yellow speech bubble icon.

7.2.5 Assistant de voyage (chatbot)

L'interface de l'assistant de voyage (`chat.php`), accessible via un bouton flottant, s'affiche dans un conteneur dédié. Elle présente une zone de dialogue pour l'interaction avec le bot.

Assistant de Voyage

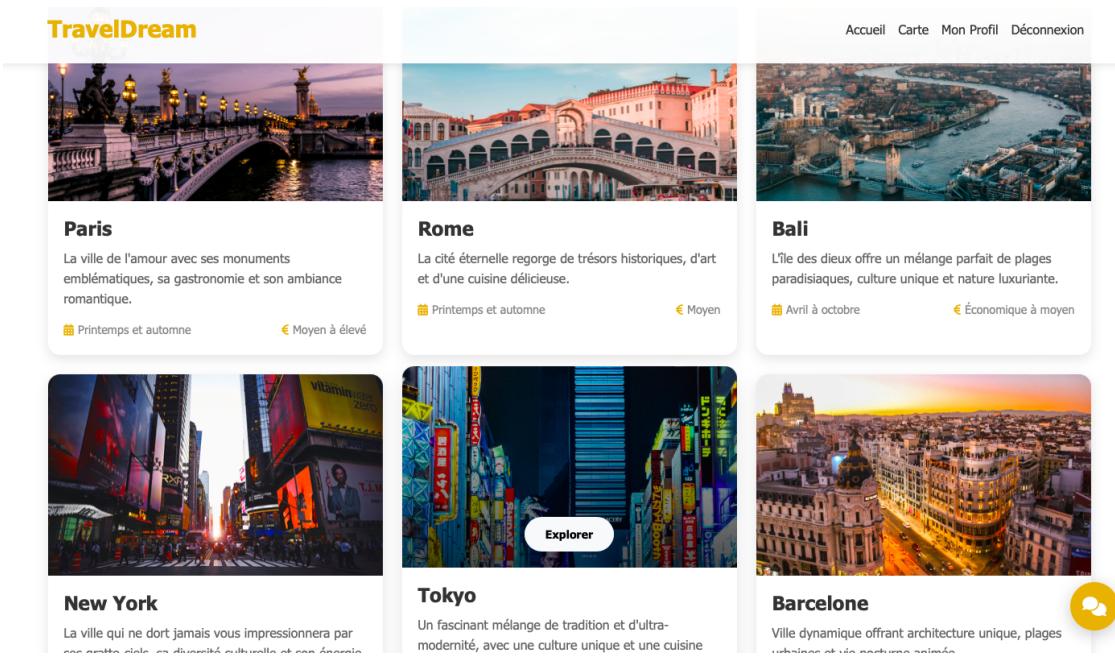
Notre assistant intelligent vous aide à trouver la destination parfaite selon vos préférences et votre budget.

 TravelDream Assistant
En ligne

Bonjour ! Je suis l'assistant TravelDream. Je vais vous aider à trouver la destination idéale pour votre prochain voyage. Quel continent souhaitez-vous explorer ? (Europe, Asie, Afrique, Amérique du Nord, Amérique du Sud, Océanie, ou 'tous')

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.



The screenshot displays the TravelDream mobile application's main screen. At the top, there is a header bar with the 'TravelDream' logo on the left and navigation links ('Accueil', 'Carte', 'Mon Profil', 'Déconnexion') on the right. Below the header, there is a yellow banner featuring a robot icon and the text 'TravelDream Assistant En ligne'. The main content area is a grid of six cards, each representing a travel destination:

- Paris**: Shows a bridge at night. Text: 'La ville de l'amour avec ses monuments emblématiques, sa gastronomie et son ambiance romantique.' Buttons: 'Printemps et automne' and 'Moyen à élevé'.
- Rome**: Shows the Rialto Bridge. Text: 'La cité éternelle regorge de trésors historiques, d'art et d'une cuisine délicieuse.' Buttons: 'Printemps et automne' and 'Moyen'.
- Bali**: Shows an aerial view of a coastal city. Text: 'L'île des dieux offre un mélange parfait de plages paradisiaques, culture unique et nature luxuriante.' Buttons: 'Avril à octobre' and 'Économique à moyen'.
- New York**: Shows a street scene with bright billboards. Text: 'La ville qui ne dort jamais vous impressionnera par son énergie vibrante, sa diversité culturelle et son énergie'. Buttons: 'Printemps et automne' and 'Moyen à élevé'.
- Tokyo**: Shows a dense urban area with many signs. Text: 'Un fascinant mélange de tradition et d'ultra-modernité, avec une culture unique et une cuisine'. Button: 'Explorer'.
- Barcelone**: Shows a cityscape at sunset. Text: 'Ville dynamique offrant architecture unique, plages sableuses et une gastronomie animée'. Buttons: 'Printemps et automne' and 'Économique à moyen'.

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 (géré par Bootstrap).
- Cartes de voyage et de destination : s'adaptent en colonnes (par exemple, 1 colonne sur mobile, 2 ou 3 sur écrans plus larges).
- 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 (conceptuels)** : vérification du bon fonctionnement de chaque composant PHP individuel (fonctions utilitaires, logique de traitement d'un script spécifique).
- **Tests d'intégration** : validation des interactions entre les différents modules (par exemple, authentification puis création d'un voyage, partage d'un voyage et vérification de l'accès par un autre utilisateur).
- **Tests fonctionnels** : contrôle du respect des exigences pour chaque fonctionnalité (création de compte, connexion, ajout d'un voyage, ajout d'une activité à un voyage, partage, etc.).
- **Tests d'interface utilisateur** : vérification de l'apparence, de la réactivité (responsive design) et du comportement sur divers navigateurs et appareils.
- **Tests de sécurité** : identification des vulnérabilités, notamment sur l'authentification, la gestion des sessions, la protection contre les injections SQL et XSS.

8.1.2 Environnement de test

Les tests ont été réalisés dans un environnement de développement local :

- Serveur local (XAMPP/WAMP/MAMP) avec PHP et MySQL.
- Base de données de test peuplée avec des jeux de données variés et réalistes.
- Navigateurs : Chrome, Firefox, Safari, Edge.
- Outils de développement des navigateurs pour simuler différents appareils et tailles d'écran.

8.1.3 Outils et méthodes

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

- **Tests manuels** : exécution de scénarios d'utilisation typiques (créer un compte, planifier un voyage complet, partager, consulter) avec documentation des résultats et des anomalies.

- **Validation W3C** : vérification de la conformité du HTML et du CSS générés aux standards du web.
- **Outils de développement des navigateurs** : inspection du DOM, débogage JavaScript, analyse des requêtes réseau, et analyse des performances de chargement.
- **Checklists fonctionnelles** : listes de contrôle basées sur les besoins fonctionnels pour s'assurer que chaque fonctionnalité est testée.

8.2 Tests fonctionnels

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

- **Inscription** : création de compte avec email unique, hachage du mot de passe.
- **Connexion** : accès avec identifiants valides, refus avec identifiants invalides, création de session.
- **Création de voyage** : enregistrement correct des informations de base et de tous les éléments associés (checklist, transports, etc.) dans la base de données.
- **Modification de voyage** : mise à jour correcte des informations, y compris la suppression et l'ajout d'éléments dynamiques.
- **Partage de voyage** : un propriétaire peut partager un voyage, les utilisateurs partagés peuvent le voir dans `mesvoyages.php` et `voyage_detail.php`.
- **Chatbot** : capacité à poser des questions et à recevoir des recommandations de destinations pertinentes.
- **Carte interactive** : recherche de lieux et géolocalisation fonctionnelles.

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 (avec activités, logements)** → **Modification du voyage** → **Visualisation détaillée** → **Partage avec un autre utilisateur** → **Déconnexion**.
- **Connexion d'un deuxième utilisateur** → **Vérification de l'accès au voyage partagé (liste et détails)** → **Tentative de modification (devrait être impossible si non propriétaire)** → **Déconnexion**.
- **Recherche de destination sur destination.php** → **Planification d'un voyage à partir de cette destination** → **Complétion des détails du voyage**.
- **Utilisation de la carte interactive pour trouver un lieu** → **Utilisation du chatbot pour des recommandations**.

8.3.2 Intégrité des données

L'intégrité des données a été contrôlée pour :

- La création, modification et suppression (logique de "delete-then-insert") des voyages et de leurs éléments associés (activités, logements, etc.).
- La vérification des contraintes d'intégrité référentielle (clés étrangères entre Voyage et Activite, Logement, etc., et entre Activite et ticket_activite).
- La cohérence des données entre les différentes tables après chaque opération CRUD.
- La gestion correcte des partages dans la table voyage_partage.

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 : Des différences mineures d'affichage des éléments d'interface (notamment avec les formulaires complexes et les mises en page Flexbox/Grid) ont été observées entre les navigateurs.

Solution : L'utilisation de Bootstrap 5 a grandement aidé à minimiser ces problèmes grâce à ses composants et son système de grille normalisés. Des tests réguliers sur Chrome, Firefox, Safari et Edge ont permis d'identifier et de corriger les écarts restants, parfois en utilisant des réinitialisations CSS (normalize.css implicitement inclus par Bootstrap) ou des ajustements CSS spécifiques.

9.1.2 Gestion dynamique des formulaires complexes

Problème : Le formulaire de création/modification de voyage (`creation_voyage.php`) contient de nombreuses sections dynamiques (checklist, transports, activités, etc.) où l'utilisateur peut ajouter ou supprimer des éléments. La gestion des noms des champs (`name="transport[index][type]"`) et la synchronisation avec le backend pour la sauvegarde étaient complexes.

Solution :

- **JavaScript côté client** : Du JavaScript a été utilisé pour cloner les templates HTML des éléments, incrémenter les index dans les noms des champs, et gérer la suppression d'éléments.
- **Traitements côté serveur (`creation_voyage_envoyer.php`)** : Le PHP a été structuré pour boucler sur les tableaux de données reçus (par exemple `$_POST['transport']`). La stratégie "supprimer toutes les anciennes entrées puis insérer les nouvelles" pour chaque sous-section (transports, logements, etc.) a simplifié la logique de mise à jour, évitant de devoir déterminer quels éléments étaient nouveaux, modifiés ou supprimés individuellement.

9.1.3 Intégration des API externes (Leaflet/Nominatim)

Problème : Assurer le chargement correct de la carte Leaflet, la gestion des appels à l'API Nominatim pour le géocodage (limites de taux, gestion des erreurs) et l'affichage des résultats de manière conviviale.

Solution :

- **Chargement asynchrone/Lazy loading (conceptuel)** : Pour la carte, s'assurer qu'elle ne bloque pas le rendu initial de la page. L'initialisation de la carte dans `map.php` est effectuée après le chargement du DOM.
- **Gestion des erreurs** : Des blocs `try...catch` en JavaScript pour les appels `fetch` à Nominatim et des messages d'alerte pour l'utilisateur en cas d'échec.
- **Debounce/Throttle** : Implémentation de fonctions `debounce` et `throttle` dans `map.php` pour optimiser les appels à l'API de recherche lors de la saisie et pour les animations au défilement.

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.

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 Gestion collaborative des voyages

Actuellement, le partage de voyage est en mode "lecture seule" pour le destinataire. **Description** : Permettre au propriétaire d'un voyage d'inviter d'autres utilisateurs à collaborer sur la planification. Cela impliquerait la gestion de droits d'édition (qui peut modifier quoi). **Implémentation envisagée** :

- Ajout d'un champ de permission dans la table `voyage_partage` (ex : 'read', 'edit').
- Adaptation des formulaires de `creation_voyage.php` et du traitement dans `creation_voyage_envoyer` pour vérifier les permissions avant toute modification par un utilisateur partagé.
- Interface pour le propriétaire pour gérer les permissions des collaborateurs.

10.1.3 Gestion des dépenses et budget

La table `dépense` existe dans la base de données mais n'est pas encore exploitée. **Description** : Permettre aux utilisateurs de définir un budget pour leur voyage et de suivre leurs dépenses par catégorie.

Implémentation envisagée :

- Ajout d'une section "Dépenses" dans le formulaire de `creation_voyage.php` pour ajouter des dépenses (catégorie, montant, date).
- Modification de `creation_voyage_envoyer.php` pour sauvegarder ces dépenses.
- Affichage d'un résumé des dépenses et comparaison avec le budget sur `voyage_detail.php`.
- Graphiques pour visualiser la répartition des dépenses.

10.1.4 Amélioration des performances

- **Cache** : Mettre en place un système de cache (côté serveur avec Redis ou Memcached, ou côté client) pour les données fréquemment accédées.
- **Optimisation des requêtes SQL** : Analyse et optimisation des requêtes complexes, ajout d'index pertinents sur les tables de la base de données.
- **Lazy loading** : Généraliser le lazy loading pour les images et autres contenus lourds sur toutes les pages.
- **Minification** : Minifier les fichiers CSS et JavaScript pour réduire leur taille.
- **CDN** : Utiliser un Content Delivery Network pour servir les ressources statiques (CSS, JS, images) plus rapidement.

10.1.5 Renforcement de la sécurité

Mesures envisagées :

- Authentification à deux facteurs (2FA).
- Audit de sécurité régulier et corrections des vulnérabilités potentielles.
- Chiffrement plus poussé de certaines données sensibles si nécessaire (au-delà du hachage des mots de passe).
- Protection CSRF (Cross-Site Request Forgery) en utilisant des tokens dans les formulaires.
- Rate limiting sur les tentatives de connexion et autres actions sensibles pour prévenir les attaques par force brute.

10.1.6 Intelligence artificielle et personnalisation avancée

Objectif : Offrir une assistance encore plus intelligente et personnalisée.

Applications possibles :

- Améliorer l'assistant virtuel avec du Natural Language Processing (NLP) plus avancé pour comprendre des requêtes plus complexes.
- Suggestions proactives basées sur les voyages précédents de l'utilisateur ou les tendances populaires.
- Analyse prédictive des meilleures périodes pour voyager ou des fluctuations de prix (nécessiterait des sources de données externes).
- Génération automatique d'itinéraires optimisés en fonction des activités sélectionnées et des contraintes de temps.

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 titres, dates et destinations.
- Organisation détaillée des transports, hébergements, activités (avec gestion des tickets), restaurants, transports urbains et checklists avant départ.
- Partage des plans de voyage avec d'autres utilisateurs pour consultation.
- 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 organisée en dossiers (`utils`, `config`, `static`) 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, et tous les éléments d'un voyage). L'interface utilisateur, à la fois esthétique et fonctionnelle, offre une expérience intuitive sur tous les appareils grâce à une approche responsive design et à l'utilisation de Bootstrap. Les tests fonctionnels et d'intégration 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 HTML/CSS/JS, backend PHP).
- Conception et implémentation de bases de données relationnelles (MySQL).
- Programmation PHP procédurale et orientée fonctionnalités.
- Sécurité web (protection contre injections SQL via PDO, XSS via `htmlspecialchars`, hachage de mots de passe).
- Intégration d'APIs tierces (Leaflet.js, Nominatim).
- Responsive design avec Bootstrap.
- Développement de fonctionnalités dynamiques avec JavaScript (gestion de formulaires, AJAX pour le partage).
- Tests manuels et débogage d'applications web.

Compétences transversales :

- Gestion de projet (découpage en phases, suivi).
- Travail en équipe (collaboration, communication).
- Analyse des besoins et conception de solutions.
- Résolution de problèmes techniques.
- Rédaction de documentation technique (rapport).
- Conception de l'expérience utilisateur (UX) et de l'interface utilisateur (UI) de base.
- Veille technologique sur les outils et frameworks web.

11.3 Perspectives

TravelDream représente une base solide qui pourrait évoluer dans plusieurs directions. Les améliorations futures identifiées (application mobile, gestion collaborative, gestion des dépenses, IA avancée,

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'intégration de services de réservation externes (vols, hôtels, activités).
- L'analyse de données pour identifier des tendances et des préférences de voyage
- L'utilisation de technologies plus modernes pour le frontend (frameworks JavaScript comme Vue.js ou React) ou backend (frameworks PHP comme Symfony ou Laravel) pour des projets de plus grande envergure.

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.