



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de **HONORIS UNITED UNIVERSITIES**

Thème

Conception et réalisation d'une application web de Réservation des Hotels

MÉMOIRE DE FIN DE MODULE

Pour l'application de Framework Django

Encadré par:

Professeur : CHAIMAE

AZROUMAH LI

Réalisé par :

Cherkaoui Salma

Aya Kammah

2023-2024

Remerciements

Nous tenons en premier lieu à exprimer notre profonde gratitude à Allah, le Très-Haut, pour nous avoir octroyé la santé, le courage, et la patience nécessaires pour mener à bien ce projet.

Nous souhaitons également adresser nos plus sincères remerciements à nos chers parents pour leur immense soutien et sacrifice tout au long de notre parcours de formation.

Nous exprimons également notre gratitude envers notre professeur et tuteur pédagogique, Madame CHAIMAE AZROUMAHLLI, pour sa soutien et sa encouragement infaillibles tout au long de ce projet de fin de fin de module.

Nous exprimons également toute notre gratitude envers les membres du jury qui ont manifesté un grand intérêt pour notre travail et qui nous feront l'honneur de l'évaluer.

Enfin, nous souhaitons adresser nos remerciements les plus sincères à tous les enseignants du département IIR pour leur précieuse collaboration durant la semestre.

Nous tenons à remercier toutes les personnes qui nous ont aidés de près ou de loin durant notre parcours de formation.

Table des matières

Remerciements.....	2
Chapitre 1 : Présentation de projet :.....	6
Contexte du Projet :.....	6
Chapitre 2 : Analyse et conception informatiques.....	7
Introduction.....	7
Diagramme des cas d'utilisation :.....	8
Diagramme de séquences	8
Diagramme de séquence d'inscription à site	8
Diagramme de Classes	9
Représentation des tables en Base de données (MySQL).....	10
Conclusion	10
Chapitre 3 : Mise en œuvre informatique	11
Introduction.....	11
MVT.....	11
Technologies choisies	12
Architectures Applicative et Technique	13
.....	13
Hotel-Reservation-Django-React :	13
Contient l'ensemble du projet intégrant le backend (Django) et le frontend (React)	13
Accounts :.....	13
Gère les comptes utilisateurs.....	13
Les urls utilisé :.....	13
Hotel_app :	14

les urls:	14
<i>les vues</i> :	14
<i>lookup_fieled</i> :	14
<i>serialisers</i> :	16
<i>Django Admin</i> :	16
Hotel_reservation_site :	16
Projet Django principal qui contient les URLs des applications	16
Présentation de sessions d'utilisation de notre projet.....	20
Conclusion	24

Introduction générale

Dans un monde où la mobilité est devenue la norme, notre projet d'application de réservation d'hôtels s'inscrit dans une démarche d'optimisation de l'expérience de voyage. En exploitant les puissants frameworks Django et React, notre application vise à simplifier et à rendre plus convivial le processus de réservation, offrant aux utilisateurs une interface moderne et réactive pour explorer et réserver des hébergements en toute facilité. Notre objectif est de repousser les limites de l'efficacité dans le domaine de la réservation d'hôtels, en combinant la robustesse côté serveur de Django avec la réactivité côté client de React pour créer une solution innovante et adaptée aux besoins actuels des voyageurs.

Chapitre 1 : Présentation de projet :

Contexte du Projet :

Notre projet s'inscrit dans le domaine du tourisme et de l'hôtellerie, offrant une solution moderne de réservation d'hôtels. Face à la croissance constante du secteur du voyage, il devient impératif de proposer une plateforme conviviale et efficace permettant aux utilisateurs de trouver et de réserver des hébergements adaptés à leurs besoins. Nous visons à simplifier ce processus en créant une application intuitive qui facilite la découverte d'hôtels, la réservation de chambres et la gestion efficace de l'inventaire pour les administrateurs.

Fonctionnalités Clés du Projet

Consultation des Hôtels :

Les utilisateurs peuvent parcourir les hôtels disponibles, visualiser leurs caractéristiques, et obtenir des informations détaillées sur les chambres offertes.

Authentification et Création de Compte :

Avant de pouvoir réserver une chambre, les utilisateurs doivent s'authentifier et créer un compte. Cela garantit une expérience personnalisée et sécurisée.

Réservation de Chambres :

Les utilisateurs authentifiés ont la possibilité de réserver une chambre en fonction de leur disponibilité, de leurs préférences de catégorie, de prix et de capacité.

Filtrage des Chambres :

Les utilisateurs peuvent filtrer les chambres disponibles en fonction de critères tels que la catégorie, le prix et la capacité, facilitant ainsi la recherche de l'hébergement idéal.

Gestion des Réservations :

Les utilisateurs peuvent consulter et gérer leurs réservations en cours, fournissant une visibilité sur leurs séjours planifiés.

Check-out par l'Admin :

Seul l'administrateur peut effectuer le check-out des chambres, rendant les chambres disponibles pour de nouvelles réservations une fois le séjour du client terminé.

Ces fonctionnalités sont conçues pour offrir une expérience complète et fluide aux utilisateurs tout en simplifiant la gestion pour les administrateurs de l'application.

Chapitre 2 : Analyse et conception informatiques

Introduction

Le chapitre d'Analyse et Conception Informatiques constitue une étape cruciale dans le processus de développement d'un système informatique. Cette phase permet de définir les besoins et les fonctionnalités requises pour répondre aux exigences des utilisateurs et de concevoir une solution technique adéquate.

Dans ce chapitre, nous allons décrire les spécifications fonctionnelles et techniques du système en se basant sur l'étude préalable effectuée et en prenant en compte les besoins des acteurs impliqués. Nous allons également identifier les différents acteurs et leurs rôles, ainsi que les différents scénarios d'utilisation du système. Pour cela, nous allons utiliser plusieurs outils et diagrammes de modélisation tels que les use cases, les diagrammes de séquence et le diagramme de classes.

L'objectif de ce chapitre est de fournir une vue d'ensemble claire et détaillée de la solution envisagée afin de guider les étapes de développement et de mise en place du système.

Diagramme des cas d'utilisation :

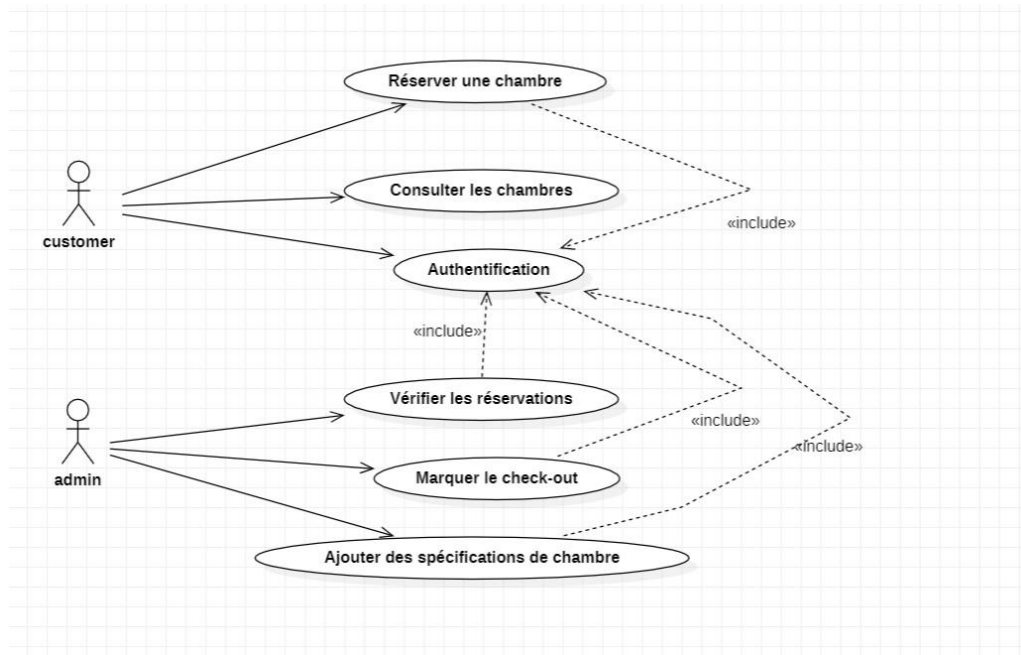


Figure 1: Diagramme de cas d'utilisation de l'application HOTEL

Diagramme de séquences

Diagramme de séquence d'inscription à site

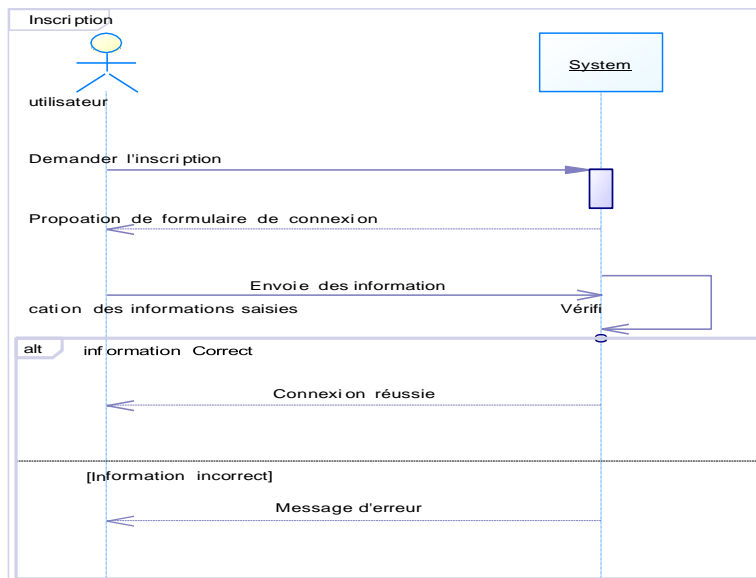


Figure 2: Diagramme de séquence d'inscription dans le site HOTEL

Diagramme de Classes

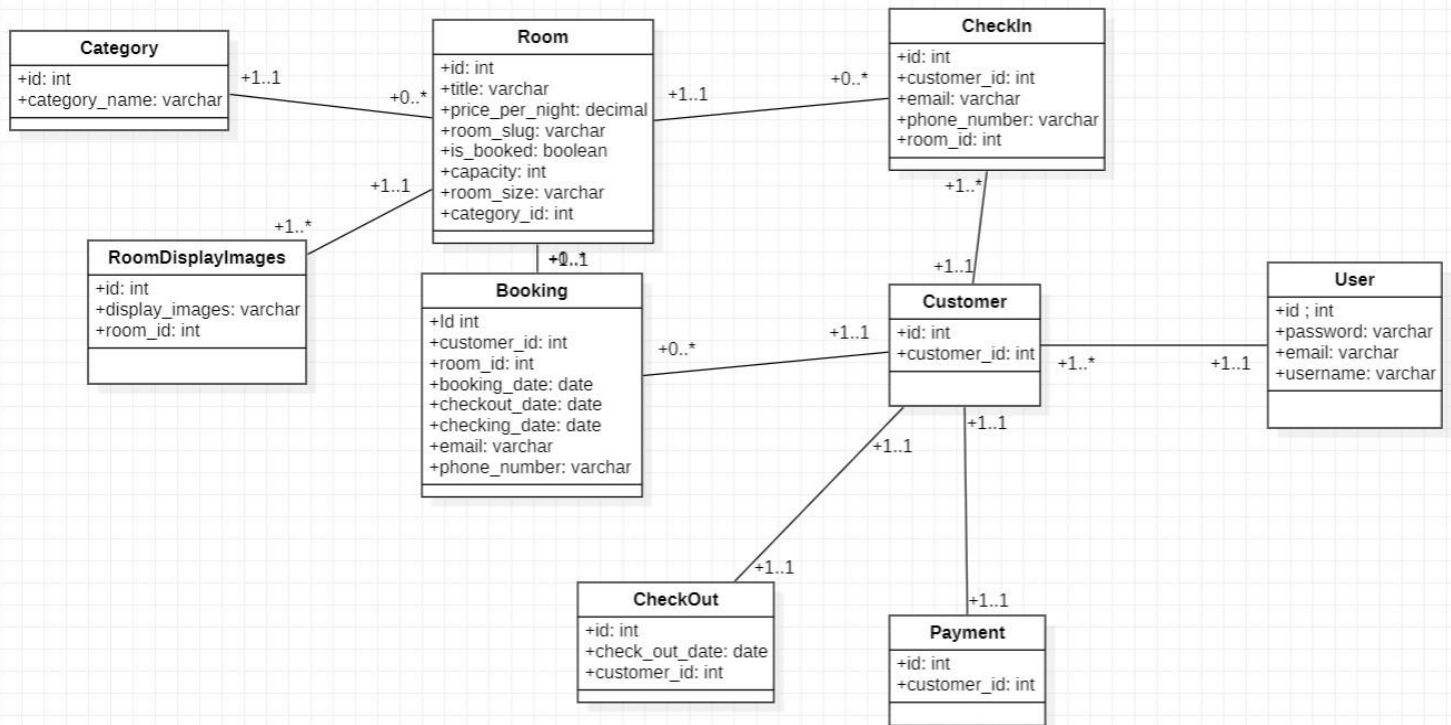


Figure 3: Diagramme de Classe d'appliacion HOTEL

Représentation des tables en Base de données (MySQL)

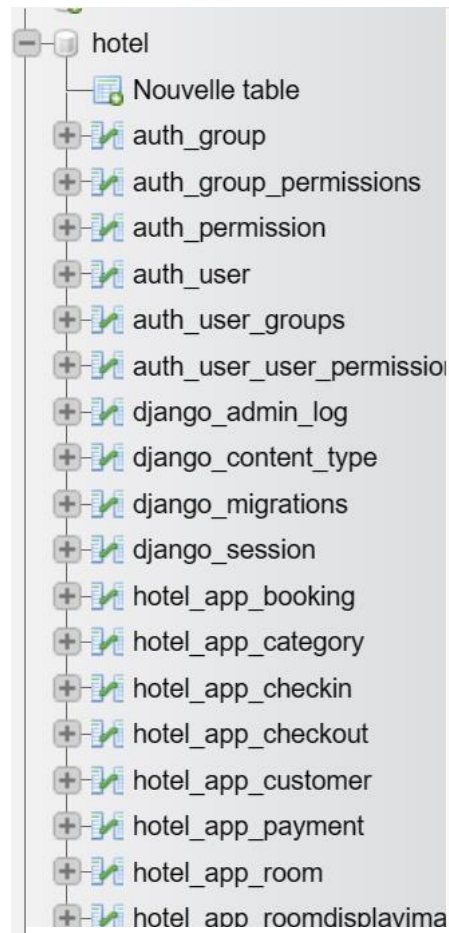


Figure 4: les tables de bases de données

Conclusion

En conclusion, l'analyse et la conception informatiques sont des étapes cruciales dans la réalisation d'un projet informatique. À travers la définition des spécifications, la modélisation des acteurs et des scénarios, et la création de différents diagrammes, nous avons pu identifier les besoins et les fonctionnalités attendus du système. Cela permettra une meilleure compréhension des attentes des utilisateurs et une mise en place plus efficace du projet. Le diagramme de classes en particulier, permettra de définir les classes du système et les relations entre elles, pour une implémentation efficace et modulaire. L'ensemble de ces travaux permettra de fournir une base solide pour la phase de développement du projet.

Chapitre 3 : Mise en œuvre informatique

Introduction

Dans ce chapitre, nous allons explorer les différentes architectures applicatives et techniques ainsi que l'architecture de développement du projet, qui est basée sur le modèle MVT (Modèle-Vue-Contrôleur). Nous allons également présenter les technologies sélectionnées pour la réalisation de ce site web, notamment le Django et React

Le but de cette partie est de fournir des détails sur les choix techniques et les différentes étapes de la mise en œuvre informatique afin d'assurer la qualité et la performance du site web.

MVT

Modèle (Model) : Dans le cas de DRF, le modèle représente la logique métier et les données de l'application, généralement définies par des modèles Django qui correspondent aux tables de la base de données. Ces modèles interagissent avec la base de données pour récupérer ou stocker des informations.

Vue (View) : La vue dans DRF correspond aux vues Django. Ces vues gèrent la logique métier de l'application en traitant les requêtes HTTP, récupérant ou modifiant les données via les modèles, et renvoyant les résultats sous forme de réponses HTTP.

Template (Template) : Bien que la notion de "template" soit plus centrée sur la génération de pages HTML complètes dans le contexte de Django traditionnel, dans le cas de DRF, les templates peuvent être interprétés comme la manière dont les données sont présentées sous forme de réponse, généralement sous forme de données JSON pour les API REST.

Dans notre cas, avec React en tant que framework frontend, nous maintenons la séparation des préoccupations en utilisant React pour la vue, tandis que Django Rest Framework (MVT) gère le modèle et la logique de contrôle.

Cela permet une architecture claire et modulaire, facilitant le développement, la maintenance et l'évolutivité de notre application.

Technologies choisies



MySQL est un système de gestion de base de données relationnelle open source qui utilise le langage SQL. Il permet de stocker et de récupérer des données structurées de manière efficace, offrant des fonctionnalités telles que la création de tables, la gestion des relations, et la possibilité de réaliser des opérations avancées comme les jointures.



Django Rest Framework (DRF) est une extension de Django qui simplifie la création d'API REST. Il offre des fonctionnalités telles que la sérialisation des données, la gestion des autorisations, et facilite le développement d'applications web avec des interfaces API efficaces.



React est une bibliothèque JavaScript qui facilite la création d'interfaces utilisateur interactives et dynamiques pour les applications web. Elle se distingue par l'utilisation de composants réutilisables et une approche déclarative, permettant des mises à jour de l'interface de manière efficace.



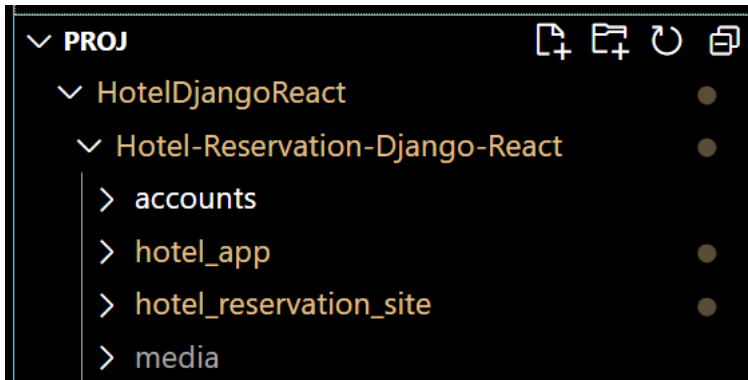
Git est un système de contrôle de version pour suivre les modifications dans le code source d'un projet. Il permet une gestion efficace des versions, facilite la collaboration entre les développeurs et offre des fonctionnalités comme le suivi des modifications et la gestion des branches.



GitHub est également une plateforme de gestion de code source très populaire. Elle permet de stocker des dépôts de code et de travailler en collaboration avec d'autres développeurs. GitHub est souvent utilisé pour les projets open source et offre une interface conviviale pour l'exploration, la contribution et le partage de code.

Architectures Applicative et Technique

La structure de projet coté Backend :

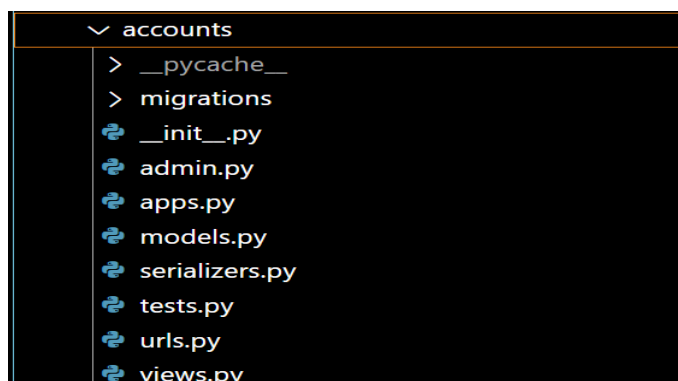


Hotel-Reservation-Django-React :

Contient l'ensemble du projet intégrant le backend (Django) et le frontend (React)

Accounts :

Gère les comptes utilisateurs

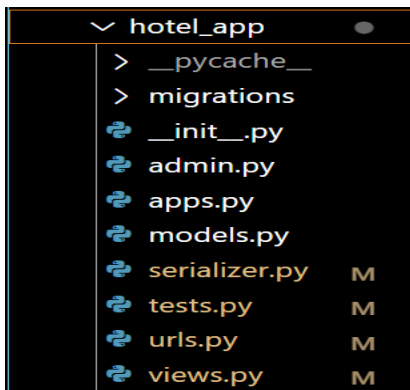


Les urls utilisé :

```
urlpatterns = [
    path('register/', UserView.as_view()), # l'url pour crée un compte d'utilisateur

    path('login/', MyTokenObtainPairView.as_view(), name='token_obtain_pair'), # pour
connecter un utilisateur déjà existe
    path('refresh_token/', jwt_views.TokenRefreshView.as_view(),
name='token_refresh'),
]
```

Hotel app :



les urls:

```
urlpatterns = [
    path('get_room_list/', RoomView.as_view(), name="room_list"),
    path('get_a_room_detail/<str:room_slug>', RoomDetailView.as_view(),
name="single_room"),
    path('book/', BookingCreateAPIView.as_view(), name='book_room'),
    path('checkout/', CheckoutView.as_view(), name="checkout"),
    path('get_current_checked_in_rooms/', CheckedInView.as_view(),
name="checked_in_rooms"),
    path('getcategories/', get_categories, name='get_categories'),
]
```

les vues :

par exemple la vue RoomDetailView :

```
class RoomDetailView(RetrieveAPIView):
    serializer_class = RoomSerializer
    queryset = Room.objects.all()
    lookup_field = 'room_slug'
```

lookup_fieled :

utilisé pour spécifier le champ qui doit être utilisé pour rechercher une instance spécifique de Room

Et pour les modèles utilisé :

```
class Room(models.Model):
    title = models.CharField(max_length=30)
    category = models.ForeignKey('Category', on_delete=models.CASCADE)
    price_per_night = models.DecimalField(max_digits=8, decimal_places=3)
    room_slug = models.SlugField()
    is_booked = models.BooleanField(default=False)
    capacity = models.IntegerField()
    room_size = models.CharField(max_length=5)
    cover_image = models.ImageField(upload_to=room_images_upload_path)
    featured = models.BooleanField(default=False)

    def __str__(self):
```

```

        return self.title

class Category(models.Model):
    category_name = models.CharField(max_length=30)

    def __str__(self):
        return self.category_name

class Customer(models.Model):
    customer = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.customer

class Booking(models.Model):
    customer = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey('Room', on_delete=models.CASCADE)
    booking_date = models.DateTimeField(auto_now_add=True)
    checking_date = models.DateTimeField(blank=True, null=True)
    checkout_date = models.DateTimeField(null=True, blank=True)
    phone_number = models.CharField(max_length=14, null=True)
    email = models.EmailField()

    def __str__(self):
        return self.customer.username

class Payment(models.Model):
    customer = models.ForeignKey('Customer', on_delete=models.CASCADE)

    def __str__(self):
        return self.customer

class CheckIn(models.Model):
    customer = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey('Room', on_delete=models.CASCADE)
    phone_number = models.CharField(max_length=14, null=True)
    email = models.EmailField(null=True)

    def __str__(self):
        return self.room.room_slug

class CheckOut(models.Model):
    customer = models.ForeignKey('Customer', on_delete=models.CASCADE)
    check_out_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.customer

```

```
class RoomDisplayImages(models.Model):
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    display_images = models.ImageField(upload_to=room_display_images_upload_path)

    def __str__(self):
        return self.room.room_slug
```

serialisers:

les serializers sont utilisés pour convertir les objets complexes, tels que les modèles Django, en formats facilement traitables comme le JSON, et vice versa , facilitant ainsi la communication entre le backend (Django) et le frontend (React) via des API REST

-exemple d'utilisation dans le modèle Room:

```
class RoomSerializer(serializers.ModelSerializer):
    category_name = serializers.CharField(source='category.category_name')

    class Meta:
        model = Room
        fields = '__all__'
```

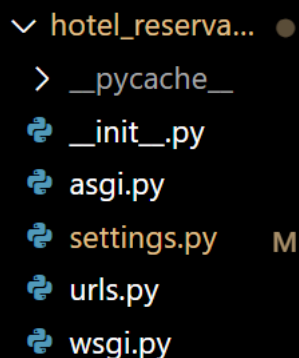
Django Admin:

Django Admin est une interface d'administration qui simplifie la gestion des données de notre application via une interface web. Les enregistrements avec `admin.site.register()` permettent de rendre nos modèles administrables dans Django Admin. Par exemple, `admin.site.register(Room, RoomAdmin)` enregistre le modèle Room, avec une personnalisation possible via RoomAdmin, pour une gestion aisée dans l'interface d'administration. De manière similaire, les autres enregistrements rendent les modèles Category, Customer, Booking, Payment, CheckIn, CheckOut, et RoomDisplayImages administrables dans Django Admin.

```
admin.site.register(Room, RoomAdmin)
admin.site.register(Category)
admin.site.register(Customer)
admin.site.register(Booking)
admin.site.register(Payment)
admin.site.register(CheckIn)
admin.site.register(CheckOut)
admin.site.register(RoomDisplayImages)
```

Hotel reservation site :

Projet Django principal qui contient les URLs des applications



```

✓ hotel_reserva... ●
  > __pycache__
  🌀 __init__.py
  🌀 asgi.py
  🌀 settings.py M
  🌀 urls.py
  🌀 wsgi.py
```


Bibliothèque essentielle :

corsheaders :

est une bibliothèque Django qui résout les problèmes de CORS (Cross-Origin Resource Sharing) en ajoutant les en-têtes CORS nécessaires aux réponses de notre application Django. Elle permet à notre backend Django d'accepter des requêtes AJAX depuis des domaines différents, facilitant ainsi l'intégration avec des frontends hébergés sur d'autres domaines.

python-decouple est une bibliothèque Python qui simplifie la gestion des paramètres de configuration en utilisant un fichier spécifique (par exemple, `.env`).

Elle permet à notre application de stocker des configurations sensibles, comme des clés secrètes ou des identifiants de base de données, en dehors du code source. Cela améliore la sécurité, la portabilité et la maintenabilité de notre application en évitant l'inclusion directe d'informations sensibles dans le code.

Les urls;

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('hotel/', include("hotel_app.urls")),
    path('api/', include("rest_framework.urls")),
    path('accounts/', include('accounts.urls')),
]
```

La structure de projet coté Frontend :

```
> node_modules
> public
> src
> venv
```

public :

Ce dossier contient les fichiers statiques qui ne changent pas fréquemment (par exemple, les images, les fichiers de favicon, etc.).

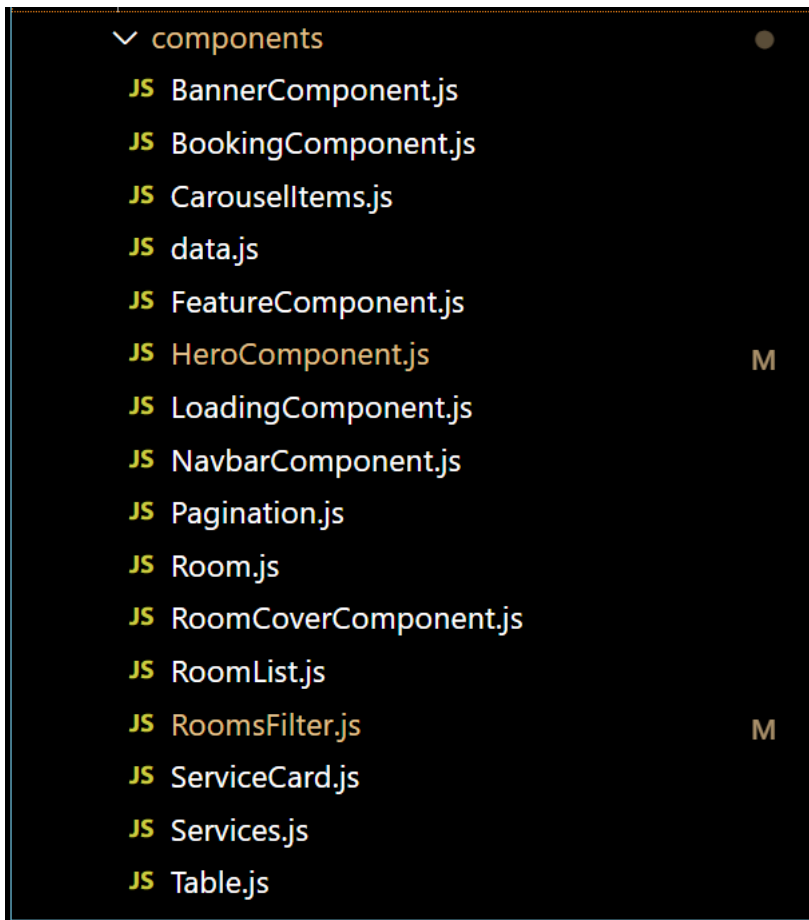
Le fichier index.html dans ce dossier est le point d'entrée de votre application React.

src :

components :

Ce dossier contient des composants réutilisables qui sont utilisés dans plusieurs parties de l'application.

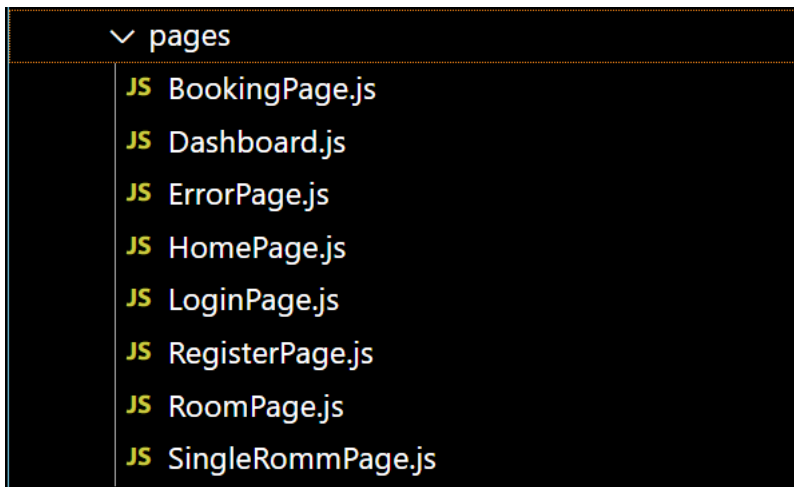
Chaque composant peut avoir son propre dossier avec un fichier JavaScript/JSX, un fichier CSS...



pages :

Ce dossier contient des composants qui représentent des pages spécifiques de votre application.

Chaque page peut avoir son propre dossier avec un fichier JavaScript/JSX, un fichier CSS.

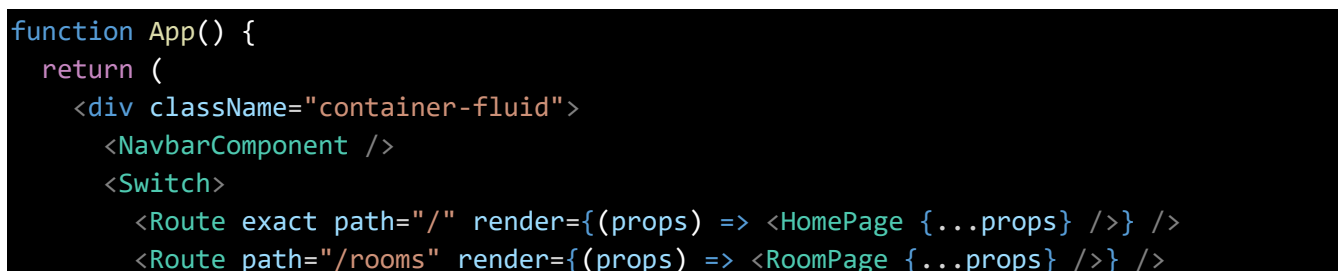


assets :

Des fichiers tels que des images, des icônes ou d'autres médias peuvent être stockés ici.

App.js :

Le composant principal qui rassemble tous les autres composants et définit la structure générale de l'application.



```

    <Route
      path="/single-room/:room_slug"
      render={(props) => <SingleRoomPage {...props} />}
    />
    <Route
      path="/book/:room_id"
      render={(props) => <BookingPage {...props} />}
    />
    <Route path="/login" render={(props) => <LoginPage {...props} />} />
    <Route
      path="/register"
      render={(props) => <RegisterPage {...props} />}
    />
    <Route path="/dashboard" render={(props) => <Dashboard {...props}/>}/>
    <Route render={(props) => <ErrorPage {...props} />} />
  </Switch>
</div>
);
}

```

index.js :

Le point d'entrée de l'application React, où le composant racine est rendu dans le DOM.

index.css :

Le fichier CSS global pour l'application.

Bibliothèque essentielle :

Axios :

```
import axios from "axios";
```

est une bibliothèque JavaScript populaire utilisée pour effectuer des requêtes HTTP depuis un navigateur ou depuis Node.js. Elle simplifie la gestion des requêtes HTTP et offre des fonctionnalités telles que la gestion des erreurs, la conversion automatique des données JSON, la prise en charge des promesses, etc. Axios est souvent utilisé dans le développement d'applications front-end, en particulier avec des frameworks comme React, pour interagir avec des API distantes.

Présentation de sessions d'utilisation de notre projet

La page d'Accueil de HOTEL

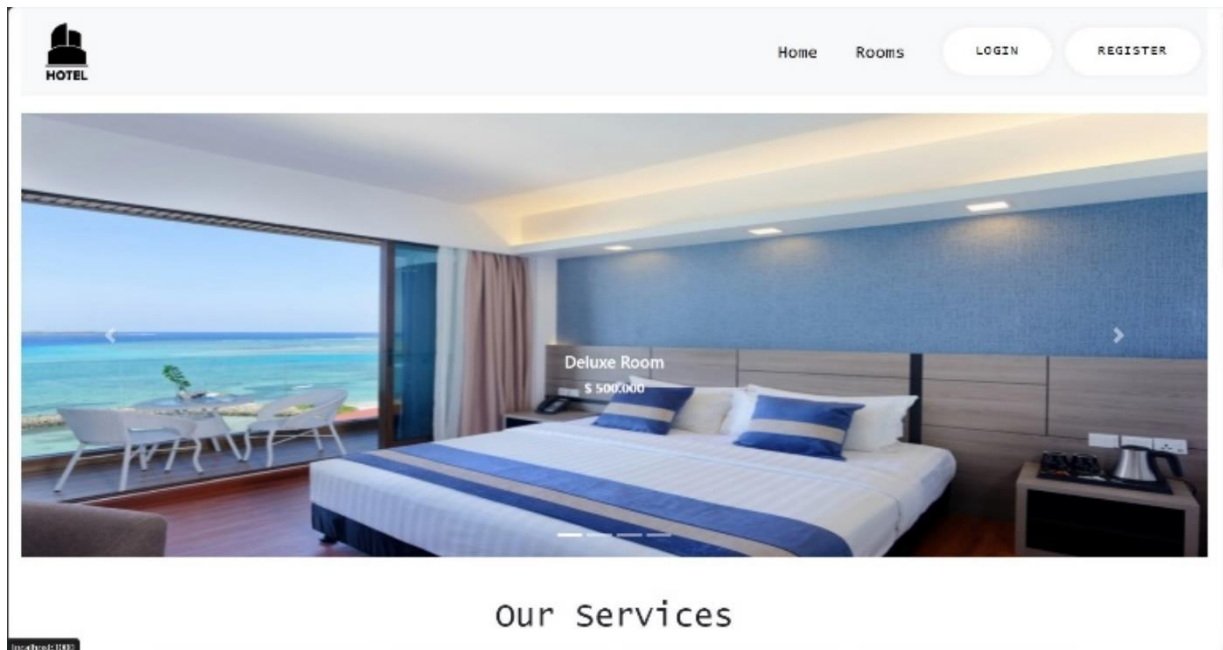


Figure 5: la page d'accueil de HOTEL

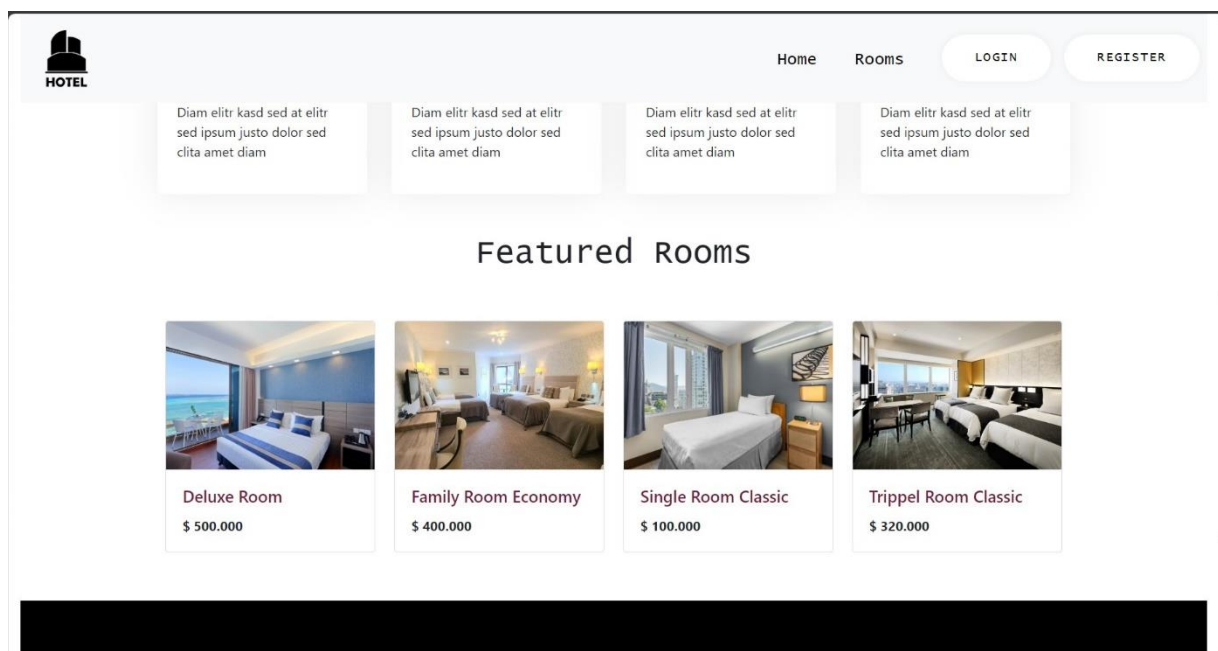


Figure 6: Featured Rooms

La page rooms represente les différents rooms disponible dans l'application avec ces caractéristiques(prix,type,...)
Et donne aussi la possibilité aussi de filtrer les caractéristique des chambres selon les besoins de client Comme Category, Ccapacity, le prix, et la disponibilité de la chambre.

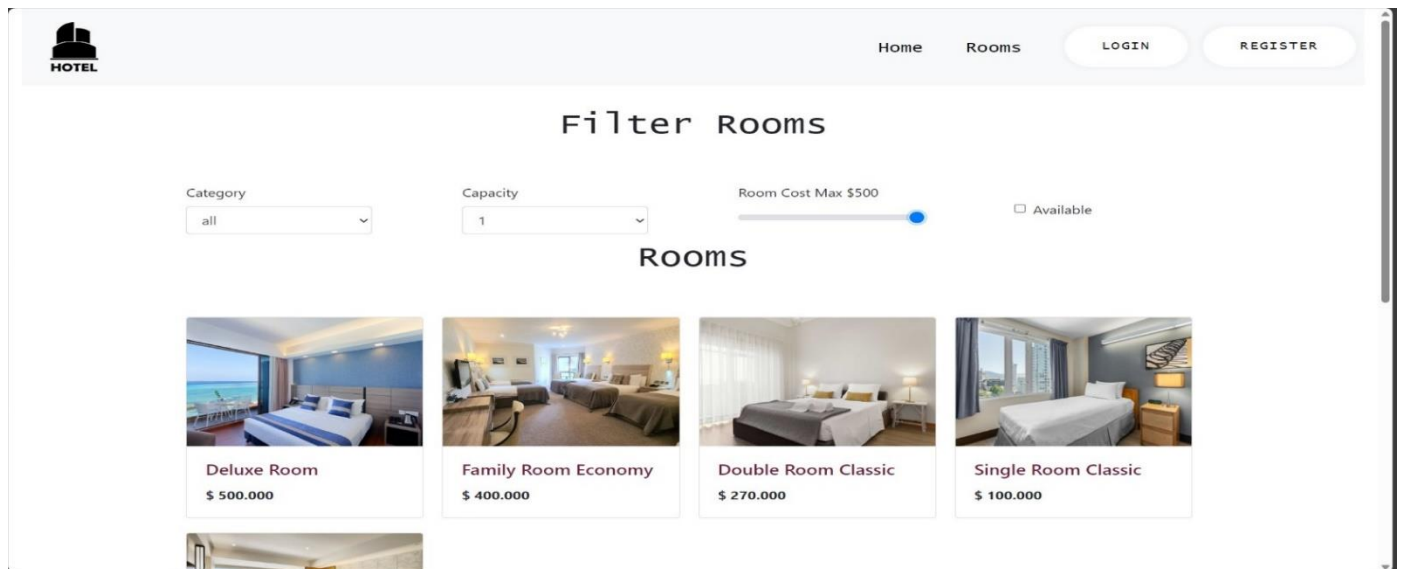


Figure 7:Page des rooms

La page d'authentification des utilisateur qui permet un utilisateur d'authentifier si il a déjà un compte sinon il faut crée un compte d'abord.

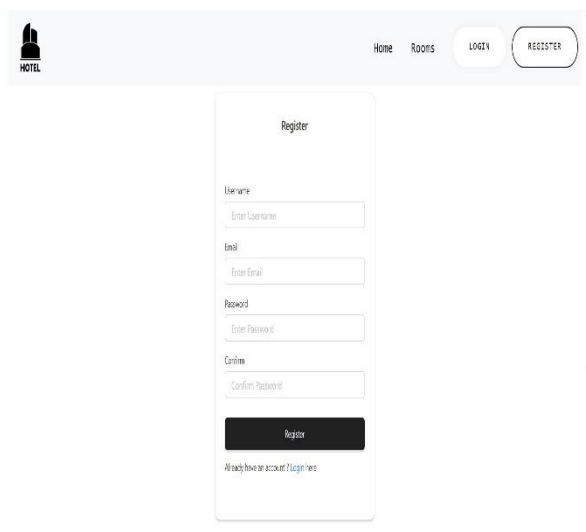


Figure 8:La Page d'enregistrement

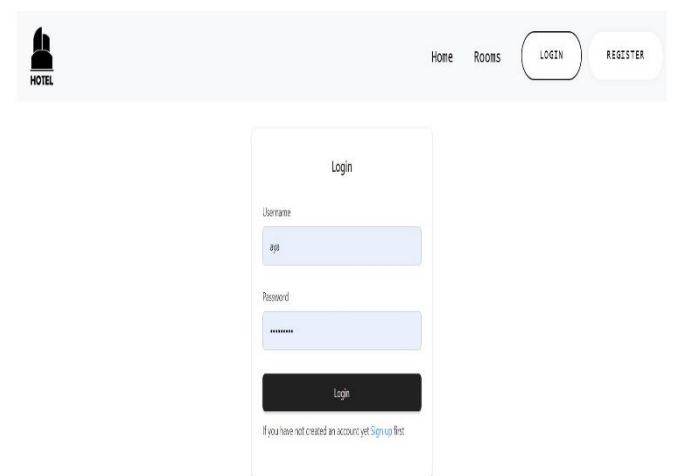



Figure 9:Page d'authentification

on clique sur le boutons book Room et implémenter les informations de réservation.



HomeRoomsLogout

Goto Room

Email

Enter Email

Phone Number

Enter Phone Number

Checking Date

jj/mm/aaaa --:--

Checkout Date

jj/mm/aaaa --:--

Book

Figure 10:Page d'implémentation des informations des réservations

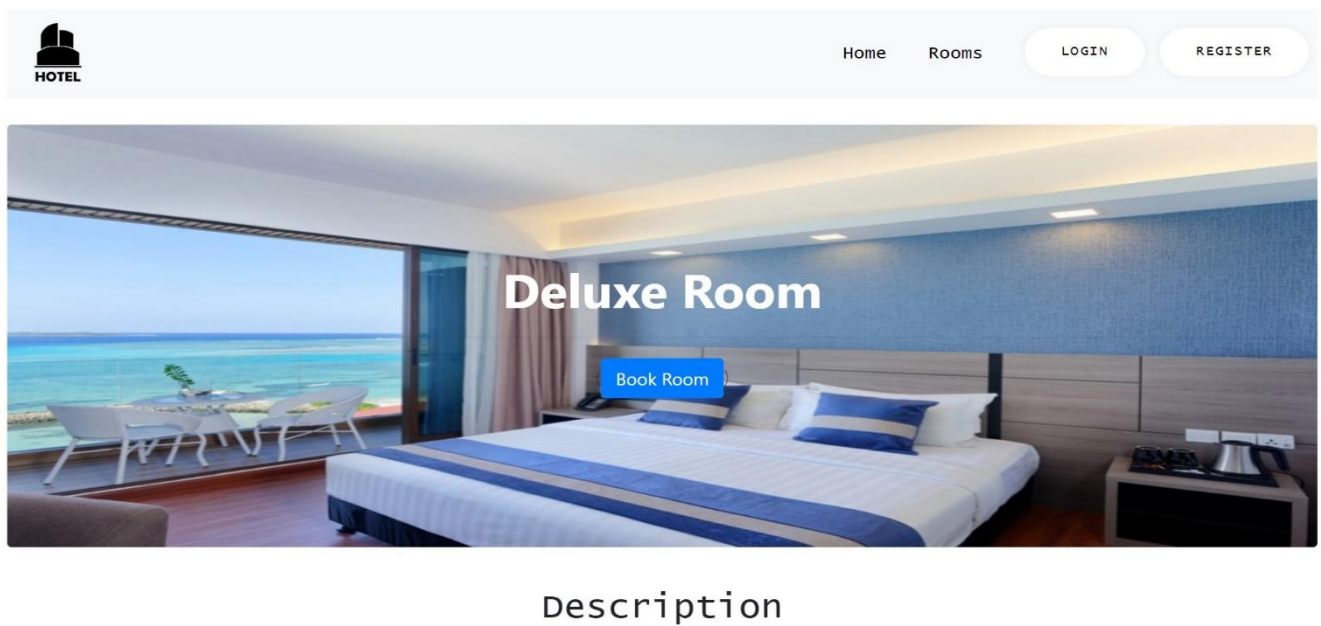


Figure 11:Reserver un Room

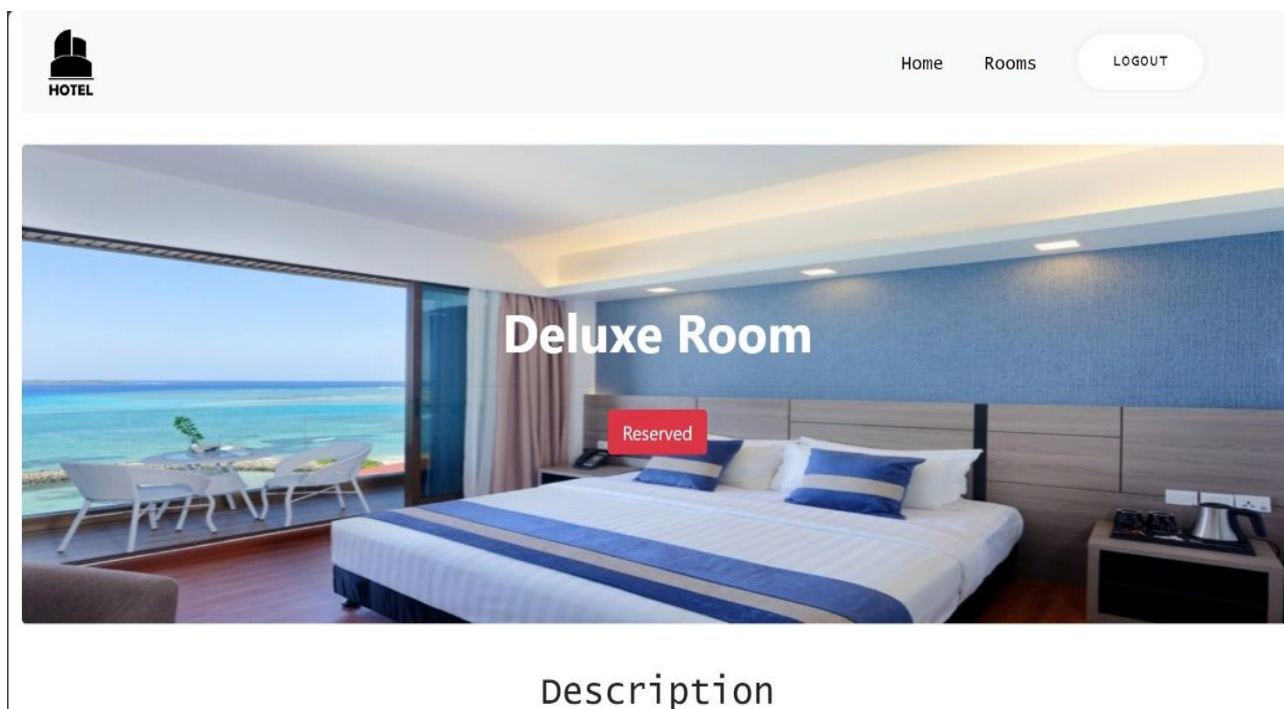



Figure 12: Vérifier la réservation

Après l'authentification de l'administrateur peut effectuer le check-out des chambres, rendant les chambres disponibles pour de nouvelles réservations une fois le séjour du client terminé



[Home](#)
[Rooms](#)
[Dashboard](#)

LOGOUT

List of Booked Rooms

#	Room	Booked By	Phone Number	Action
1	family_room	hanane	0623252928	<button>Checkout</button>
2	deluxe-room	aya	0767409836	<button>Checkout</button>
3	single_room	salma	0624262528	<button>Checkout</button>

Figure 13: Dashboard de l'administrateur

Conclusion

En conclusion, notre projet d'application de réservation d'hôtels a été réalisé avec succès en combinant les frameworks Django et React. Cette approche a permis de créer une application robuste et interactive, exploitant la puissance du backend Django pour la gestion des données et du frontend React pour une expérience utilisateur dynamique. La séparation des préoccupations entre le backend et le frontend a facilité le développement et la maintenance, tandis que l'attention portée à la sécurité et à la communication entre les différentes parties a assuré une application fonctionnelle et sécurisée. Ce projet représente une convergence réussie de technologies modernes pour répondre aux exigences d'une application de réservation d'hôtels.

Table des figures

Figure 1: Diagramme de cas d'utilisation de l'application HOTEL	8
Figure 2:Diagramme de séquence d'inscription dans le site HOTEL	8
Figure 3:Diagramme de Classe d'appliaction HOTEL	9
Figure 4:les tables de bases de données	10
Figure 5:la page d'accueil de HOTEL.....	20
Figure 6:Featured Rooms	20
Figure 7:Page des rooms	21
Figure 8:La Page d'enregistrement	21
Figure 9:Page d'authentification	21
Figure 10:Page d'implémentation des informations des réservations	22
Figure 11:Reserver un Room	22
Figure 12:Vérifier la réservation	23
Figure 13:Dashboard de l'administrateur	23

