www.isl.be

INSTITUT SAINT-LAURENT
enseignement de promotion sociale

rue Saint-Laurent 33 - 4000 LIEGE
04 223 11 31

Cahier des charges

BOX ToGo

Marco-Cornilio PARASMO

Année 2022 - 2023

Table des matières

1. Introduction

1.1. Présentation du sujet

Cette plateforme numérique vise à mettre en relation les propriétaires d'espaces de rangement inutilisés et ceux qui cherchent des solutions de stockage. Avec une interface conviviale et sécurisée, elle simplifie la location, optimise l'usage d'espaces souvent délaissés et encourage la constitution d'une communauté autour du partage d'espace de stockage.

1.2. Objectifs du site

Faciliter la connexion : Mettre en relation les propriétaires d'espaces de rangement inutilisés avec ceux qui ont besoin de tels espaces pour stocker leurs biens.
Optimiser l'utilisation des espaces de rangement : Encourager l'utilisation efficace des espaces de rangement inutilisés, en offrant aux propriétaires un moyen de monétiser ces espaces.
Offrir une variété d'options : Proposer une large gamme d'espaces de stockage dans différentes régions et à des prix variés, pour répondre à une variété de besoins de stockage.
Assurer la facilité d'utilisation : Offrir une interface utilisateur intuitive qui facilite le processus de location, y compris la recherche d'espaces disponibles, la communication entre les parties et le processus de paiement.
Promouvoir la sécurité et la confiance : Fournir un environnement sécurisé pour les transactions et offrir des garanties pour les locataires et les propriétaires.
Favoriser une communauté : Créer un espace où les propriétaires et les locataires peuvent partager leurs expériences, donner et recevoir des conseils, et développer des relations basées sur des besoins communs.
Assurer le support client : Fournir un service d'assistance pour répondre aux questions, résoudre les problèmes et améliorer l'expérience utilisateur globale.

1.3. Public cible

Propriétaires d'espaces inutilisés : Ce groupe comprend les individus ou les entreprises qui possèdent des espaces inutilisés tels que des caves, des entrepôts, des locaux commerciaux, des abris de jardin, etc. Ils cherchent à maximiser l'utilisation de ces espaces et à générer des revenus supplémentaires en les louant.
Individus cherchant une source de revenu supplémentaire : Ce groupe comprend des personnes qui peuvent ne pas avoir d'espace inutilisé mais sont disposées à investir dans l'achat ou la location d'espaces à des fins de sous-location. Ces individus cherchent à générer des revenus passifs par le biais de la location d'espaces de stockage.

Locataires à la recherche d'espaces de stockage : Ce groupe comprend les individus ou les entreprises qui ont un besoin supplémentaire d'espace de stockage pour leurs biens. Cela peut inclure les étudiants, les jeunes professionnels, les petites entreprises, les collectionneurs, les déménageurs, etc.
Secteur de l'économie de partage : La plateforme peut également intéresser les acteurs de l'économie de partage, qui cherchent à optimiser l'utilisation des ressources disponibles et à favoriser des modes de consommation plus durables.

2. Exigences fonctionnelles

Utilisateur non connecté :

L'utilisateur peut naviguer sur le site sans s'authentifier.
L'utilisateur peut créer un compte (s'inscrire) sur le site.
L'utilisateur peut se connecter (s'authentifier) sur le site.
L'utilisateur peut se connecter (s'authentifier) sur le site a l'aide de google.
L'utilisateur peut choisir la langue du site.
L'utilisateur peut consulter le profil d'un propriétaire.
L'utilisateur peut consulter les détails d'une offre.
L'utilisateur peut visualiser l'emplacement d'un espace de stockage sur une carte.
Visualiser les avis et notes des propriétaires.
L'utilisateur peut effectuer des recherches d'espaces disponibles.

Utilisateur connecté :

L'utilisateur authentifié peut gérer son profil.
L'utilisateur authentifié peut modifier son mot de passe.
L'utilisateur authentifié peut ajouter un espace de stockage à ses favoris.
L'utilisateur authentifié peut rédiger un commentaire sur un espace de stockage.
L'utilisateur authentifié peut noter un espace de stockage.
Gérer ses réservations (visualiser les réservations passées et actuelles).
Consulter l'historique des espaces loués.
Réserver un espace de stockage directement depuis le site.
L'utilisateur authentifié peut contacter un propriétaire.

Propriétaire :

Le propriétaire authentifié peut ajouter une annonce.
Le propriétaire authentifié peut mettre à jour les détails de son espace de stockage.
Le propriétaire authentifié peut gérer la version multilingue de son profil.
Communiquer directement avec l'utilisateur par le biais d'un système de messagerie interne.

Administrateur du site :

Suivre les métriques du site (nombre d'utilisateurs, nombre de réservations, etc.).
L'administrateur peut gérer les commentaires (supprimer des commentaires signalés, etc.).
L'administrateur peut gérer les annonces (supprimer des annonces signalées, etc.).
L'administrateur peut gérer les utilisateurs (activer/désactiver des comptes, gérer des signalements, etc.).

2.1.1 Visiteur

Un visiteur non authentifié peut explorer le site et effectuer des recherches parmi les espaces de stockage proposés, en filtrant par type d'espace, localité ou code postal.
Un visiteur non authentifié peut examiner les informations détaillées d'une annonce, y compris sa description, les équipements disponibles, son adresse précise et son emplacement sur une carte interactive. De plus, des images de l'annonce ainsi que les tarifs sont consultables.
Un visiteur non authentifié a la faculté d'accéder au profil d'un propriétaire d'annonce pour y lire les commentaires et évaluations liés à ses espaces de stockage.
Un visiteur non authentifié peut paramétrer la langue de l'interface parmi trois options : anglais, néerlandais et français.
Un visiteur non authentifié a l'option de créer un compte sur la plateforme en utilisant soit son adresse e-mail, soit son compte Google.

2.1.2 Utilisateur

Un utilisateur authentifié a la capacité de personnaliser les éléments de son profil, y compris son nom, prénom, avatar, adresse e-mail, numéros de téléphone, et date de naissance.
Un utilisateur authentifié peut mettre à jour son mot de passe.
Un utilisateur authentifié peut ajouter un espace de stockage à sa liste de favoris.
Un utilisateur authentifié a la possibilité de laisser un avis et d'attribuer une note à un espace de stockage.
Un utilisateur authentifié peut effectuer une réservation d'espace de stockage.
Un utilisateur authentifié peut administrer ses réservations et consulter l'historique de celles-ci.
Un utilisateur authentifié peut entrer en communication avec le propriétaire d'un espace de stockage.

2.1.3 Propriétaire

Un propriétaire a la possibilité de publier une nouvelle annonce, pour laquelle diverses informations sont requises : le type du bien, son état, le prix demandé, la surface de stockage disponible, les dimensions de l'entrée et la localisation du bien.
Un propriétaire a la faculté de mettre à jour les caractéristiques de son espace de stockage à tout moment.

Un propriétaire peut gérer la version multilingue de son profil, lui permettant d'atteindre une audience plus large.
Un propriétaire peut entrer en contact avec les utilisateurs intéressés grâce à un système de messagerie intégré.
Un propriétaire a la possibilité de retirer son annonce de la plateforme à sa discrétion.

2.1.4 Administrateur

Un administrateur a la capacité de suivre le nombre total d'utilisateurs inscrits sur la plateforme.
Un administrateur a la capacité de suivre le nombre total d'annonces mises en ligne.
Un administrateur a la capacité de suivre le nombre total de réservations effectuées sur le site.
Un administrateur a la capacité de suivre le nombre total de commentaires déposés.
Un administrateur a le pouvoir de supprimer une annonce de la plateforme.
Un administrateur a le pouvoir de supprimer un compte utilisateur.

2.2. Règles métiers

Un mot de passe doit comporter un minimum de 7 caractères.
Un mot de passe doit être une combinaison de caractères alphabétiques et numériques.
Après quatre tentatives de connexion infructueuses, un compte est automatiquement verrouillé, qu'il s'agisse d'un compte de Prestataire ou d'Internaute.
La publication d'une nouvelle annonce fait passer le statut d'un utilisateur de «simple utilisateur» à «propriétaire».
Pour publier une annonce, il est impératif de spécifier le type du bien, son prix, sa surface en mètres carrés, ainsi que son adresse.

2.3. Interface

Le site est multilingue et offre la possibilité d'afficher le contenu en anglais, en français et en néerlandais.
L'affichage des annonces ainsi que les résultats de recherche doivent être paginés pour faciliter la navigation.
L'interface doit être épurée et centrée sur la mise en avant des annonces disponibles.
Le mot de passe doit être hashé et chiffré pour garantir la sécurité des données utilisateur.
Le site doit être responsive, s'adaptant ainsi aux écrans de desktop, de tablettes et de smartphones.

3. Exigences non fonctionnelles

3.1. Conception

La réécriture des URL est un élément clé pour améliorer l'expérience utilisateur ainsi que le référencement du site. Des URL propres, simples à comprendre et descriptives non seulement rendent le site plus navigable pour les utilisateurs, mais elles sont

```
#[Route('/owner/profile/{id}', name: 'profile_page')]
```

Le layout est adapté aux desktops, tablettes et smartphones

Avec la diversité des appareils utilisés pour accéder au Web, il est indispensable de concevoir un site adaptatif. Le design responsive garantit que le site offre une expérience utilisateur optimale quel que soit le type de dispositif utilisé. Pour atteindre cet objectif, nous utiliserons des techniques CSS modernes comme les Media Queries, pour assurer que l'interface s'ajuste dynamiquement à la taille de l'écran.

Un nom de domaine doit être déposé

Le choix et l'enregistrement d'un nom de domaine sont cruciaux pour l'identité et la crédibilité du site. Il est recommandé de choisir un nom de domaine qui est non

1. GaragaLoc	5. StockLoc	9. BoxToGo	13. StockageToGo
2. EspaceStock	6. ProPartage	10. StockInBox	
3. StockezMoi	7. PartageBox	11. ProStockage	
4. BoxPartage	8. LocaBox	12. EspaceEnPlus	

services

3.2. Acteurs

Il existe trois profils d'utilisateurs sur le site.
L'Administrateur a pour mission de gérer et superviser le site dans son ensemble.
Le Propriétaire, qu'il soit particulier ou indépendant, possède un espace inutilisé qu'il souhaite mettre en location. Dans le climat économique actuel, l'optimisation de l'espace est un aspect crucial.
L'Utilisateur Simple est toute personne qui navigue sur le site, qu'elle soit en mode consultation, qu'elle envisage de s'inscrire ou qu'elle soit déjà inscrite et puisse accéder à tous les services proposés.

Quant à l'Utilisateur de Base, il représente le point de départ de chacun des profils. Un visiteur devient un Utilisateur Simple dès son inscription. S'il publie une annonce, il est promu au statut de Propriétaire. Ce même utilisateur peut également se voir attribuer le rôle d'Administrateur.

4. Cas d'utilisation

4.1 Résumé

Priorité 1	Priorité 3
CU01 Visualiser la page d'accueil	CU10 Visualiser la localisation sur une carte
CU02 Consulter la description d'une annonces	CU11 Contacter un propriétaire
CU03 Rechercher d'une espace de stockage	CU12 Changer de mot de passe
CU04 Consulter la fiche signalétique d'un espaces	CU13 Ajouter une annonce
CU05 Consulter la fiche signalétique d'un propriétaire	CU14 Choisir la langue

Priorité 2	Priorité 4
CU06 S'inscrire	CU15 Accéder à l'interface d'administration
CU07 S'authentifier	CU16 Surveiller les indicateurs de performance du site
CU08 Gérer la fiche Utilisateur	CU17 Ajouter des types d'espaces
CU09 Gérer la fiche d'une annonce	CU18 ajouter des équipements additionnels

4.2 priorité 1

CU01 Visualiser la page d'accueil : Cette action correspond à la présentation de la page principale du site BoxToGo lorsque l'utilisateur accède au site via un lien spécifique.
CU02 Voir le détail d'une annonce : L'utilisateur a la possibilité, à n'importe quel moment, de lire la description et de visualiser la photo liée à une catégorie d'espaces.
CU03 Recherche d'un espace de stockage : Cette fonctionnalité offre la facilité de localiser une annonce en utilisant un filtre basé sur le type de bien, avec l'option d'affiner la recherche en indiquant le code postal ou la ville. Il est également possible d'intégrer un filtre supplémentaire pour sélectionner le statut du propriétaire, qu'il soit particulier ou professionnel.
CU04 Visualiser la fiche détaillée d'un espace : L'utilisateur a la possibilité de consulter la fiche descriptive d'une annonce. Cette fiche comprend une galerie de photos, une carte pour localiser le bien, une énumération des équipements disponibles sur le lieu, ainsi qu'un tableau regroupant les commentaires et les évaluations que le bien a reçus.
CU05 Examiner la fiche détaillée d'un propriétaire : L'utilisateur peut accéder à la fiche informative d'un propriétaire. Cette fiche, en plus des informations de base, contient la liste des annonces qui lui sont attribuées ainsi qu'un formulaire pour le contacter.

4.3 priorité 2

CU06 S'inscrire : Tout utilisateur, qu'il soit simple ou propriétaire, peut s'inscrire sur le site. L'inscription, que ce soit en tant qu'utilisateur simple ou en tant que propriétaire, ne nécessite pas de confirmation ; le nouvel utilisateur n'a pas besoin de valider son inscription.
CU07 S'authentifier : L'utilisateur se connecte au site en saisissant son adresse e-mail et le mot de passe de son choix. Il a également la possibilité de s'authentifier via Google.
CU08 Gérer la fiche Utilisateur : L'utilisateur a la possibilité de mettre à jour ses propres informations contenues dans sa fiche utilisateur.
CU09 Gérer la fiche d'une annonce : Le propriétaire dispose de la possibilité de mettre à jour son annonce en modifiant divers éléments tels que le titre, la description, le prix, les catégories, l'adresse et les équipements associés au bien.

4.4 priorité 3

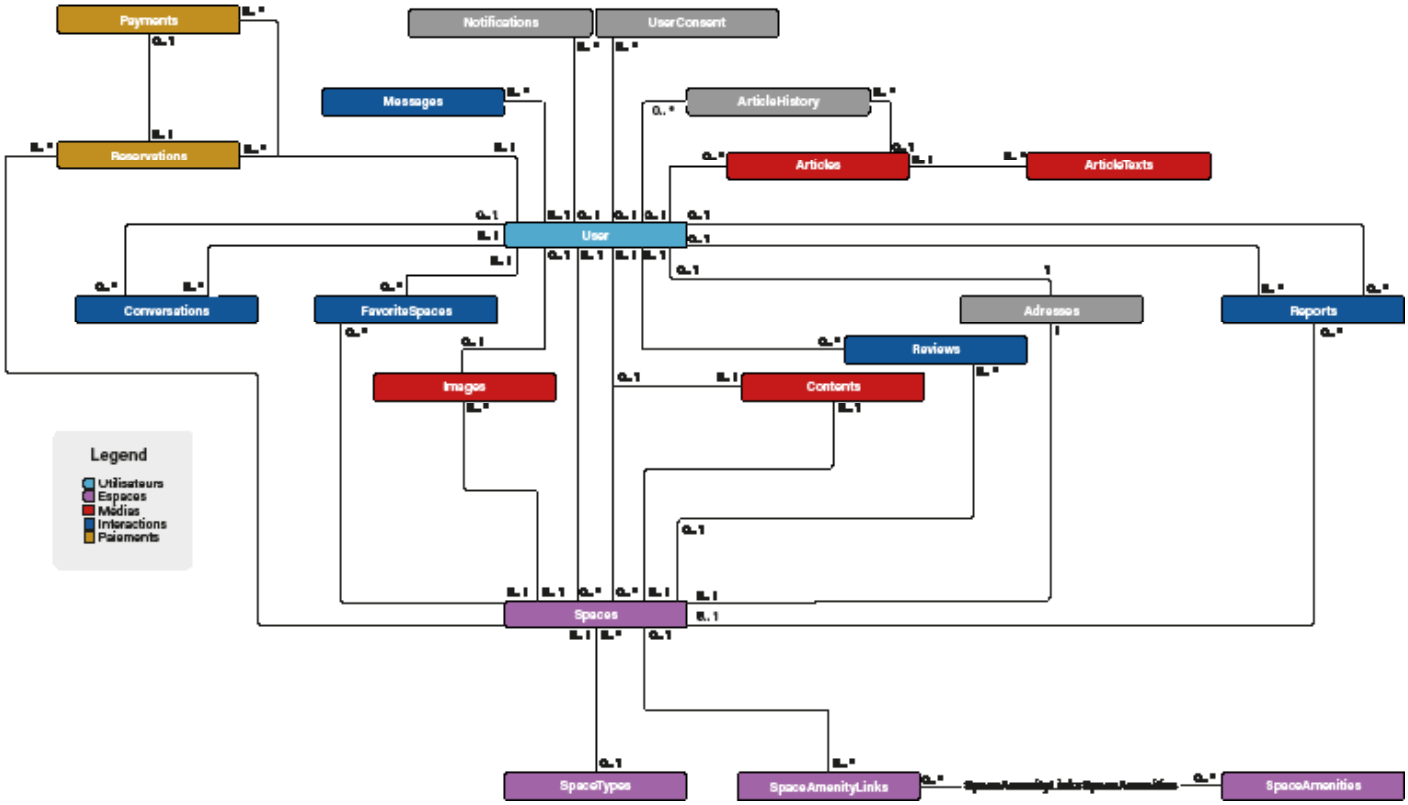
CU10 Visualiser la localisation sur une carte : L'utilisateur a la possibilité de voir l'emplacement géographique de l'adresse du bien sur une carte.
CU11 Contacter un propriétaire : Pour entrer en contact avec un propriétaire, l'utilisateur utilise un formulaire standard disponible sur la page de profil du propriétaire. Après l'envoi du premier message, l'utilisateur peut poursuivre la conversation via l'onglet «Messages».
CU12 Changer de mot de passe : Une fois authentifié, l'utilisateur a la possibilité de modifier son mot de passe. Pour minimiser les risques d'erreur, il devra saisir le nouveau mot de passe deux fois.
CU13 Gestion des annonces par le propriétaire : Le propriétaire dispose des options pour ajouter une nouvelle annonce, en supprimer une déjà existante ou en effectuer des modifications.
CU14 Sélection de la langue : Que l'utilisateur soit authentifié ou non, il doit pouvoir modifier la langue d'affichage du site à tout moment.

4.5 priorité 4

CU15 Accès à l'interface administrative : Possibilité d'accéder à un panneau d'administration.
CU16 Suivi des métriques de performance : L'administrateur peut consulter divers indicateurs pour évaluer les performances du site.
CU17 Intégration de nouveaux types d'espaces : L'administrateur dispose de l'option d'ajouter de nouveaux types d'espaces à la plateforme.
CU18 Ajout d'équipements supplémentaires : L'administrateur a la capacité d'intégrer de nouveaux équipements qui peuvent être inclus dans la fiche signalétique des biens sur le site.

5. Description de la base de données

5.1. Schéma conceptuel



5.1.1 User

Un utilisateur est une personne ayant créé un compte sur la plateforme, à qui est attribué le rôle de «ROLE_USER». S'il choisit de mettre un espace en location, il obtient le statut de propriétaire et reçoit le rôle «ROLE_OWNER». Un autre rôle, «ROLE_ADMIN», est spécifiquement réservé aux administrateurs de la plateforme.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
email	VARCHAR	Obligatoire
password	VARCHAR	Obligatoire
roles	VARCHAR	ENUM
createAt	DATETIME_IMMUTABLE	Obligatoire
givenName	VARCHAR	Obligatoire string(255)
familyName	VARCHAR	Obligatoire string(255)
birthDate	DATETIME_IMMUTABLE	Obligatoire
picture	VARCHAR	Nullable string(255)
status	VARCHAR	Nullable string(255)
gender	VARCHAR	Nullable string(255)
phoneNumber	VARCHAR	Nullable string(255)
language	VARCHAR	Obligatoire string(255) ('FR', 'EN', 'DE')
appearance	VARCHAR	Obligatoire string(255)
preference	ARRAY	Nullable
googleId	VARCHAR	Nullable string(255)
failedAuthCount	INTEGER	Obligatoire, auto-incrémentée
adresse_id	INTEGER	Clé étrangère vers la table `Addresses`
image_id	INTEGER	Clé étrangère vers la table `ImageSpaces`
content_id	INTEGER	Clé étrangère vers la table `Contents`

5.1.2 SpaceType

La table « SpaceTypes » permet à un administrateur d'ajouter de nouveaux types d'espaces que la plateforme peut accueillir. Elle contient un champ pour la désignation en anglais et un autre pour celle en français.

Nom	Types de données	Contraintes
id	INTEGER	Clé primaire, auto-incrémentée
nameEn	VARCHAR	Nullable string(255)
nameFr	VARCHAR	Nullable string(255)

5.1.3 SpaceAmenities

La table permet à un administrateur d'ajouter de nouveaux équipements, lesquels seront intégrés à une liste afin d'aider les propriétaires à détailler les caractéristiques de leur espace.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
nameEn	VARCHAR	Nullable string(255)
nameFr	VARCHAR	Nullable string(255)

5.1.4 Spaces

La table « Spaces » sert à intégrer un nouvel espace sur la plateforme. Elle entretient deux relations avec la table « User » afin de distinguer entre le propriétaire et le locataire de l'espace.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
price	DECIMAL	Obligatoire
surface	INTEGER	Obligatoire
entryWidth	INTEGER	Nullable
entryLength	INTEGER	Nullable
floorLevel	VARCHAR	Nullable string(255)
conditionStatus	VARCHAR	Nullable string(255)
availabilityStartDate	DATETIME_IMMUTABLE	Obligatoire
availabilityEndDate	DATETIME_IMMUTABLE	Obligatoire
status	VARCHAR	Obligatoire string(255)
isPublished	BOOLEAN	Obligatoire
reference	VARCHAR	Obligatoire string(255)
type_id	INTEGER	Clé étrangère vers la table `SpaceTypes`
adresse_id	INTEGER	Clé étrangère vers la table `Addresses`
content_id	INTEGER	Clé étrangère vers la table `Contents`
owned_by_user_id	INTEGER	Clé étrangère vers la table `User`
rented_by_user_id	INTEGER	Clé étrangère vers la table `User`

5.1.5 Addresses

Pour le moment, la table « Addresses »contient les champs suivants : country, city, street, streetNumber, et postalCode. Les utilisateurs sont actuellement libres de les remplir à leur guise. Cependant, cette approche est susceptible d'évoluer lors des prochaines mises à jour de la plateforme.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
country	VARCHAR	Obligatoire string(50)
city	VARCHAR	Obligatoire
street	VARCHAR	Obligatoire
streetNumber	VARCHAR	Obligatoire string(20)
postalCode	VARCHAR	Obligatoire string(20)

5.1.6 Images

La table « Images » est destinée à conserver à la fois l'image de profil d'un utilisateur et la galerie d'images associée à une annonce. Par défaut, la photo de profil est configurée avec un sortOrder de 21. Afin d'éviter toute confusion avec cette dernière, la galerie d'images d'une annonce est limitée à 20 images.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
imagePath	VARCHAR	Obligatoire string(255)
createAt	DATETIME_IMMUTABLE	Obligatoire
sortOrder	INTEGER	Obligatoire
spaces_id	INTEGER	Clé étrangère reliant à la table `Spaces`

5.1.7 SpaceAmenityLinks

Cette table sert de liaison entre « Spaces » et « SpaceAmenities » afin d'indiquer tous les équipements présents dans l'espace concerné.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
spaces_id	INTEGER	Clé étrangère reliant à la table `Spaces`

5.1.8 Conversations

Cette table est destinée à gérer la discussion entre deux utilisateurs. Elle entretient deux relations avec la table User dans le but de distinguer l'expéditeur du message de son destinataire.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
content	TEXT	Nullable
sent_by_user_id	INTEGER	Clé étrangère reliant à la table `User`
received_by_user_id	INTEGER	Clé étrangère reliant à la table `User`

5.1.9 FavoriteSpaces

Cette table offre la possibilité aux utilisateurs d'ajouter un espace à leurs favoris.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
user_id	INTEGER	Clé étrangère reliant à la table `User`
space_id	INTEGER	Clé étrangère reliant à la table `Spaces`

5.1.10 Reviews

Cette table permet aux utilisateurs de laisser un avis sur un espace. Dans les prochaines mises à jour, une contrainte sera ajoutée pour limiter cette fonctionnalité aux utilisateurs ayant préalablement réservé l'espace en question.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
rating	INTEGER	Nullable
comment	TEXT	Obligatoire
spaces_id	INTEGER	Clé étrangère reliant à la table `Spaces`
user_id	INTEGER	Clé étrangère reliant à la table `User`

5.1.11 Reservations

Lorsqu'un utilisateur souhaite réserver un espace, une vérification initiale est effectuée via la table « Reservations » pour s'assurer que l'espace n'est pas déjà réservé. Si tel est le cas, le statut est modifié en «failed». Autrement, la procédure continue vers la table « Payments ».

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
dateStart	DATETIME_IMMUTABLE	Obligatoire
dateEnd	DATETIME_IMMUTABLE	Obligatoire
reference	VARCHAR	Obligatoire string(255)
status	VARCHAR	Obligatoire string(255)
price	VARCHAR	Obligatoire string(255)
user_id	INTEGER	Clé étrangère reliant à la table `User`
space_id	INTEGER	Clé étrangère reliant à la table `Spaces`
payment_id	INTEGER	Clé étrangère reliant à la table `Payments`

5.1.12 Payments

Cette table gère les paiements sur la plateforme en utilisant Stripe en mode «checkout». Autrement dit, le processus de paiement est délégué à Stripe, et nous récupérons ensuite un token et un identifiant pour confirmer que la transaction s'est bien déroulée.

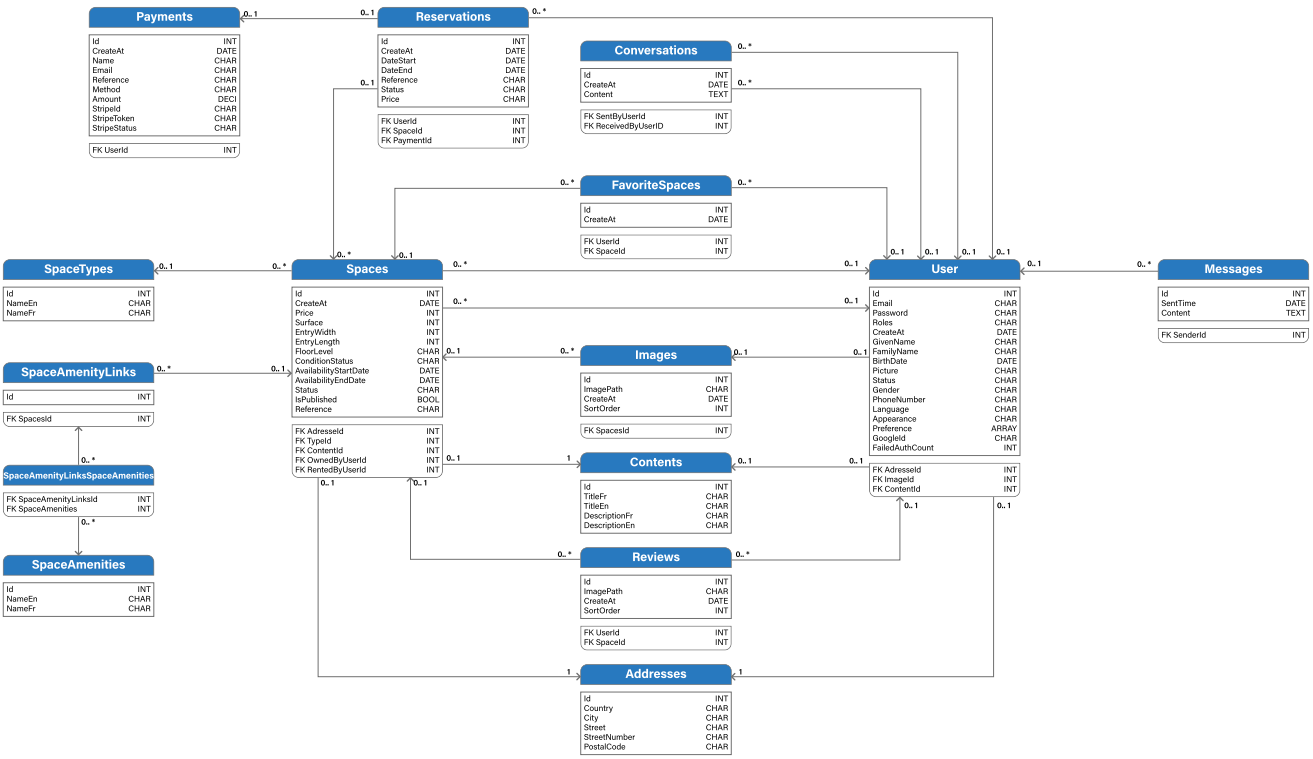
Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
createAt	DATETIME_IMMUTABLE	Obligatoire
name	VARCHAR	Obligatoire string(255)
email	VARCHAR	Obligatoire string(255)
reference	VARCHAR	Obligatoire string(255)
method	VARCHAR	Obligatoire string(255) ex. carte de crédit, PayPal
amount	DECIMAL	Obligatoire
stripeId	VARCHAR	Obligatoire string(255)
stripeToken	VARCHAR	Obligatoire string(255)
stripeStatus	VARCHAR	Obligatoire string(255)
user_id	INTEGER	Clé étrangère reliant à la table `User`

5.1.13 Contents

La table Contents a pour but d'attribuer un titre et une description à un espace. Chaque champ comporte une version en français et une en anglais, permettant ainsi au propriétaire de gérer le multilinguisme. Cette table est également utilisée pour fournir une description associée à l'utilisateur.

Nom	Types de données	Contraintes
ID	INTEGER	Clé primaire, auto-incrémentée
title_fr	VARCHAR	Obligatoire string(200)
title_en	VARCHAR	Obligatoire string(200)
title_ne	VARCHAR	Obligatoire string(200)
description_fr	TEXT	Obligatoire
description_en	TEXT	Obligatoire
description_ne	TEXT	Obligatoire

5.1.14 Schéma physique



6. Présentation des problèmes et solutions envisagées

6.1. Problèmes rencontrés

6.1.1 Formulaires

Mon premier problème a surgi lorsque j'ai tenté de créer le formulaire permettant d'ajouter une annonce sur la plateforme. La difficulté résidait dans la nécessité de gérer entre quatre et cinq entités en relation. Avant de procéder à l'enregistrement, il était impératif de m'assurer que toutes les données étaient correctes et cohérentes.

6.1.2 Adobe XD

Ensuite, ayant commencé à coder le site sans disposer de maquettes préalables, j'ai rapidement constaté la difficulté de travailler sans une vision claire de chaque page. À plusieurs reprises, j'ai dû apporter des modifications, pour ensuite changer d'avis. Cette approche m'a donné l'impression de stagner, plutôt que de progresser de manière efficace.

6.1.3 Messagerie

La mise en place du système de messagerie a représenté un défi particulier pour moi. Initialement, je n'avais aucune expertise dans ce domaine et j'ignorais s'il existait des outils spécifiques ou des bonnes pratiques pour aborder ce type de fonctionnalité.

6.2. Solutions proposées

6.2.1 Formulaires

Pour résoudre le problème lié à l'ajout d'une annonce via un formulaire impliquant plusieurs entités, j'ai opté pour la création d'un modèle unique qui rassemble les cinq formulaires correspondant à chaque entité nécessaire. L'avantage de cette méthode est que tous les formulaires sont consolidés en une seule interface, permettant ainsi une soumission globale via un unique bouton.

```php
class FormAddNewSpaceModel {
    private Spaces $space;
    private Addresses $adresse;
    private SpaceAmenityLinks $amenity;
    private Images $image;
    private Contents $content;

    public function __construct()
    {
        $this->space = new Spaces();
        $this->adresse = new Addresses();
        $this->amenity = new SpaceAmenityLinks();
        $this->image = new Images();
        $this->content = new Contents();
    }
```

Pour améliorer l'expérience utilisateur, j'ai ensuite utilisé JavaScript pour diviser ce formulaire global en plusieurs étapes. Des contraintes ont été mises en place entre chaque étape du formulaire pour garantir que les informations essentielles soient correctement saisies par l'utilisateur. Pour mettre davantage l'accent sur ces champs cruciaux, j'ai ajouté la classe 'focus' à ceux que j'ai jugés obligatoires.

```
{{ form_row(form.type, {'row_attr': {'class': 'listing__template-step__select'}, 'attr': {'class': 'focus'}}) }}
```

À l'aide de JavaScript, j'ai mis en place un mécanisme qui empêche l'utilisateur de passer à l'étape suivante tant que les champs marqués avec la classe 'focus' ne sont pas remplis.

```
const focusElements = currentSection.querySelectorAll('.focus');
let allFieldsFilled = true; // indicateur pour vérifier si tous les champs sont remplis

for (let focusedElement of focusElements) {
    if (!focusedElement.value) {
        allFieldsFilled = false; // mettez l'indicateur à false si un champ est vide
        console.log('Élément avec la classe focus trouvé et est vide:', focusedElement);

        focusedElement.classList.add('highlight-focus');

        setTimeout(() => {
            focusedElement.classList.remove('highlight-focus');
        }, 1000);

        break;
    }
}
```

La soumission finale n'est alors possible qu'après la validation de toutes ces étapes, ce qui est également géré en JavaScript.

```
// Vérifiez s'il y a une section suivante
const nextSectionIndex = Array.from(sections).indexOf(currentSection) + 1;
if (nextSectionIndex < sections.length) {
    sections[nextSectionIndex].classList.add('show');
} else {
    // Si on est à la dernière section, soumettre le formulaire
    form.submit();
}
```

6.2.2 Adobe XD

Pour pallier le problème de développement sans plan directeur, j'ai utilisé Adobe XD pour concevoir une maquette du site. Cependant, cette démarche m'a révélé un autre défi majeur : des lacunes dans la conception de la base de données. En cours de maquettage, j'ai découvert des éléments auxquels je n'avais pas pensé auparavant. Cette expérience m'a enseigné l'importance de consacrer davantage de temps à la phase de conception de la base de données avant de plonger dans le code. Je prendrai cela en compte pour mes futurs projets.

6.2.3 Messagerie

Pour l'implémentation du système de messagerie, j'ai découvert qu'il y a diverses méthodes pour le réaliser. Étant donné que c'est ma première tentative dans ce domaine, j'ai choisi d'adopter une approche rudimentaire. Je vais donc utiliser AJAX pour actualiser la page toutes les 10 secondes afin de vérifier la présence de nouveaux messages.

Pour structurer efficacement le code, j'ai envisagé de diviser la logique en trois routes distinctes :

Interface de Messagerie: La première route sert à afficher l'interface utilisateur globale de la messagerie.

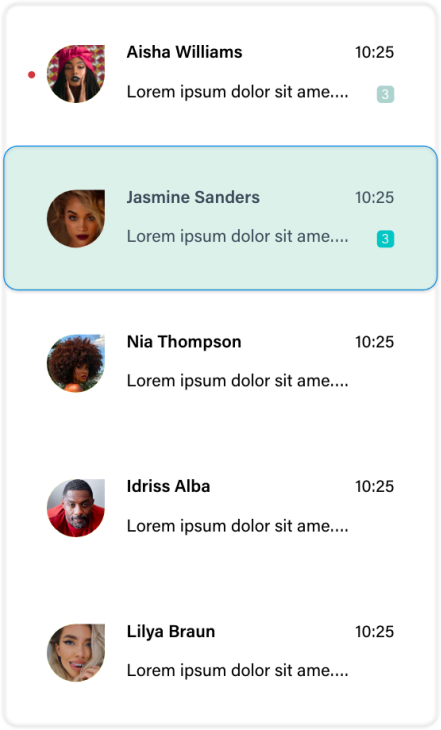
```
#[Route('/talks', name: 'app_talks')]
```

Contenu des Conversations: La deuxième route a pour but de récupérer et d'afficher les messages associés à une conversation spécifique.

```
#[Route('/talks/{id}', name: 'app_current_talk')]
```

Vérification de nouveaux messages : La dernière route a pour but de vérifier régulièrement s'il y a des messages récents dans la conversation en cours. Ce contrôle est fait toutes les 10 secondes en utilisant une requête AJAX.

```
#[Route('/talks/{id}/fetch', name: 'app_fetch_talks', methods: ['POST'])]
```



Pour éviter la redondance de code dans deux routes différentes concernant l'affichage du contenu de la barre latérale, j'ai opté pour la gestion de cette logique via un EventSubscriber. J'ai développé deux méthodes qui récupèrent tous les messages de l'utilisateur connecté. La première méthode a pour fonction de trier les conversations en les regroupant, afin de faciliter la navigation au sein des échanges. La deuxième méthode récupère tous les interlocuteurs par groupe pour les afficher dans la barre latérale.

```php
public function onKernelRequest(RequestEvent $event): void
    {
        $route = $event->getRequest()->attributes->get('_route');
        $token = $this->tokenStorage->getToken();

        if ($token) {
            $currentUser = $token->getUser();

            if (in_array($route, SidebarMessageSubscriber::ROUTES)) {
                // Récupération de tous les messages reçus par l'utilisateur actuel
                $receivedMessages = $this->repository->findBy(['receivedByUser' => $currentUser]);
                // Récupération de tous les messages envoyés par l'utilisateur actuel
                $sentMessages = $this->repository->findBy(['sentByUser' => $currentUser]);

                $conversations = $this->groupTalksByConversation($receivedMessages, $sentMessages);

                $interlocutors = $this->transformConversationsForSidebar($conversations, $currentUser->getId());
                // dd($interlocutors);
                $this->twig->addGlobal('interlocutors', $interlocutors);
            }
        }
    }
```

Dans ma première route, je commence par vérifier si l'utilisateur a déjà eu des conversations. Si c'est le cas, j'obtiens l'identifiant de la dernière personne avec qui il a échangé des messages afin de le rediriger vers la deuxième route. En revanche, s'il n'a pas encore eu de conversations, l'interface de dialogue s'affiche avec un message indiquant qu'il n'y a pas encore de discussions en cours.

```php
$currentInterlocutorId = $this->redirectToLatestConversation();

if ($currentInterlocutorId) {
    return $this->redirectToRoute('app_current_talk', ['id' => $currentInterlocutorId], Response::HTTP_SEE_OTHER);
}
```

La deuxième route a pour objectif de récupérer la conversation correspondant à l'identifiant de l'utilisateur reçu en paramètre.
Quant à la troisième route, elle utilise une requête AJAX pour effectuer une vérification toutes les 10 secondes afin de voir si l'utilisateur a reçu un nouveau message. Si c'est le cas, la page est automatiquement rafraîchie. Cette dernière fonctionnalité sera améliorée ultérieurement : au lieu de rafraîchir toute la page, le nouveau message sera récupéré et injecté directement dans la vue.

```php
$conversations = $this->getConversationsWithInterlocutor($currentInterlocutor);

$lastConversation = end($conversations['conversations']);
$lastMessageTime = $lastConversation->getCreateAt();

// Date actuelle
$currentTime = new \DateTimeImmutable('now');

// Calculer la différence en secondes entre la dernière conversation et la date actuelle
$diffInSeconds = $currentTime->getTimestamp() - $lastMessageTime->getTimestamp();

// Comparer la différence en secondes
if ($diffInSeconds < 10) {
    return $this->json(['message' => 'new message']);
```

7. Conclusion

Je suis fier du chemin que j'ai parcouru et des compétences que j'ai acquises jusqu'à présent. Néanmoins, je réalise que je n'en suis qu'au début de mon voyage en tant que développeur. Chaque jour apporte son lot d'apprentissages et ce projet m'a particulièrement éclairé sur la polyvalence requise pour être un développeur complet. Il ne s'agit pas seulement de coder; il faut également comprendre la structure d'une base de données, connaître les différentes méthodes de résolution de problèmes et être capable de choisir l'approche la plus optimale parmi les nombreuses disponibles. De plus, des outils de gestion de projet comme Trello ne sont pas simplement des options, ils sont essentiels pour mener à bien toute tâche complexe. En somme, être un bon développeur exige une palette de compétences bien plus large que je ne l'avais imaginé, et j'ai hâte de continuer à me perfectionner dans tous ces domaines.