
UTChat

Étudiant: Rodrigo Uriel ZUNIGA TELLEZ

Responsable UV: Ahmed LOUNIS

Responsable TD: Ghada Jaber

CHAPITRE 1

RESUME

Ce rapport permet d'expliquer le fonctionnement d'une application web des chats. On utilise le modèle MVC (Section 2) dans l'architecture Web et les WebSockets (Section 2.1) afin d'établir une connexion avec le serveur du chat.

L'objectif est de créer une application Web qui permet à différentes personnes de créer des salles de discussion avec un horaire défini. Dans ces salles de discussion, les créateurs pourront ajouter des invités existants dans le système (en utilisant leur nickname).

C'est pourquoi dans un premier temps j'expliquerai la conception du projet, puis je montrerai l'installation de l'application ainsi qu'un scénario d'utilisation complet.

Enfin, je donnerai une petite conclusion et une réflexion personnelle sur l'importance dans le développement de ce type d'architecture et une éventuelle approche future que ce projet pourrait avoir à l'échelle industrielle.

TABLE DES MATIÈRES

1	Resume	2
2	Introduction	5
2.1	MVC - MODEL VUE CONTROLLER	5
2.2	WebSockets	6
3	Conception	7
3.1	UML - BDD	7
3.2	Model	8
3.2.1	User	8
3.2.2	Chat	8
3.3	Vue	9
3.4	Controller	9
3.4.1	Validation	9
3.4.2	Validation Register	9
3.4.3	Edit User	9
3.4.4	Delete User	10
3.4.5	Register Chat	10
3.4.6	Edit Chat	10
3.4.7	Disconnect	10
4	Installation	11
4.1	Clone	11
4.2	Import BDD	11
5	Utilisation	13
5.1	Login	13
5.2	Register	13
5.3	Home	14
5.4	My Chats	14
5.4.1	Create Chat	14
5.4.2	Modify Chat	15
5.5	Mes Invitations	16
5.6	Chat	16

5.7	Admin	16
6	Conclusion	17
7	Annexes	18
7.1	Singleton	18
7.2	Record Base	19

CHAPITRE 2

INTRODUCTION

Dans cette section, on expliquera des concepts de base dans le but de bien comprendre le projet.

Premièrement, on expliquera le modèle MVC (Section 2.1) et ensuite on passera à parler sur les WebSockets (Section 2.1) qui font la connexion entre le chat et le serveur qui gère les connections.

2.1 MVC - MODEL VUE CONTROLLER

Le MVC, est une technique de développement avancée devenue un design pattern, qui découpe l'application en 3 couches principales, nommées Modèle, Vue et Contrôleur. La distinction de ces couches :

- Facilite l'organisation des sources du proje.
- Permet à chaque corps de métier de travailler en parallèle sur les sources qui leur sont dédiées.
- Réduit l'impact des modifications pour minimiser les risques d'erreur et les régressions.

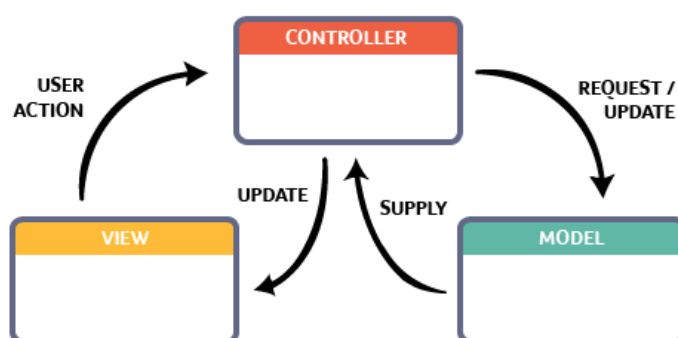


FIGURE 2.1 – MVC

Modèle : Cette partie gère les données de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.

Vue : Cette partie se concentre sur l’affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu’elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

Contrôleur : Cette partie gère la logique du code qui prend des décisions. C’est en quelque sorte l’intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C’est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d’accès).

2.2 WebSockets

Le protocole WebSocket est un protocole réseau basé sur le protocole TCP. Celui-ci définit la façon dont les données sont échangées entre les réseaux. En raison de sa fiabilité et de son efficacité, il est utilisé par presque tous les clients. TCP établit une connexion entre deux points finaux de communication qu’on appelle des sockets. Ainsi, une communication bidirectionnelle s’établit entre les données.

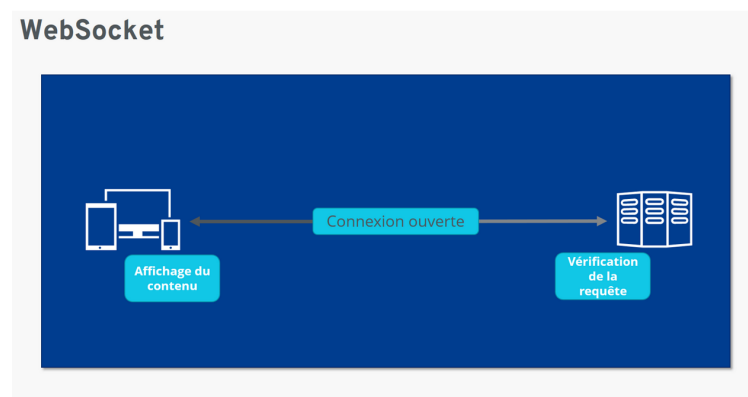


FIGURE 2.2 – WebSocket

L’utilisation d’un WebSocket permet la consultation dynamique en temps réel d’un site Web. Avec le protocole WebSocket, il suffit au client d’établir la connexion avec un serveur Web. La connexion entre le client et le serveur s’établit grâce à la phase de handshake du protocole WebSocket. Dans ce cas, le client envoie toutes les identifications nécessaires à l’échange de données au serveur.

CHAPITRE 3

CONCEPTION

Une fois compris l'utilisation du modèle MVC (Section 2) et des Websockets (Section 2.1) on peut maintenant expliquer la conception du projet prévu au niveau de chaque étape de l'architecture.

Tout d'abord, il est important de montrer un diagramme **UML** de la Base de Données. Ensuite on pourra montrer comment a été divisé le projet en classes.

3.1 UML - BDD

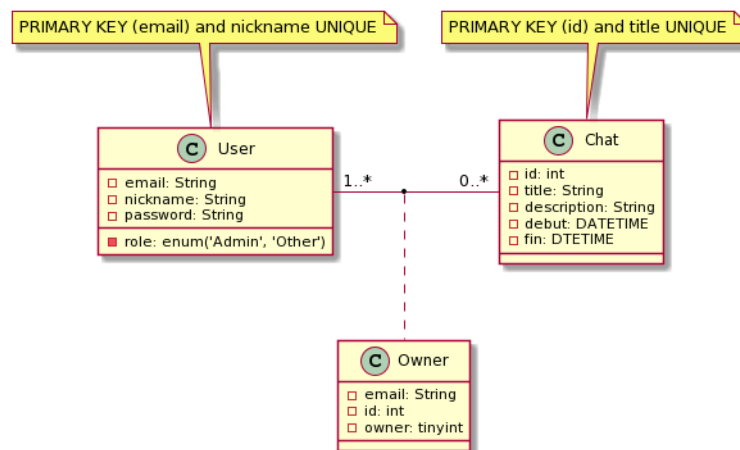


FIGURE 3.1 – UML - BDD

On utilise 3 tableaux :

- **User** : Gère tous les utilisateurs dans le système.
- **Chat** : Gère les chats créés par les utilisateurs.
- **Owner** : Permet de lier à chaque chat la liste des utilisateurs. Soit invité, soit créateur.

3.2 Model

Comme on a vu dans l'introduction, le model va permettre à l'application de créer des Utilisateurs et Chats.

3.2.1 User

La class User (Java Bean) hérite directement du Active Record Base class (Section 7.1).

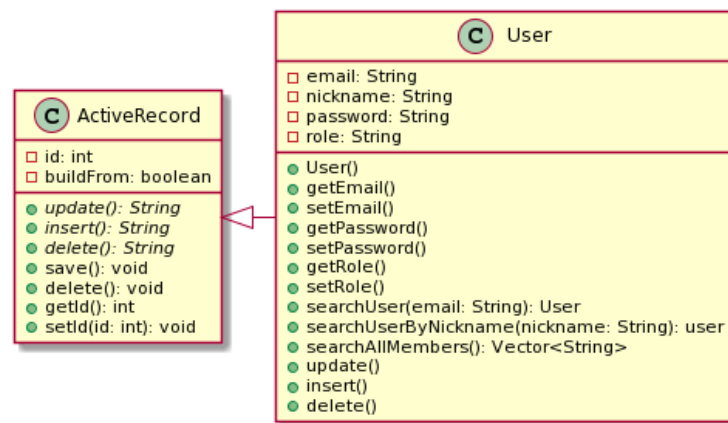


FIGURE 3.2 – User UML

User doit surcharger les méthodes *update*, *insert* et *delete* afin de redéfinir le comportement de la superclasse. D'une autre part, les fonctions pour chercher des utilisateurs selon le nickname ou l'email aident à la page web à gestion les utilisateurs en cherchant dans la BDD.

3.2.2 Chat

La class Chat (Java Bean) hérite directement du Active Record Base class (Section 7.1).

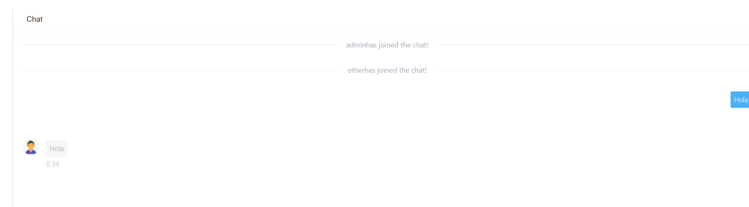


FIGURE 3.3 – Chat UML

La class User doit surcharger les méthodes *update*, *insert* et *delete* afin de redéfinir le comportement de la superclasse. D'une autre part, les fonctions pour chercher des chats selon le titre ou la liste complète des chats par utilisateur, font des requêtes à la BDD.

3.3 Vue

Dans une application web, les vues sont faites avec HTML, plus spécifiquement, dans ce projet, la partie Vue est développée en utilisant des fichiers JSP qui permettent de joindre le code Java à l'intérieur des balises HTML. Ça permet de créer des Web pages Dynamiques.

Cela n'a pas beaucoup de sens de décrire chacune des vues utilisées dans ce rapport, plus tard, dans le scénario d'utilisation, elles seront mieux décrites.

3.4 Controller

Les contrôleurs de cette application sont utilisés pour valider les utilisateurs, enregistrer des informations dans la base de données, les modifier ou les supprimer, ainsi que pour déconnecter les users.

3.4.1 Validation

Une fois que les utilisateurs ont été ajoutés à la base de données, ce servlet permet de faire une requête à la base de données pour savoir si les informations d'identification saisies sont correctes et correspondent à n'importe quel utilisateur.

En cas de succès, il redirige vers la page d'accueil, en cas d'échec, il revient à la page de connexion.

Les variables de session sont définies pour être utilisées dans toute l'application jusqu'à la fermeture de la session.

3.4.2 Validation Register

Lors de l'inscription d'un utilisateur, certaines informations sont demandées à la personne, comme c'est le cas de son adresse e-mail, de son pseudo et de son mot de passe.

Une requête est faite au serveur pour savoir si l'utilisateur à enregistrer est valide (s'il existe déjà, il retourne à la page d'accueil pour informer que ledit utilisateur existe déjà dans la base de données) et enfin la session est démarrée avec le nouvel enregistrement.

3.4.3 Edit User

D'autre part, les administrateurs doivent pouvoir modifier les utilisateurs existants. C'est pourquoi ce contrôleur aide en effectuant une UPDATE de l'enregistrement sélectionné et en modifiant les champs souhaités.

3.4.4 Delete User

Les admins peuvent aussi supprimer des utilisateurs, une requete du type DELETE sur la BDD est effectue lorsqu'on sélectionne un user dans l'application web.

L'Active Record (Section 7.1) permet d'exécuter ce requête.

3.4.5 Register Chat

Lors de l'enregistrement d'un chat, il est nécessaire de faire une série d'insertions dans la base de données. Il existe un tableau qui relie les utilisateurs aux chats, c'est pourquoi chaque fois que nous créons un chat, nous devons sélectionner les invités et les ajouter un par un à la classe d'association Owner (Voir 3.1).

3.4.6 Edit Chat

Il est important, au même titre que les membres, de modifier les chats créés.

Cette option n'est accessible qu'aux créateurs et ils peuvent modifier les dates de début et de fin prévues.

3.4.7 Disconnect

De la même manière que pour la connexion il doit y avoir un contrôleur chargé de déconnecter l'utilisateur actuel.

Ce servlet vous permet d'effacer les variables de session précédemment définies et de vous rediriger vers la page de connexion.

CHAPITRE 4

INSTALLATION

Ce chapitre cherche à expliquer comment utiliser ce petit chat en utilisant le dépôt Git.

4.1 Clone

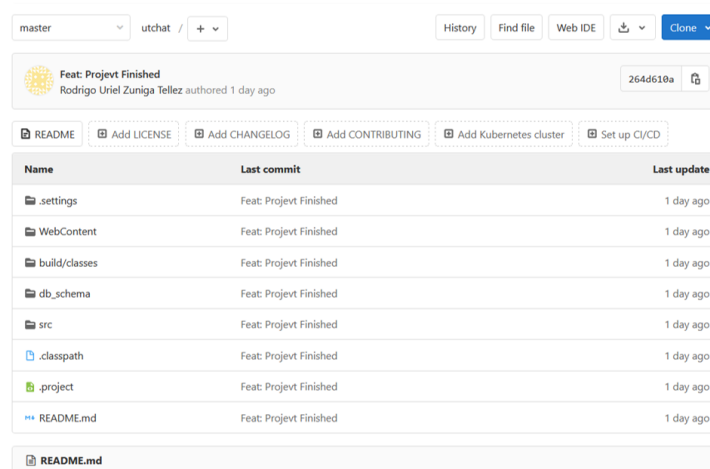


FIGURE 4.1 – Git Clone

Pour cloner ce projet, vous devez utiliser la commande `Git clone` suivie de l'URL du projet.

Une fois cela fait, un dépôt sera créé sur votre ordinateur avec tous les fichiers du projet sur la branche **master**.

4.2 Import BDD

Deuxièmement on doit créer une base de données en faisant l'import du fichier *chat.sql* qui est loué dans le répertoire *db schema* (Figure 4.2) dans la racine du projet. Ce fichier contient la création

de la BDD et les 3 tableaux utilisée (Section 3.1) ainsi comme 2 utilisateurs *admin* et *other* par défaut.

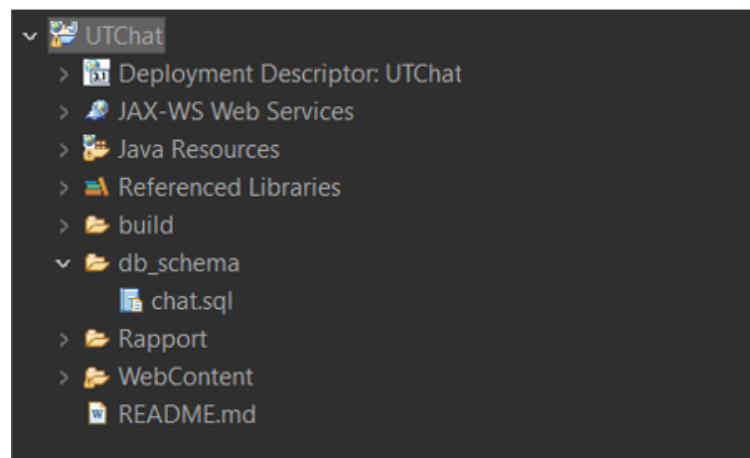


FIGURE 4.2 – Répertoire db schema

Après avoir effectué les deux étapes précédentes, nous pouvons passer au scénario d'utilisation.

CHAPITRE 5

UTILISATION

Après avoir compris la conception et les concepts de base, on peut maintenant passer à l'explication de l'application en général.

5.1 Login

Au démarrage de l'application, la première chose que nous verrons sera la fenêtre de *login*. Ici, nous pouvons utiliser l'un des deux utilisateurs par défaut ajoutés à la base de données **Admin** ou **Other**.

Si vous souhaitez ajouter un nouvel utilisateur, cliquez simplement sur le lien en bas pour accéder à la page d'Register (Section 5.1).

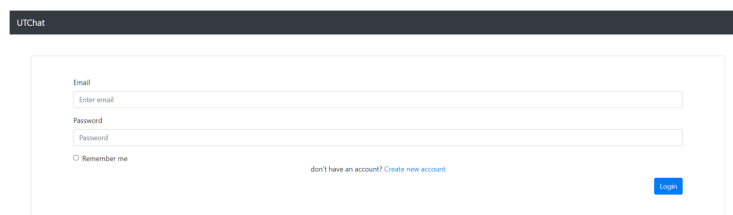
The image shows a web browser window with a dark header bar labeled "UTChat". Below the header is a white login form. The form contains two input fields: "Email" with a placeholder "Enter email" and "Password" with a placeholder "Password". Below the password field is a checkbox labeled "Remember me". At the bottom of the form, there is a link that says "don't have an account? Create new account" and a blue "Login" button.

FIGURE 5.1 – Login

5.2 Register

Pour enregistrer un utilisateur, il est nécessaire de saisir une adresse e-mail au format valide ainsi que de confirmer le mot de passe pour éviter tout type d'erreur. Ce n'est que lorsque les mots de passe correspondent et que les champs sont remplis que le bouton d'envoi sera activé.

Registration form with the following fields:

- Email: Enter email
- Username: Enter email
- Password: Password
- Password: Password
- Password: Password

Register button

FIGURE 5.2 – Register

5.3 Home

Une fois entré, nous verrons un menu avec 3 options ou 2 selon nos droits en tant qu'utilisateur. Les administrateurs auront accès à l'option d'admin (Section 5.6) tandis que les autres n'auront accès qu'à leurs invitations et à leurs chats de discussions créées.

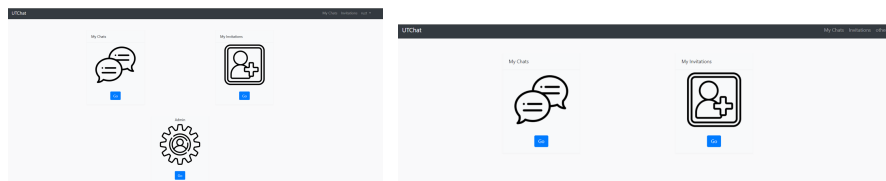


FIGURE 5.3 – Home Page

5.4 My Chats

Les discussions sont le moteur de cette application. Les utilisateurs doivent pouvoir créer, modifier et entrer dans le canal de discussion chaque fois que possible dans les intervalles de temps choisis.

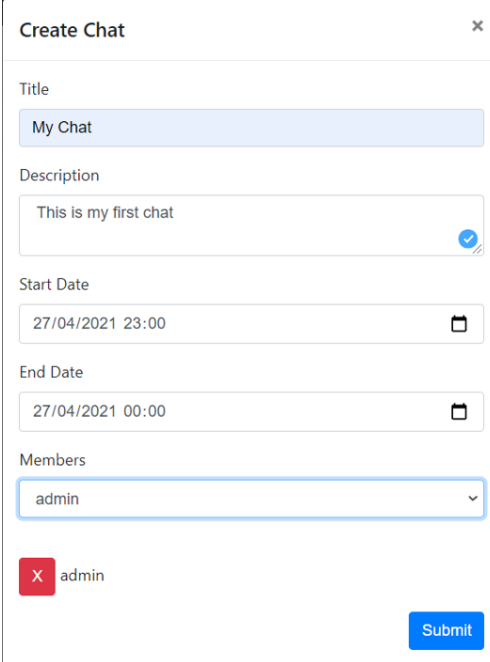
La figure suivante montre la page qui contient un tableau avec tous les chats liés à ce créateur.

UTChat						
My Chats						Create Chat
	Title	Description	Debut	Fin	Edit	Enter
1	MyChat	Thisismyfirstchat	2020-12-30T20:12	2021-12-28T00:00	Edit	Enter

FIGURE 5.4 – My Chats

5.4.1 Create Chat

Pour créer un chat, il est nécessaire de saisir un nom de chat qui sera unique, une description et les dates de début et de fin avec leur heure respective (**Attention, google chrome prend en charge l'entrée Datetime qui est utilisée dans toute l'application.**)

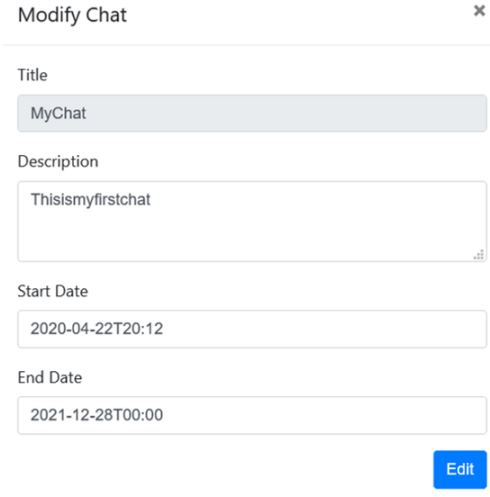


The 'Create Chat' form is a modal window with a title bar 'Create Chat' and a close button. It contains several input fields: 'Title' with the value 'My Chat', 'Description' with the value 'This is my first chat', 'Start Date' with the value '27/04/2021 23:00', and 'End Date' with the value '27/04/2021 00:00'. There is a 'Members' dropdown menu with 'admin' selected. Below the dropdown, there is a red square with a white 'X' and the text 'admin'. A blue 'Submit' button is at the bottom right.

FIGURE 5.5 – Create Chat

5.4.2 Modify Chat

Les chats doivent être modifiables, si à tout moment le créateur souhaite modifier la date de début ou de fin, il doit pouvoir le faire simplement en cliquant sur le bouton bleu dans la liste des chats puis en modifiant le champ souhaité. Lors de la soumission du nouveau formulaire, les modifications seront enregistrées dans la base de données.



The 'Modify Chat' form is a modal window with a title bar 'Modify Chat' and a close button. It contains several input fields: 'Title' with the value 'MyChat', 'Description' with the value 'Thisismyfirstchat', 'Start Date' with the value '2020-04-22T20:12', and 'End Date' with the value '2021-12-28T00:00'. A blue 'Edit' button is at the bottom right.

FIGURE 5.6 – Modify Chat

5.5 Mes Invitations

Tous les chats qui ne sont pas créés par vous-même apparaîtront dans la section des invitations. Cette nouvelle page est similaire à celle de la section précédente à la différence que nous ne pouvons pas modifier les discussions, la seule chose que nous pouvons faire est d'entrer dans la salle lorsque le temps est entre les intervalles de début et de fin.

My Invitations					
	Title	Description	Debut	Fin	Enter
1	MyChat	Thisismyfirstchat	2020-12-30 - 20:12:00	2021-12-28 - 00:00:00	<button>Enter</button>

FIGURE 5.7 – Mes Invitations

5.6 Chat

Lorsque nous entrons dans une salle de discussion, nous pouvons envoyer et recevoir des messages en nous connectant aux WebSockets.

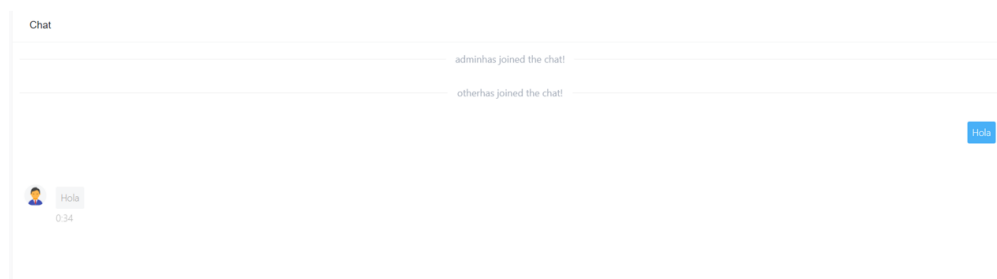


FIGURE 5.8 – Chat

5.7 Admin

Seuls les administrateurs auront accès à cette section, ils pourront voir ici la liste de tous les utilisateurs avec leurs correspondantes, pouvant modifier chacune des données qui leur sont liées directement dans la base de données.

Email	Nickname	Role	Edit	Delete
admin@hotmail.com	admin	Admin	<button>Edit</button>	<button>X</button>
cesar@hotmail.com	cesar	Other	<button>Edit</button>	<button>X</button>
other@hotmail.com	other	Other	<button>Edit</button>	<button>X</button>

FIGURE 5.9 – Admin

CHAPITRE 6

CONCLUSION

L'architecture MVC est l'une des plus utilisées par les développeurs web, car elle permet de structurer le projet en 3 grandes sections qui ont des objectifs différents et communiquent entre elles, offrant de grandes performances et un environnement formidable pour l'utilisateur.

D'autre part, l'utilisation de Java comme langage permet de gérer le Backend d'une application en utilisant le parallélisme et la puissance offerts par toutes ses bibliothèques.

L'utilisation de conceptions parentales telles que Singleton et Active Record est essentielle pour l'architecture parfaite du système car elles vous permettent de gérer des instances uniques et d'hériter des classes enfants pour le modèle au sein du MVC.

7.1 Singleton

Le modèle Singleton est l'un des modèles de conception les plus simples de Java. Ce type de modèle de conception relève du modèle de création car ce modèle fournit l'un des meilleurs moyens de créer un objet.

Ce modèle implique une seule classe qui est chargée de créer un objet tout en s'assurant qu'un seul objet est créé. Cette classe fournit un moyen d'accéder à son seul objet auquel on peut accéder directement sans avoir besoin d'instancier l'objet de la classe.

Nous allons créer une classe `SingleObject`. La classe `SingleObject` a son constructeur privé et une instance statique d'elle-même.

La classe `SingleObject` fournit une méthode statique pour obtenir son instance statique vers le monde extérieur. `SingletonPatternDemo`, notre classe de démonstration utilisera la classe `SingleObject` pour obtenir un objet `SingleObject`.

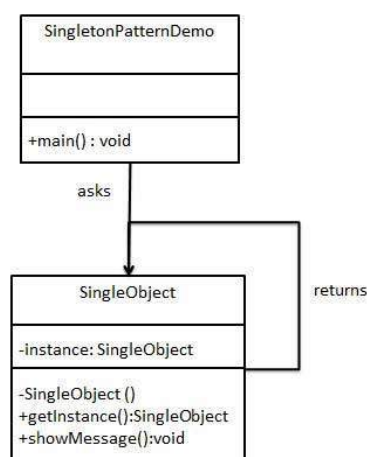


FIGURE 7.1 – Singleton

7.2 Record Base

Même si DAO peut gérer n'importe quelle tâche relative à la base de données, il est fort probable qu'une grande partie du temps de développement soit consacré à l'écriture de requêtes SQL pour faire des opérations CRUD (create, read, update and delete) courantes. De plus il est difficile de maintenir un code quand celui est parsemé de requêtes SQL. Active Record est là pour régler ces problèmes.

Active Record (AR) est une technique très utilisée de Object-Relational Mapping (ORM). Chaque classe AR représente une table dans la base, dont les attributs représente une propriété de cette classe AR, et une instance AR représente une ligne de cette table. Les opérations couramment utilisées sont implémentées dans des méthodes de cette classe. Ainsi il est possible d'accéder aux données d'une manière orientée objet.

TABLE DES FIGURES

2.1	MVC	5
2.2	WebSocket	6
3.1	UML - BDD	7
3.2	User UML	8
3.3	Chat UML	8
4.1	Git Clone	11
4.2	Repertoire db schema	12
5.1	Login	13
5.2	Register	14
5.3	Home Page	14
5.4	My Chats	14
5.5	Create Chat	15
5.6	Modify Chat	15
5.7	Mes Invitations	16
5.8	Chat	16
5.9	Admin	16
7.1	Singleton	18