

**UNIVERSITÉ MOULOUD MAMMERI TIZI-OUZOU**  
**FACULTÉ DU GÉNIE ÉLECTRIQUE ET D'INFORMATIQUE**  
**DEPARTEMENT D'INFORMATIQUE**



## Rapport projet dev web



**Réalisé Par :**

- CHIKHAOUI Romayssa (CPI)
- DJERROUDI Lyna (SI)
- KACID Feriel (SI)
- REZZOUK Nada (CPI)

**Année universitaire : 2025/2026**

# Sommaire

1. Introduction:	2
1.1 Contexte et Objectifs	2
1.2 Description du Projet	2
2. Diagrammes UML:	3
2.1 Diagramme de contexte:	3
2.2 Diagramme de cas d'utilisation :	4
3. Architecture du projet	5
3.1 Frontend	6
3.1.1. Pages et fonctionnalités	6
3.1.2. Communication avec le backend	6
3.2 Backend (Node.js + Express)	7
3.2.1. Gestion des rôles et autorisations	7
3.2.2. CRUD pour les entités principales	7
3.2.3. API RESTful	7
3.3 Base de données (MySQL)	7
Collections principales	7
3.4 Communication entre les composants	9
3.4.1 Frontend ↔ Backend	9
3.4.2 Backend ↔ Base de données	9
4. Réalisation:	10
4.1 Installation et Configuration	10
4.2 Création de la structure des fichiers backend	11
4.3 Création de la structure des fichiers frontend	15
5. Technologies utilisées	17
5.1 Frontend - EJS& CSS	17
5.2 Backend - Node.js & Express	18
5.3 Base de données - MySQL	18
5.4 API RESTful	19
6. Captures Des Pages et Fonctionnalités du site	20

# **1.Introduction:**

## **1.1 Contexte et Objectifs:**

La transformation numérique actuelle a conduit à une adoption massive des solutions en ligne dans différents secteurs d'activité. Dans le domaine de la location d'espaces, les plateformes numériques facilitent la mise en relation entre les propriétaires et les clients à la recherche de salles pour organiser des réunions, des événements ou des ateliers. Malgré cette évolution, trouver un espace répondant précisément aux besoins des utilisateurs peut encore représenter une difficulté.

Ce projet a pour objectif de concevoir une plateforme intuitive dédiée à la réservation de salles, permettant de connecter efficacement les propriétaires aux clients potentiels. Grâce à un système de gestion structuré et performant, la plateforme vise à améliorer la visibilité des salles disponibles à la location tout en simplifiant le processus de réservation et en offrant une meilleure expérience utilisateur.

### **Les principaux objectifs du projet sont :**

- Mettre à disposition une vue d'ensemble des salles disponibles à travers des fonctionnalités de recherche et de filtrage, notamment selon des critères tels que la capacité.
- Intégrer un système de géolocalisation permettant de visualiser les salles sur une carte interactive.
- Offrir une expérience utilisateur fluide en concevant des interfaces ergonomiques et adaptées aux différents rôles de la plateforme (client, propriétaire et administrateur).
- Garantir la sécurité et la protection des données des utilisateurs et du système.

## **1.2 Description du Projet:**

La plateforme de réservation de salles offre plusieurs fonctionnalités adaptées aux différents types d'utilisateurs :

### **● Pour les visiteurs :**

Lorsqu'un utilisateur accède à la page d'accueil, il peut :

- Rechercher et filtrer les salles disponibles selon plusieurs critères tels que la capacité (nombre de personnes), la localisation et le nom.
- Consulter les salles disponibles sans avoir la possibilité de les réserver ni de laisser des commentaires.
- Visualiser l'emplacement des salles sur une carte interactive grâce à l'intégration d'une API de localisation.
- S'inscrire en tant que client ou propriétaire.
- Se connecter afin d'accéder à l'espace client ou propriétaire.

### ● Pour les clients :

Lorsqu'un visiteur se connecte en tant que client, il peut :

- Rechercher et filtrer les salles disponibles selon la capacité, la localisation et le nom.
- Visualiser la localisation des salles sur une carte interactive.
- Effectuer la réservation d'une salle.
- Laisser des avis et des commentaires après une réservation.
- Consulter l'historique de ses réservations.

### ● Pour les propriétaires :

Lorsqu'un utilisateur se connecte en tant que propriétaire, il peut :

- Gérer ses salles en ajoutant, modifiant ou supprimant des annonces.
- Consulter les avis et commentaires laissés par les clients.
- Accéder à une page de statistiques lui permettant de visualiser le nombre de réservations ainsi que les revenus générés.

### ● Pour les administrateurs :

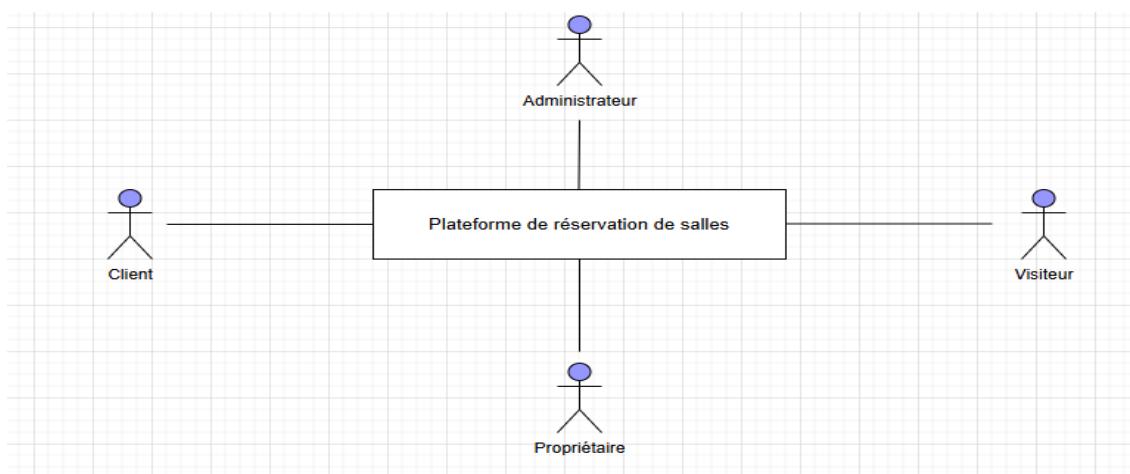
- Gérer les comptes utilisateurs (activation et désactivation).
- Supprimer des annonces de salles si nécessaire.
- Superviser et modérer les avis et commentaires publiés sur la plateforme.

## 2. Diagrammes UML:

### 2.1 Diagramme de contexte :

Ce diagramme représente les interactions entre le système et son environnement, il permet de délimiter le champ d'étude du système.

Après identification des acteurs de notre système on obtient le diagramme suivant :



**Fig 1:Diagramme de contexte**

## 2.2 Diagramme de cas d'utilisation :

Ce diagramme montre les différentes actions que les utilisateurs peuvent effectuer sur la plateforme, selon leur rôle (visiteur, client, propriétaire, administrateur).



**Fig 2:Diagramme de cas d'utilisation**

### **3. Architecture du projet:**

Le projet est structuré selon une architecture **frontend-backend** clairement séparée, permettant une meilleure évolutivité ainsi qu'une maintenance facilitée. Chaque composant du système est responsable d'un ensemble de fonctionnalités spécifiques et communique avec les autres à travers des interfaces bien définies basées sur des API.

#### **Frontend (EJS / CSS):**

Le frontend est développé à l'aide de EJS (Embedded JavaScript Templates), permettant la création de pages web dynamiques et interactives. Cette technologie facilite l'intégration de données côté serveur directement dans les vues.

La mise en forme et l'aspect visuel de l'application sont assurés à l'aide de CSS, ce qui permet de concevoir une interface utilisateur claire, moderne et ergonomique. L'interface est adaptée aux différents rôles du système, à savoir le visiteur, le client, le propriétaire et l'administrateur, afin d'offrir une expérience utilisateur fluide et intuitive.

#### **Backend:**

Le backend repose sur une **API RESTful** développée avec **Node.js et Express**. Il assure la gestion de la logique métier, le traitement des requêtes, l'authentification des utilisateurs ainsi que la communication avec la base de données.

#### **Base de données:**

La base de données utilisée est **MySQL**, choisie pour sa fiabilité et sa capacité à gérer efficacement des données structurées. Elle permet de stocker et d'organiser les informations relatives aux utilisateurs, aux salles, aux réservations et aux avis.

#### **3.1 Frontend (EJS):**

L'interface utilisateur est développée à l'aide de **EJS (Embedded JavaScript Templates)** et permet d'offrir une expérience adaptée à chaque type d'utilisateur de la plateforme.

##### **3.1.1 Pages et fonctionnalités:**

###### **• Pages publiques pour les visiteurs :**

Les visiteurs ont accès à une page d'accueil présentant les salles disponibles avec leurs informations principales telles que la localisation, la capacité et le prix. À ce niveau, aucune authentification n'est requise.

- **Interfaces spécifiques selon le rôle utilisateur :**

- **Client** :

- Rechercher et filtrer les salles selon différents critères (prix, capacité, localisation).
    - Visualiser les salles sur une carte interactive grâce à Leaflet (bibliothèque JavaScript open-source) utilisant OpenStreetMap (cartographie libre), permettant l'affichage des marqueurs GPS (latitude/longitude) stockés dans la base de données.
    - Effectuer la réservation d'une salle.
    - Consulter l'historique de ses réservations.
    - Laisser des avis et des commentaires sur les salles réservées.

- **Propriétaire** :

- Ajouter des salles avec leurs descriptions, capacité, prix et localisation géographique (définie à l'aide d'une carte interactive).
    - Modifier ou supprimer les salles existantes.
    - Consulter les avis et commentaires laissés par les clients.
    - Accéder à des statistiques comprenant le nombre de réservations, les revenus générés et les avis reçus.

- **Administrateur** :

- Gérer les comptes utilisateurs (activation et désactivation).
    - Supprimer des annonces de salles si nécessaire.
    - Superviser et modérer les avis et commentaires publiés sur la plateforme.

### **3.1.2 Communication avec le backend:**

Le frontend communique avec le backend via des **routes RESTful** gérées par Express. Les données sont transmises et affichées dynamiquement à travers les templates EJS.

### **3.2 Backend (Node.js + Express):**

Le backend constitue le cœur de la logique métier de la plateforme et assure la communication entre le frontend et la base de données.

### **3.2.1 Gestion des rôles et des autorisations**

- Chaque utilisateur se voit attribuer un rôle spécifique : Visiteur, Client, Propriétaire ou Administrateur.
- Les autorisations d'accès aux différentes fonctionnalités sont gérées via **JWT (JSON Web Tokens)**, garantissant la sécurité des opérations selon le rôle de l'utilisateur.

### **3.2.2 CRUD pour les entités principales**

- **Salles :**

- Les propriétaires peuvent créer, consulter, mettre à jour et supprimer leurs salles.
- Les clients et les visiteurs peuvent uniquement consulter les informations des salles disponibles.

- **Utilisateurs :**

- L'administrateur peut gérer les comptes de tous les utilisateurs, notamment activer ou désactiver un compte.

- **Réservations :**

- Les clients peuvent réserver des salles.
- L'historique des réservations est conservé et les statistiques associées permettent aux propriétaires de suivre les réservations de leurs salles.

- **Avis et commentaires :**

- Les clients peuvent laisser des avis sur les salles qu'ils ont réservées.
- Les propriétaires peuvent consulter ces avis et commentaires.
- L'administrateur a la possibilité de supprimer les avis jugés inappropriés.

### **3.2.3 API RESTful:**

Le backend expose des **endpoints clairs et organisés** pour effectuer les opérations CRUD, gérer l'authentification et manipuler les différentes données de la plateforme.

### **3.3 Base de données (MySQL):**

La base de données utilisée pour ce projet est **MySQL**, qui permet de stocker et gérer toutes les informations nécessaires au bon fonctionnement de la plateforme.

## **Tables principales:**

### **1. Users :**

- Contient les informations relatives aux utilisateurs (Clients, Propriétaires, Administrateurs).
- Inclut les rôles, les identifiants de connexion et les données personnelles (nom, email, mot de passe, numéro de téléphone etc.).

### **2. Rooms :**

- Stocke les informations des salles proposées à la location, notamment :
  - La description (titre et détails).
  - La localisation géographique et wilaya (pour l'affichage sur une carte).
  - Le prix, la capacité, l'équipement et les photos.

### **3. Bookings :**

- Enregistre les réservations effectuées par les clients.
- Contient les informations sur la salle réservée, le client, le prix, la date de réservation et le statut.

### **4. Reviews :**

- Stocke les avis et commentaires laissés par les clients après leurs réservations.

## **3.4 Communication entre les composants:**

### **3.4.1 Frontend ↔ Backend:**

- Le frontend envoie des requêtes HTTP au backend via des **API RESTful**.
- Les réponses du backend contiennent les données au format **JSON**, par exemple la liste des salles disponibles ou les détails d'une réservation.

### **3.4.2 Backend ↔ Base de données**

- Le backend utilise des requêtes **SQL** pour interagir avec la base de données MySQL.
- Les opérations réalisées incluent la récupération, l'insertion, la mise à jour et la suppression des données dans les différentes tables de la base.

## **4.Réalisation:**

### **4.1 Installation et Configuration:**

Voici le résumé des installations pour notre plateforme utilisant Node.js, Express, EJS et MySQL (via WAMP Server) :

#### **1. Installation des outils requis:**

- Node.js (avec npm intégré) : Téléchargement depuis nodejs.org
- WAMP Server (Windows Apache MySQL PHP) :
  - Installer Apache (serveur web local)
  - Installer MySQL (base de données)
  - Installer phpMyAdmin (interface web pour gérer MySQL)

#### **2. Configuration WAMP et MySQL:**

Lancer WAMPServer (icône verte dans la barre des tâches)

Création de la base de données via phpMyAdmin :

- Accéder à : <http://localhost/phpmyadmin>
- Se connecter (root / avec mot de passe)
- Créer la BDD : EventSpace

**Structure des tables créées :**

- `users` (id, name, email, password, type, phone, created\_at)
- `rooms` (id, name, description, capacity, price, Wilaya, latitude, longitude, image, equipment, owner\_id)
- `bookings` (id, room\_id, user\_id, start\_date, end\_date, participants, purpose, status)
- `reviews` (id, booking\_id, room\_id, user\_id, rating, comment, date)

#### **3. Initialisation du projet Node.js:**

**Installation des dépendances :**

```
npm install express ejs mysql2 multer bcryptjs jsonwebtoken cookie-parser method-override  
dotenv
```

## Packages utilisés :

- `express` : Framework serveur web (port 3000)
- `ejs` : Moteur de templates
- `mysql2` : Driver MySQL pour Node.js (connexion à la BDD WAMP)
- `multer` : Upload des images des salles
- `bryptjs` : Hashage des mots de passe
- `jsonwebtoken` : Authentification JWT
- `cookie-parser` : Gestion des cookies
- `method-override` : PUT/DELETE via formulaires
- `dotenv` : Variables d'environnement (DB\_HOST, DB\_USER, DB\_PASS)

## 4.2 Crédit de la structure du dossier :

```
Projet-Dev-Web/
|
├── .gitignore
├── app.js
├── package.json
├── package-lock.json
└── setup-admin.js
|
├── node_modules/      (ignoré par git)
├── .env                (ignoré par git)
├── test-db.js          (ignoré par git)
├── create-owner.js     (ignoré par git)
├── create-admin.js     (ignoré par git)
├── create-user.js      (ignoré par git)
└── db.js               (ignoré par git)
|
├── public/
│   ├── css/
│   │   ├── style.css
│   │   ├── style.css.bak
│   │   └── custom.css
│
│   ├── js/
│   │   ├── rooms.js
│   │   └── animations.js
│
│   ├── images/           (vide ou contient des images)
│
└── uploads/            (ignoré par git)
|
└── views/
    ├── admin/
    │   └── dashboard.ejs
```

```

    └── admin/
        └── dashboard.ejs

    └── client/
        ├── dashboard.ejs
        └── bookings-history.ejs

    └── owner/
        ├── dashboard.ejs
        ├── dashboard.temp.ejs
        ├── dashboard.temp2.ejs
        ├── add-room.ejs
        └── edit-room.ejs

    └── partials/
        ├── header.ejs
        └── footer.ejs

    └── reviews/
        └── create.ejs

    └── book-room.ejs
    └── index.ejs
    └── index.ejs.bak
    └── login.ejs
    └── register.ejs
    └── rooms.ejs

```

#### **4.2.1 Fichiers racine:**

- `.gitignore` : Liste des fichiers/dossiers à exclure du suivi Git (node\_modules, .env, uploads, etc.)
- `app.js` : Fichier principal de l'application Node.js, contient la configuration du serveur Express, les routes et la logique backend
- `package.json` : Fichier de configuration npm contenant les dépendances du projet et les scripts
- `package-lock.json` : Verrouille les versions exactes des dépendances installées
- `setup-admin.js` : Script d'initialisation pour créer un compte administrateur

#### **4.2.2. Dossiers/fichiers ignorés par Git (dans .gitignore):**

- `node_modules/` : Dépendances du projet téléchargées via npm (bibliothèques Node.js comme Express, EJS, etc.)

- `.env` : Variables d'environnement sensibles (clés API, mots de passe de base de données, secrets de session)
- `uploads/` : Dossier contenant les fichiers uploadés par les utilisateurs (images de salles)
- `.DS_Store` : Fichier système macOS (métadonnées de dossier)
- `test-db.js, create-owner.js, create-admin.js, create-user.js, db.js` : Scripts de test et de configuration de base de données

#### **4.2.3. public/ - Ressources statiques accessibles au client:**

- `css/` : Feuilles de style CSS
  - `style.css` : Styles principaux de l'application
  - `custom.css` : Styles personnalisés et thème sombre
  - `style.css.bak` : Backup de l'ancien fichier CSS
- `js/` : Scripts JavaScript côté client
  - `rooms.js` : Gestion des interactions sur la page des salles (filtres, animations)
  - `animations.js` : Animations AOS, compteurs, scroll effects, parallaxe
- `images/` : Images statiques du site (vide actuellement)
- `uploads/` : Images uploadées dynamiquement par les propriétaires (ignoré par Git)

#### **4.2.5. views/ - Templates EJS (pages HTML dynamiques):**

##### views/admin/ - Interface administrateur:

- `dashboard.ejs` : Tableau de bord admin avec gestion des utilisateurs, salles et avis

##### views/client/ - Interface client:

- `dashboard.ejs` : Tableau de bord client pour rechercher des salles et gérer les réservations
- `bookings-history.ejs` : Historique des réservations (en cours et passées)

##### views/owner/ - Interface propriétaire:

- `dashboard.ejs` : Tableau de bord propriétaire listant ses salles
- `add-room.ejs` : Formulaire d'ajout d'une nouvelle salle
- `edit-room.ejs` : Formulaire de modification d'une salle existante
- `dashboard.temp.ejs, dashboard.temp2.ejs` : Fichiers temporaires de modifications

##### views/partials/ - Composants réutilisables:

- `header.ejs` : En-tête commun (navbar, liens CSS/JS)
- `footer.ejs` : Pied de page commun (infos de contact, liens sociaux)

## views/reviews/ - Avis et commentaires:

- `create.ejs` : Formulaire de création d'un avis sur une salle

## views/ - Pages principales:

- `index.ejs` : Page d'accueil avec hero section et présentation
- `rooms.ejs` : Page listant toutes les salles disponibles avec filtres
- `book-room.ejs` : Page de réservation d'une salle
- `login.ejs` : Page de connexion
- `register.ejs` : Page d'inscription (choix entre client/propriétaire)
- `index.ejs.bak` : Backup de l'ancienne page d'accueil

## 5. Technologies utilisées:

Dans ce projet, plusieurs technologies modernes ont été employées pour construire une application web performante et évolutive. Ces technologies permettent une gestion fluide des données, une interface utilisateur dynamique et une architecture backend robuste. Voici les principales technologies utilisées :

### 5.1 Frontend – EJS & CSS:

#### 1) EJS (Embedded JavaScript Templates)



EJS est un moteur de templates pour Node.js qui permet de générer des pages HTML dynamiques côté serveur. Il permet de créer des vues modulaires et réutilisables et de gérer l'affichage des données selon le rôle de l'utilisateur.

- **Rôle dans le projet :**

EJS a été utilisé pour construire les pages interactives de l'application, adaptées à chaque type d'utilisateur (Visiteur, Client, Propriétaire, Administrateur). Les données provenant du backend sont intégrées directement dans les pages HTML, permettant une interface réactive sans rechargement complet de la page.

- **Avantages :**

- Facilité d'intégration avec Express
- Génération dynamique des pages
- Réutilisation des templates pour différents rôles

## 2) CSS (Cascading Style Sheets)



CSS est un langage de feuilles de style utilisé pour décrire la présentation et la mise en forme des pages web. Il permet de contrôler l'apparence visuelle des éléments HTML tels que les couleurs, les polices, la disposition, les marges et la responsivité de l'interface.

- **Rôle dans le projet :**

CSS a été utilisé pour assurer une interface utilisateur claire, cohérente et ergonomique. Il permet d'adapter l'affichage des pages aux différents types d'utilisateurs (Visiteur, Client, Propriétaire, Administrateur) en améliorant la lisibilité et l'expérience utilisateur. Grâce au CSS, l'application offre un design moderne et structuré.

- **Avantages :**

- Amélioration de l'expérience utilisateur
- Séparation du contenu et de la présentation
- Personnalisation et uniformité de l'interface
- Compatibilité avec tous les navigateurs

## 5.2 Backend – Node.js & Express:

### 1) Node.js:



Node.js est un environnement d'exécution JavaScript côté serveur qui permet d'exécuter du code JavaScript en dehors du navigateur. Il est basé sur le moteur V8 de Google Chrome et est idéal pour gérer des applications web performantes et des requêtes simultanées grâce à son modèle asynchrone et non-bloquant.

- **Rôle dans le projet :**

Node.js constitue l'environnement backend, exécutant la logique serveur et gérant les interactions avec la base de données MySQL.

- **Avantages :**

- Haute performance et gestion asynchrone des requêtes
- Large écosystème de bibliothèques via npm
- Adapté aux applications nécessitant un grand nombre de connexions simultanées

## 2) Express.js:



Express est un framework minimaliste pour Node.js qui facilite le développement d'applications web et d'API. Il permet de gérer facilement les routes, les requêtes HTTP et les middlewares.

- **Rôle dans le projet :**

Express a été utilisé pour gérer les routes du backend, la logique métier, l'authentification et les interactions avec la base de données MySQL.

- **Avantages :**

- Légèreté et simplicité
- Architecture modulaire avec middlewares
- Extensible pour ajouter de nouvelles fonctionnalités

## 5.3 Base de données – MySQL



MySQL est une base de données relationnelle permettant de stocker et organiser les données de manière structurée. Elle est fiable et adaptée pour gérer les informations des utilisateurs, des salles, des réservations et des avis.

- **Rôle dans le projet :**

MySQL a été utilisé pour stocker toutes les données essentielles de la plateforme. Les tables principales sont :

- **Users** : informations des utilisateurs et rôles

- **Rooms** : informations sur les salles (description, localisation, capacité, prix, photos)

- **Réservations** : historique des réservations et statistiques

- **Reviews** : avis et commentaires des clients

- **Avantages :**

- Gestion structurée des données
  - Fiabilité et sécurité des informations
  - Support des requêtes complexes via SQL

## 5.4 API RESTful

L'API RESTful a été construite avec **Node.js** et **Express** pour permettre l'échange de données entre le frontend EJS et le backend. Cette architecture assure une séparation claire entre l'interface utilisateur et la logique côté serveur.

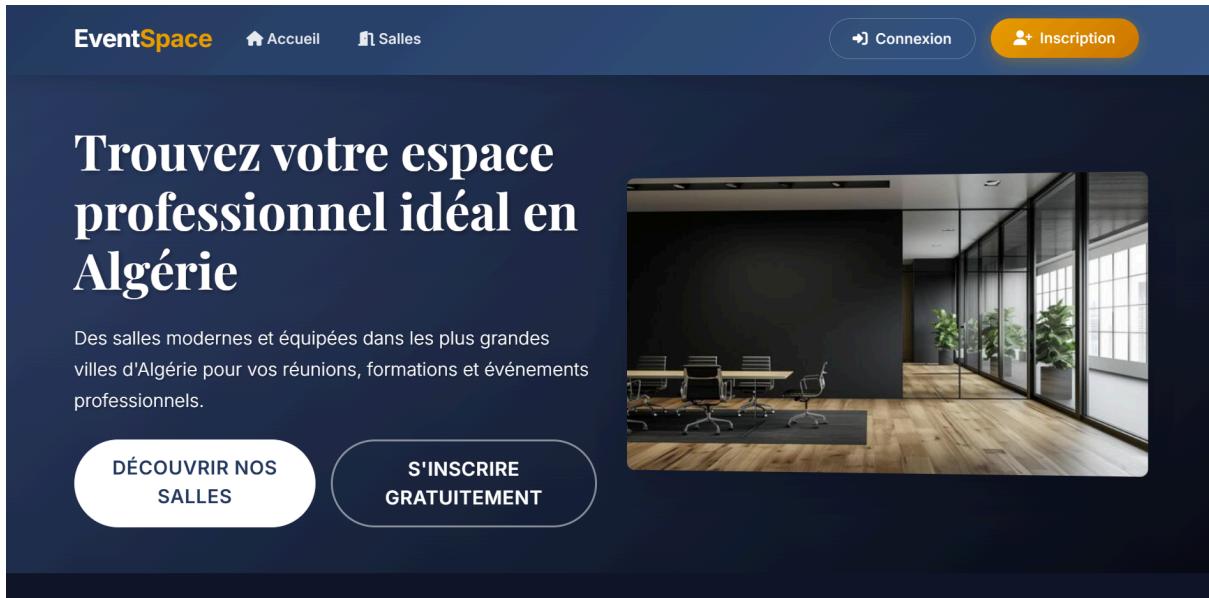
- **Fonctionnalités :**

- Gestion des opérations CRUD sur les entités principales
  - Authentification et autorisations selon les rôles utilisateurs via JWT
  - Communication sécurisée avec la base de données MySQL

- **Rôle dans le projet :**

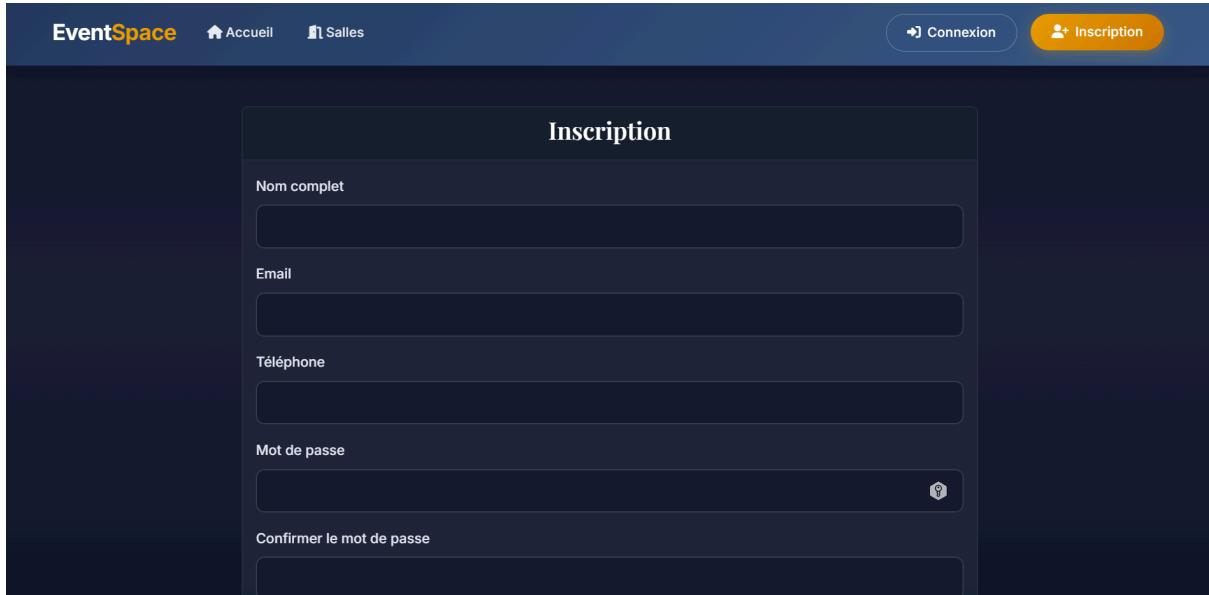
L'API RESTful sert de passerelle entre le frontend et la base de données, gérant toutes les interactions et assurant l'accès aux données en fonction des rôles utilisateurs.

## 6. Captures Des Pages et Fonctionnalités du site



The screenshot shows the homepage of EventSpace. At the top, there is a dark blue header bar with the logo "EventSpace" on the left, followed by navigation links for "Accueil" and "Salles". On the right side of the header are "Connexion" and "Inscription" buttons. The main content area has a dark background with white text. A large heading reads "Trouvez votre espace professionnel idéal en Algérie". Below it, a subtext states: "Des salles modernes et équipées dans les plus grandes villes d'Algérie pour vos réunions, formations et événements professionnels." Two prominent buttons are centered: "DÉCOUVRIR NOS SALLES" on the left and "S'INSCRIRE GRATUITEMENT" on the right. To the right of the text is a photograph of a modern conference room with a long table, several chairs, and large windows overlooking an office exterior.

Page Accueil



The screenshot shows the "Inscription" (Registration) page. The header is identical to the homepage, featuring the "EventSpace" logo and navigation links. The main form is titled "Inscription" at the top. It contains five input fields: "Nom complet" (Last Name), "Email", "Téléphone" (Phone Number), "Mot de passe" (Password), and "Confirmer le mot de passe" (Confirm Password). A small "G" icon with a lock symbol is positioned next to the password fields. The entire form is set against a dark background.

Formulaire inscription (client / propriétaire)

Formulaire connexion (client / propriétaire)

Espace Admin

**EventSpace** Accueil Salles Dashboard Rzk N Déconnexion

Rechercher Carte Réservations Mes avis + Nouvelle réservation

## Mon Espace Client

Gérez vos réservations et découvrez de nouvelles salles

8 Salles disponibles 1 Réservations actives 2 Avis publiés

Rechercher une salle  
Trouvez l'espace parfait pour vos besoins

8 salles trouvées

Espace client

**EventSpace** Accueil Salles Dashboard OWNER Déconnexion

## Mes salles

+ Ajouter une salle

**show**  
big room for your show and conferences  
📍 Tizi Ouzou, dbk  
Capacité: 30 personnes  
Prix: 1500.00DA / jour

Modifier Supprimer

Espace propriétaire

**EventSpace** Accueil Salles Connexion Inscription

## Nos salles disponibles

Capacité min.
Prix max/jour
Wilaya

Équipements

Filtrer
 Réinitialiser

Voir les salles disponibles (Filtrage de salles)

**EventSpace** Accueil Salles Dashboard RZK N Déconnexion

**show**  
big room for your show and conferences  
**Capacité:** 30 personnes  
**Prix:** 1500.00 DA / jour  
**Équipements:**  
wifi projector sound

Avis des clients

### Réserver cette salle

**Date de début**

**Date de fin**

**Nombre de participants**  
  
Maximum 30 personnes

**Objet de la réservation**

**Prix total: 0 DA**

Confirmer la réservation

Espace réserver une salle ( client )