

SocSport

# socsport Documentation

# Choix sujet

"SocSport" est une application innovante visant à faciliter la vie active et sociale. Elle permet aux utilisateurs d'organiser facilement des matchs sportifs avec amis, collègues et passionnés. En favorisant un mode de vie sain, renforçant les liens sociaux, facilitant l'organisation d'événements sportifs, créant des communautés et offrant des opportunités commerciales, SocSport se positionne comme bien plus qu'une simple application, intégrant également des aspects technologiques modernes pour une expérience utilisateur optimale.

# Concurrence

Des applications comme SportyHQ, OpenSports, Beballer et Strava peuvent être utilisées pour organiser des événements sportifs et rassembler des communautés autour du sport.

SocSport se distingue de la concurrence en offrant une expérience unique et conviviale. En effet, notre application permet une réservation gratuite de terrains avec l'ajout facile d'événements sportifs personnalisés. Grâce à une technologie de localisation avancée, les utilisateurs peuvent trouver rapidement les terrains les plus proches. L'interface intuitive, les fonctionnalités sociales avancées et les retours positifs des utilisateurs renforcent notre engagement envers une communauté sportive dynamique. Optez pour SocSport pour une expérience inégalée dans l'organisation d'activités sportives.

# Modèles

# Installation de SocSport

# 0. Prérequis

Assurez-vous d'avoir Python installé sur votre machine avec une version compatible (Python 3.12 ou supérieur recommandé). Vous pouvez vérifier si Python est installé en exécutant `python --version` dans votre terminal. Dans le cas où Python n'est pas installé, vous pouvez le faire en suivant les instructions sur le site officiel de Python : [Installation de Python](#).

# 1. Installation de Flask

Vous pouvez installer Flask en lançant votre terminal de commande en tant qu'administrateur puis en entrant cette ligne de commande :

```
pip install Flask
```



## 2. Installation de Node.js

Si vous n'avez pas Node.js installé sur votre système, vous pouvez dans un premier temps le télécharger sur le site officiel : [Installation de Node.js](#).

### 3. Clonage du projet

Maintenant que Node.js et Flask sont installés, vous pouvez cloner le dépôt Github de SocSport sur votre machine locale en utilisant la commande suivante :

```
git clone https://github.com/mfrj22/socsport.git
```

## 4. Installation des librairies Python

Pour notre projet, nous avons utilisé différentes bibliothèques Python :

- SQLAlchemy permet d'intégrer une base de données SQL dans notre application Flask.

```
pip install Flask-SQLAlchemy
```

- PyMySQL permet d'assurer la connectivité et la communication avec notre base de données MySQL.

```
pip install pymysql
```

- Cryptography est utilisé pour renforcer la sécurité.

```
pip install cryptography
```

- Geopy gère les opérations de géolocalisation (calcul de distances entre des points géographiques).

```
pip install geopy
```

- PyTest permet d'exécuter des tests unitaires.

```
pip install pytest
```

## 5. Installation des modules React

Pour le bon fonctionnement de notre application, nous avons ajouté des modules React :

```
npm install  
npm install react-router-dom  
npm install react-slick slick-carousel
```

## 6. Lancement de l'Application

Vous êtes maintenant prêt à lancer l'application ! Utilisez la commande suivante pour démarrer le serveur de développement :

```
npm start
```

Pour lancer l'API Flask, il vous faudra lancer un terminal depuis la racine du projet puis lancer cette ligne de commande :

```
python app.py
```

## 7. Diagramme de classe

```
@startuml
class Ville {
    + id: Integer
    + nom: String
    + code_postal: Integer
    + departement: String
}

class Terrain {
    + id: Integer
    + nom: String
    + adresse: String
    + latitude: Float
    + longitude: Float
    + horaire_ouverture: Time
    + horaire_fermeture: Time
}

class Evenement {
    + id: Integer
    + nom: String
    + date: Date
    + heure_debut: Time
    + heure_fin: Time
    + nb_participants: Integer
}

class Sport {
    + id: Integer
    + name: String
}

class Reservation {
    + id: Integer
    + nom_participant: String
    + prenom_participant: String
    + email_participant: String
    + tel_participant: String
}

Ville "1" -- "*" Terrain : a
Terrain "1" -- "*" Evenement : a
Evenement "1" -- "0..1" Terrain : a
Evenement "1" -- "*" Reservation : a
Sport "1" -- "*" Terrain : a
Sport "1" -- "*" Reservation : a
@enduml
```

## 8. Diagramme de séquence

```
@startuml
actor User
participant Client
participant FlaskApp
database Database

User -> Client: Send POST request /nearest-fields
activate Client

Client -> FlaskApp: Handle POST /nearest-fields
activate FlaskApp

FlaskApp -> Database: Query all terrains
activate Database

Database --> FlaskApp: Return terrain data
deactivate Database

FlaskApp --> Client: Return JSON response
deactivate FlaskApp

Client --> User: Return nearest fields data
deactivate Client
@enduml
```

## 9. Diagramme entité/association

```
@startuml
entity "Ville" {
  + id: Integer
  nom: String
  code_postal: Integer
  departement: String
}

entity "Terrain" {
  + id: Integer
  nom: String
  adresse: String
  latitude: Float
  longitude: Float
  horaire_ouverture: Time
  horaire_fermeture: Time
}

entity "Evenement" {
  + id: Integer
  nom: String
  date: Date
  heure_debut: Time
  heure_fin: Time
  nb_participants: Integer
}

entity "Sport" {
  + id: Integer
  name: String
}

entity "Reservation" {
  + id: Integer
  nom_participant: String
  prenom_participant: String
  email_participant: String
  tel_participant: String
}

Ville --{ Terrain
Terrain --{ Evenement
Evenement }--|> Terrain
Evenement --{ Reservation
Sport --{ Terrain
Sport --{ Reservation
@enduml
```