2022

Documentation Abalone

PROJET D'ARCHITECTURE WEB

FLORIAN LEBECQUE

ABALONE

Projet d'architecture web

Introduction

Le principe du site est de permettre à deux joueurs de s'affronter en temps réel sur une partie du jeu de plateau Abalone. Les joueurs pourront créer des parties privées un publique qui pourrons donc être rejoints par d'autre joueur. Ceux-ci pourront communiquer à l'aide d'un tchat.

(Le jeu en lui-même a déjà été programmé en javascript, il faut néanmoins faire quelque modification pour le rendre multijoueur)

https://fr.wikipedia.org/wiki/Abalone_(jeu)

Contents

Spécification

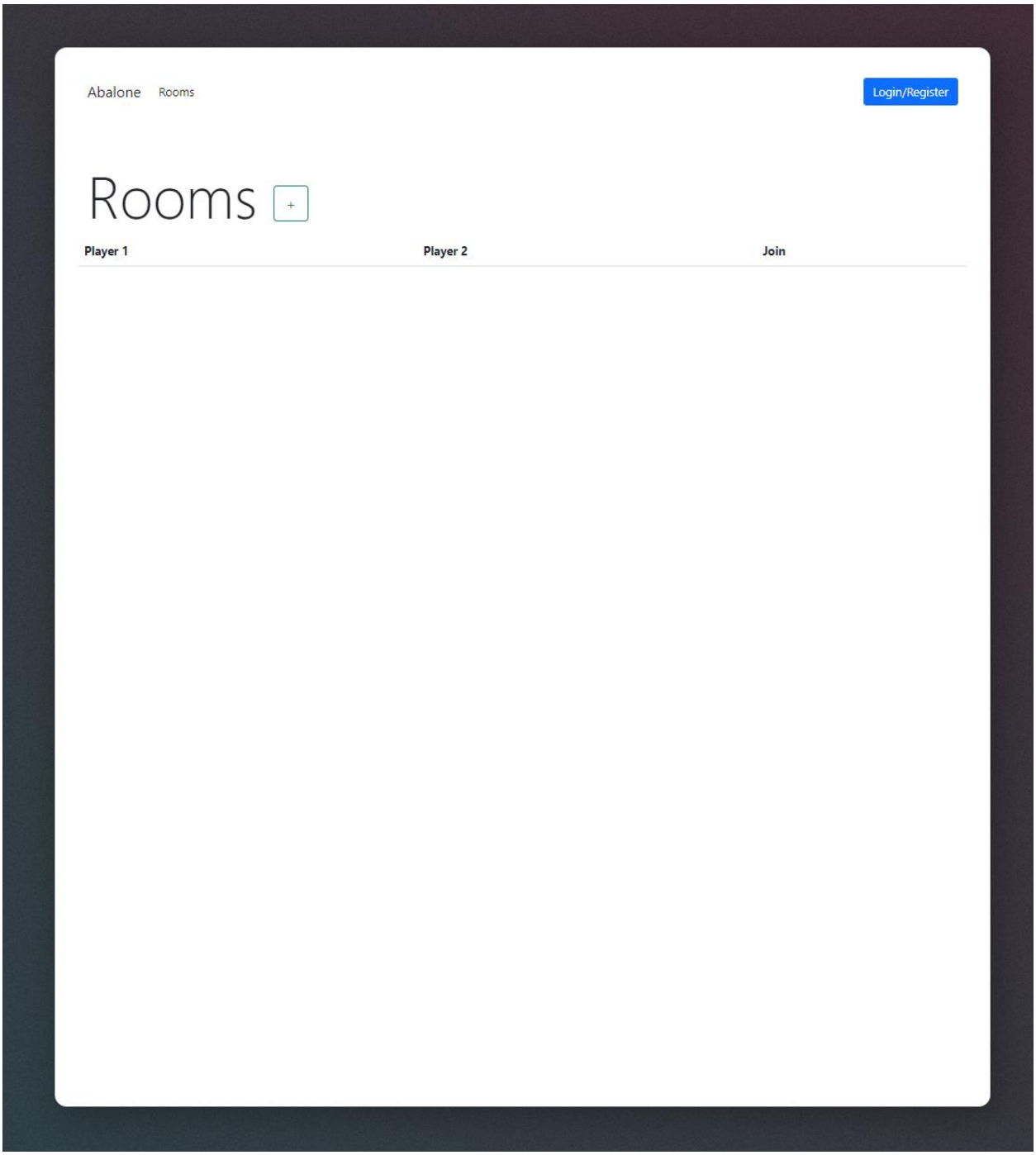
Acteur

Acteurs	Rôles
Joueur	Personne qui interagie avec le site

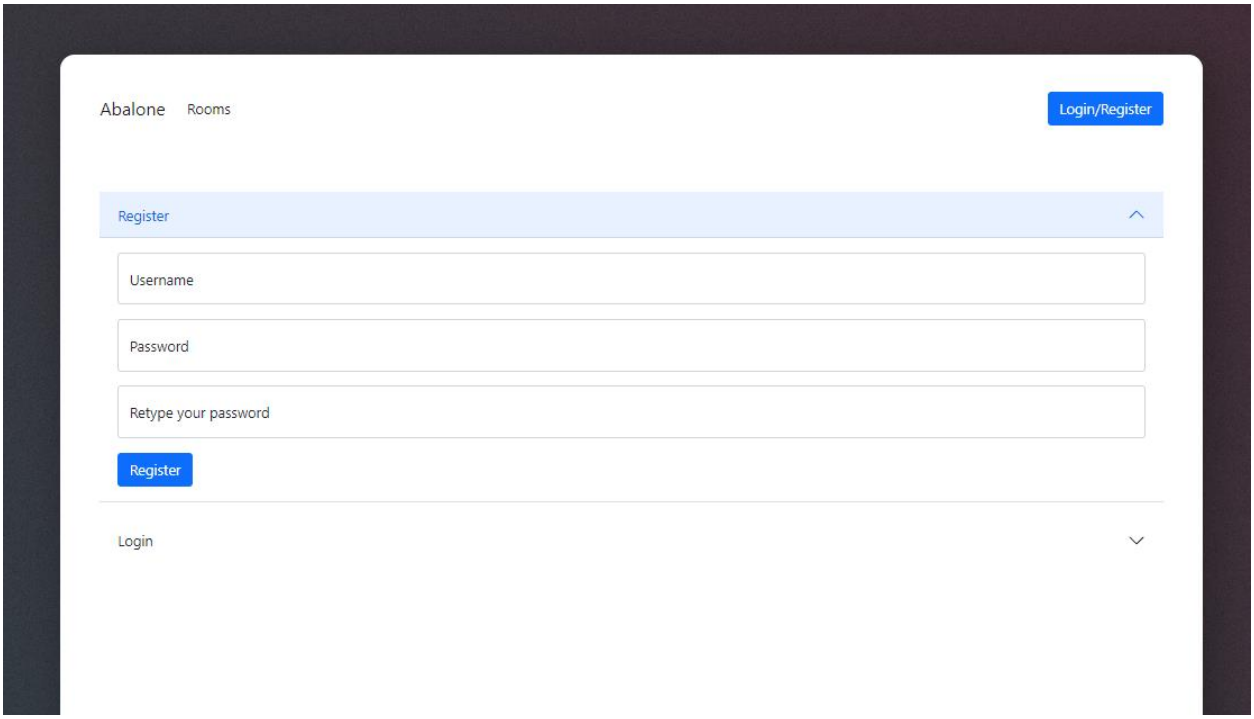
Users stories

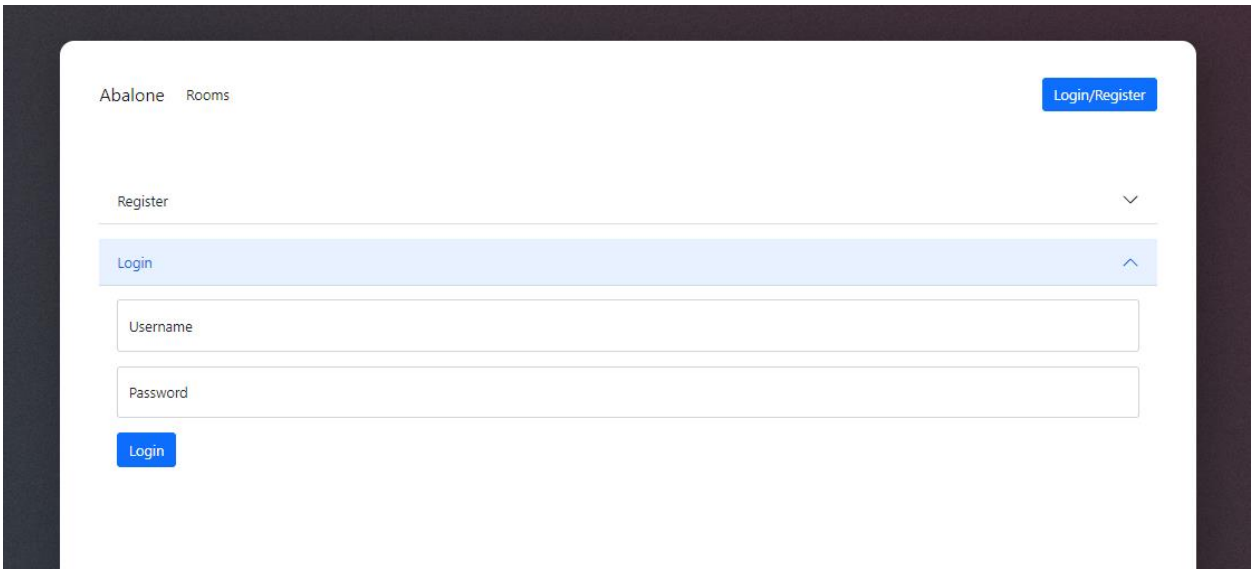
Users' stories	Priorité	État
Gestion du compte		
En tant que joueur je veux pouvoir me créer un compte pour pouvoir me connecter sur le site	2	V
En tant que joueur je veux pouvoir me connecter pour pouvoir utiliser le site	2	V
Gestion sociale		
En tant que joueur je veux pouvoir ajouter une personne en ami afin de faire une partie contre elle	2	X
En tant que joueur je veux pouvoir supprimer une personne de mes amis pour faire le tri	2	X
Gestion du jeu		
En tant que joueur je veux pouvoir créer une partie afin qu'un autre joueur m'affronte	1	V
En tant que joueur je veux pourvoir rejoindre une partie afin d'affronter un autre joueur	1	V
En tant que joueur je veux pouvoir protéger une partie pour pas que n'importe qui puisse rejoindre la partie	3	X
En tant que joueur je veux pouvoir inviter un ami à faire une partie contre moi	2	X
En tant que joueur je veux pouvoir envoyer un message à mon adversaire afin de lui communiquer mon ressenti.	2	V
En tant que joueur je veux pouvoir jouer lorsque c'est mon tour	1	V
En tant que joueur je veux voir en temps réel les coups de mon adversaire afin d'avoir une partie fluide	1	V
En tant que joueur je veux voir une liste des différentes parties disponible afin de rejoindre une	1	V
En tant que joueur je veux voir un écran de fin de partie afin d'avoir une conclusion satisfaisante à la partie	1	V
En tant que joueur je veux être sûr que mon adversaire ne triche pas afin d'avoir des parties équitables	3	V
En tant que joueur je veux avoir un historique de mes parties	2	X

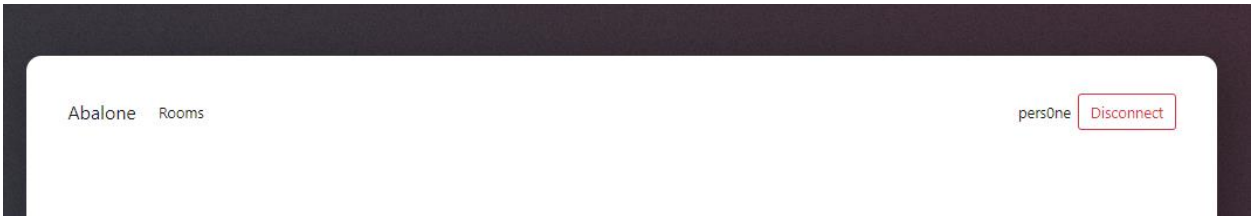
Captures

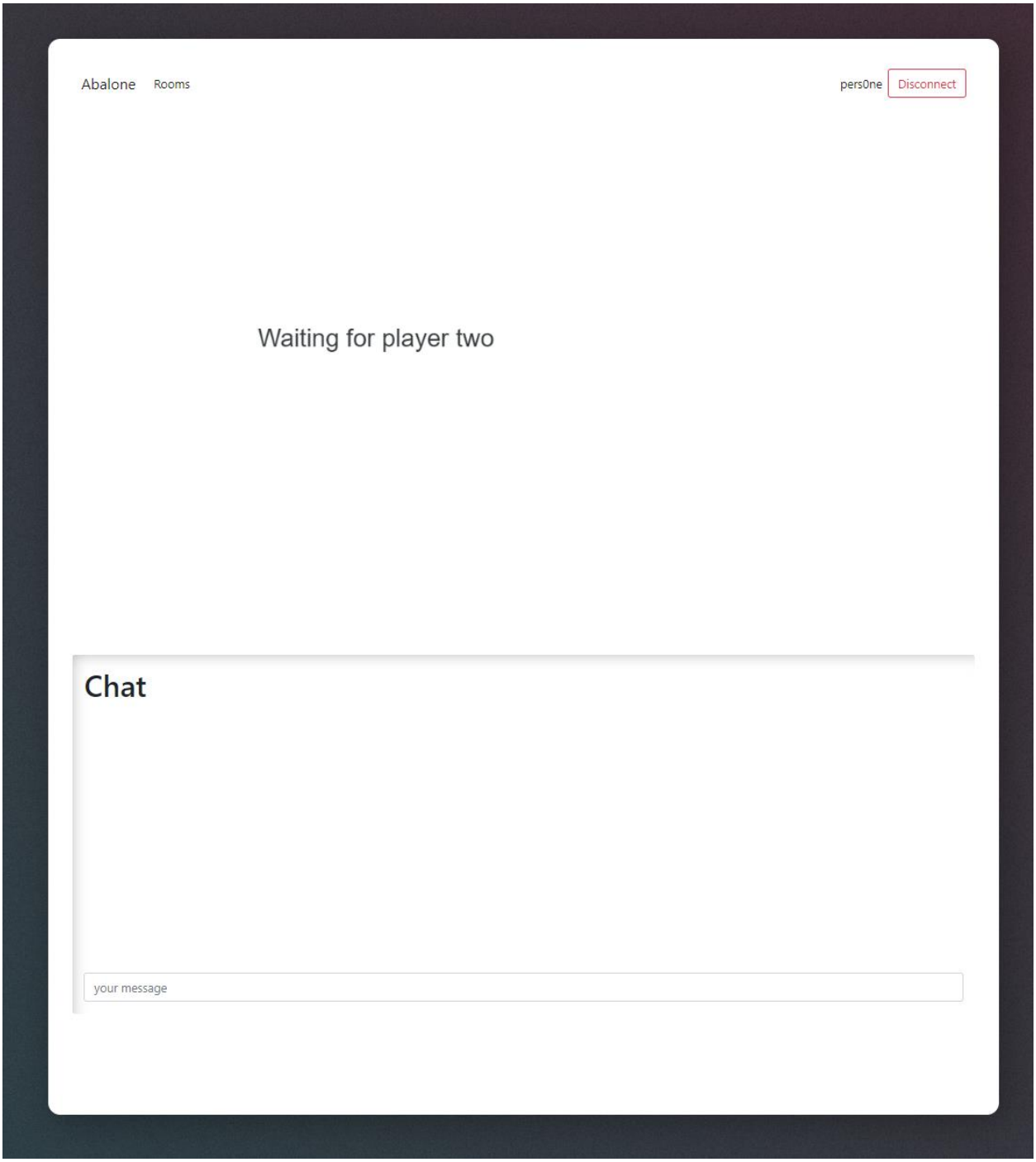

Abalone
Rooms
Login/Register
Rooms
+
Player 1
Player 2
Join

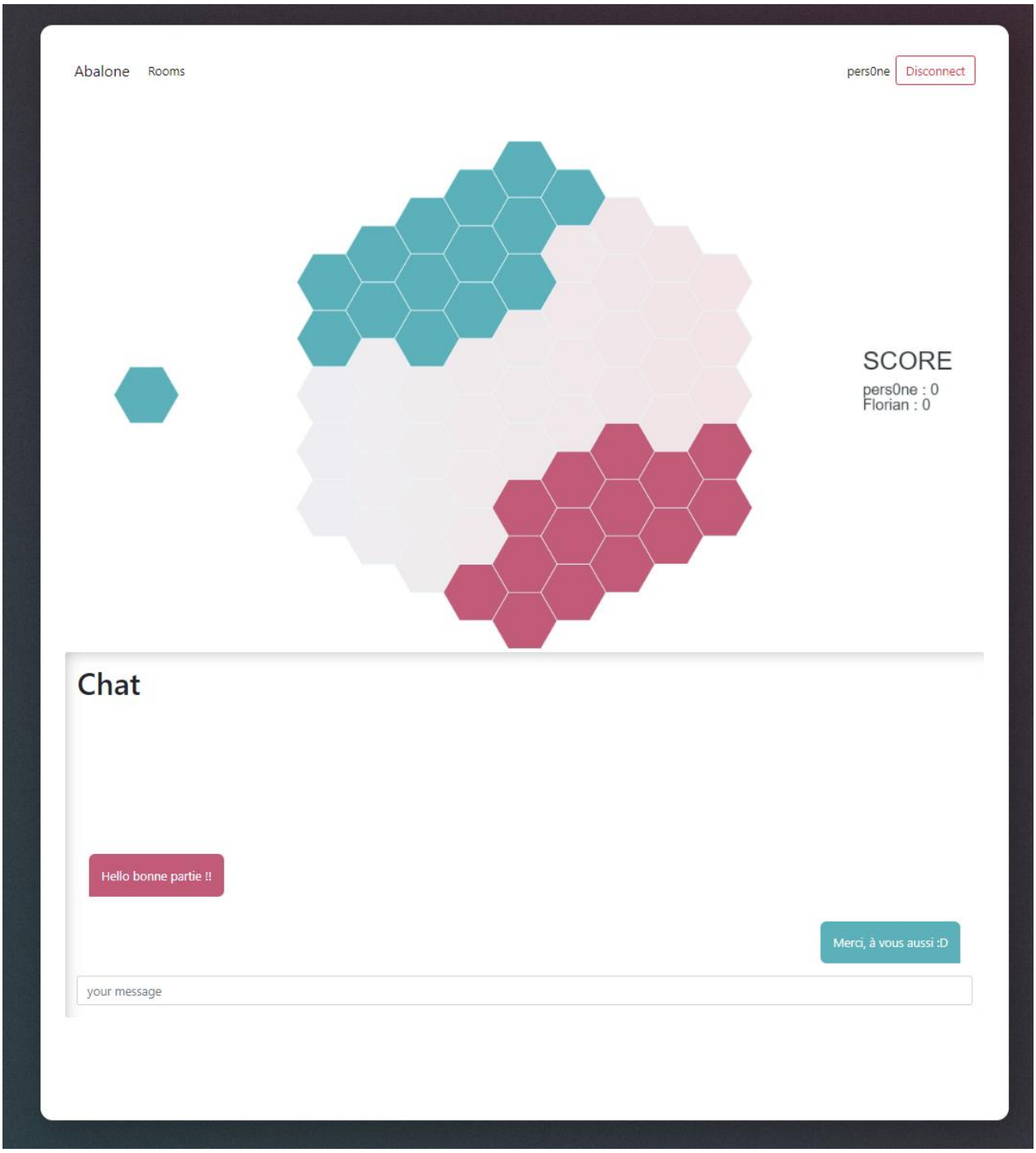

Rooms
+
Player 1
Player 2
Join
Florian
Join


Abalone
Rooms
Login/Register
Register
Username
Password
Retype your password
Register
Login


Abalone
Rooms
Login/Register
Register
Login
Username
Password
Login


Abalone
Rooms
pers0ne
Disconnect


Abalone
Rooms
pers0ne
Disconnect
Waiting for player two
Chat
your message


Abalone
Rooms
pers0ne
Disconnect
SCORE
pers0ne : 0
Florian : 0
Chat
Hello bonne partie !!
Merci, à vous aussi :D
your message

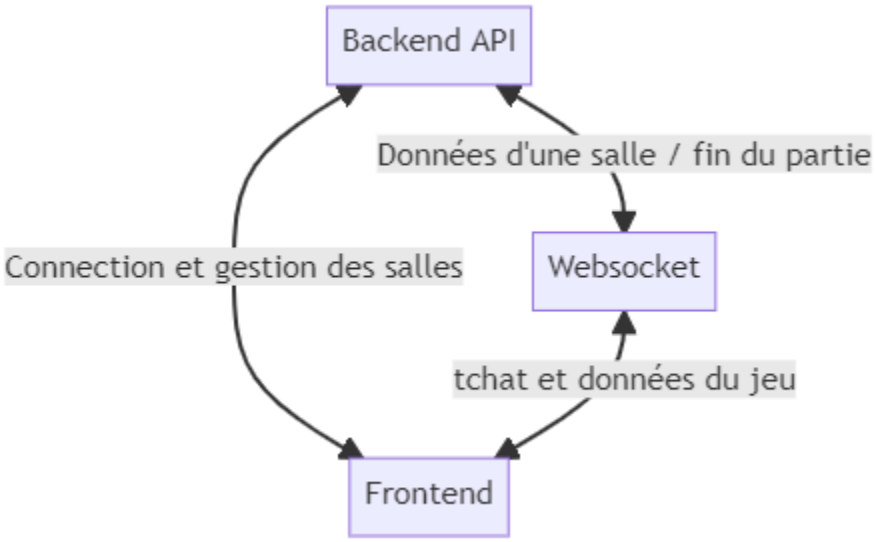
Architectures

Le projet est constitué de 3 parties, l'API, le serveur web socket et le serveur frontend.

Le serveur API sert de backend, il reprend absolument toute la logique de la gestion des utilisateurs, la gestion de la base de données, etc…

Le serveur web socket, lui ne gère que la transmission des données du jeu entre deux utilisateurs et le tchat en temps réel.

Le serveur frontend lui permet d'afficher le site web et contient la logique du jeu abalone en javascript.



API

La totalité de l'api utilise JSON.

Room

{
id:"",
player_1:"",
player_2:"",
passwd :"",
}

Méthode	Route	Paramètres	Sortie
Get	rooms/		Renvoie un tableau contenant la liste des salles de jeu

Post	rooms/add	{
		player : « name »
		token : « token »
		}
			Crée une salle de jeux
			
			Renvoie un objet room

Post	rooms/join	{
		player : « name »
		token : « token »
		roomId : « room id »
		}
			Permet de rejoindre une salle de jeu
			
			Renvoie un objet room

Post	rooms/start	{
		player : « name »
		roomId : « room id »
		}
			Permet démarrer une partie
			
			Renvoie vrai ou faut

Post	rooms/end	{
		player : « name »
		roomId : « room id »
		}
			Supprime la salle de jeu
			
			Renvoie vrai ou faut

Get	`user/s/{search}`	Le paramètre « search » peut être un nom ou une partie de nom	Renvoie une liste d'user
			
			`[`
			`  {`
			`  id: « id »`
			`  name : « name »`
			`  }`
			`]`

Get	`user/follow`	Les paramètres de d'authentification sont placés dans le header Il faut le header « name » et le header « token »	Renvoie une liste des utilisateurs qu'on suit
Get	`user/i/{id}`	Id d'un utilisateur	Renvoie un utilisateur

Post	`user/login`	`{`
		`  name: « name »,`
		`  password: « password »`
		`}`
			Renvoie un utilisateur avec un champ token

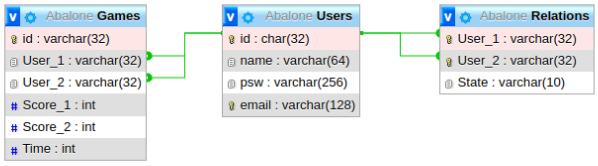
Post	`user/register`	`{`
		`  name: « name »,`
		`  password: « password »`
		`}`
			Renvoie un utilisateur

Post	`user/follow`	Il faut avoir l'authentification dans le header
		
		Il faut aussi avoir l'id d'utilisateur que l'on souhaite ajouter
		
		`{`
		`  id : « id »`
		`}`
			Renvoie vrai ou faut

Delete	`user/follow`	Il faut avoir l'authentification dans le header	Renvoie vrai ou faut

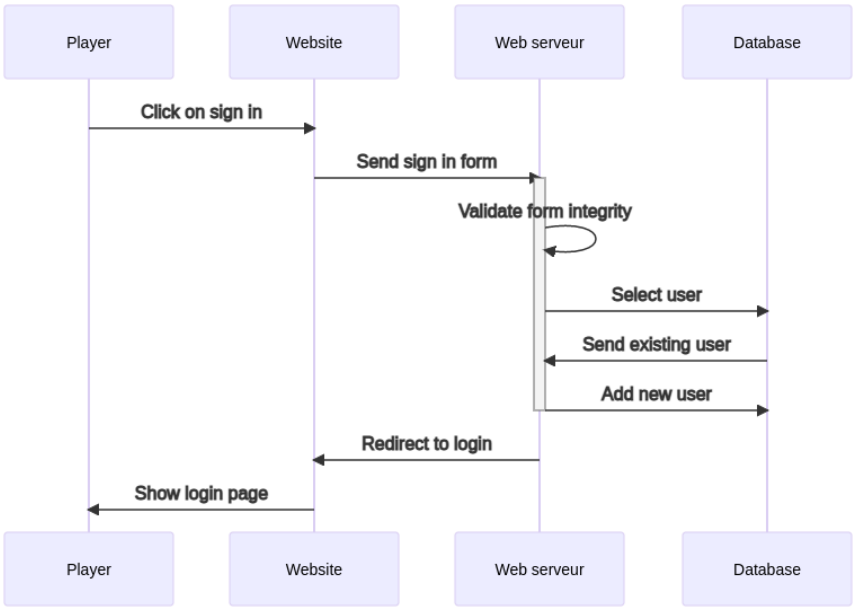
Base de données

Le but est d'enregistrer le strict minimum d'information (plus simple pour le GPRD). On ne sauvegarde pas les messages, cela demanderait du chiffrement. Un historique des messages n'est pas nécessaire (le site n'étant pas un réseau social)

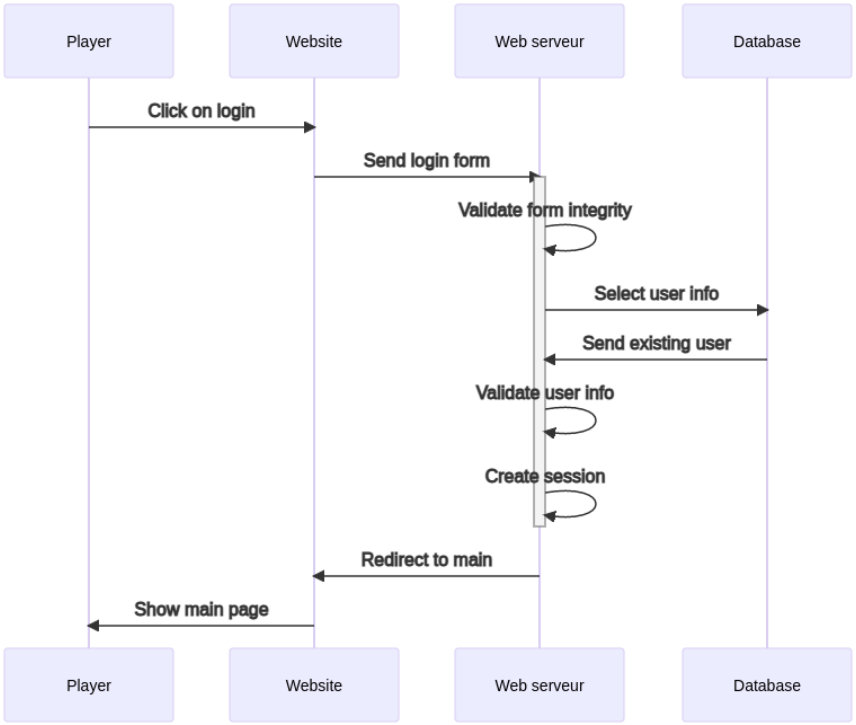


Diagrammes de séquence

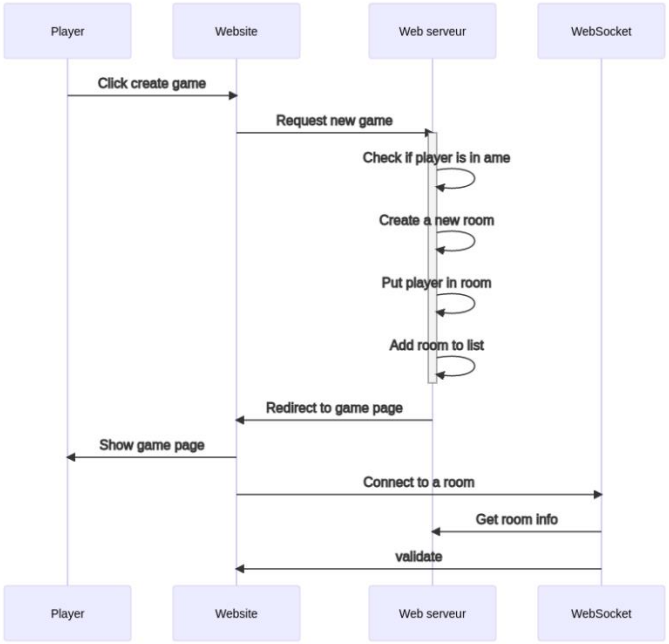
Création de compte



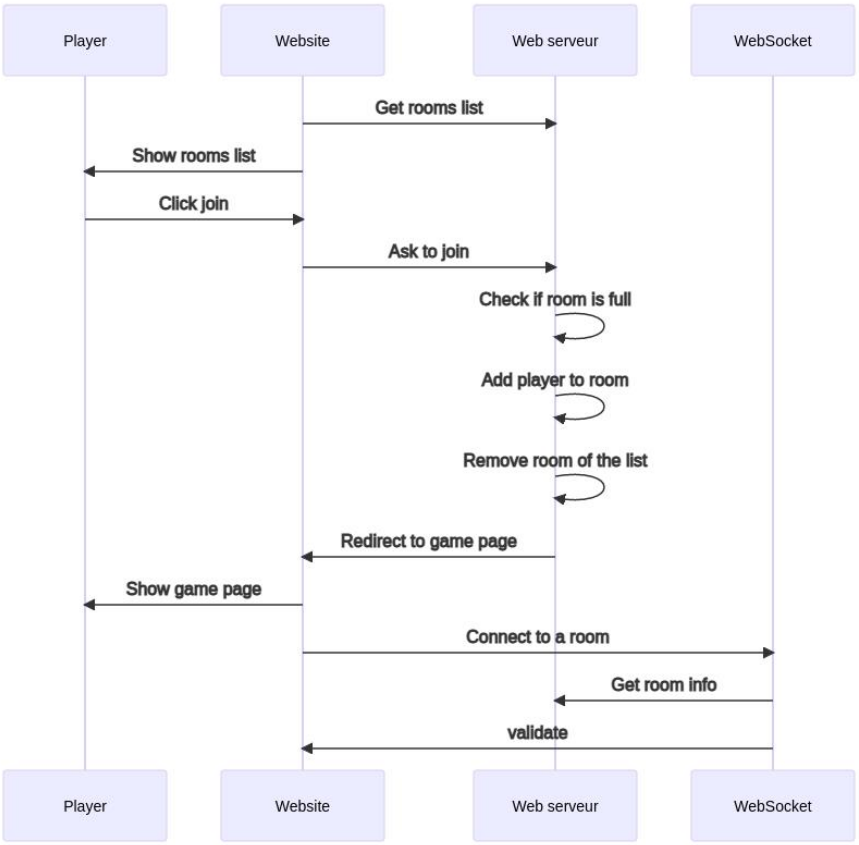
Connection



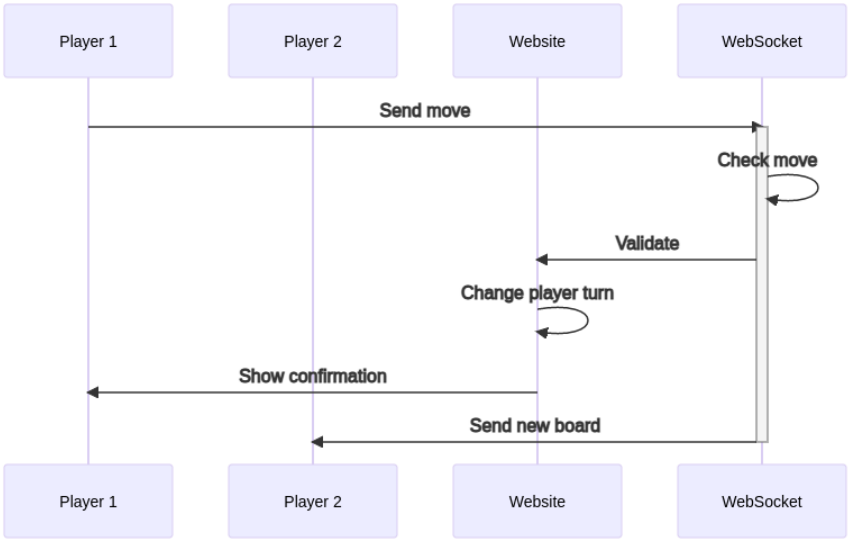
Créer un salon



Rejoindre une partie



Jouer un coup



Fin de partie