## Dossier de synthèse

Pour le titre de Développeur Web et Web Mobile

Niveau 5

# « Au Paradis d'Eve »

## Réservation de chambre d'hôtes

Davina HOUQUET

2024

# Sommaire

# Présentation du projet

## Présentation personnelle

Je m'appelle Davina et j'ai 24 ans. Après l'obtention de mon premier diplôme d'éducatrice sportive à l'Université de Strasbourg en 2019, j'ai exercé dans le milieu médico-social auprès d'un public en situation de handicap mental jusqu'en décembre 2022. À la suite de quoi j'ai entamé une reconversion professionnelle en 2023 dans le développement web chez Elan Formation.

Bien que le milieu éducatif et social soit très enrichissant, j'ai décidé de me tourner vers le développement web qui dispose d'une nature évolutive et correspond à mon besoin de diversité, de renouvellement. En ce sens, l'aspect moins conventionnel de ce domaine m'a également permis de me projeter dans une carrière de développeuse web.

Dans le cadre de ma reconversion professionnelle, j'ai été orienté vers l'organisme de formation Elan à Strasbourg, qui propose la formation de développeur web et web mobile. Avant d'intégrer cette formation, j'ai effectué une remise à niveau de trois mois m'ayant permis d'acquérir des bases d'algorithmique, de HTML/CSS et de PHP.

## Présentation du projet

Le projet que je présente dans ce dossier se matérialise sous la forme d'un **site de réservation de chambres d'hôte**, permettant aux visiteurs de réserver leurs séjours dans une maison.

Le nom choisi pour ce projet « Au Paradis d'Eve », trouve son inspiration dans l'hôtel « Paradis d'Ouvéa » situé sur l'île natale de la personne associée à ce projet.

Ce projet vise à **proposer des chambres d'hôtes aux touristes de la région du Pays de Bitche** et a été réalisé en collaboration avec la personne associée. Ses idées et ses souhaits ont guidé ma démarche, me donnant une vision claire de la direction à suivre.

L'application est composée d'une présentation globale de l'établissement, d'une description des alentours, puis d'une description détaillée de chaque espace de vie, à partir de laquelle il est possible d'accéder au service de réservation.

### Concernant l'utilisateur…

L'utilisateur peut créer la **réservation d'une chambre**, sans nécessairement devoir créer un compte. Il peut se **créer un compte** et le supprimer. Il peut **consulter l'historique de ses réservations**, ainsi que

celles à venir, depuis son compte. Il a également accès aux détails de ses réservations grâce au fichier PDF généré lors d'une finalisation de réservation.

L'utilisateur peut laisser un avis et une note sur l'espace réservé lorsque sa réservation est terminée, qui apparaîtra ensuite dans son détail. Il peut retrouver l'ensemble des avis qu'il a laissé sur son profil. Il peut contacter la responsable de l'établissement via un formulaire de contact sur la page d'accueil.

### Concernant l'administrateur…

L'administrateur peut modifier tous les textes de la page d'accueil, ainsi que ceux dans la partie « Alentours ». Il peut créer, modifier ou supprimer des espaces de vie, ainsi qu'y ajouter des images. Il peut gérer les options proposées dans la réservation, en ajouter ou en supprimer. L'administrateur peut accéder à l'ensemble des réservations à venir de tous les clients confondus.

# Cahier des charges

## Barre de navigation et footer

La barre de navigation et le footer sont deux éléments communs à l'intégralité des pages. Le premier étant en haut de page, et le deuxième en bas de page.

La barre de navigation contient les noms cliquables de toutes les autres pages (Accueil, Alentours, Réservation, Chambres et Espaces de vie, Inscription, Connexion, Déconnexion).

Le footer contient quant à lui les liens cliquables vers les questions pratiques, mentions légales et conditions générales d'utilisation.

## Page d'accueil

La page d'accueil se compose d'abord d'informations générales sur l'établissement, ses fonctions et sa localisation, avec un lien vers la page de réservation. Une image en fond, conformément à la maquette Figma.

Au scroll, une nouvelle section apparaît, faisant apparaître les chambres avec une photo et leur prix. En continuant, une nouvelle section contenant d'abord un effet parallax, puis une photo des alentours avec un texte descriptif. Enfin, les informations de l'établissement, ainsi qu'un formulaire de contact se trouvent en bas de page, au-dessus du footer.

Dans une mesure de respect du RGPD, le formulaire de contact n'oblige pas l'utilisateur à entrer son nom, prénom et son numéro de téléphone, mais uniquement son courriel.

## Inscription/Connexion

L'utilisateur peut créer un compte, en choisissant un pseudo, en entrant une adresse mail valide, et en renseignant et confirmant un mot de passe. Dans un souci de respect du RGPD, une case à cocher est également présente dans le formulaire, afin d'informer l'utilisateur sur l'utilisation de ses données et d'obtenir son consentement.

## Alentours

La page des alentours contient cinq propositions de sorties rédigées par l'administrateur, disposant chacune d'un titre, d'une description, d'un budget approximatif et chacune illustrées par une image.

## Espaces de vie et détails

Une page de choix est mise à disposition de l'utilisateur cliquant sur l'onglet « Chambres et espaces de vie », sur laquelle se trouvent tous les espaces de l'établissement. Chaque espace est représenté avec une photo en arrière-plan et le nom de l'espace en premier-plan.

Au clic d'un espace, l'utilisateur arrive sur la page de détails de celui-ci. Composée en premier lieu de son nom, de ses caractéristiques (taille, capacité d'accueil en nombre de personnes, et avec le logo wifi s'il en dispose), suivi d'une description. A côté se trouve un carrousel de photos, ainsi qu'un calendrier permettant une vue globale de ses disponibilités s'il s'agit d'un espace réservable. En dessous se trouvent les cartes cliquables des autres espaces. Enfin, si des avis ont été laissés par des utilisateurs, ils sont consultables à la fin de la page.

## Réservation

La page de réservation contient l'ensemble des chambres, apparaissent sous forme de carte, reprenant le nom de la chambre, ses caractéristiques, sa description, ainsi que son prix.

Le formulaire de réservation se compose de deux à trois parties. La première partie étant de renseigner les informations personnelles nécessaires, telles que le prénom, nom, numéro de téléphone, nombre de personnes, dates de début et fin de séjour, en ajoutant ou non des options (détaillées de leur tarif). Si les conditions sont remplies, la deuxième partie de la réservation sera soit de poursuivre en se connectant ou en créant un compte, soit de poursuivre sans créer de compte. La troisième partie de la réservation permettra d'abord de visualiser un résumé de la première partie, avec les dates, le nombre de personnes, la chambre sélectionnée et ses caractéristiques, ainsi qu'un formulaire de coordonnées à remplir. Ce dernier contient un champ d'adresse mail, une adresse postale, ainsi qu'une vue du prix total.

### Avis

L'onglet « Avis » aboutit sur une page affichant les avis laissés par les utilisateurs ayant passé un séjour dans l'établissement, classés par date, du plus récent au plus ancien.

### Compte utilisateur

Chaque utilisateur se voit attribuer l'un des deux rôles définis sur la plateforme : administrateur ou utilisateur. Cette distinction permet une segmentation claire des responsabilités et des fonctionnalités accessibles à chaque catégorie d'utilisateur. L'administrateur peut effectuer les mêmes opérations mais pourra, en plus, superviser et gérer efficacement l'ensemble de la plateforme.

L'utilisateur disposant d'un compte pourra retrouver dans l'onglet portant le nom de son pseudo différentes informations et fonctionnalités. D'abord, ce dernier aura la possibilité de modifier son pseudo, son courriel ou son mot de passe, ainsi que celle de pouvoir supprimer son compte. Il aura également la possibilité de consulter ses réservations passées, ses réservations en cours et celles à venir, avec pour chacune, leurs dates, prix et détails, comme les options sélectionnées et leur tarif. Depuis chaque réservation, il est possible de télécharger une facture détaillée en PDF, et de laisser un avis sur la chambre réservée.

## Liste des compétences couvertes par REAC[1]

### Développer la partie front-end d'une application web ou web mobile sécurisée

- Installer et configurer son environnement de travail en fonction du projet web ou web mobile
- Maquetter des interfaces utilisateur web ou web mobile
- Réaliser des interfaces utilisateur statiques web ou web mobile
- Développer la partie dynamique des interfaces utilisateur web ou web mobile

### Développer la partie back-end d'une application web ou web mobile sécurisée

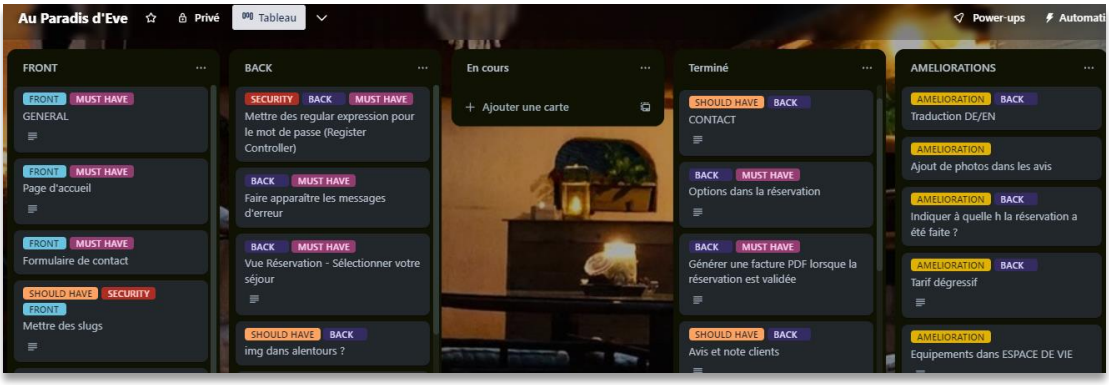
- Mettre en place une base de données relationnelle
- Développer des composants d'accès aux données SQL et NoSQL
- Développer des composants métier coté serveur
- Documenter le déploiement d'une application dynamique web ou web mobile

[1] Référentiel Emploi Activité Compétences

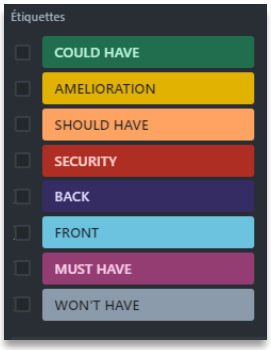
# Gestion de projet

## Trello, méthode MOSCOW et Kanban

Pour organiser mon projet, j'ai utilisé l'outil de gestion de projet en ligne Trello en m'inspirant de la méthode **Kanban**. J'ai créé un tableau avec des listes représentant les différentes étapes, telles que "Front-end", "Back-end", "En cours" et "Terminé". Les tâches front-end et back-end ont été décomposées en cartes, que j'ai ensuite organisées en fonction de la priorité établie avec la méthode **MoSCoW**. En créant Cela m'a permis d'avoir une vue d'ensemble du travail restant et de celui déjà effectué, tout en évitant de perdre le fil des tâches en cours.




Pour hiérarchiser les cartes, j'ai utilisé la méthode **MoSCoW** en attribuant des étiquettes personnalisées dans Trello. Chaque tâche a été étiquetée en fonction de sa priorité, que ce soit "**Must-have**" pour les éléments absolument nécessaires au bon fonctionnement de l'application, "**Should have this if at all possible**" pour les véritables valeurs ajoutées qui peuvent être implémentées dans le temps, après les points prioritaires, "**Could-have this if it does not affect anything else**" pour les éléments qu'il est bien d'avoir, les petits plus qui contribuent à la satisfaction client, ou "**Won't-have this time but would like in the future**" pour les tâches exclues du projet lors de la phase actuelle de traitement, mais qui pourraient être traitées dans une prochaine intégration. Chaque étiquette dispose de sa couleur représentative en fonction de l'importance de la tâche à effectuer. En filtrant les tâches par ces étiquettes, j'ai pu visualiser facilement les priorités et m'assurer que l'équipe se concentrait sur les éléments essentiels du projet.



### Méthode agile : SCRUM

La méthode agile **Scrum** [2]est destinée à la gestion de projets informatiques. Le principe de Scrum est de pouvoir modifier la direction prise par le projet au fur et à mesure de son avancement. La mêlée est donc une phase essentielle dans la gestion de projet. Si les conditions de réussite ne sont pas remplies, alors il faut réorienter le projet pour repartir sur de meilleures bases.

Son fonctionnement repose sur le principe des **sprints**[3]. Outre le sprint, il existe deux autres événements Scrum, les réunions debout quotidiennes et les rétrospectives de sprint. Les réunions debout quotidiennes ont lieu tous les jours, comme leur nom le suggère. En 15 minutes, elles permettent à l'équipe Scrum d'interagir et de coordonner ses tâches pour la journée. Quant aux rétrospectives de sprint, elles sont organisées par le Scrum master à chaque fin de sprint.

La méthode Scrum a été utilisée dans ce projet, à ceci près que des messages quotidiens en début de journée listant les tâches à effectuer pour la journée ont remplacé les réunions d'équipe. Un point régulier a été effectué avec la personne associée afin d'évaluer la direction du projet.

### Méthode agile MVP : Minimum Viable Product

Produit Minimum Viable : Il s'agit d'un concept issu de la méthodologie Lean Startup et du développement agile. Un MVP est la version la plus simple d'un produit qui satisfait aux besoins de base des utilisateurs tout en permettant aux créateurs de recueillir rapidement des retours et des données sur le produit.

## Langages et technologies utilisées

### Langages

Les choix technologiques opérés dans le cadre du développement de l'application impliquent l'utilisation de langages spécifiques pour répondre aux divers besoins fonctionnels. Ces langages incluent :

- **HTML** (*HyperText Markup Language*) : Utilisé comme langage de balisage, HTML est conçu pour représenter de manière structurée les pages web et leur contenu.

[2]Traduction française : « mêlée », tient son nom du monde du rugby.
[3] Cycles de travail de deux semaines à l'issue desquels un livrable est attendu

- CSS (*Cascading Style Sheet*) : En tant que langage informatique, CSS est employé pour décrire la présentation des documents HTML à travers des paramètres de style, assurant ainsi une mise en forme cohérente et esthétique.

- PHP (*PHP Hypertext Preprocessor*) : En tant que langage de script open source interprété côté serveur, PHP joue un rôle crucial dans le développement de sites web dynamiques. Sa longue histoire et sa large adoption en font un choix robuste pour la manipulation des données et la génération de contenu dynamique sur le serveur.

- Javascript: En tant que langage de programmation de script, Javascript est principalement utilisé pour créer des pages web interactives afin d'améliorer l'expérience utilisateur.

- SQL : Structured Query Language, utilisé dans le Système de Gestion de Base de Données MySQL afin de communiquer avec la base de données.

# Environnement de travail

## VSCode

Pour développer « Au Paradis d'Eve », j'ai utilisé l'outil VSCode qui est un éditeur de code source gratuit développé par Microsoft et qui permet d'installer de nombreuses extensions selon le besoin.

Installation : il est possible d'installer VSCode depuis le playstore ou depuis leur site internet en fonction du système d'exploitation utilisé.

Exemples d'extensions utilisées dans ce projet :

- « Auto Close Tag » de Jun Han, qui permet de clore automatiquement une balise HTML
- « Auto Rename Tag » de Jun Han, qui permet de renommer automatiquement la deuxième balise lorsque sa paire est modifiée
- « Indent-rainbow » de Oderwat, qui permet de clarifier l'indentation en utilisant des couleurs
- « Material Icon Theme » de Philipp Kief, permettant de visualiser des icônes dans l'arborescence du projet
- « Twig pack » de Bajdzis, qui permet l'auto-completion dans les fichiers twig, ainsi qu'une syntaxe adaptée

# Git

L'application a été développée à l'aide de l'outil Git via la plateforme Github. Cet outil permet un suivi précis de l'évolution du code source tout au long du processus de développement. Chaque modification apportée au code est enregistrée, ce qui facilite la gestion des versions et la possibilité de revenir à des états antérieurs en cas de besoin. Cela m'a assuré une meilleure stabilité et un contrôle rigoureux sur le code de l'application.

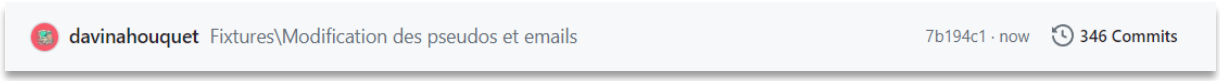
Pour utiliser Git sur Windows, il est nécessaire d'installer Git Bash[4], qui est une application pour les environnements Windows qui fournit une couche d'émulation pour une expérience de ligne de commande Git.

Exemples de certaines commandes fréquemment utilisées :

- Pour définir le mail d'un utilisateur : `git config --global user.email exemple@exemple.com`
- Pour créer un nouveau dépôt GIT : `git init`
- Pour afficher la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés : `git status`
- Pour fusionner toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local : `git pull`
- Pour enregistrer les changements qui ne doivent pas être commit immédiatement : `git stash`

## Commit

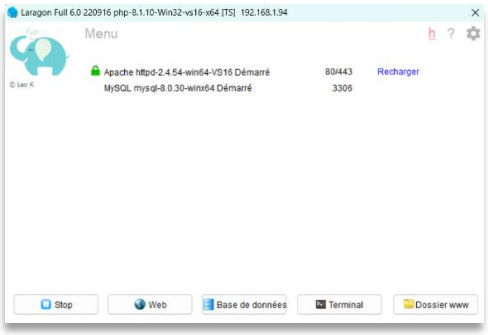
Un commit est une unité de travail participant à un projet. Il a la charge d'enregistrer de petits groupes de modifications significatives sur les fichiers, accompagné d'un message de validation décrivant brièvement ces modifications. Ce projet, par exemple, dispose d'un nombre élevé de commits (400+). En effet, j'ai régulièrement enregistré les modifications apportées pour avoir un suivi à jour à distance. Cela m'a permis d'avoir une vue d'ensemble des modifications apportées, et d'avoir une solution sur un dépôt distant en cas de perte de mon projet en local.




Dans cet exemple, on constate en premier lieu l'auteur du commit, suivi du message accompagnant les modifications, et du côté droit l'identifiant du commit, la date à laquelle il a été effectué, et enfin le nombre total de commits.

[4] Bash est l'acronyme de Bourne Again Shell.

## Laragon




**Laragon** est un environnement de développement web pour Windows qui permet de créer et de gérer un serveur web depuis un ordinateur personnel. Il offre un ensemble d'outils tels que des serveurs **Apache** (serveur web open-source), Nginx, MySQL, PHP. Cela permet de créer et de tester des applications web localement.

Il dispose également d'un menu SQL qui permet de gérer des composants de base de données MySQL. Il donne accès aux outils de gestion des données tels que HeidiSQL ou PhpMyAdmin.

Installation : l'installation de Laragon s'éffectue depuis leur site internet, en exécuter leur fichier exe.

## HeidiSQL

HeidiSQL est un outil de gestion de base de données open-source qui permet aux utilisateurs de se connecter et d'interagir avec différentes bases de données. Il permet de travailler sur des bases de données sans avoir à utiliser des commandes SQL en ligne de commande.

## Bundles

Un bundle[5] est un morceau d'application dans l'univers Symfony. En fait, cela correspond à une brique logicielle, c'est-à-dire un ensemble cohérent pour une fonctionnalité donnée. Dans ce projet par exemple, des bundles supplémentaires ont été installés tels que *MakerBundle*, *VerifyEmailBundle*, *CalendarBundle* ou *DomPdf*.

Installation :

- Bundle de vérification d'email : *VerifyEmailBundle*
  Ce bundle permet de vérifier l'adresse email d'un utilisateur à son inscription. Pour l'installer, il faut effectuer la ligne de commande suivante : `composer require symfonycasts/verify-email-bundle`
- Génération de PDF : *DomPdf*
  Ce bundle permet de convertir une vue HTML en PDF. Pour l'installer, il faudra exécuter la ligne de commande : `composer require dompdf/dompdf`
- Maker Bundle : MakerBundle

[5] Traduction française : *paquet*

*MakerBundle* aide à la création de contrôleurs, de formulaires ou de classes. Pour l'installer, il faut effectuer la ligne de commande : `composer require --dev symfony/maker-bundle`

Dans ce projet, j'ai créé la majorité des entités, contrôleurs et formulaires grâce au *MakerBundle*, grâce aux commandes `symfony console make :entity`, `symfony console make :controller` ou `symfony console make :form`.

- CalendarBundle : CalendarBundle permet de générer des calendriers et de gérer des événements à l'intérieur. Pour l'installer, il faut effectuer la ligne de commande : `composer require tattali/calendar-bundle`.

## Symfony (Scoop, composer, Symfony CLI)

### Composer


Composer est un **logiciel gestionnaire de dépendances libre écrit en PHP**. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.

Installation : sur le site internet de Composer, il est nécessaire de télécharger le fichier *.exe* qui se trouve en première page de l'onglet *Download*.

### Scoop

Scoop est un installateur en ligne de commande open source pour Windows. Il est utilisé pour installer divers outils et applications en ligne de commande sur Windows.

- `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`
- `> Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression`

### Symfony CLI

Symfony CLI est un ensemble d'outils en ligne de commande facilitant le développement avec le framework Symfony.

Installation : Il faut télécharger le fichier *.exe* depuis le site internet et utiliser la commande suivante avec Scoop pour installer Symfony CLI:

- `scoop install symfony`

### Symfony

Pour ce projet, j'ai utilisé le framework Symfony. Symfony est un framework (cadre de travail **ou cadriciel en français**) open-source PHP, basé sur le *design pattern* [6]MVP (Modèle – Vue – Presentateur), ressemblant à la structure MVC - Modèle Vue Contrôleur (« MVC »). Le design pattern MVP sera développé dans la partie *Architecture* de ce dossier.

### Doctrine

Le framework Symfony utilise par défaut **l'ORM Doctrine** qui permet d'interagir avec la base de données plus facilement. Nous traiterons ce sujet dans la partie Doctrine.

### ORM

L'acronyme ORM provient du terme anglais ***Object Relational Mapping***, qui est une approche de la programmation qui permet de convertir les données entre des types de systèmes incompatibles.

## Librairies externes

**Bootstrap** : Bootstrap est un framework CSS. Un framework correspond à un ensemble de librairies regroupées dans un but précis et possédant des règles internes que doivent suivre les utilisateurs. J'ai utilisé Bootstrap dans ce projet, afin de mettre en forme les formulaires ainsi que les tableaux d'affichage des réservations.

# Règlement Général de Protection des Données

## Définition

Le Règlement Général sur la Protection des Données (RGPD) est une loi européenne qui vise à protéger la vie privée et les droits des individus en ce qui concerne le traitement de leurs données personnelles.

## Principes mis en œuvre dans ce projet

Certains principes ont guidé ma démarche dans le développement de cette application :

- **Transparence**

  Le traitement des données personnelles doit être licite, équitable et transparent pour la personne concernée. Lors de l'inscription, les utilisateurs ont une case à cocher, indiquant qu'ils

[6] Patron de conception

sont conscients du traitement des données que l'on compte effectuer. Aussi, ils peuvent retrouver ces informations dans les liens du *footer* accessibles n'importe où dans l'application.

- **Consentement**

  La confirmation de l'acceptation de l'utilisateur est demandée à l'inscription, via une case à cocher, afin de s'assurer qu'il est conscient de l'utilisation qui sera faite de ses données.

- **Finalité limitée**

  Les données personnelles ne doivent être collectées que pour des finalités spécifiques, explicites et légitimes.

- **Minimisation des données**

  On ne doit collecter que les informations nécessaires à la réalisation de l'objectif prévu. Par exemple, dans ce projet, je ne demande que les informations absolument nécessaires à la finalité prévue, c'est-à-dire la réservation.

- **Exactitude**

  Les données personnelles doivent être exactes et tenues à jour. L'utilisateur peut mettre à jour ou supprimer les données le concernant.

- **Intégrité et confidentialité**

  Les données personnelles doivent être traitées de manière à garantir la sécurité appropriée, y compris la protection contre le traitement non autorisé ou illégal et contre la perte, la destruction ou les dommages accidentels. Dans ce projet par exemple, des mesures de sécurité ont été prises pour protéger les données de l'utilisateur.

- **Responsabilité**

  Le responsable du traitement (l'entité qui collecte et traite les données) est responsable de la conformité avec ces principes et doit être en mesure de démontrer cette conformité.

# Search Engine Optimization

## Définition

Le SEO, ou *Search Engine Optimization* (Optimisation pour les moteurs de recherche), est un ensemble de techniques et de bonnes pratiques visant à améliorer la visibilité d'un site web dans les résultats des moteurs de recherche.

## Principes mis en œuvre dans ce projet

Tout comme le RGPD évoqué précédemment, ses principes ont permis d'aiguiller le développement de cette application.

- Contenu de qualité
- Performances
- Mots-clés et importants
- Expérience utilisateur : Responsive web design
- Netlinking
- Réseaux sociaux
- Sécurité du site

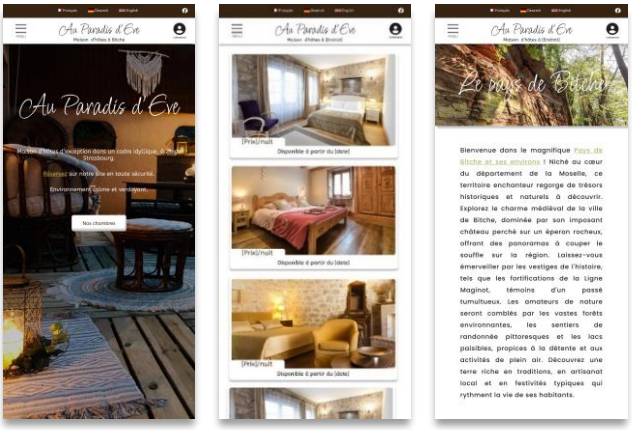
# Maquettage

## Maquettes version mobile et version desktop

La maquette, élaborée pour les versions mobile et desktop en utilisant l'outil **Figma**, incarne la fusion entre le cahier des charges et la charte graphique.

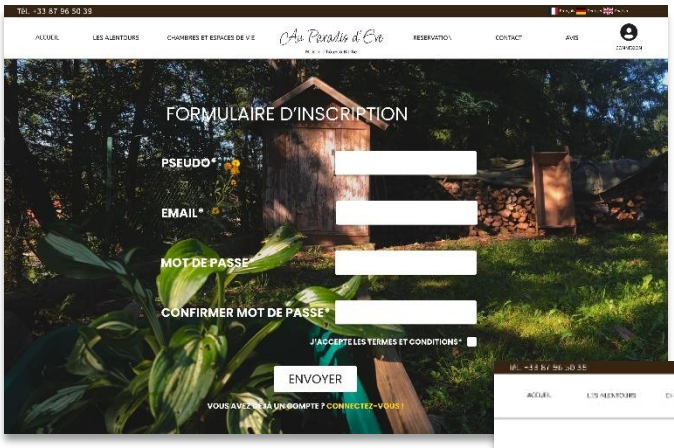
### Version mobile : Mobile first design

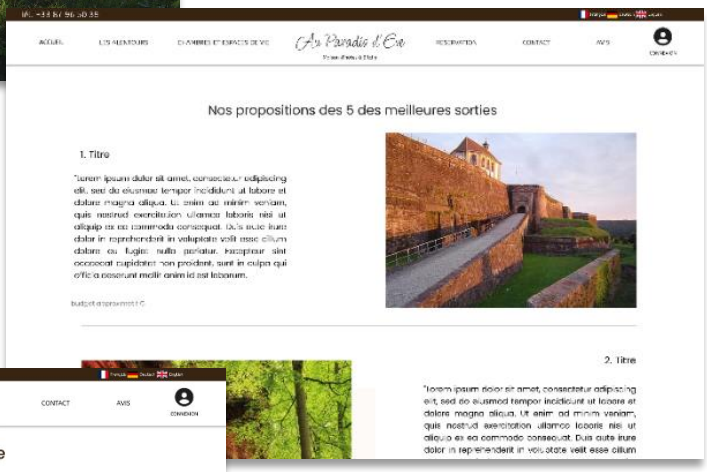
*Maquette de la page d'accueil avec formulaire de contact*

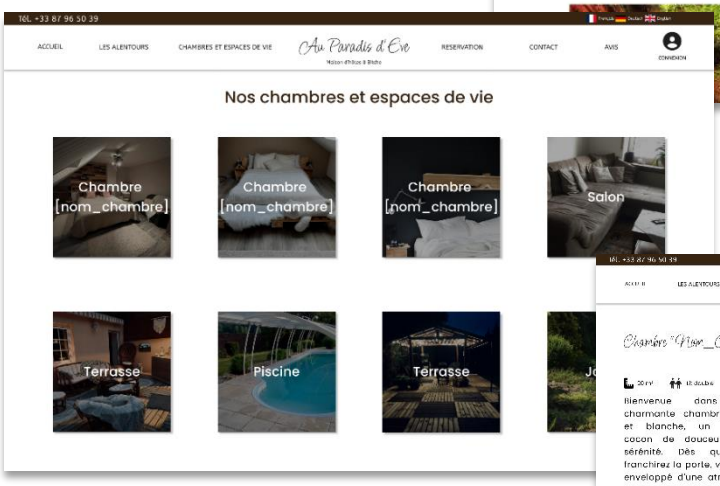


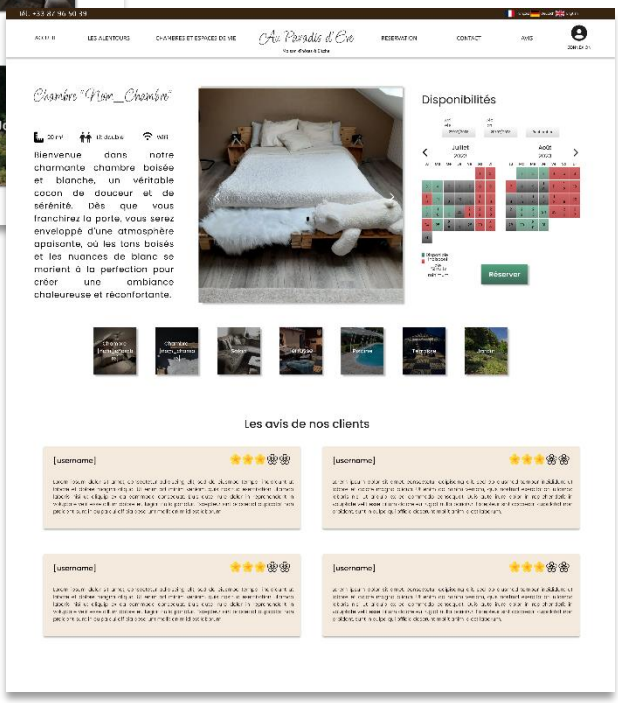
## Version bureau

*Formulaire d'inscription, Page des alentours, Page d'accès à la présentation de chaque espace et Page de détails d'un espace*



# UX/UI

### UX *User Expérience* : l'Expérience Utilisateur

Elle désigne la facilité avec laquelle un individu va utiliser le qu'on lui met à disposition, dans notre cas, il s'agit d'un site internet. L'UX dépend de nombreux facteurs ergonomiques et techniques (webdesign, responsive, etc.). Elle revêt aussi une dimension émotionnelle puisqu'elle implique l'ensemble des ressentis de l'internaute lorsqu'il navigue sur les pages du site web.

### UI *User Interface* : l'Interface Utilisateur

L'interface utilisateur se rapporte donc à l'environnement graphique dans lequel évolue l'utilisateur d'un produit, d'un site web ou d'une application par exemple. La mission de l'UI designer consiste à créer une interface agréable et pratique, facile à prendre en main. Ainsi, l'UI design fait partie de l'UX design, en cela qu'il travaille à donner la meilleure expérience possible à l'utilisateur, mais il s'attache plus particulièrement aux éléments perceptibles : éléments graphiques, boutons, navigation, typographie…

### Benchmark

Pour avoir une idée approximative des éléments nécessaires à faire apparaître sur les différentes pages, je me suis inspirée d'un site web de réservation déjà existant, à savoir de celui du Manoir de La Croix Saint Louis, mais également de sites très fréquentés comme Airbnb ou Booking.

# Charte graphique

L'aspect visuel du site a été conçu représenter l'identité de l'établissement : **chaleureuse**, **naturelle**, accueillante.

### LES COULEURS

La palette de couleurs a été créée dans le but de refléter l'essence naturelle du Pays de Bitche, en particulier à travers le choix de tons marrons, verts et beige en harmonie avec l'atmosphère de la maison et des chambres. L'objectif est de proposer une interface incitant les visiteurs à explorer le site, en facilitant un accès intuitif et logique aux différentes fonctionnalités.

Le site a été imaginé dans une volonté de rendre l'expérience utilisateur agréable et fluide, en respectant le design et les coloris désirés.

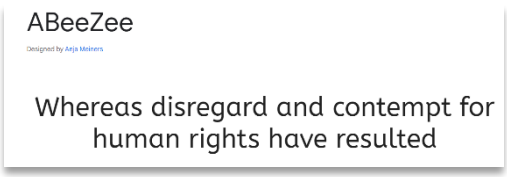
```css
:root {
    --dark-brown: #372412;
    --font-color: #1E1E1E;
    --box-shadow: rgb(78, 77, 77);
    --white: rgb(255, 255, 255);
    --hover: #FFA800;
    --clear-card: rgba(195, 105, 20, 0.106);
    --dark-green: rgb(30, 105, 30);
}
```

## POLICE D'ECRITURE

Les polices d'écriture quant à elles ont été sélectionnées pour établir une proximité plus affirmée avec l'utilisateur. Importées depuis Google Fonts dans mon fichier « base.css », elles ont pu être utilisées dans l'ensemble du projet afin de conserver une cohérence sur les différentes pages accessibles à l'utilisateur.

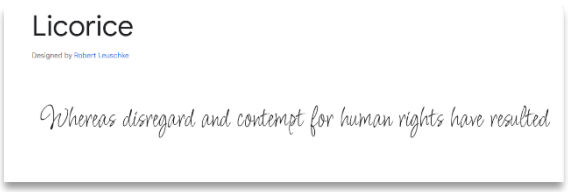
public > css > # base.css > :root

```css
@import url('https://fonts.googleapis.com/css2?family=Licorice&display=swap');
@import url('https://fonts.googleapis.com/css2?family=ABeeZee&family=Licorice&display=swap');
```




Pour la barre de navigation, les paragraphes et certains titres en majuscule, j'ai utilisé la police d'écriture AbeeZee , de Anja Meiners.

Pour les titres importants, j'ai utilisé la police d'écriture Licorice, de Robert Leuschke. Il s'agit d'une typographie manuscrite qui permet de rappeler l'objectif de proposer une expérience humaine et accueillante de façon moins formelle.




## LE LOGO

Cette dernière m'a par ailleurs permis d'apporter un maximum de cohérence en l'utilisant comme le logo de l'établissement.

Au Paradis d'Eve

Celui-ci est visible sur la page d'accueil, mais l'est également toujours sur la barre de navigation, et permet, au clic, un retour à la page d'accueil.

### IMAGES ET MEDIAS

Les images et médias proviennent directement de l'établissement pour les espaces et les chambres, et de sources diverses libres de droits sur internet pour les alentours. Elles ont été sélectionnées dans l'objectif d'illustrer les différentes sections et apporter plus de détails et de visibilité sur le séjour proposé et soutenir la volonté de proposer une navigation agréable pour l'utilisateur.

### FONDS ET TEXTURES

J'ai intégré des fonds images dans certaines sections pour offrir une expérience visuelle immersive et personnelle à mes visiteurs. Ces images sont sélectionnées pour refléter l'identité de l'établissement, en conservant un contraste élevé pour conserver une lisibilité claire du texte. La page d'accueil par exemple, est constituée d'une image de fond avec un effet parallax pour inciter les utilisateurs à découvrir la suite des sections. Cette approche vise à instaurer une atmosphère visuelle distinctive et à améliorer l'expérience utilisateur.

Dans d'autres sections, j'ai opté pour des fonds blancs afin de garantir une clarté visuelle, mettant en avant le contenu de manière nette et lisible. Cette approche assure une lisibilité optimale des textes tout en créant une sensation d'espace et de légèreté. Je privilégie les fonds blancs pour mettre en valeur des informations ou des fonctionnalités importantes. Cette simplicité visuelle favorise une navigation intuitive et agréable.

J'ai utilisé des ombrages sur certains éléments et à la fin de certaines sections pour créer une profondeur visuelle et accentuer l'aspect chaleureux et lumineux du site. Ces ombrages subtils donnent l'impression que les éléments sont légèrement surélevés et offrent une sensation de profondeur. L'objectif étant de renforcer la convivialité, de guider le regard des utilisateurs vers des zones spécifiques.

### DISPOSITION ET MISE EN PAGE

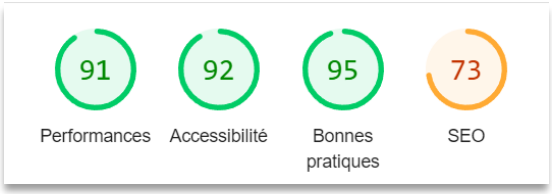
Ce projet a été réalisé dans l'objectif de permettre à l'utilisateur de ne pas se sentir « perdu », de toujours savoir où il se trouve et comment il peut accéder aux différentes fonctions. Ainsi, la structure des pages garde la même logique dans l'ensemble du projet. Le titre le plus grand et le plus important se trouve en haut de page à gauche, suivi d'un sous-titre s'il y en a, sinon du contenu cliquable. Les contenus cliquables sont annotés dessus ou en dessous afin que l'utilisateur puisse savoir à quel contenu il s'apprête à accéder. Les différents contenus peuvent être accompagnés d'illustrations afin d'offrir un

aperçu concret des éléments cités dans les textes. L'ensemble est volontairement espacé afin d'aérer le contenu et de faciliter la prise d'information.

## Accessibilité

La charte graphique a d'ailleurs participé au respect des règles d'accessibilité, auxquelles j'ai accordé une attention particulière dans la réalisation de ce projet.

### Contraste




Les choix de couleurs et de contrastes ont été soigneusement considérés pour garantir une lisibilité optimale. Les combinaisons de couleurs utilisées respectent les normes d'accessibilité, assurant une expérience visuelle claire et facile à comprendre pour tous les utilisateurs.

### Textes de remplacement sur les images

Chaque image du site est accompagnée de textes alternatifs (attributs "alt") significatifs. Ces descriptions textuelles permettent aux utilisateurs malvoyants ou utilisant des lecteurs d'écran de comprendre le contenu des images, renforçant ainsi l'accessibilité de notre site.
Lors de l'ajout d'un espace par exemple, l'administrateur doit remplir un champ de type texte *Texte de remplacement* qui lui permet de décrire les images qu'il souhaite affiche.

### Navigation intuitive

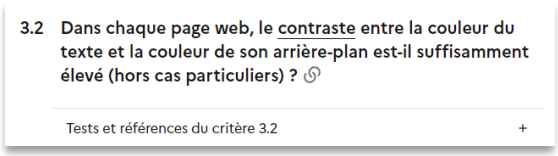
La navigation sur le site a été conçue de manière à être logique et intuitive. Les utilisateurs peuvent parcourir facilement le contenu, accéder aux différentes sections et comprendre la structure du site, contribuant ainsi à une expérience utilisateur accessible.
Un aspect crucial de l'UX est **la règle des 3 clics**, qui suggère que les utilisateurs devraient pouvoir accéder à n'importe quelle information ou fonctionnalité du site en trois clics ou moins.
Cette règle repose sur l'idée que plus le nombre de clics nécessaire pour atteindre une destination spécifique est faible, plus l'expérience utilisateur est fluide.

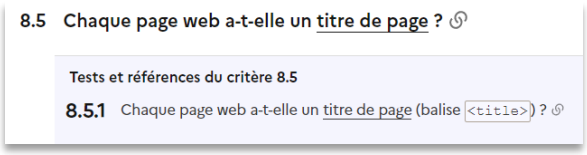
### Référentiel général d'amélioration de l'accessibilité (RGAA)

Le Référentiel général d'amélioration de l'accessibilité découle de l'obligation d'accessibilité imposée par l'article 47 de la loi du 11 février 2005 pour « l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées » et permet de définir, en France,

les modalités techniques d'accessibilité des services en ligne de l'État, des collectivités territoriales et des établissements publics.

Il contient les obligations légales à respecter dans la création de contenu web.




## Breakpoints et mediaqueries

**Breakpoints** : Les valeurs des points de rupture, en français, sont des points spécifiques dans la feuille de style (CSS) **où le design de la page web est modifié en fonction de la taille de l'écran** du dispositif utilisé pour visualiser la page. Les breakpoints CSS permettent d'adapter dynamiquement le contenu en fonction de la taille de l'écran, améliorant ainsi l'expérience utilisateur sur différents appareils tels que les ordinateurs de bureau, les tablettes et les smartphones.

**Media Queries** : Les media queries, requêtes média en français, sont une fonctionnalité du langage de style CSS qui **permet de définir des règles de style conditionnelles basées sur les caractéristiques du média ou de l'appareil utilisé pour afficher la page**.

Ce projet a été réalisé en **Mobile First Design**, qui consiste à aller plus loin que simplement adapter le contenu desktop a une version mobile. Le site a été pensé pour les écrans de taille mobile. Dans les faits, cela implique que les règles appliquées sont, de base, applicables pour les tailles d'écrans de téléphone, et que les breakpoints retenus sont de plus en plus grands.

Mobile-First Design

**75 % du trafic en France se fait désormais sur un smartphone.**

https://www.radiofrance.fr/franceinter/internet-75-du-trafic-en-france-se-fait-desormais-sur-un-smartphone-3249187

Exemple de media queries dans ce projet :

```
/* Breakpoint 1 : Smartphone */
@media (min-width: 360px) and (max-width: 767px) {
```

```
/* Breakpoint 2 : Tablette Portrait - PC avec naviga
@media (min-width: 768px) and (max-width: 999px) {
```

```
/* Breakpoint 3 : Tablette Paysage - PC Laptop - Tout
@media (min-width: 1000px) and (max-width: 1279px) {
```

# Modélisation

## Méthode Merise

La méthode Merise est une **approche utilisée pour concevoir des systèmes informatiques**. Elle se divise généralement en quatre grandes étapes :

- **Comprendre les besoins** : On commence par comprendre ce que les utilisateurs veulent que le système fasse
- **Définir les données** : On identifie les types d'informations nécessaires et comment elles sont liées les unes aux autres
- **Décrire les actions** : On spécifie les différentes actions que le système doit effectuer sur ces données.
- **Concevoir le système** : On crée des modèles qui représentent graphiquement la structure des données et la façon dont le système va les traiter.

## Modèle Conceptuel de Données

Le MCD (modèle conceptuel de données) est un schéma conceptuel permettant de modéliser la problématique à traiter lors d'un système d'information. Il s'agit d'une représentation claire des données du système d'information à concevoir. Cette représentation permet également de concevoir les relations entre ces données.

Le modèle conceptuel de données élaboré pour ce projet s'aligne étroitement sur la structure et les éléments de ma maquette Figma ainsi que sur les principes du Règlement Général de Protection des Données évoqué précédemment, créant une représentation visuelle des données nécessaires.

Voir annexe page 58.

### Les entités et cardinalités

Les cardinalités sont des couples de valeur que l'on trouve entre chaque entité et ses associations liées. Donc, pour une association de 2 entités, il y a 4 cardinalités à indiquer (2 de chaque côté). Il y a trois valeurs typiques : 0, 1 et N (plusieurs). Pour les associations à 2 entités, ce sont des valeurs qui permettent d'indiquer combien de fois au minimum et au maximum une occurrence d'entité peut être liée à une autre occurrence d'entité.

J'ai identifié et créé plusieurs entités qui correspondent aux composants clés de la maquette Figma. Chaque entité représente une partie significative de l'interface utilisateur et est étroitement liée aux éléments visuels et fonctionnels que les utilisateurs utiliseront.

Dans ce projet de réservation de chambres d'hôtes, j'ai créé six entités :

- **Espace** : qui possède une ou plusieurs images, qui est relié à une catégorie et qui est lié ou non à plusieurs réservations, est donc conceptualisé en cardinalité :
  - **OneToMany** à l'entité Image
  - **ManyToOne** à l'entité Catégorie
  - **OneToMany** à l'entité Réservation
- **Image** : reliée à l'entité espace, qui peut avoir une ou plusieurs images
- **Catégorie** : reliée à un ou plusieurs espaces, elle implique qu'un espace ne peut avoir qu'une catégorie
  - **OneToMany**
- **Réservation** : entité clé de ce projet, elle contient toutes les informations d'un utilisateur nécessaire à la fonction de réservation d'un espace, elle est reliée en cardinalité :
  - **ManyToOne** à l'entité Espace
- **Option** : l'entité Réservation peut contenir, ou pas, une collection d'options si celles-ci sont sélectionnées lors de la réservation
  - **ManyToMany** à l'entité Réservation
- **User** : l'entité User concerne directement l'utilisateur et est relié ou non à une réservation puisqu'il est possible de réserver sans créer de compte
  - **OneToMany** à l'entité Réservation

### Attributs

Les attributs de chaque entité ont été définis en fonction des informations nécessaires pour rendre fidèlement la maquette dans une structure de base de données. Ces attributs capturent les détails spécifiques correspondant aux composants visuels de la maquette. Chaque entité dispose d'un identifiant.

- **Espace** : dispose d'un nom, d'une taille, d'une description
- **Image** : possède un attribut lien et un attribut alt pour les utiliser dans les balises *<img>*
- **Catégorie** : dispose uniquement d'un nom en plus de son identifiant
- **Réservation** : l'entité *Reservation* comporte en attributs toutes les informations nécessaires à l'enregistrement d'une réservation, tels qu'un prénom, nom, les dates de début et de fin de séjour, le prix, un email, une adresse de facturation, un lien vers la facture en PDF, ainsi qu'une note et un avis, si l'auteur de la réservation souhaite en laisser après la fin de séjour
- **Option** : contient un attribut nom, description et tarif en plus de son identifiant, et permet d'être afficher avec l'entité *Reservation* à laquelle elle est reliée
- **User** : détient toutes les informations nécessaires à la création d'un compte, à savoir un rôle définit à utilisateur *[ROLE_USER]* par défaut, un pseudo, un email, et un mot de passe. Le reste des attributs n'est pas obligatoire, mais peut être enregistré si l'utilisateur le souhaite pour des réservations futures. Il peut en faire le choix lors d'une réservation en cochant une case.

## Modèle Logique de Données

Un **Modèle Logique de Données** (MLD) est un plan qui montre comment différents objets représentés comme des tables, qui sont liés les uns aux autres. On y inclut aussi les tables associatives (ou table de liaison), qui aident à connecter ces objets. Chaque rangée dans ces tables est comme une instance particulière de ces objets. En résumé, le MLD décrit la façon dont les objets, les tables, les tables associatives et les instances sont organisés et interconnectés dans un système informatique.

Réalisé avec l'outil Looping[7], j'ai réalisé le modèle conceptuel de données de cette application en prenant en compte que la cardinalité *ManyToMany* créerait une table associative, qui se traduit en Symfony par une entité à part entière.

Voir annexe page 59.

# Architecture de l'application

## Design pattern

### Design pattern :

**Patron de conception** : arrangement caractéristique de modules, reconnu comme bonne pratique pour résoudre un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.

#### En quoi est-ce une bonne pratique d'utiliser un design pattern ?

- **Réutilisabilité du Code** : code qui a déjà été testé et validé dans d'autres contextes réutilisables
- **Standardisation** : approche standardisée de la résolution de problèmes de conception
- **Facilitation** du Développement Évolutif : permet d'ajouter de nouvelles fonctionnalités sans avoir à réécrire une grande partie du code

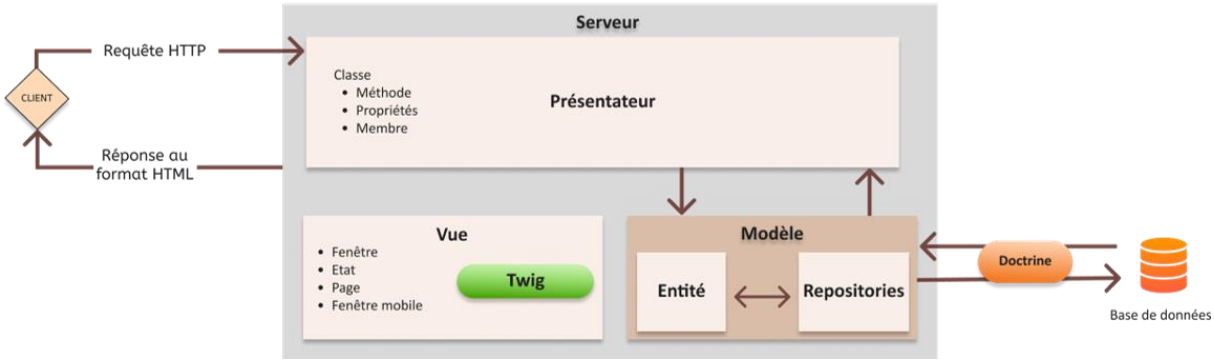
### Design pattern MVP

[7] Logiciel de modélisation conceptuelle de données qui permet d'organiser et de structurer des données

**Modèle** : La couche modèle gère la logique métier et les interactions avec la base de données. Symfony utilise Doctrine, un outil ORM pour simplifier la manipulation des données en les représentant sous forme d'objets PHP.

**Vue** : La vue est quant à elle responsable de l'affichage des données. Symfony permet de générer des vues grâce au moteur de template Twig, qui facilite l'intégration des données.

**Présentateur** : Le présentateur -ou Manager- gère la logique de traitement des requêtes HTTP en interagissant avec le modèle pour récupérer des données et en les envoyant à la vue pour affichage.




La principale différence entre la structure Modèle-Vue-Controller et Modèle-Vue-Présentateur réside dans la manière dont la communication s'effectue entre la vue et le modèle. **Dans Modèle-Vue-Présentateur, la vue est plus passive et n'a pas de référence directe au modèle**, tandis que le présentateur agit comme un intermédiaire.

Dans ce projet par exemple, il a été nécessaire de créer des **requêtes DQL** *Doctrine Query Language* (Langage de requête de données) afin de récupérer des champs précis de la base de données dont nous verrons l'exemple plus tard dans ce dossier.

## Requête HTTP

Une **requête HTTP** *Hypertext Transfer Protocol* est une demande envoyée par un client à un serveur pour obtenir des informations ou effectuer une action sur le serveur.

Une requête http est généralement composée :

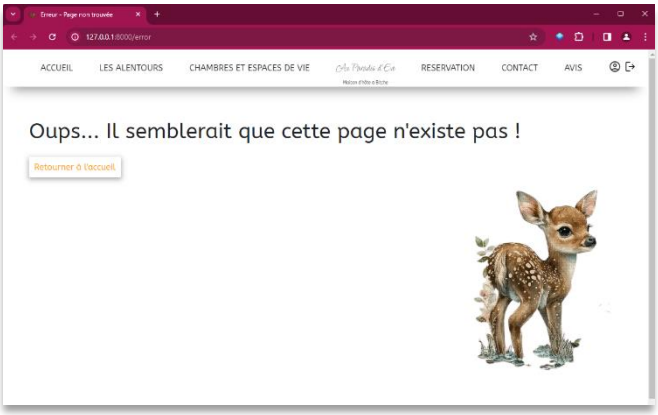
- D'une **méthode** qui indique quel type d'action le client demande à effectuer
  Exemple : GET pour récupérer des données, POST pour soumettre des données comme dans un formulaire par exemple.
- D'une **URL** : Une URL (en Anglais *Uniform Resource Locator*) est l'adresse d'un site web.
- D'un **en-tête** avec :

  - Un Host : Indique le nom de domaine du serveur.
  - User-Agent : Informations sur le navigateur ou l'agent utilisateur.
  - Accept : Les types de médias que le client peut traiter.
  - Content-Type : Le type de média du corps de la requête (pour les requêtes POST, PUT, etc.).
  - Authorization : Informations d'identification pour accéder à une ressource protégée.
- D'un **corps de requête** avec :
  - Optionnel pour les requêtes GET, HEAD, DELETE, etc.
  - Contient les données à envoyer au serveur, généralement pour les requêtes POST, PUT, PATCH, etc.
  - Le format du corps dépend du type de média spécifié dans l'en-tête Content-Type.

## Codes de réponse HTTP

Les codes de statut de réponse HTTP indiquent si une requête HTTP a été exécutée avec succès ou non. Les réponses sont regroupées en cinq classes, voici quelques exemples pour chacune d'entre elle :

- **Réponses informatives**
  - **100 : Continue**
  - 101 : Switching Protocols
  - 102 : Processing
  - 103 : Early Hints
- **Réponses de succès**
  - **200 : OK**
  - 201 : Created
  - 202 : Accepted
  - 203 : Non-Authoritative Information
  - 204 : No Content
- **Messages de redirection**
  - **300 : Multiple Choices**
  - 301 : Moved Permanently
  - 302 : Found
  - 303 : See Other
- **Réponses d'erreur côté client**
  - **400 : Bad Request**
  - 401 : Unauthorized
  - 402 : Payment Required
  - 403 : Forbidden
  - 404 : Not Found




*Redirection en cas d'erreur côté client*

- 405 : Method Not Allowed

- **Réponses d'erreur côté serveur**
  - **500 : Internal Server Error**
  - 501 : Not Implemented
  - 502 : Bad Gateway

## Couches de l'application

Voici une explication des différentes couches et des étapes traversées lorsqu'une requête HTTP est effectuée :

- **Serveur Web** : La requête HTTP est d'abord traitée par le serveur web (par exemple, Apache, Nginx), qui dirige la requête vers l'application Symfony.
- **Front Controller** (Contrôleur frontal) : L'index.php dans le dossier public de l'application est le front controller. C'est le point d'entrée unique de toutes les requêtes HTTP. Le front controller charge le Kernel (le noyau) de Symfony, qui initialise l'application.
- **Kernel Symfony** : Le kernel est responsable de la gestion de l'ensemble du cycle de vie de la requête. Il initialise l'application, charge les *bundles* (modules), et gère l'injection de dépendances. Il instancie le gestionnaire de requêtes (*RequestHandler*), qui gère le traitement de la requête.
- **Event Dispatcher** (Gestionnaire d'événements) : Symfony utilise un système d'événements. Lorsqu'une requête est reçue, des événements sont déclenchés. Les écouteurs d'événements (*event listeners*) peuvent être attachés pour réagir à ces événements.
- **Routeur** : Le routeur prend en charge la résolution de l'URI de la requête en un contrôleur et une action spécifique. Il utilise les définitions de routes définies dans le fichier *routes.yaml*. Si la route est trouvée, le routeur renvoie le contrôleur et l'action correspondants.
- **Contrôleur** : Le contrôleur est responsable du traitement de la requête. Il effectue les actions nécessaires, comme récupérer des données à partir du modèle (si nécessaire) et retourner une réponse. Les contrôleurs sont des classes PHP qui étendent généralement la classe AbstractController de Symfony.
- **Modèle** (*Model*) : Le modèle représente la logique métier de l'application. Il peut s'agir d'accéder à une base de données, de récupérer des données, de les traiter, et de les retourner au contrôleur.
- **Vue** (*View*) : La vue est responsable de l'affichage des données au client. Dans Symfony, les vues sont des fichiers Twig rendus par le contrôleur.
- Le contrôleur retourne une réponse, qui peut inclure une vue rendue.

```php
return $this->render('reservation/new.html.twig', [
    'form' => $form,
    'chambre' => $chambre
]);
```

- **Réponse** (*Response*) : La réponse générée par le contrôleur est renvoyée au client. Elle peut inclure des en-têtes HTTP, du contenu HTML, des redirections, des codes d'état, etc.

### Programmation Orientée Objet (POO)

La programmation orientée objet est une façon d'approcher la programmation informatique permettant d'optimiser et de réutiliser du code, en faisant interagir des objets entre eux, grâce aux principes **d'encapsulation**, **d'héritage** et de **polymorphisme**.

- **Héritage** : relation hiérarchique entre les classes, permet de réutiliser les attributs et méthodes
- **Polymorphisme** : concerne les méthodes des objets : peuvent avoir des comportements différents
- **Encapsulation** : regroupe des données en empêchant leur accè s autrement que par les services proposés

  - **Objet** : regroupe à la fois des données, des attributs et des méthodes
  - **Classe** : un « moule » pour créer des objets sur le même modèle

### Doctrine


Doctrine est l'**ORM** (*Object Relation Mapper*) de Symfony qui permet de gérer manipuler et de récupérer des tables de données de la même façon qu'un objet quelconque, donc en gardant le langage PHP.

# Sécurité

### Les failles de sécurité

#### Authentication

L'authentification est le processus de vérification de l'identité d'un utilisateur pour garantir qu'il est qui il prétend être. Cela est généralement accompli en utilisant des informations d'identification, telles qu'un nom d'utilisateur et un mot de passe.

Le framework Symfony étant utilisé dans ce projet, il permet une gestion simplifiée de l'authentification. Un jeton *token* est généré côté serveur, en session, et côté utilisateur. La connexion n'aboutit que si les deux jetons correspondent lors de la comparaison.

En PHP natif, une façon de générer un jeton est d'utiliser la fonction ***bin2hex*** qui permet de convertir des données binaires en représentation hexadécimale.

### Autorisations

La section access_control permet de définir des règles d'accès spécifiques basées sur le chemin (URL) et les rôles nécessaires. La section role_hierarchy permet de définir une hiérarchie entre les rôles, indiquant comment un rôle peut hériter des permissions d'un autre rôle.

Dans cet exemple :

```yaml
# Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/profile, roles: ROLE_USER }
role_hierarchy:
    ROLE_ADMIN: ROLE_USER
```

```php
#[Route('/admin/espace/new', name: 'new_espace')]
#[Route('/admin/espace/edit/{id}', name: 'edit_espace')]
public function new_edit_espace(Espace $espace = null, I
```

ROLE_ADMIN: ROLE_USER : Cela signifie que le rôle ROLE_ADMIN hérite des permissions du rôle ROLE_USER. Les pages commençant par "/admin" sont accessibles uniquement aux utilisateurs ayant le rôle ROLE_ADMIN. Les pages commençant par "/profile" sont accessibles aux utilisateurs ayant le rôle ROLE_USER. Les utilisateurs ayant le rôle ROLE_ADMIN ont également toutes les permissions associées au rôle ROLE_USER.

### Faille Upload

Le principe de l'attaque est d'uploader un fichier qui contient du code malveillant ou un code PHP de sa création. Si la faille est présente, alors le fichier atterrira sur le serveur. Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute. Afin de contrer cela, j'ai appliqué des contraintes de tailles et de type dans le formulaire Symfony permettant de télécharger des fichiers, en l'occurrence il s'agit du formulaire d'ajout d'un Espace. Ainsi, il n'est pas possible d'ajouter autre chose que des images d'une taille maximale de 5 Mo.

Aussi, une protection supplémentaire est appliquée dans le contrôleur lorsque le formulaire est soumis. En effet, lorsque les données sont récupérées, un appel au service UploadImageService est effectué, et permet entre autre de donner un nom de fichier unique et haché avant de l'enregistrer.

### Protection contre les attaques par injection

- CSRF : Cross Site Request Forgery

Il s'agit d'une attaque qui exploite la confiance qu'un site accorde à un utilisateur authentifié. Elle consiste à forcer un utilisateur à exécuter des actions indésirables sans son consentement lorsqu'il est déjà authentifié sur un site web.

Elle peut entraîner des actions non autorisées de la part d'un utilisateur légitime. Par exemple, si un utilisateur est connecté à un site A et visite un site malveillant B, le site B peut déclencher des actions (comme la modification d'un mot de passe, le changement d'une adresse e-mail, etc.) sur le site A sans que l'utilisateur en soit conscient.

Comment s'en prémunir ?

- Avec Symfony :

  Symfony dispose d'un composant de sécurité intégré qui génère et vérifie automatiquement les jetons CSRF pour les formulaires.

```
new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
```

```
<input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
```

- En PHP natif :

  En gérant manuellement les jetons CSRF : comme évoqué précédemment, il est possible de générer manuellement un jeton CSRF dans les formulaires PHP à l'aide de la fonction bin2hex.

### La faille XSS : Cross Site Scripting

Il s'agit pour un attaquant d'injecter du code malveillant via les paramètres d'entrée côté client. Exemple : du code JavaScript.

Elle permet de détourner la logique d'une application web et permet par exemple **le vol de cookies ou de jetons de session**, l'altération de données ou de contenus, l'exécution d'un malware. Les possibilités d'exploitation d'une faille XSS sont nombreuses, presque infinies avec des conséquences souvent critiques voire irréversibles. Retenez juste qu'un attaquant peut potentiellement réaliser toute action via le navigateur web d'une victime.

Comment s'en prémunir ?

Pour prévenir les attaques XSS, il faut partir du principe que les données reçues par une application web ne peuvent pas être considérées comme « toujours » sûres ! Il est donc important d'implémenter des mesures de protection pour traiter toutes les données venant de l'extérieur. Ainsi, tout contenu doit être filtré, validé et encodé avant d'être utilisé par l'application.

➢ Avec Symfony :

- Auto-échappement des variables dans les templates : Par défaut, Symfony utilise le moteur de templates Twig qui effectue l'échappement automatique des variables pour éviter l'injection de code malveillant.
- Utilisation de l'encodage de caractères par défaut : Symfony utilise UTF-8 comme encodage de caractères par défaut.
- Validation des formulaires et filtrage des données : Symfony propose des composants de formulaire qui effectuent la validation des données côté serveur.

➢ En PHP natif :

- En utilisant par exemple la fonction *htmlspecialchars* qui permet de convertir les caractères spéciaux en entités HTML, ou la fonction *htmlentities* qui effectue une tâche similaire en convertissant un ensemble plus large de caractères en entités HTML
- Par l'utilisation de filtres *filter_var* qui permettent de filtrer une variable avec des paramètres spécifiques. Par exemple en utilisant *FILTER_SANITIZE_SPECIAL_CHARS*, on assainit les données d'entrée en remplaçant ou en supprimant les caractères spéciaux.

### Attaque par force brut

L'attaque par force brute est une méthode d'attaque qui consiste à déchiffrer un mot de **passe en essayant systématiquement toutes les combinaisons possibles**. Cela implique généralement des tentatives répétées et automatisées pour deviner le mot de passe.

Elle peut compromettre la sécurité des comptes en ligne, des systèmes informatiques et d'autres ressources protégées par mot de passe. Si un attaquant réussit, il peut accéder à des données sensibles, compromettre la confidentialité et causer des dommages importants.

Comment s'en protéger ?

➢ Avec Symfony :

Verrouillage du compte : Configurer le composant Security de Symfony pour verrouiller temporairement un compte après un certain nombre de tentatives infructueuses.

```
login_throttling:
    max_attempts: 3
```

Too many failed login attempts, please try again in 1 minute.

➢ En PHP natif :

Délais d'attente entre les tentatives : Introduire des délais d'attente entre les tentatives de connexion pour ralentir le processus d'attaque.

### L'attaque par dictionnaire

L'attaque par dictionnaire est une méthode utilisée pour pénétrer dans un ordinateur ou un serveur protégé par un mot de passe, **en essayant systématiquement tous les mots d'un dictionnaire donné**. Elle fait partie des attaques par force brute.

Si un attaquant réussit à deviner le mot de passe correct, il peut accéder au compte ciblé.

### Comment s'en protéger ?

En suivant les recommandations de la CNIL, qui suggère de mettre en place un mot de passe d'au moins douze caractères, et qui doit contenir un nombre, une majuscule, un signe de ponctuation ou un caractère spécial.

- Avec Symfony :

  Il est possible de mettre des contraintes Constraints, livré avec le Validator Symfony, en lui précisant les recommandations souhaitées. En l'occurrence, j'ai suivi celles de la CNIL afin de permettre aux utilisateurs de se protéger contre cette attaque.

```
'constraints' => [
    new NotBlank([
        'message' => 'Veuillez entrer un mot de passe',
    ]),
    new Length([
        'min' => 12,
        'minMessage' => 'Votre mot de passe doit contenir 12 caractères minimum',
        'max' => 4096,
    ]),
    new Regex([
        'pattern' => '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]+$/',
        'message' => 'Le mot de passe doit contenir au moins une minuscule, une majuscule, un chiffre et un caractère spécial.'
    ]),
],
```

- En PHP natif :

  Il est possible d'utiliser la fonction native de PHP *pregmatch* qui effectue une recherche de correspondance avec une expression rationnelle standard.

## Injection SQL

C'est le fait d'insérer des commandes SQL malveillantes dans une requête SQL via les entrées de données de l'application.

Cela peut permettre à l'attaquant d'interagir de manière non autorisée avec la base de données, de récupérer des données sensibles, voire de supprimer/modifier des données.

### Comment s'en protéger ?

- Avec Symfony :

- Utilisation des requêtes préparées dans le Repository :
  - **Préparation** : La requête SQL est préparée en tant que modèle avec des espaces réservés pour les valeurs que vous souhaitez insérer. Ces espaces réservés sont

généralement représentés par des symboles comme ? ou des noms de paramètres spécifiques au langage de programmation.

- **Paramètres** : Les valeurs réelles à insérer dans la requête sont liées (associées) aux espaces réservés. Cette opération est effectuée de manière sécurisée, en s'assurant que les valeurs sont traitées correctement pour éviter toute interprétation malveillante.
- **Exécution** : La requête préparée, avec les valeurs liées, est ensuite exécutée. Cela permet d'envoyer la requête à la base de données tout en garantissant que les données utilisateur ne peuvent pas être interprétées comme du code SQL malveillant.

```
$entityManager->persist($reservation);
$entityManager->flush();
```

- En précisant les types de données attendues, dans les formulaires notamment. Si la donnée attendue est un est un texte, l'utilisation de contraintes de validation appropriées, telles que Length ou Regex, dans les formulaires Symfony permet de s'assurer que la donnée entrée est conforme aux attentes.

➢ En PHP natif :

- Requêtes préparées avec PDO
- Filtrage et validation des entrées en précisant les types de données attendus. S'il s'agit d'un texte, d'un booléen, le filtre vérifiera que la donnée entrée dans le champ correspond à au type de donnée de la base de données.

### Gestion sécurisée des erreurs

Dans le cadre du développement de notre application Symfony, j'ai mis en place une manière efficace de communiquer avec les utilisateurs. J'utilise la fonction addFlash, méthode fournie par le composant Session de Symfony, pour afficher des messages clairs et informatifs après qu'ils ont effectué une action. Par exemple, un message "Succès" s'affiche après une opération réussie, tandis qu'un message "Erreur" alerte l'utilisateur en cas de problème.

De plus, pour améliorer l'expérience utilisateur en cas de route non trouvée, j'ai créé une page d'erreur personnalisée. Maintenant, si un utilisateur tente d'accéder à une page qui n'existe pas, il est redirigé vers une page d'erreur dédiée avec un message explicatif.

### Empreinte Numérique :

L'empreinte numérique (ou *fingerprint* en anglais) se réfère à une signature numérique associée à une entité particulière, telle qu'un fichier, un document, ou même un périphérique. Dans le contexte de la sécurité, l'empreinte numérique est souvent utilisée pour vérifier l'intégrité des données.

Par exemple, lors du téléchargement d'un fichier, son empreinte numérique peut être générée à l'aide d'un algorithme de hachage, puis cette empreinte est souvent publiée à côté du fichier sur le site web. Les utilisateurs peuvent alors vérifier l'intégrité du fichier en comparant l'empreinte numérique générée localement avec celle publiée.

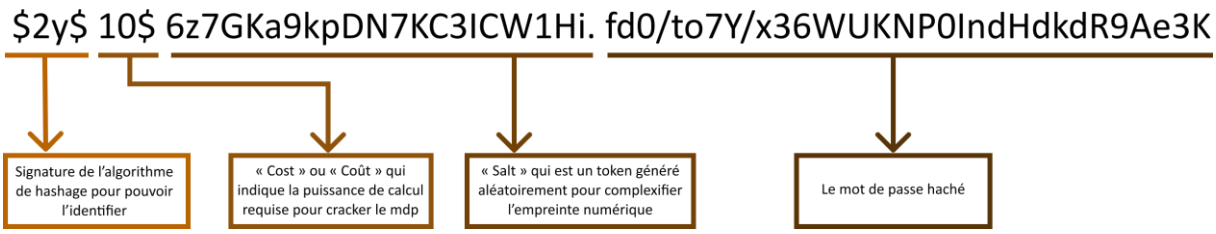
### Hachage :

Le hachage est un processus de transformation de données en une chaîne de caractères alphanumériques, généralement de longueur fixe, qui est représentative de ces données. Les algorithmes de hachage sont largement utilisés en sécurité informatique, par exemple pour stocker des mots de passe de manière sécurisée.

Lorsqu'un mot de passe est stocké dans une base de données, il ne doit pas être stocké en texte brut pour des raisons de sécurité. Au lieu de cela, seul son hachage est stocké. Lorsqu'un utilisateur tente de se connecter, le mot de passe qu'il fournit est haché et comparé au hachage stocké. Cela ajoute une couche de sécurité, car même si la base de données est compromise, les mots de passe réels ne sont pas directement exposés.

Il est essentiel de choisir des algorithmes de hachage robustes et adaptés, comme bcrypt, par exemple. De plus, l'utilisation de techniques comme le sel *salt* peut renforcer la sécurité des mots de passe hachés.

Exemple d'empreinte numérique réalisée avec l'algorithme fort bcrypt :




# Fonctionnalités phares du projet

## Fonctionnalité phare n°1 : La réservation

La fonctionnalité au cœur de ce projet est la réservation. En effet, l'application doit permettre à un utilisateur de visualiser les chambres disponibles et de réserver une chambre au sein de l'établissement.

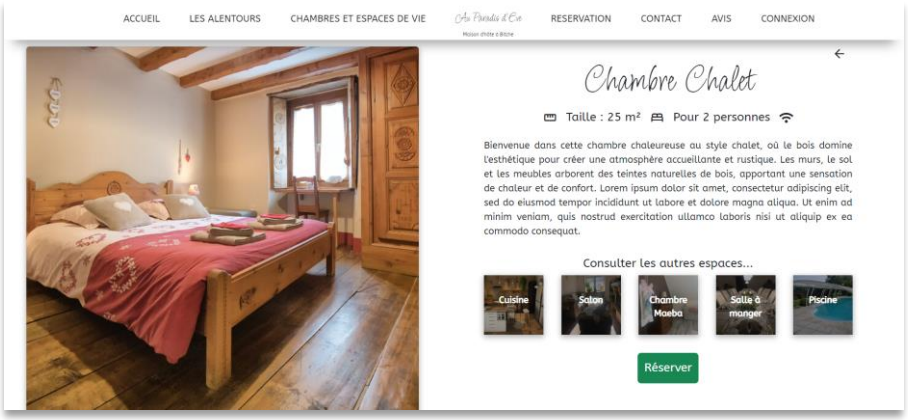
### Liste des espaces disponibles à la réservation : les chambres

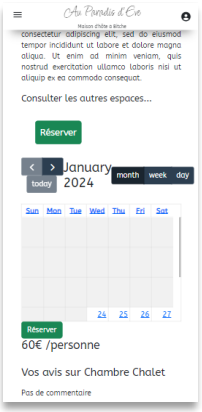
Dans l'onglet réservation de la barre de navigation, l'utilisateur retrouve une liste de chambres uniquement, sous forme de cartes avec une image de celle-ci, son nom suivi de ses informations techniques ainsi que de sa description, les conditions de réservation et sont tarif.

Il est possible de voir la vue de détail de la chambre lorsqu'on clique sur la photo et d'accéder à la fonctionnalité de réservation lorsqu'on clique sur le bouton réserver.


Cette fonctionnalité est également accessible depuis la vue de détail de la chambre.

## La vue de détails de la chambre




La fonctionnalité est accessible à deux reprises dans cette vue, une fois en dessous de la description, et une fois en-dessous du calendrier, où est affiché le prix de l'espace.

## Le bouton « Réserver » : l'accès à la fonctionnalité

```
<a href="{{ path('new_reservation', {'id' :espace.id})}}" type="button" class="btn btn-success">Réserver</a>
```

Le bouton menant à la fonctionnalité de réservation contient l'identifiant de l'espace à réserver, puisqu'il est nécessaire pour qu'on puisse bien réserver l'espace en question. Cet identifiant est d'ailleurs attendu en paramètres de la méthode, dans la route on peut constater qu'il y a un */{id}* attendu, qui correspond à une entité *Espace $espace*, qui est le premier paramètre entre les parenthèses de la fonction.

```
// Ajouter une réservation
#[Route('/reservation/new/{id}', name: 'new_reservation')]
public function newReservation( Espace $espace, Reservation $reservation = null, EntityManagerInterface $entityManager, ReservationRepository
$reservationRepository, Request $request)
```

La première étape lorsqu'on entre dans cette fonction est de vérifier si l'utilisateur qui l'effectue est connecté, auquel cas il accède directement au formulaire de réservation. Si non, il fait face à une page de choix. En effet, il a le choix de poursuivre en tant qu'invité, ou de se connecter. L'objectif est de ne pas contraindre l'utilisateur à devoir se créer un compte et de lui offrir une expérience utilisateur positive.

```php
if(!$this->getUser() && !$request->query->get('acceptInvited')){
    return $this->redirectToRoute('app_choix', ['id' => $espace->getId()]);
}
```

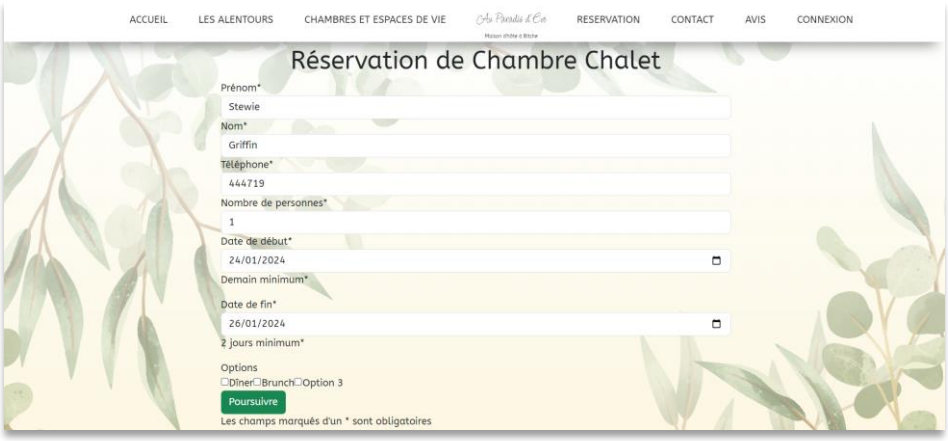

Si l'utilisateur décide de poursuivre en se connectant, il sera redirigé vers la page de connexion, avec un lien d'inscription s'il n'est pas encore inscrit.

Et s'il décide de poursuivre sa réservation en tant qu'invité, il accède alors directement au formulaire de réservation.

## Les formulaires

Le formulaire en question dispose de deux étapes distinctes :

La **première étape** consiste à remplir le prénom, nom, numéro de téléphone, nombre de personnes, ainsi que les dates et les options.




- La **deuxième étape** offre d'abord un résumé de la réservation en cours. Puis elle vérifie les conditions de réservations, ce qui permet d'être sûr qu'aucune autre réservation n'a été effectuée entre temps. Ensuite, les coordonnées de l'utilisateur sont demandées, c'est-à-dire l'email et l'adresse physique de facturation, qui figureront dans le résumé de sa réservation.


A noter que l'utilisateur connecté à la possibilité d'enregistrer ses informations grâce à une case à cocher *checkbox* en fin de formulaire et qui permettra de préremplir les champs lors d'une éventuelle prochaine réservation.

Après avoir terminé le parcours de réservation, celle-ci prend le statut confirmée '*CONFIRMEE'* en base de données et un mail récapitulatif est envoyé au client, avec un lien pour procéder à une éventuelle annulation, ainsi qu'un lien vers les conditions générales d'utilisation.

L'utilisateur connecté peut retrouver ses réservations en cours, à venir et passées sur son profil.

### L'annulation

Si l'utilisateur décide d'annuler sa réservation, il a la possibilité de retrouver cette fonction soit dans le mail de confirmation, soit dans son profil à côté du résumé de sa réservation à venir.

Lorsqu'il fait appel à cette fonction, celle-ci, avant d'annuler la réservation, vérifie, grâce au service de réservation *ReservationService*, la différence entre la date du jour et la date de début du supposé séjour. En fonction de cette différence, le statut de la réservation change de confirmée 'CONFIRMEE ' à :

- *'A REMBOURSER'* : concerne les réservations dont la différence est au minimum de 30 jours. Elles sont dont remboursable à 100%. Ce statut est également applicable si c'est l'administrateur qui annule la réservation, peu importe la date d'annulation.
- *'A REMBOURSER PARTIELLEMENT'* : concerne les réservations dont la différence est de 20 jours, elles méritent un rembourser partiel, de 30%.
- *'ANNULEE'* : concerne les réservations survenues 7 jours ou moins avant la date supposée du séjour et ne sont pas remboursables.

# Fonctionnalité phare n°2 : L'interface administration

L'application offre de nombreuses possibilités d'actions à l'utilisateur disposant du rôle administrateur *ROLE_ADMIN*.

En effet, pour commencer, il a la possibilité de personnaliser les textes de la page d'accueil qui se trouvent dans un fichier json.

Ensuite, dans l'onglet des alentours, il a la possibilité de personnaliser les sorties proposées. Il peut modifier les sorties, leur titre, leur description, le tarif proposé, et insérer un lien externe pour modifier l'image à afficher en lien avec la sortie.

```php
// Edition des textes des alentours
#[Route('/admin/edit/sortie/{id}', name: 'edit_alentours')]
public function editSortie(Request $request, EntityManagerInterface $entityManager, $id): Response
{
    $jsonFilePath = '../public/json/alentours.json';
    $jsonData = file_get_contents($jsonFilePath);
    $data = json_decode($jsonData, true);

    $sortieToEdit = null;
    foreach ($data['sorties'] as $sortie) {
        if ($sortie['id_sortie'] == $id) {
            $sortieToEdit = $sortie;
            break;
        }
    }
    $form = $this->createForm(SortieType::class, $sortieToEdit);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $updatedData = $form->getData();

        foreach ($data['sorties'] as &$sortie) {
            if ($sortie['id_sortie'] == $id) {
                $sortie = $updatedData;
                break;
            }
        }
        $updatedJson = json_encode($data, JSON_PRETTY_PRINT);
        file_put_contents($jsonFilePath, $updatedJson);

        $this->addFlash('success', 'Les informations ont été mises à jour avec succès.');
        return $this->redirectToRoute('app_alentours');
```

```php
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
    ->add('id_sortie', HiddenType::class)
    ->add('nom_sortie', TextType::class, [
        'label' => 'Titre de la sortie',
        'attr' => ['class' => 'form-control'],
    ])
    ->add('description_sortie', TextareaType::class, [
        'label' => 'Description',
        'attr' => ['class' => 'form-control'],
    ])
    ->add('tarif_sortie', NumberType::class, [
        'label' => 'Tarif approximatif',
        'attr' => ['class' => 'form-control'],
    ])
    ->add('image_sortie', UrlType::class, [
        'label' => 'Image',
        'attr' => ['class' => 'form-control'],
        'required' => false
    ])
    ->add('credit', TextType::class, [
        'label' => 'Crédit photo',
        'attr' => ['class' => 'form-control'],
    ])
    ->add('save', SubmitType::class, [
        'label' => 'Enregistrer',
        'attr' => ['class' => 'btn btn-success'],
    ]);
}
```

Dans l'onglet des différents espaces de vie, il peut en ajouter, en supprimer ou en modifier. Il peut modifier, ou ajouter lorsqu'il s'agit d'un ajout, le nom de l'espace, texte, les spécifications techniques, la catégorie d'un espace et les images qui y sont liées avec leur texte de remplacement *alt*.

## Service de téléchargement d'image

Pour cette partie, implémenter un service de téléchargement d'image a été nécessaire afin de permettre à l'administrateur de choisir les images qui catégoriseront l'espace en question, et ainsi améliorer l'expérience utilisateur qui pourra avoir une vue détaillée des différents espaces de l'établissement, en toute sécurité. Ainsi, le service de téléchargement d'image est appelé dans le contrôleur de l'entité Espace, au niveau de la fonction de création ou d'édition d'un espace de vie.

```php
->add('imageFiles', FileType::class, [
    'label' => 'Image(s)',
    'multiple' => true,
    'mapped' => false,
    'required' => false,
    'attr' => ['class' => 'form-control'],
    'help' => 'Formats autorisés : JPEG, PNG, WEBP | Taille maximale : 5 Mo',
    'constraints' => [
        new All([
            'constraints' => [
                new File([
                    'maxSize' => '5M', // Limite la taille à 5 Mo
                    'mimeTypes' => ['image/jpeg', 'image/png', 'image/webp'],
                    'mimeTypesMessage' => 'Veuillez télécharger une image valide (JPEG, PNG, WEBP)',
                ]),
            ],
        ]),
    ],
])
```

```php
->add('altImage', TextareaType::class, [
    'label' => 'Texte de remplacement',
    'mapped' => false,
    'required' => false,
    'attr' => ['class' => 'form-control']
])
->add('valider', SubmitType::class, [
    'attr' => ['class' => 'btn btn-success']
]);
```

On peut notamment remarquer les différentes contraintes appliquées au champ *FileType*.

**Contraintes** : Dans Symfony, les contraintes (*constraints*) sont utilisées pour définir des règles de validation sur les propriétés des objets.

Ce formulaire est traité dans le fichier *AdminController*, en récupérant dans un premier temps les données entrées dans le formulaire, puis en faisant à son tour appel au service *ImageUploadService*.

```php
if ($form->isSubmitted() && $form->isValid()) {

    $espace = $form->getData();
    $images = $form['imageFiles']->getData();
    $altImage = $form['altImage']->getData();

    if($altImage == null){
        $altImage = 'Texte de remplacement indisponible';
    }

    $this->imageUploadService->uploadImages($images, $altImage, $espace);

    $entityManager->persist($espace);
    $entityManager->flush();

    $this->addFlash('success', 'Espace ajouté');

    return $this->redirectToRoute('app_espace');
```

Le service de téléchargement d'image permettra, pour chaque image téléchargée, de faire plusieurs choses:

- D'abord, de donner au fichier téléchargé un **identifiant unique** grâce à la fonction *uniqid()* et de le stocker sur le serveur qu'utilise l'application. Attribuer un ID unique lors de l'upload d'une image permet une identification claire, évite les conflits de noms, facilite la gestion et la sécurité des fichiers.

```php
foreach ($imageFiles as $imageFile) {

    $newFileName = md5(uniqid()) . '.webp'; // Nouvelle extension en WebP
    // Convertir l'image en WebP
    $image = imagecreatefromstring(file_get_contents($imageFile->getPathname()));
    imagewebp($image, $this->parameterBag->get('images_directory') . '/' . $newFileName,
    80); // 80 = qualité
    $image = new Image();
    $image->setLienImage($newFileName);
```

- La fonction *md5* qui se situe juste avant d'attribuer un id unique au fichier téléchargé est une fonction de hachage qui prend une chaîne de caractères en entrée et génère une empreinte (hash) de 32 caractères hexadécimaux. Cette fonction utilise l'algorithme de hachage MD5 (*Message Digest Algorithm 5*), qui produit une valeur de hachage fixe de 128 bits.
- L'utilisation de la fonction *md5* et *uniqid* offre l'avantage d'une longueur de nom de fichier fixe et cohérente et sont rapides d'exécution.
- La fonction *imagecreatefromstring*() est une fonction de la bibliothèque GD de PHP. Elle prend en paramètre une chaîne de données image et crée une ressource image à partir de cette chaîne. *file_get_contents($imageFile->getPathname())* récupère le contenu de l'image à partir du fichier spécifié par *$imageFile->getPathname()*. Cela peut être un fichier JPEG, PNG, GIF… Ensuite, *imagecreatefromstring()* prend ce contenu et le transforme en une ressource image que PHP peut manipuler.

```
    $image->setEspace($espace);
    $image->setAltImage($altImage);
    $espace->addImage($image);
    $this->entityManager->persist($image);
    $this->entityManager->persist($espace);
}
$this->entityManager->flush();
```

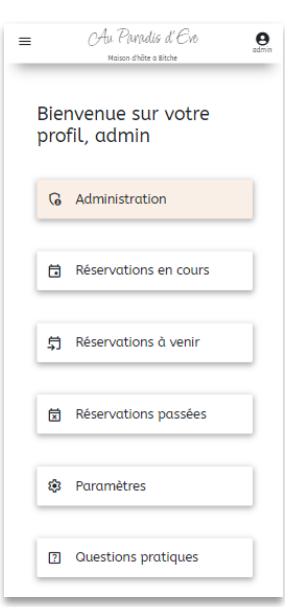
Ensuite, l'espace en question est mis en lien avec les images téléchargées, ainsi que le texte de remplacement, et sont persistés en base de données.

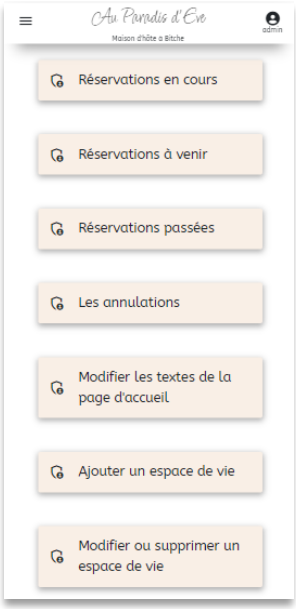
```
<!-- Formulaire avec enctype="multipart/form-data" -->
<form action="upload.php" method="post" enctype="multipart/form-data">
    Sélectionnez un fichier à télécharger :
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Télécharger" name="submit">
</form>
```

```
//upload.php
// Vérifie si la requête est de type POST et si un fichier a été soumis
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_FILES["fileToUpload"])) {
    // Répertoire de destination pour les fichiers téléchargés
    $targetDir = "uploads/";

    // Chemin complet du fichier de destination
    $targetFile = $targetDir . basename($_FILES["fileToUpload"]["name"]);

    // Vérifie si le fichier existe déjà
    if (file_exists($targetFile)) {
        echo "Le fichier existe déjà.";
    } else {
        // Déplace le fichier du répertoire temporaire vers le répertoire de destination
        if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $targetFile)) {
            echo "Le fichier a été téléchargé avec succès.";
        } else {
            echo "Une erreur s'est produite lors du téléchargement du fichier.";
        }
    }
}
```

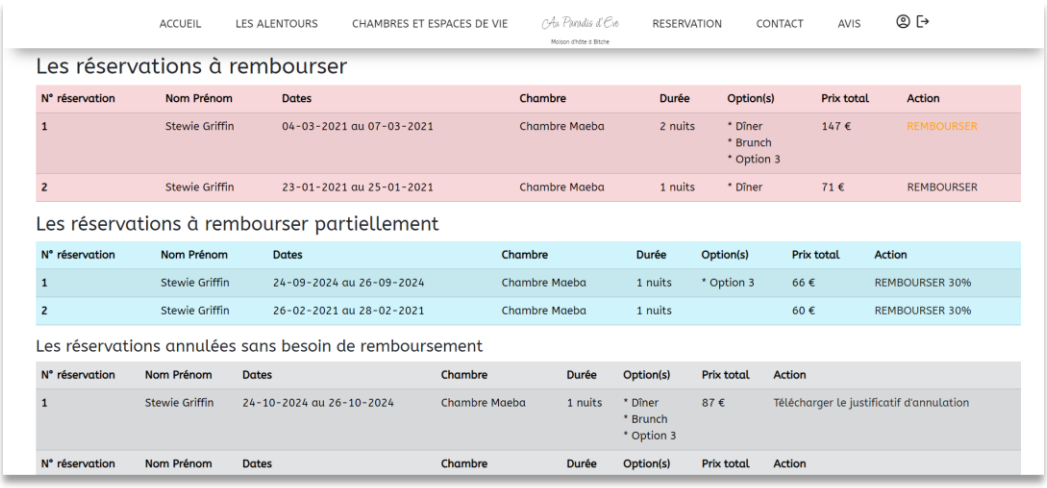
En PHP natif, l'upload de fichiers implique l'utilisation du formulaire HTML avec l'attribut "*enctype="multipart/form-data*" qui indique la manière dont les données du formulaire doivent être encodées lorsqu'elles sont soumises. Ensuite, il est nécessaire de créer un script PHP dans lequel une vérification sera faite pour constater si une requête POST a été soumise et si un fichier a été inclus dans la requête.

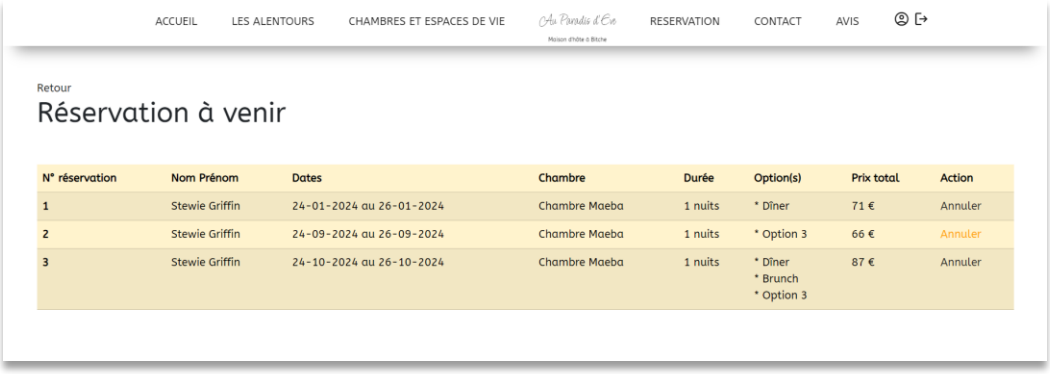
## Les réservations




L'administrateur, dans la partie *Administration*, a une vue d'ensemble de toutes les réservations. Celles qui sont en cours, celles à venir, celles qui sont passées et celles qui sont annulées. Le tableau des annulations lui permet par exemple d'avoir un aperçu des réservations à rembourser entièrement, partiellement, ou pas du tout par exemple.




Un utilisateur connecté quant à lui, a également accès à ses réservations. Pour récupérer les réservations à venir d'un utilisateur en particulier, j'ai eu l'opportunité de créer une requête DQL supplémentaire.

Le langage DQL *Data Query Language* consiste en des instructions permettant de récupérer les données stockées dans des bases de données relationnelles. Les applications logicielles utilisent la commande SELECT pour filtrer et renvoyer des résultats spécifiques depuis un tableau SQL *Structured Query Langage*. *ReservationRepository.php*

```php
74      // Récupère les réservations à venir d'un utilisateur
75      public function findReservationsAVenir(User $user): array
76      {
77          $now = new \DateTime();
78          $confirmee = 'CONFIRMEE';
79
80          return $this->createQueryBuilder('r')
81              ->andWhere('r.user = :user')
82              ->andWhere('r.date_debut > :now')
83              ->andWhere('r.statut = :confirmee' )
84              ->setParameter('now', $now)
85              ->setParameter('user', $user)
86              ->setParameter('confirmee', $confirmee)
87              ->orderBy('r.date_debut', 'ASC')
88              ->getQuery()
89              ->getResult()
90          ;
91      }
```

Dans cette illustration, la requête consiste à récupérer l'objet *Reservation* sous certaines conditions traduites par des *andWhere*. Ici, je ne souhaite récupérer que les réservations d'un user en particulier *:user*, dont le paramètre est défini ensuite, grâce à *setParameter,* par la variable *$user*, qui à son tour sera définit dans le contrôleur grâce à l'argument attendu *$user* de la requête personnalisée. La condition suivante consiste à ne récupérer que les réservations dont le statut de la réservation est confirmé *$confirmee*.

L'utilisation de cette requête personnalisée se fait dans le contrôleur relié à la vue dans laquelle je souhaite afficher ces données récupérées, dans ce cas précis, il s'agit de *UserController*. En effet, à la ligne 303, je définis l'attribut attendu de la requête personnalisée, c'est-à-dire l'utilisateur *$user* duquel je souhaite récupérer les réservations. Je crée ensuite, à la ligne 304, une variable accueillant les résultats de la requête, puis je fais appel au repository *ReservationRepository* dans lequel j'ai créée la requête, et j'appelle la requête *findReservationsAVenir* en lui indiquant entre parenthèses l'attribut attendu *$user* définit plus tôt.

*UserController.php*

```
300     #[Route('/user/reservations/aVenir', name: 'reservations_a_venir')]
301     public function reservationsAVenirDirectory(ReservationRepository $reservationRepository, User $user)
302     {
303         $user = $this->getUser();
304         $reservationsAVenir = $reservationRepository->findReservationsAVenir($user);
305         $toutesReservationsAVenir = $reservationRepository->findToutesReservationsAVenir();
306 
307         return $this->render('user/reservations/a_venir.html.twig', [
308             'reservationsAVenir' => $reservationsAVenir,
309             'toutesReservationsAVenir' => $toutesReservationsAVenir
310         ]);
311     }
```

Enfin, le rendu de cette requête se fait dans la vue *a_venir.html.twig* reliée au contrôleur grâce au *render* apparaissant sur la capture ci-dessus à la ligne 307.

*templates/user/reservations/a_venir.html.twig*

```
36      {% if reservationsAVenir %}
37          <h5>Vos réservations à venir</h5>
38          <p>Préparez votre séjour</p>
39          {% for reservationAVenir in reservationsAVenir %}
40          <p>Votre séjour du {{reservationAVenir.dateDebutFr}} au {{reservationAVenir.dateFinFr}}
            dans {{reservationAVenir.espace}}, au nom de  {{reservationAVenir.prenom}}
            {{reservationAVenir.nom}}</p>
41          <p>Durée totale : {{reservationAVenir.duree}} nuits</p>
42          <p>Prix total : {{reservationAVenir.prixTotal}} €</p>
43          {% if reservationAVenir.options is not empty %}
44          <ul>
45              {% for option in reservationAVenir.options %}
46                  <li>{{ option.nom }}</li>
47                  <li>{{ option.tarif }} €</li>
48              {% endfor %}
49          </ul>
50          {% else %}
51              <p>Pas d'option</p>
52          {% endif %}
53              <a href="{{ path('annuler_reservation', {'id' : reservationAVenir.id}) }}">Annuler ma
            réservation</a>
54          {% endfor %}
55          {% else %}
56          <h5>Pas de réservations à venir</h5>
57          <a href="{{ path('app_reservation') }}">Réserver</a>
58      {% endif %}
```

La capture d'écran ci-contre provient de cette vue en question. La première condition consiste à vérifier si des réservations à venir existent, pour que dans le cas contraire, je puisse préciser à l'utilisateur qu'il n'y en a aucune et lui proposer de réserver. Ensuite, après le titre et la ligne de texte permettant de contextualiser la nature de la réservation affichée, j'ai effectué une boucle sur l'ensemble des réservations à venir trouvées de l'utilisateur connecté, et pour chacune d'entre elle afficher ses détails, à savoir les dates de début et de fin, dans quel espace, à quel nom, la durée totale, les options s'il y en a et le prix total. Chaque réservation à venir dispose également d'une possibilité d'annulation, matérialisée ici par le *lien Annuler ma réservation*.

Si je n'avais pas utilisé Doctrine, j'aurais pu exécuter la requête suivante en langage SQL *Structured Query Langage* :

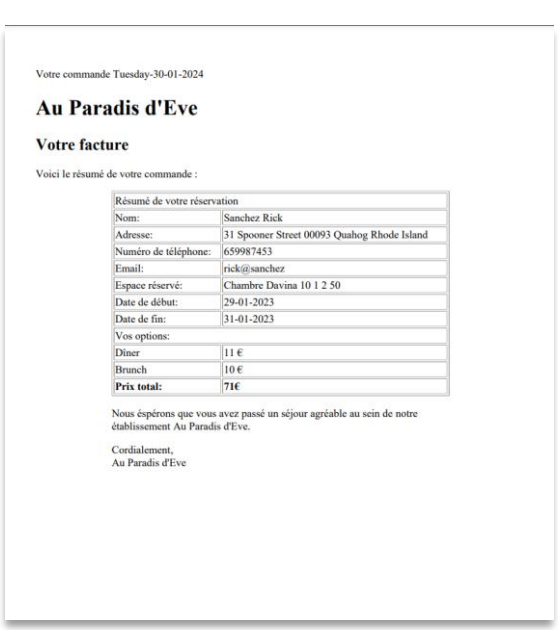
```sql
SELECT *
FROM reservation
WHERE date_debut >= CURRENT_DATE
AND statut = 'CONFIRMEE'
AND user_id = 1
ORDER BY date_debut ASC
```

*La même requête en langage SQL, en admettant que nous voulions récupérer les réservations de l'utilisateur ayant l'id 1.*

### DomPDF

Lorsque la date de fin de séjour est passée, les réservations passées apparaissent dans l'onglet Réservations passées, chez le client et chez l'administrateur, à côté desquelles un récapitulatif du séjour est téléchargeable en PDF.

Pour convertir une vue HTML en PDF, j'ai utilisé la bibliothèque open-source **Dompdf** qui permet de convertir des fichiers HTML en fichiers PDF.



Votre commande Tuesday-30-01-2024

**Au Paradis d'Eve**

**Votre facture**

Voici le résumé de votre commande :

| Résumé de votre réservation | |
|---|---|
| Nom: | Sanchez Rick |
| Adresse: | 31 Spooner Street 00093 Quahog Rhode Island |
| Numéro de téléphone: | 659987453 |
| Email: | rick@sanchez |
| Espace réservé: | Chambre Davina 10 1 2 50 |
| Date de début: | 29-01-2023 |
| Date de fin: | 31-01-2023 |
| Vos options: | |
| Dîner | 11 € |
| Brunch | 10 € |
| **Prix total:** | **71€** |

Nous espérons que vous avez passé un séjour agréable au sein de notre établissement Au Paradis d'Eve.

Cordialement,
Au Paradis d'Eve

templates > pdf_generator > index.html.twig > ...

```twig
{% block title %}Votre commande{% endblock %}
{% block body %}

{{currentDate}}

<h1>Au Paradis d'Eve</h1>
<h2>Votre facture</h2>
<p>Voici le résumé de votre commande : </p>
<div style="margin: 0 auto;display: block;width: 500px;">
    <table width="100%" border="1">
        <tr>
            <td colspan="2">
                Résumé de votre réservation
            </td>
        </tr>
        <tr>
            <td>Nom:</td>
            <td>{{reservation.nom}} {{reservation.prenom}}</td>
        </tr>
        <tr>
            <td>Adresse:</td>
            <td> {{reservation.adresseFacturation}}</td>
        </tr>
        <tr>
            <td>Numéro de téléphone:</td>
            <td>{{reservation.telephone}}</td>
        </tr>
        <tr>
            <td>Email:</td>
            <td>{{reservation.email}}</td>
        </tr>
        <tr>
            <td>Espace réservé:</td>
            <td>{{reservation.espace}}</td>
        </tr>
        <tr>
            <td>Date de début:</td>
            <td>{{reservation.dateDebutFr}}</td>
        </tr>
```

### Mailer

Lorsqu'une réservation obtient le statut de « confirmée », un mail de confirmation est envoyé sur l'adresse mail que l'utilisateur a fourni dans le formulaire, ce qui lui permet de retrouver toutes les informations nécessaires et utiles pour son séjour, ainsi que la démarche à effectuer pour l'annuler. Le composant Mailer de Symfony a été utilisé dans cette démarche, il est possible de l'implémenter en exécutant la commande :`composer require symfony/mailer`.

Voir plus en annexe p55.

# Axes d'améliorations

## Traduction de l'application

La traduction de l'application en allemand serait une amélioration significative du fait de la localisation de l'établissement, qui est à la frontière très proche de l'Allemagne. La traduction anglais serait également une amélioration Des touristes anglais se présentent également dans la région.

De plus, la traduction permet l'optimisation pour les moteurs de recherche : Les moteurs de recherche valorisent souvent les sites web multilingues, ce qui pourrait avoir un impact positif sur le classement de de l'application dans les résultats de recherche internationaux.

Une façon d'y parvenir avec Symfony serait d'abord d'effectuer la ligne de commande suivante : composer require symfony/translation, qui permettrait d'installer le traducteur Symfony, puis de configurer la langue par défaut, en l'occurrence pour ce projet, il s'agirait de la langue française, dans la configuration du site.

```yaml
# config/packages/translation.yaml
framework:
    default_locale: 'en'
    translator:
        default_path: '%kernel.project_dir%/translations'
```

Ensuite, il est nécessaire de configurer les textes à traduire accompagnés de leur traduction dans le fichier. Pour y faire appel, il faudra ensuite utiliser l'interface dédiée à cette tâche.

```yaml
# translations/messages.fr.yaml
Symfony is great: J'aime Symfony
```

```php
use Symfony\Contracts\Translation\TranslatorInterface;

public function index(TranslatorInterface $translator): Response
{
    $translated = $translator->trans('Symfony is great');
```

## Tarif dégressif, saisons et bons cadeaux

Ce deuxième axe d'amélioration concerne les tarifs et différentes offres faite aux clients de l'établissement. La gestion des tarifs selon les saisons et selon le nombre de nuits réservées serait une façon de personnaliser l'expérience client et d'optimiser la politique tarifaire. En adaptant les tarifs en

fonction des saisons, l'établissement peut répondre à la demande fluctuante tout au long de l'année, maximisant ainsi les revenus pendant les périodes de forte affluence.

### Tarif dégressif

La mise en place de tarifs dégressifs en fonction du nombre de nuits réservées inciterait les clients à prolonger leur séjour, favorisant ainsi la fidélité et contribuant à une occupation plus stable des chambres.

Cela pourrait prendre la forme suivante :

```php
function calculerTarifDegressif($nombreDeNuits) {
    // Définir les paliers de réduction
    $paliers = [
        1 => 0,    // Pas de réduction pour 1 nuit
        4 => 5,    // Réduction de 5% pour 4 nuits
        6 => 10,   // Réduction de 10% pour 6 nuits
        8 => 15    // Réduction de 15% pour 8 nuits et plus
    ];

    $nombreDeNuits = $reservation->getDuree();

    // Trouver le palier applicable
    foreach ($paliers as $nuit => $reduction) {
        if ($nombreDeNuits <= $nuit) {
            return $reduction;
        }
    }

    // Par défaut, pas de réduction
    return 0;
}
// Exemple d'utilisation
$nombreDeNuitsReserves = $reservation->getDuree();
$reductionAppliquee = calculerTarifDegressif($nombreDeNuitsReserves);

echo "Réduction applicable : {$reductionAppliquee}%";
```

Commencer par définir les paliers de réductions que l'on souhaite proposer, ici il n'y aurait de tarif dégressif qu'à partir de la quatrième nuit réservée.

Ensuite, boucler sur les réservations en allant chercher leur durée qui est un attribut de l'objet *Reservation* pour le comparer au palier préalablement établi, et appliquer le nouveau tarif si une correspondance est trouvée.

Le point positif serait d'encourager les clients à prolonger leur séjour, ce qui peut augmenter la durée moyenne de séjour et maximiser le chiffre d'affaires par client.

```php
// src/Entity/Espace.php

#[ORM\Column(type: Types::FLOAT, nullable: true)]
private ?float $tarifSaisonPrintemps;

#[ORM\Column(type: Types::FLOAT, nullable: true)]
private ?float $tarifSaisonEte;
```

### Tarification en fonction du moment de l'année

La tarification saisonnière permet d'ajuster les tarifs de l'établissement en fonction de la demande fluctuante tout au long de l'année.

```php
// src/Service/TarificationSaisonniere.php

namespace App\Service;

use App\Entity\Espace;
use Doctrine\ORM\EntityManagerInterface;

class TarificationSaisonniere
{
    private $entityManager;

    public function __construct(EntityManagerInterface $entityManager)
    {
        $this->entityManager = $entityManager;
    }

    public function getTarifSaisonnier(Espace $espace, \DateTimeInterfac
    {
        // Logique pour déterminer la saison en fonction de la date et r

        // Utilisation des champs tarifSaisonPrintemps, tarifSaisonEte, etc. de l'entité Espace

        // Exemple approximatif :
        if ($date >= $espace->getDateDebutSaisonPrintemps() && $date <= $espace->getDateFinSaisonPrintemps()) {
            return $espace->getTarifSaisonPrintemps();
        } elseif ($date >= $espace->getDateDebutSaisonEte() && $date <= $espace->getDateFinSaisonEte()) {
            return $espace->getTarifSaisonEte();
        }

        // Retournez un tarif par défaut si aucune saison trouvée
        return $espace->getTarifDefaut();
    }
}
```

```php
#[Route('/reservation/new/{id}', name: 'new_reservation')]
public function newReservation(
    Espace $espace,
    Reservation $reservation = null,
    EntityManagerInterface $entityManager,
    ReservationRepository $reservationRepository,
    Request $request,
    TarificationSaisonniere $tarificationSaisonniere
) {
    // ...

    if ($form->isSubmitted() && $form->isValid()) {

        // Utilisation du service de tarification saisonnière
        $tarifSaison = $tarificationSaisonniere->getTarifSaisonnier($espace, $reservation->getDateDebut());
        $reservation->setTarifSaison($tarifSaison);

        $entityManager->persist($reservation);
        $entityManager->flush();

        // ...
    }
}
```

# Traduction

L'utilisation de formulaire ayant été essentielle dans ce projet, j'ai choisi de traduire une partie de la documentation officielle de Symfony sur les formulaires. [**1117** caractères espaces non compris]

## Texte en anglais

### Forms

Creating and processing HTML forms is hard and repetitive. You need to deal with rendering HTML form fields, validating submitted data, mapping the form data into objects and a lot more. Symfony includes a powerful form feature that provides all these features and many more for truly complex scenarios.

### Usage

The recommended workflow when working with Symfony forms is the following:

- Build the form in a Symfony controller or using a dedicated form class;
- Render the form in a template so the user can edit and submit it;
- Process the form to validate the submitted data, transform it into PHP data and do something with it (e.g. persist it in a database).

Each of these steps is explained in detail in the next sections. To make examples easier to follow, all of them assume that you're building a small Todo list application that displays "tasks".
This class is a "plain-old-PHP-object" because, so far, it has nothing to do with Symfony or any other library. It's a normal PHP object that directly solves a problem inside your application (i.e. the need to represent a task in your application). But you can also edit Doctrine entities in the same way.
This may be confusing at first, but it will feel natural to you soon enough. Besides, it simplifies code and makes "composing" and "embedding" form fields much easier to implement.

## Traduction française

### Formulaires

La création et le traitement de formulaires HTML sont difficiles et répétitifs. Vous devez gérer le rendu des champs du formulaire HTML, la validation des données soumises, le traitement des données du formulaire en objets etc. Symfony inclut une fonctionnalité de formulaire puissante qui fournit toutes ces fonctionnalités, pour des scénarios véritablement complexes.

### Utilisation

Le flux de travail recommandé pour travailler avec les formulaires Symfony est le suivant :

- Construire le formulaire dans un contrôleur Symfony ou en utilisant une classe de formulaire dédiée ;
- Rendre le formulaire en un modèle afin que l'utilisateur puisse l'éditer et le soumettre ;
- Traiter le formulaire pour valider les données soumises, les transformer en données PHP et en faire quelque chose (par exemple, les persister dans une base de données).

Chacune de ces étapes est expliquée en détail dans les sections suivantes. Pour que les exemples soient plus faciles à suivre, ils supposent tous que vous construisez une petite application de liste de tâches qui affiche des "tâches".
Cette classe est un "plain-old-PHP-object" car, jusqu'à présent, elle n'a rien à voir avec Symfony ou toute autre bibliothèque. C'est un objet PHP normal qui résout directement un problème dans votre application (i.e. le besoin de représenter une tâche dans votre application). Mais vous pouvez aussi éditer des entités Doctrine de la même manière.
Cela peut être perturbant de prime abord, mais ça vous semblera naturel assez tôt. En plus, ça simplifie le code et il est plus simple

# Conclusion et remerciements

## En conclusion

En conclusion de ce dossier dédié au projet de réservation de chambres d'hôtes, je peux affirmer avoir réalisé avec plaisir ce site web et web mobile. En l'état actuel, « Au Paradis d'Eve » répond aux demandes principales du client, à savoir présenter son établissement, les alentours et permettre aux utilisateurs de réserver une chambre.

Les axes d'améliorations évoqués plus tôt dans ce dossier me permettront de valoriser le projet.

## Mes remerciements

La conception de cette application a été façonnée grâce aux connaissances acquises durant ma formation Développeur Web et Web Mobile.

Je tiens à exprimer mes sincères remerciements à Stéphane, Quentin, Mickaël et Alexandre, formateurs chez Elan Formation, pour leur implication et leur patience tout au long de ce parcours.

Merci également à l'équipe administrative pour sa réactivité et sa bienveillance.

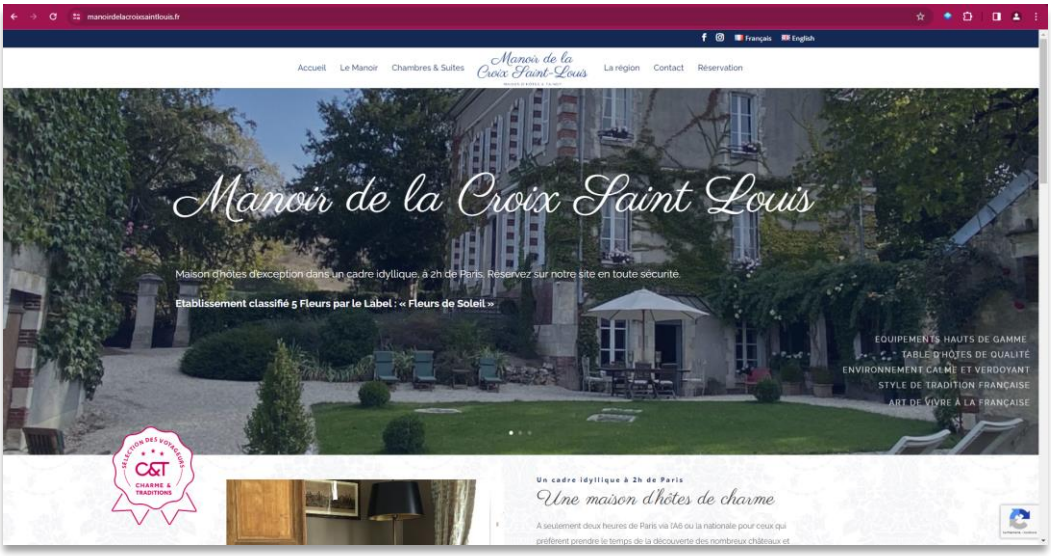
Ce projet marque la fin d'une étape significative de mon parcours de formation, mais il représente également le début, je l'espère, d'une aventure dans le monde professionnel.

Davina Houquet

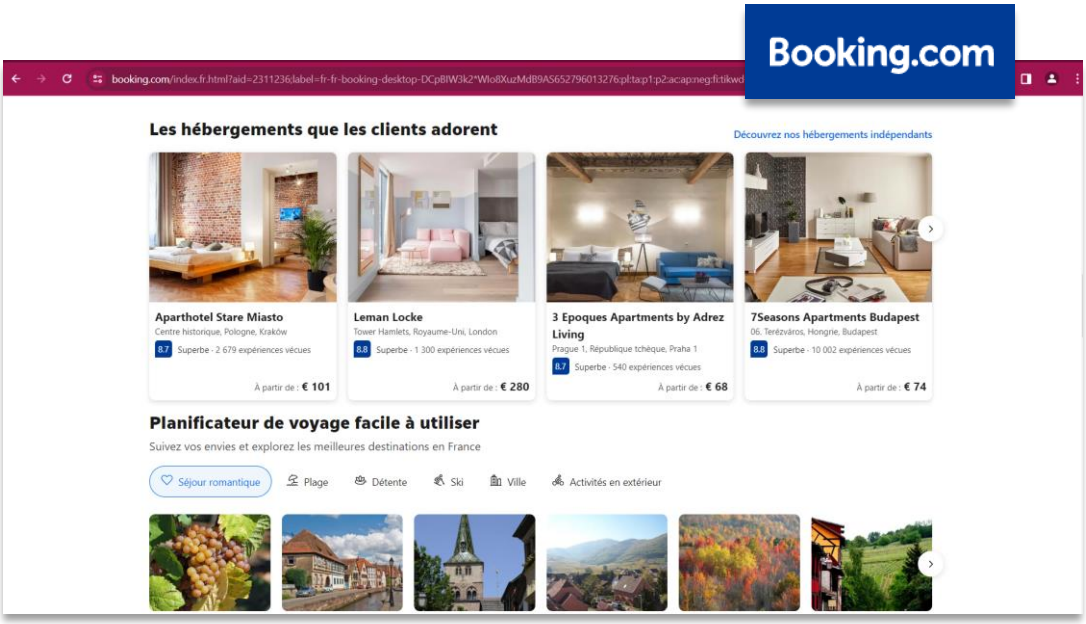
# Annexes

## BENCHMARK

Références utilisées dans la création de ce projet :



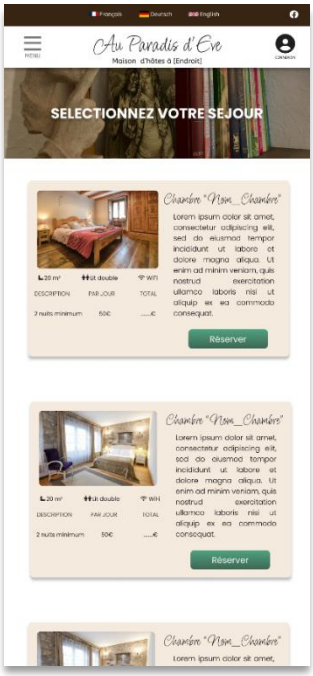

*Manoir de la Croix Saint Louis*




*Booking.com*

# Maquettage

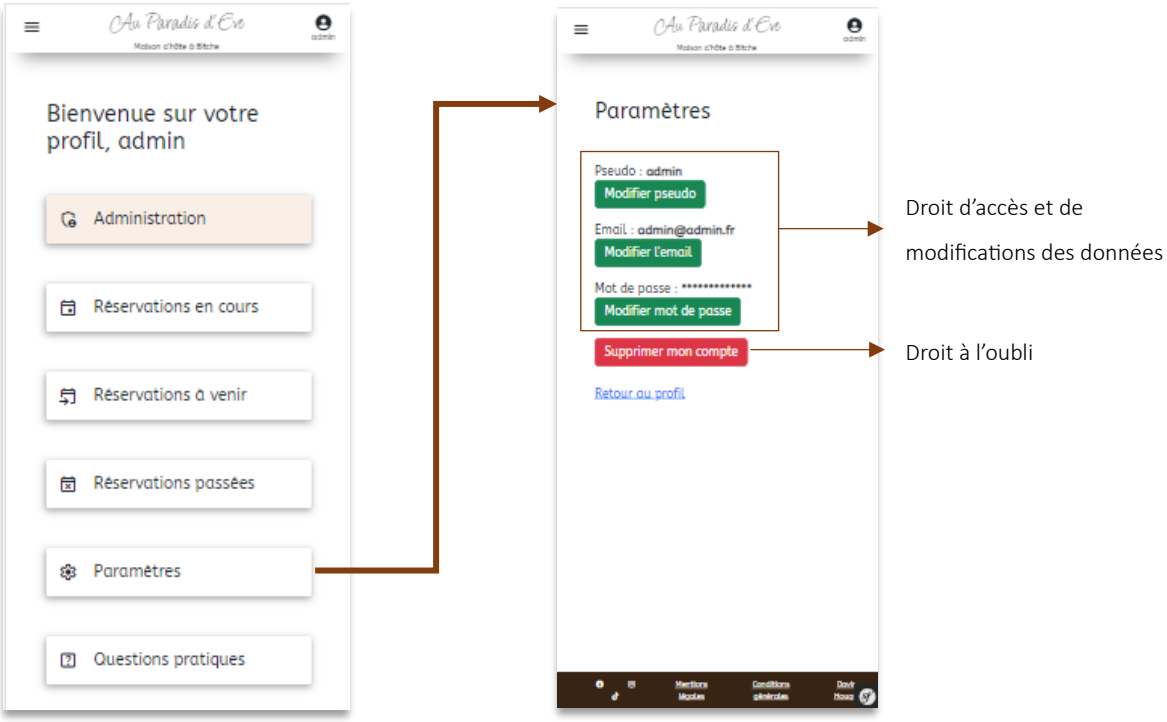
*Page de réservation et détail d'un espace*

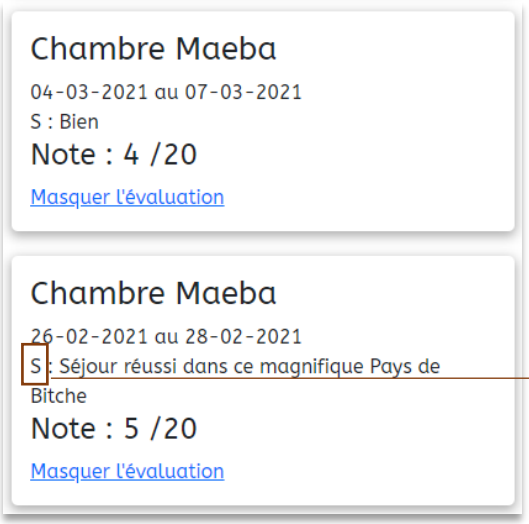





# Règlement Général sur la Protection des données

## Exactitude des données et droit à l'oubli



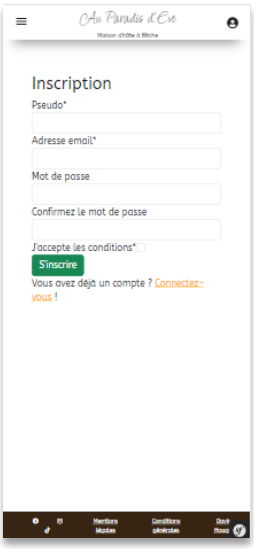

Cela permet de garantir l'anonymat aux utilisateurs.

Seule la première lettre du prénom apparaît si un avis est laissé

Seule la première lettre du prénom apparaît si un avis est laissé

## Minimisation des données

Seule la première lettre du prénom apparaît si un avis est laissé

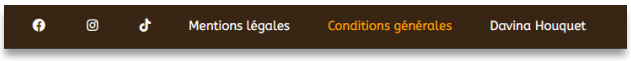


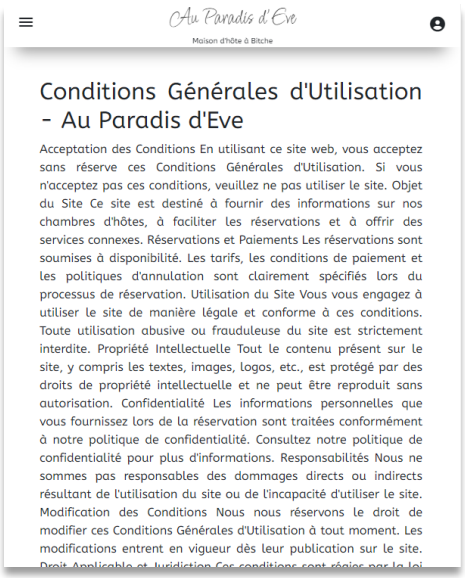

Seules les informations nécessaires et non sensibles sont demandées. L'adresse physique de facturation n'est demandée que dans le cas où l'utilisateur effectue une réservation.

Les conditions générales d'utilisation sont accessibles dans le footer si l'utilisateur souhaite les consulter.

### Transparence

Ce principe consiste à garantir la transparence tout au long du processus de traitement des données personnelles des utilisateurs. En l'occurrence, les mentions légales et les conditions générales sont accessibles dans le pied de page.




## Search Engine Optimization

### Meta description

Une meta description est une balise HTML utilisée pour fournir un résumé concis et informatif du contenu d'une page web. Cette balise est intégrée dans l'en-tête HTML d'une page et n'est pas visible sur la page elle-même, mais elle est souvent affichée dans les résultats des moteurs de recherche.

Le fichier *base.html.twig* qui est étendu dans l'ensemble des autres vues contient un bloc meta description dans son en-tête qui permet de changer la description sur chaque vue en ajoutant simplement ce bloc.

```
<meta name="description" content="{% block meta_description %}Accueil - Au Paradis d'Eve{% endblock %}">
```

*templates/base.html.twig*

Cela permet d'avoir une meta description individuelle et adaptée à chaque vue et d'améliorer le référencement général.

```
{% block meta_description %} Remplissez vos coordonnées - Coordonnées de réservation {% endblock %}
```

*Deuxième étape du formulaire de réservation reservation/new_coordonnees.html.twig*

### Balises *strong*

La balise ** est une balise HTML utilisée pour indiquer que le texte qu'elle entoure doit être rendu en tant que texte fort ou important sur une page web.

```
<p><a href="{{ path('app_reservation') }}"><strong>Réservez</strong></a> sur notre site en toute sécurité</p>
```

*accueil/index.html.twig*

```
<h1 class="licorice-title"><strong>Au Paradis d'Eve</strong></h1>
```

*accueil/index.html/twig*

### Texte de remplacement pour les images

L'attribut "*alt*" (texte alternatif), est une propriété HTML utilisée pour fournir une description textuelle d'une image sur une page web. Cet attribut est inclus dans la balise *<img>* qui est utilisée pour intégrer des images dans le code HTML d'une page.

Cet attribut est essentiel pour garantir l'accessibilité aux utilisateurs malvoyants ou non-voyants qui en dépendent pour comprendre l'information contenue dans une image. Il est également important pour améliorer l'optimisation pour les moteurs de recherche puisqu'ils utilisent également l'attribut "*alt*" pour comprendre le contenu des images.

```
<img src="{{ asset('img/altschlossfelsen.jpg') }}"
alt="L'Altschlossfelsen, nom à consonance germanique signifie :Rocher du
Vieux Château.  Barre rocheuse aux couleurs rougeâtres qui porte toutes
les particularités géologiques des Vosges du Nord.">
```

*accueil/index.html.twig*

Dans ce projet, il m'a semblé important d'apporter la possibilité à l'administrateur de proposer ses propres textes de remplacement pour apporter une description réaliste des espaces proposés. Ainsi, l'objet Image dispose d'un attribut *alt_image* qui permet de remplir un texte lors du téléchargement d'image pour un espace.

```
<img src="{{ asset('uploads/images/' ~ image.lienImage) }}" alt="{{
image.altImage|default('Chambre de l\'établissement Au Paradis
d\'Eve, lumineuse, disposant d\'un lit, d\'une commode, d\'un
fauteuil')}}">
```

*espace/ show.html.twig*

| id | espace... | lien_image | alt_image |
|---|---|---|---|
| 28 | 31 | f12727f9f4fad96df3e0f81e47fd54eb.jpg | Cuisine aux murs couleur beige. Avec un comp... |
| 29 | 32 | f9464df8a57cfd53163d63460f25e6b0.jpg | Un grand canapé confortable trône au centre, ... |
| 30 | 33 | b829d3952cc4baf2618b75d98a7489d6.jpg | La grande salle à manger peut accueillir confor.. |

*Table Image - HeidiSQL*

## Hiérarchie des titres

La hiérarchisation des balises de titre en HTML se réfère à l'utilisation des balises *<h1>, <h2>, <h3>, <h4>, <h5>, et <h6>* pour structurer le contenu d'une page web de manière hiérarchique. Ces balises de titre sont utilisées pour définir les niveaux de titre et organiser le texte de la page en sections, où *<h1>* représente le titre principal (le plus haut niveau de titre) et *<h6>* est le niveau de titre le plus bas.

```
<h1>Nos espaces communs</h1>
```

*espace/index.html.twig*

```
<h2>Nos chambres</h2>
```

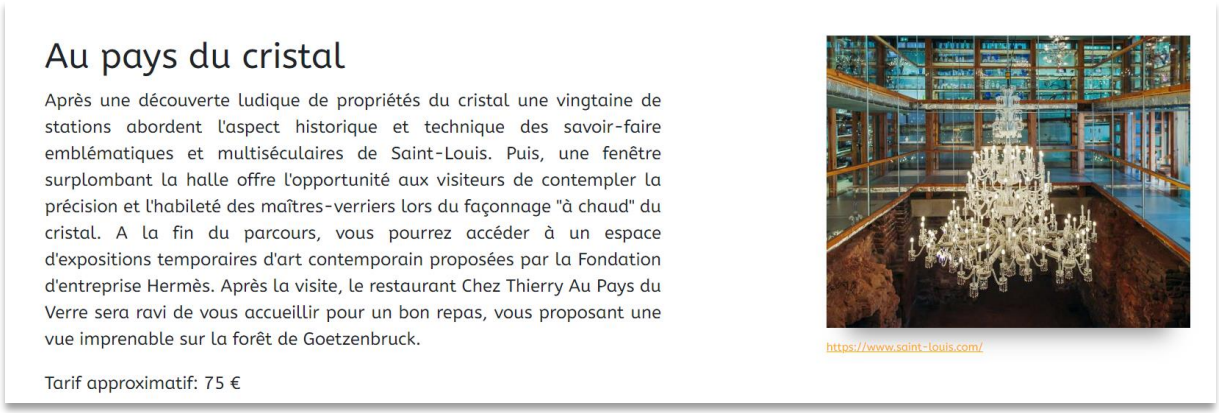
*espace/index.html.twig*

```
<h3>Les réservations à rembourser partiellement</h3>
```

*user/reservations/annulees.html.twig*

## Netlinking

Le netlinking, également connu sous le nom de backlinking ou création de liens, fait référence à la pratique de créer des liens hypertextes pointant vers une page web à partir d'autres sites. Le netlinking est une composante importante du référencement (SEO) et il joue un rôle clé dans la façon dont les moteurs de recherche évaluent la pertinence et l'autorité d'une page web.

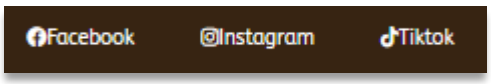




Dans l'onglet Alentours, l'administrateur a la possibilité de renvoyer vers un lien en rapport avec la sortie proposée.

## Réseaux sociaux

Les liens des réseaux sociaux rejoignent la logique du Netlinking. Les moteurs de recherche tiennent compte de l'engagement social.




*Extrait du pied de page*

## Accesibilité

### DomPDF Controller

127.0.0.1:8000/profile/pdf/generator/71

```
<a href="{{path('app_pdf_generator', {'id' : reservationPassee.id})}}" target="_blank"">PDF</a>
```

*Lien vers la génération du PDF*

```
"dompdf/dompdf": "^2.0",
```

*composer.json*

```php
<?php

namespace App\Controller;

use DateTime;
use Dompdf\Dompdf;
use App\Entity\Reservation;
use App\Repository\UserRepository;
use App\Repository\ReservationRepository;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class PdfGeneratorController extends AbstractController
{
    #[Route('/profile/pdf/generator/{id}', name: 'app_pdf_generator')]
    public function index(Reservation $reservation, ReservationRepository $
    {
        $currentDate = date('l-d-m-Y');
        $reservation = $reservationRepository->findOneBy([]);
        $data = [
            'currentDate' => $currentDate,
            'reservation' => $reservation
        ];
        $html =  $this->renderView('pdf_generator/index.html.twig', $data);
        $dompdf = new Dompdf();
        $dompdf->loadHtml($html);
        $dompdf->render();

        return new Response(
            $dompdf->stream('resume', ["Attachment" => false]),
            Response::HTTP_OK,
            ['Content-Type' => 'application/pdf']
        );
    }
}
```

*PdfGeneratorController.php*

## MAILER

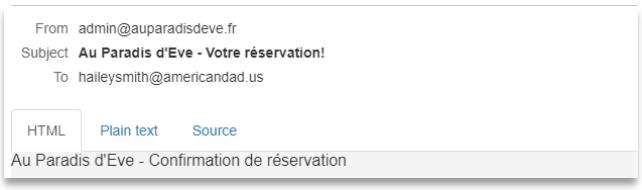
Utilisation du composant Mailer de Symfony pour envoyer un mail de confirmation de réservation

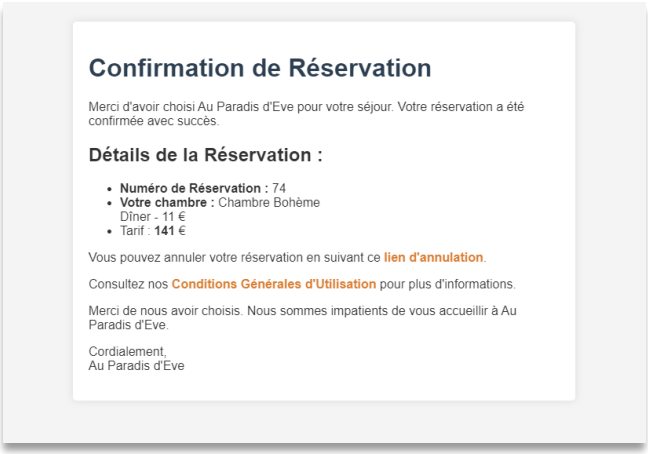
```php
$emailReservation = $reservation->getEmail();
$prenom = $reservation->getPrenom();

$email = (new Email())
    ->from('admin@auparadisdeve.fr')
    ->to($emailReservation)
    ->subject('Au Paradis d\'Eve - Votre réservation!')
    ->html($this->renderView('mailer/index.html.twig', [
        'reservation' => $reservation,
        'espace' => $espace,
    ]));
$mailer->send($email);
```

*Extrait de UserController - Envoi du mail de confirmation*

```twig
<body>
    <div class="container">
        <h1>Confirmation de Réservation</h1>
        <p>Merci d'avoir choisi Au Paradis d'Eve pour votre séjour. Votre réservation a été confirmée
        avec succès.</p>

        <h2>Détails de la Réservation :</h2>
        <ul>
            <li><strong>Numéro de Réservation :</strong> {{ reservation.id }}</li>
            <li><strong>Votre chambre :</strong> {{ espace.nomEspace }}</li>
            {% if reservation.options is not null %}
            {% for option in reservation.options %}
            {{ option.nom }} - {{ option.tarif }} €<br>
            {% endfor %}
            {% endif %}
            <li>Tarif : <strong>{{ reservation.prixTotal }}</strong> € </li>
        </ul>

        <p>Vous pouvez annuler votre réservation en suivant ce <a href="{{ path('annuler_reservation',
        {'id' : reservation.id}) }}">lien d'annulation</a>.</p>

        <p>Consultez nos <a href="{{ path('conditions') }}">Conditions Générales d'Utilisation</a> pour
        plus d'informations.</p>

        <p>Merci de nous avoir choisis. Nous sommes impatients de vous accueillir à Au Paradis d'Eve.</
        p>

        <p>Cordialement,<br>Au Paradis d'Eve</p>
    </div>
</body>
```

*Fichier HTML du mail à envoyer*

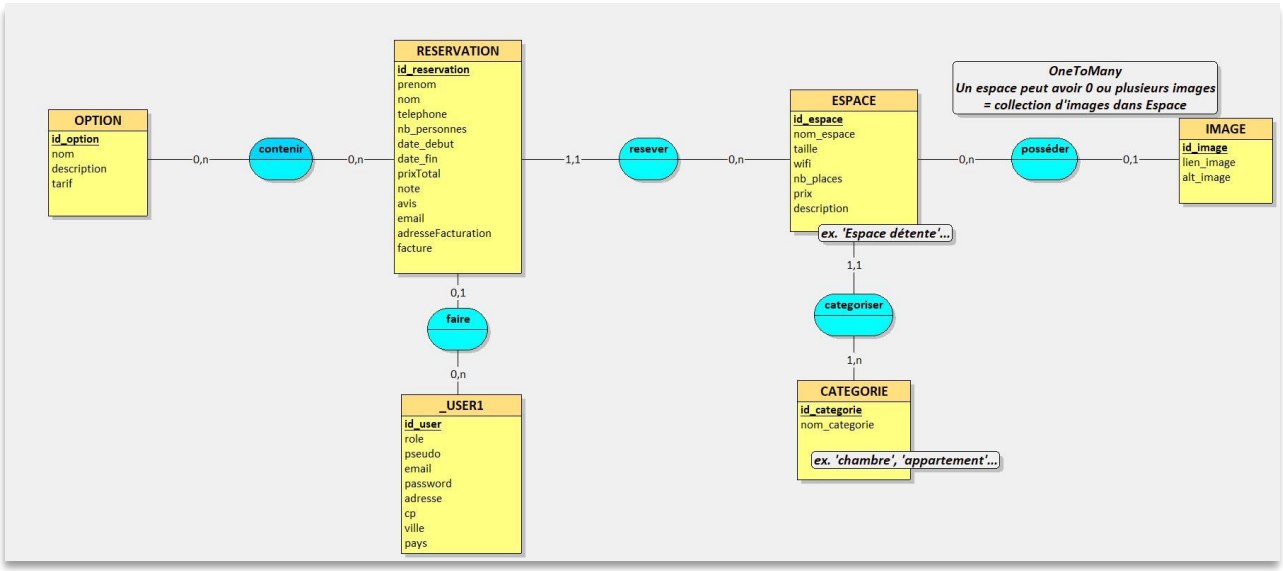



*Rendu réel du mail*

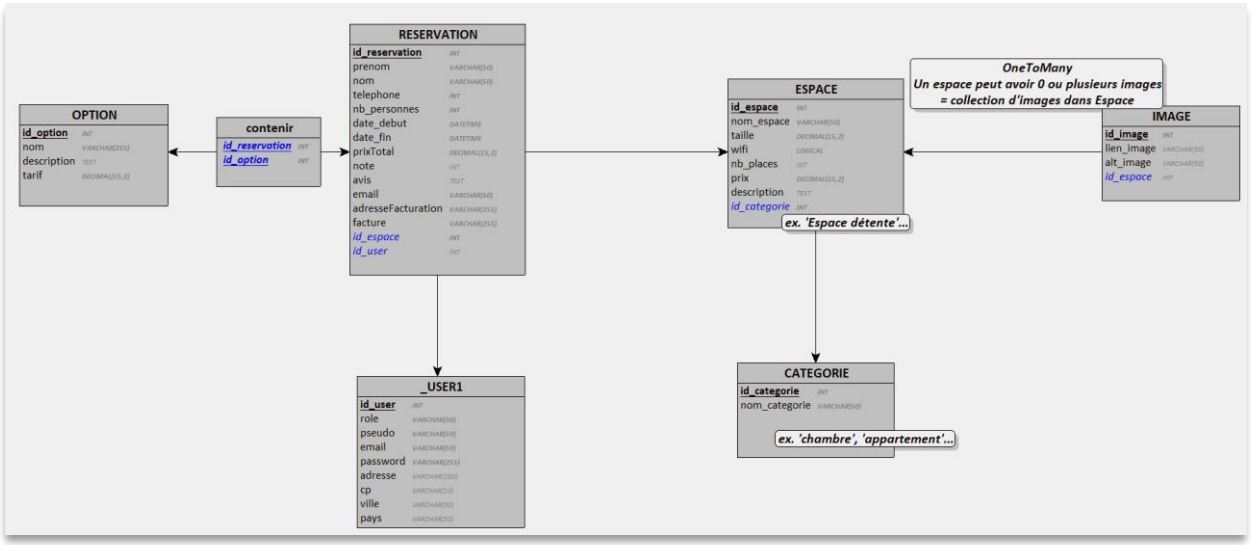
# Modèle conceptuel de données

*Modèle conceptuel de données réalisé sur Looping*[8]




# Modèle logique de données

*Modèle logique de données réalisé sur Looping*




# Certificat SSL et HTTPS

## SSL: SECURE SOCKETS LAYER

[8] Logiciel de modélisation conceptuelle de données qui permet d'organiser et de structurer des données

Le SSL est une technologie standard de sécurisation des connexions Internet par le chiffrement des données transitant entre un navigateur et un site web (ou entre deux serveurs). Durant leur transfert, les données (personnelles, financières, etc.) sont ainsi protégées des hackers qui ne peuvent ni les voir ni les détourner.

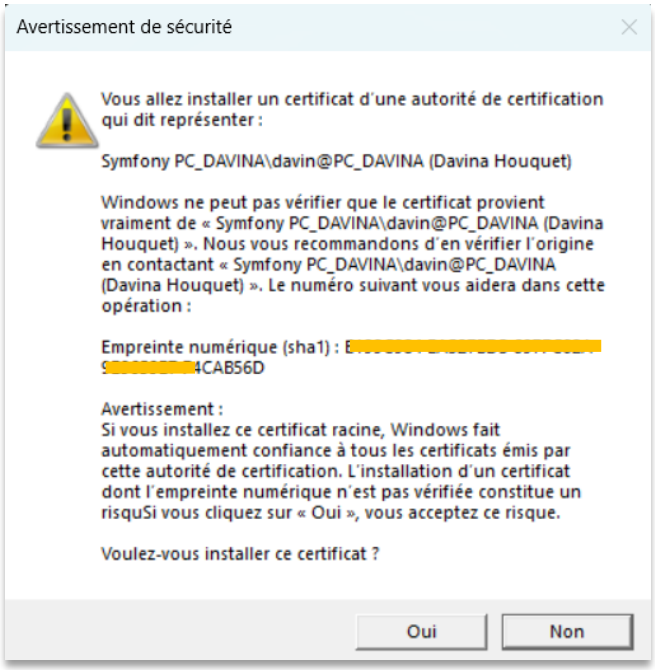
### HTTPS: HyperText Protocol Secure

Lorsqu'un site web est sécurisé par un certificat TLS/SSL, son URL commence par la mention HTTPS. Pour connaître les détails du certificat (Autorité de certification ou AC émettrice, raison sociale du propriétaire du site, etc.), les internautes peuvent cliquer sur le cadenas dans la barre d'adresse du navigateur.


### Mise en place

Il est possible d'effectuer cette mise en place en effectuant la commande `symfony.exe server :ca :install`.

```
PS C:\laragon\www\Au_Paradis_d_Eve> symfony.exe server:ca:install
You might have to enter your root password to install the local Certificate Authority certificate
The local CA is now installed in the system trust store!
Generating a default certificate for HTTPS support

 [OK] The local Certificate Authority is installed and trusted
```




Un avertissement apparaîtra alors pour confirmer l'installation du certificat. Une empreinte numérique est alors fournie.

Et lorsqu'on redémarre le serveur, on peut constater que le lien commence désormais par *https*.

```
[OK] Web server listening
     The Web server is using PHP CGI 8.1.10
     https://127.0.0.1:8000
```

https://127.0.0.1:8000/home