

# AP4 NetBeans

Ethane Zimmermann, Stephan Marcelo,  
Mateo Sage

---

## Table des matières

Présentation générale .....	2
1.1. Besoin initial .....	2
1.2. Objectifs du projet.....	2
1.3. Solution proposée .....	2
Conception de l'interface utilisateur .....	3
2.1. Plan du site / Arborescence fonctionnelle .....	3
2.2. Maquettes de l'application .....	3
Modélisation des données .....	5
3.1. Dictionnaire de données.....	5
3.2. MCD (Modèle Conceptuel de Données) .....	5
3.3. MLD (Modèle Logique de Données).....	5
Descriptif technique de l'application.....	6
4.1. Technologies et langages utilisés .....	6
4.2. Architecture de l'application .....	6
4.3. Organisation du code (MVC, DAO, etc.) .....	6
Guide d'utilisation .....	7
5.1. Lancement de l'application .....	7
5.2. Fonctionnalités principales .....	7
5.3. Scénarios d'usage (optionnel mais recommandé) .....	8
Répartition des tâches.....	13
6.1. Membres de l'équipe et rôles attribués.....	13
6.2. Description des tâches effectuées .....	13
Planification et gestion de projet .....	16
7.1. Outils de planification utilisés (Gantt) .....	16
Retour sur l'expérience projet .....	16
8.1. Déroulement global.....	16
8.2. Difficultés rencontrées .....	16
Conclusion.....	18
9.1. Résumé de l'expérience de chaque personne du groupe .....	18

# Présentation générale

## 1.1. Besoin initial

Dans le cadre de notre formation, nous avons reçu pour consigne de développer une application Java en utilisant l'environnement de développement NetBeans. Le projet porte sur la réalisation d'une application en **client lourd** permettant d'effectuer des opérations de base sur une base de données. Le besoin initial est de **gérer efficacement les utilisateurs** d'une table dédiée, en assurant les opérations de **Create, Read, Update et Delete (CRUD)**. Cette application a pour but de simuler un système de gestion d'utilisateurs, souvent utilisé dans les systèmes de gestion internes d'une entreprise comme dans le cadre D'Amset qui est une SSII (Société de Services et d'Ingénierie en Informatique) qui répond aux besoins de ses clients (organisations privées ou publiques) dans le domaine des nouvelles technologies et de l'informatique.

## 1.2. Objectifs du projet

- Développer une interface graphique conviviale en Java avec Swing
- Implémenter les fonctions CRUD complètes sur une table utilisateur dans une base de données
- Maîtriser le développement en client lourd sous NetBeans.

## 1.3. Solution proposée

L'application proposée est un programme Java en client lourd développé sous NetBeans. Elle utilise Swing pour l'interface utilisateur et se connecte à une base de données

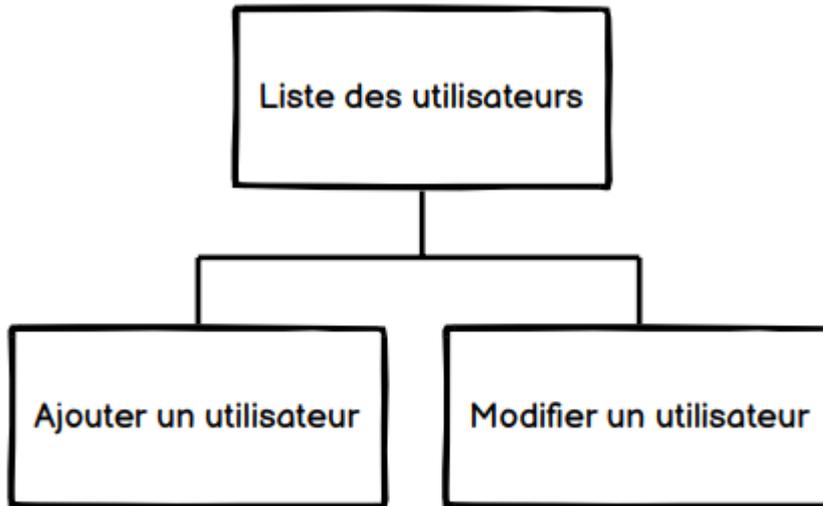
Les principales fonctionnalités incluent :

- L'ajout de nouveaux utilisateurs avec des champs comme nom, prénom, email, mot de passe, etc.
- L'affichage des utilisateurs dans un tableau interactif.
- La modification des informations d'un utilisateur existant.
- La suppression d'un utilisateur sélectionné.
- Des messages de confirmation et des alertes

L'application permet donc de gérer facilement les utilisateurs dans un environnement graphique simple

# Conception de l'interface utilisateur

## 2.1. Plan du site / Arborescence fonctionnelle



## 2.2. Maquettes de l'application

Liste des utilisateurs

Nom	Prenom	Users	Mot de passe	Emails
Zimmermann	Ethane	Zim.ethane	Eth@n3Skis975	Ethane@hotmail.com
Zimmermann	Ethane	Zim.ethane	Eth@n3Skis975	Ethane@hotmail.com
Zimmermann	Ethane	Zim.ethane	Eth@n3Skis975	Ethane@hotmail.com

### Ajouter un utilisateur

Nom

Prenom

Users

Mot de passe

Emails

### Modifier un utilisateur

Nom

Prenom

Users

Mot de passe

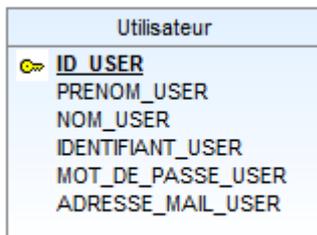
Emails

# Modélisation des données

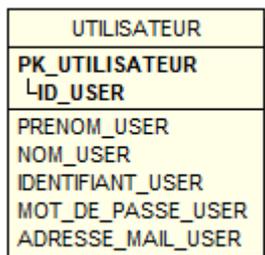
## 3.1. Dictionnaire de données

Nom du champ	Type de données	Taille	Contraintes	Description
ID_USER	INT	11	PRIMARY KEY, AUTO_INCREMENT	Identifiant unique de l'utilisateur
PRENOM_USER	VARCHAR(50)	50	NOT NULL	Nom de famille de l'utilisateur
NOM_USER	VARCHAR(50)	50	NOT NULL	Prénom de l'utilisateur
IDENTIFIANT_USER	VARCHAR(100)	100	UNIQUE, NOT NULL	Adresse email de l'utilisateur
MOT_DE_PASSE_USER	VARCHAR(50)	50	NOT NULL	Nom d'utilisateur pour la connexion
ADRESSE_MAIL_USER	VARCHAR(100)	100	NOT NULL	Mot de passe de l'utilisateur

## 3.2. MCD (Modèle Conceptuel de Données)



## 3.3. MLD (Modèle Logique de Données)



# Descriptif technique de l'application

## 4.1. Technologies et langages utilisés

L'application est développée en **Java** à l'aide de l'environnement de développement **NetBeans**. Il s'agit d'une application en **client lourd**, avec une interface graphique créée à l'aide de **Swing**.

## 4.2. Architecture de l'application

L'application adopte une architecture basée sur le modèle **MVC (Modèle – Vue – Contrôleur)**, une approche classique pour séparer la logique métier, la gestion des données et l'interface utilisateur. Cela permet une meilleure organisation du code et facilite la maintenance.

- **Modèle (Model)** : représente les entités métiers et contient la logique de manipulation des données. Par exemple, la classe Utilisateur et les classes DAO associées pour effectuer les opérations CRUD.
- **Vue (View)** : gère l'interface graphique affichée à l'utilisateur. Elle comprend les fenêtres, les formulaires et les tableaux créés avec Swing.
- **Contrôleur (Controller)** : fait le lien entre la vue et le modèle. Il intercepte les actions de l'utilisateur (clics, formulaires) et appelle les méthodes du modèle en conséquence.

## 4.3. Organisation du code (MVC, DAO, etc.)

Le code est organisé en plusieurs packages selon les rôles des classes, en suivant les principes du **MVC** et du **pattern DAO (Data Access Object)** :

- **Model** : contient les classes Java représentant les entités comme Utilisateur.
- **Dao** : contient les classes DAO telles que UtilisateurDAO qui encapsulent les opérations JDBC (insertion, mise à jour, suppression, recherche...).
- **View** : regroupe les interfaces graphiques créées avec Swing (fenêtres, formulaires, boutons...).
- **Controller** : contient les classes qui gèrent la logique de navigation et les interactions entre les vues et les modèles.

Ce découpage assure une **bonne séparation des responsabilités** et rend le projet **modulaire, réutilisable et évolutif**.

# Guide d'utilisation

## 5.1. Lancement de l'application

On lance le programme en double-cliquant sur le fichier .jar du projet, ce qui ouvre une fenêtre. Depuis cette fenêtre, on accède directement à la liste des utilisateurs. On peut alors sélectionner un utilisateur pour le modifier ou le supprimer. Il est également possible d'ajouter un nouvel utilisateur en cliquant sur le bouton "Ajouter" situé en haut à droite.

## 5.2. Fonctionnalités principales

### **Ajouter :**

Pour ajouter un utilisateur, cliquez sur le bouton en haut à droite. Une nouvelle fenêtre s'ouvrira sur la page "Ajouter un utilisateur". Remplissez le formulaire, puis cliquez sur "Ajouter" ou "Valider" pour enregistrer l'utilisateur.

### **Modifier :**

Pour modifier un utilisateur, sélectionnez l'utilisateur souhaité, puis cliquez sur le bouton "Modifier". Un message de confirmation s'affichera pour valider votre action. Si vous confirmez, une nouvelle fenêtre apparaîtra avec le formulaire prérempli contenant les informations de l'utilisateur. Vous pouvez alors modifier les informations, puis cliquer sur "Modifier" ou "Valider" pour enregistrer les changements.

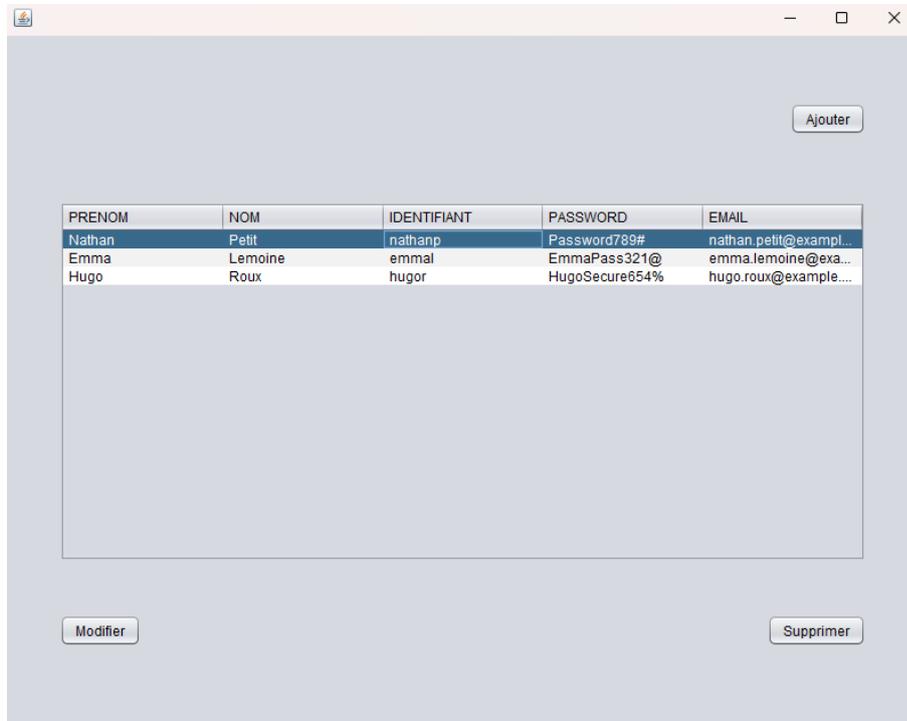
### **Supprimer :**

Pour supprimer un utilisateur, sélectionnez l'utilisateur concerné et cliquez sur le bouton "Supprimer". Un message de confirmation apparaîtra pour valider la suppression. Cliquez ensuite sur "Valider" pour supprimer définitivement l'utilisateur.

### 5.3. Scénarios d'usage (optionnel mais recommandé)

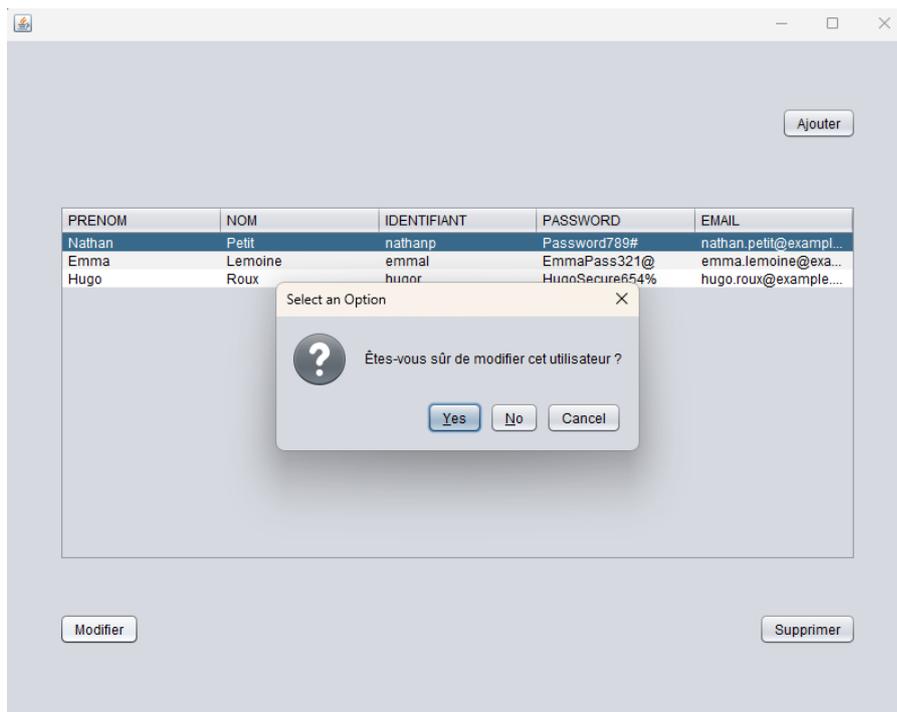
#### Modification :

Sélectionnez l'utilisateur que vous souhaitez modifier :

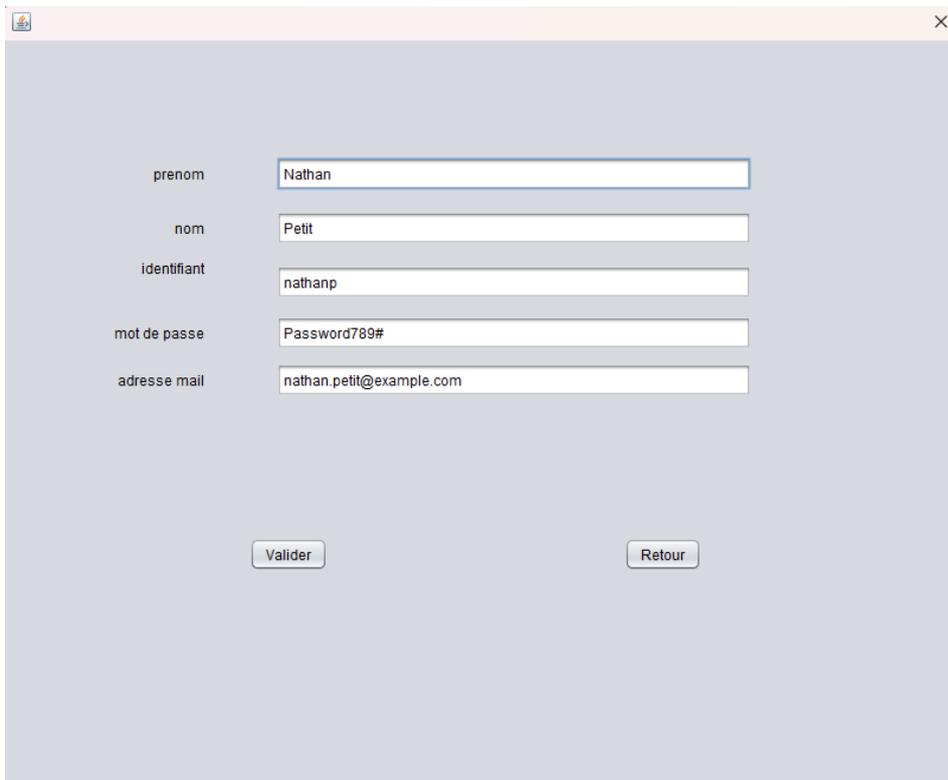


Cliquez ensuite sur le bouton **Modifier**. Un message de confirmation apparaîtra.

Cliquez sur **Yes** pour confirmer. Une nouvelle page s'ouvrira pour procéder à la modification de l'utilisateur :

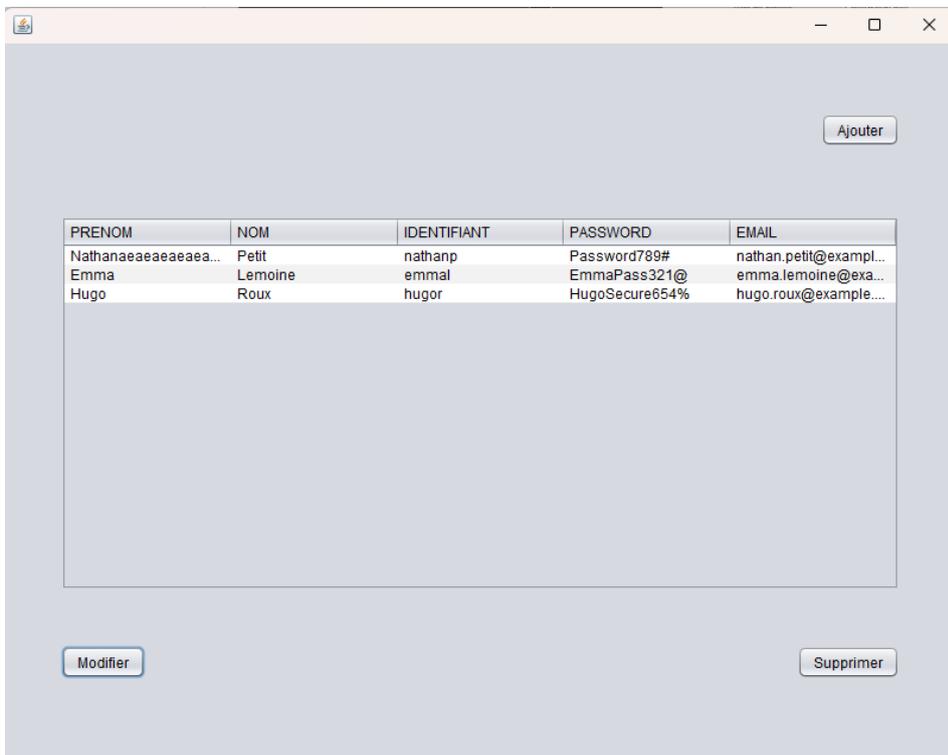


Après avoir effectué les modifications souhaitées, cliquez sur **Valider** :



A screenshot of a web application window showing a form for updating user information. The form has five input fields, each with a label to its left: 'prenom' (first name) with 'Nathan', 'nom' (last name) with 'Petit', 'identifiant' (username) with 'nathanp', 'mot de passe' (password) with 'Password789#', and 'adresse mail' (email) with 'nathan.petit@example.com'. Below the form are two buttons: 'Valider' (Validate) and 'Retour' (Return).

Vous verrez alors que votre modification a bien été prise en compte :

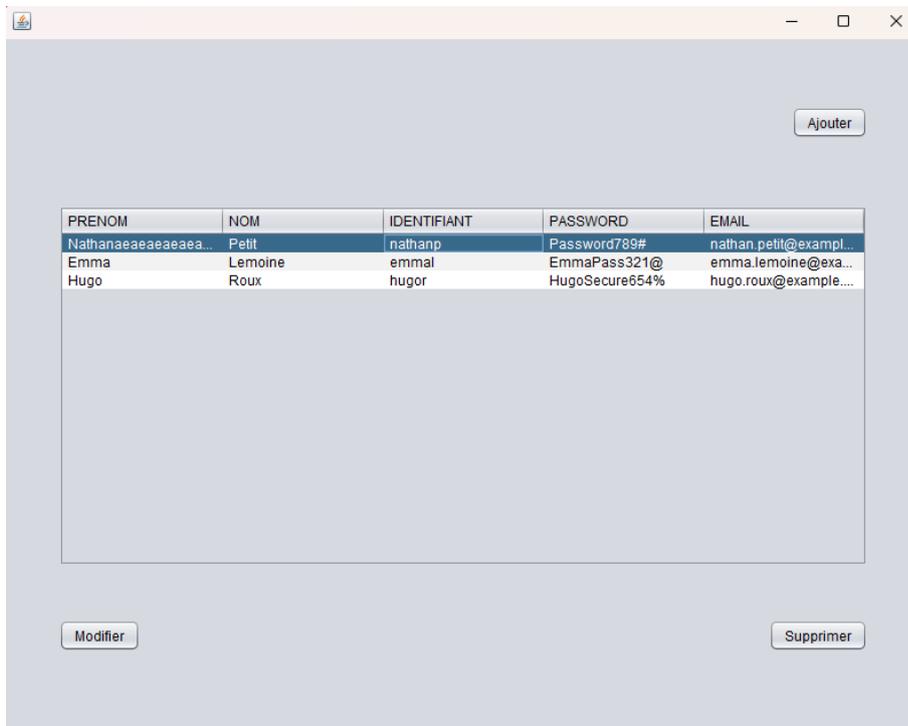


A screenshot of a web application window showing a table of user records. The table has five columns: 'PRENOM', 'NOM', 'IDENTIFIANT', 'PASSWORD', and 'EMAIL'. There are three rows of data. Below the table are three buttons: 'Ajouter' (Add) at the top right, 'Modifier' (Modify) at the bottom left, and 'Supprimer' (Delete) at the bottom right.

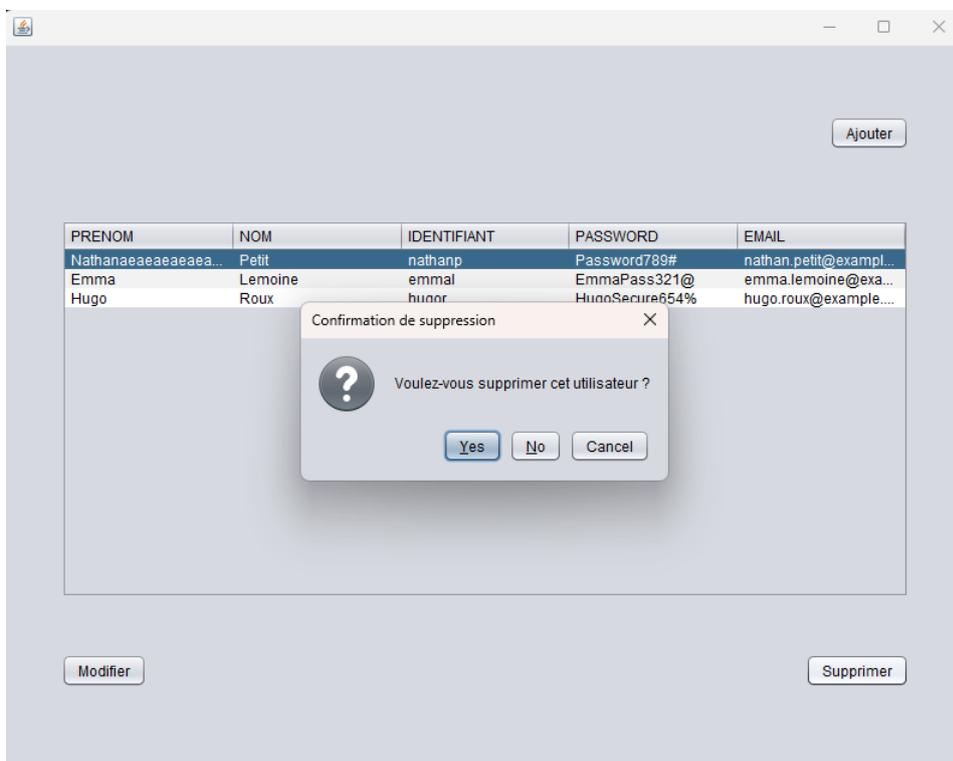
PRENOM	NOM	IDENTIFIANT	PASSWORD	EMAIL
Nathanaaaaaaaaaa...	Petit	nathanp	Password789#	nathan.petit@exampl...
Emma	Lemoine	emmal	EmmaPass321@	emma.lemoine@exa...
Hugo	Roux	hugor	HugoSecure654%	hugo.roux@example....

## Suppression :

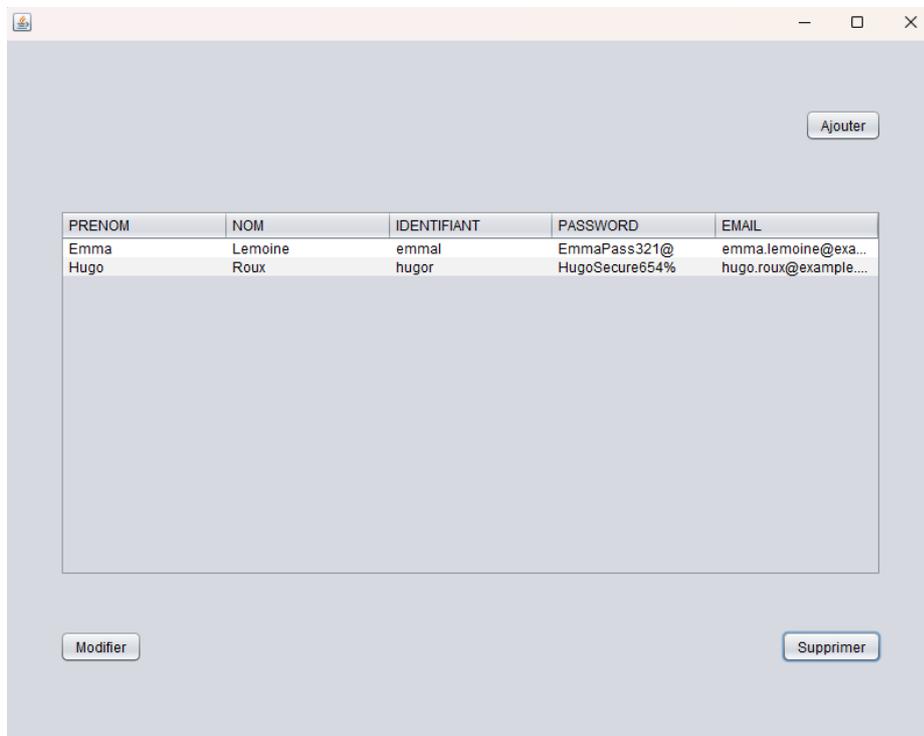
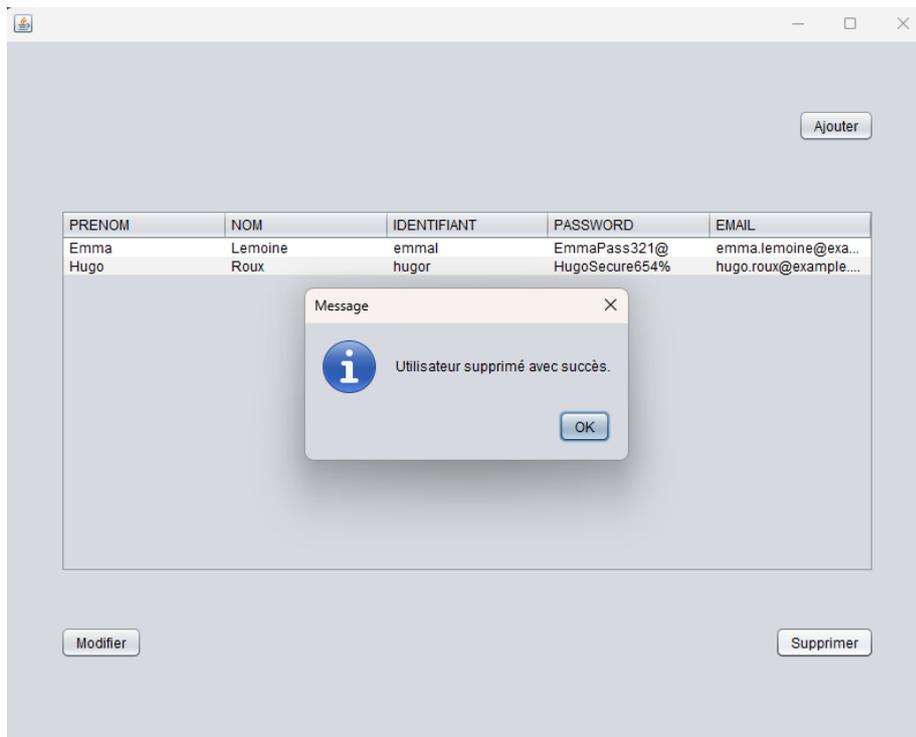
Sélectionnez l'utilisateur que vous souhaitez supprimer :



Un message de confirmation apparaîtra. Cliquez sur Oui pour confirmer la suppression :

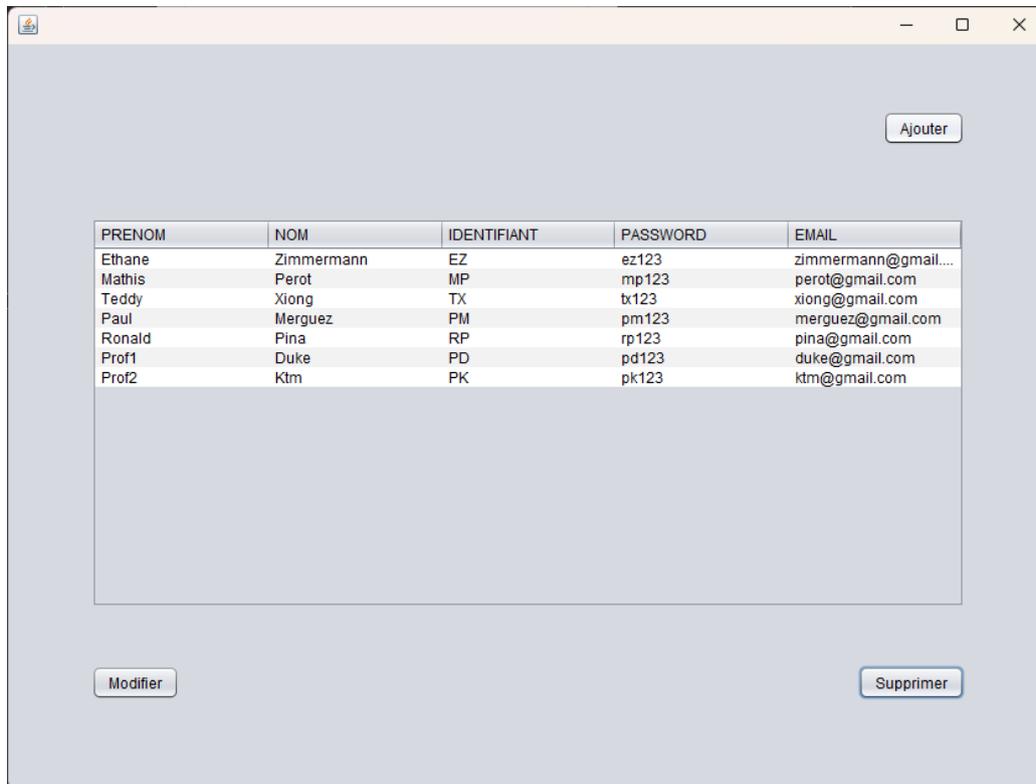


Vous verrez alors que la suppression de l'utilisateur a bien été prise en compte :



## Ajouter :

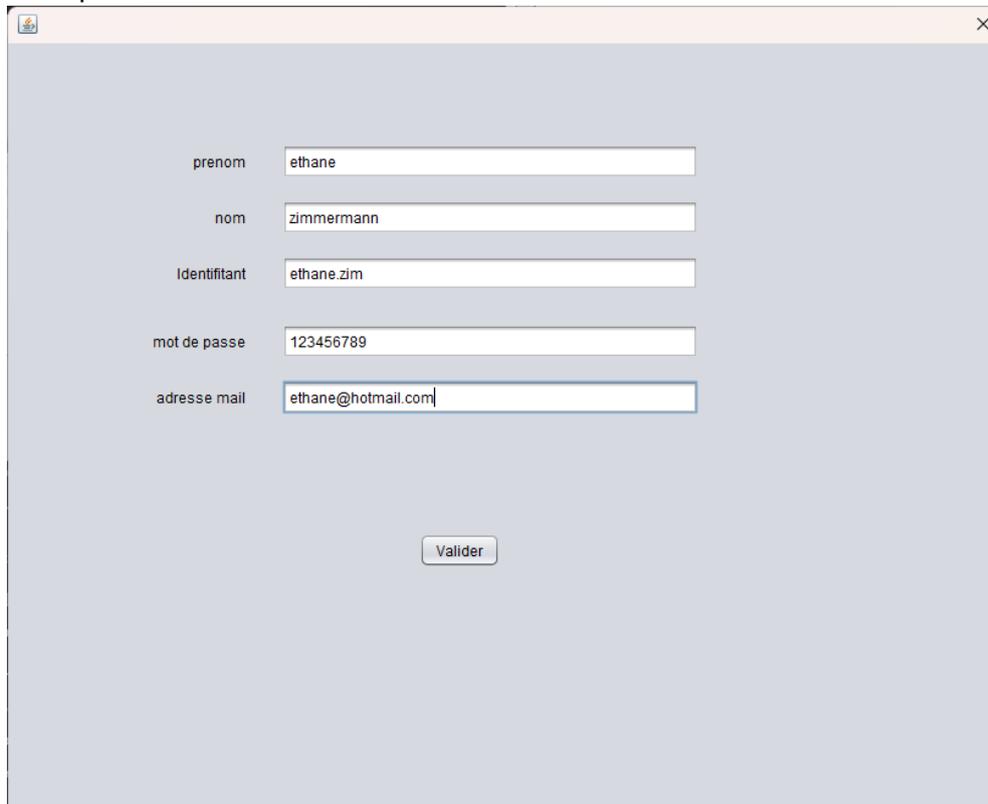
Cliquez sur le bouton Ajouter ; une nouvelle fenêtre apparaîtra avec le formulaire à remplir :



A screenshot of a web application window. At the top right is a button labeled "Ajouter". Below it is a table with five columns: PRENOM, NOM, IDENTIFIANT, PASSWORD, and EMAIL. The table contains seven rows of user data. At the bottom left is a button labeled "Modifier", and at the bottom right is a button labeled "Supprimer".

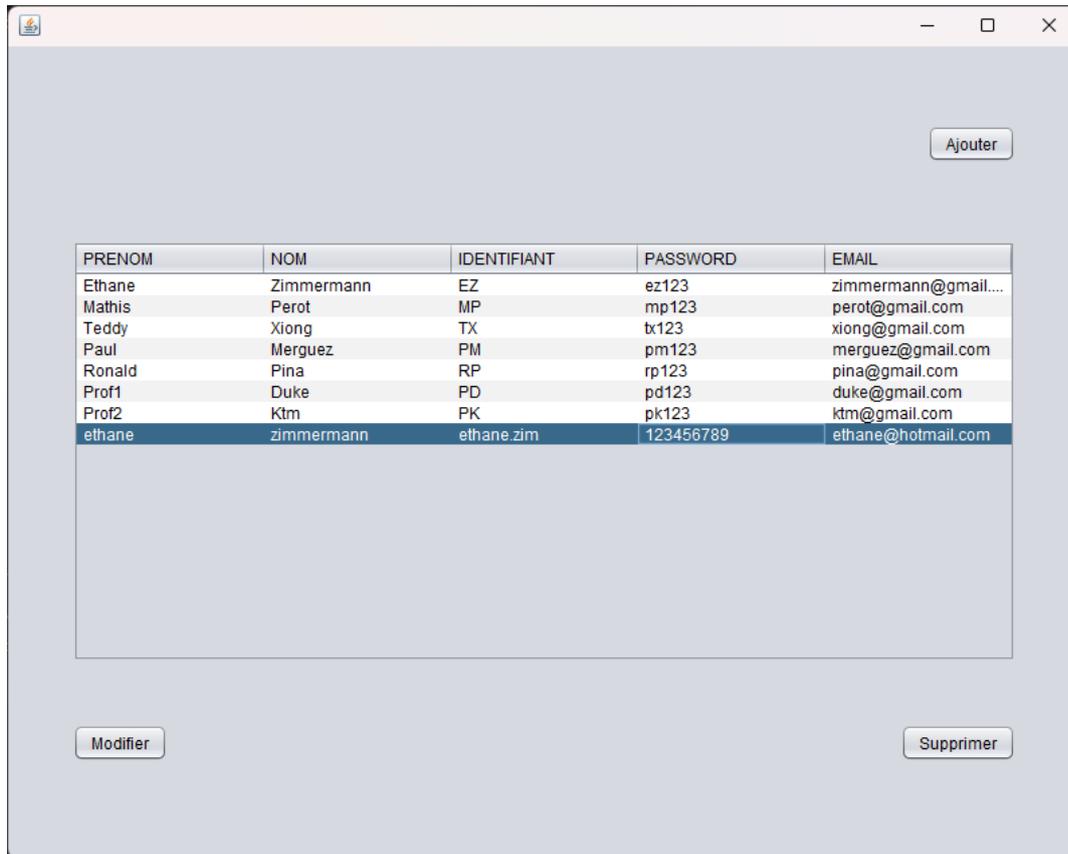
PRENOM	NOM	IDENTIFIANT	PASSWORD	EMAIL
Ethane	Zimmermann	EZ	ez123	zimmermann@gmail...
Mathis	Perot	MP	mp123	perot@gmail.com
Teddy	Xiong	TX	tx123	xiong@gmail.com
Paul	Merguez	PM	pm123	merguez@gmail.com
Ronald	Pina	RP	rp123	pina@gmail.com
Prof1	Duke	PD	pd123	duke@gmail.com
Prof2	Ktm	PK	pk123	ktm@gmail.com

Remplissez le formulaire du nouvel utilisateur :



A screenshot of a web application window showing a registration form. The form has five input fields with labels: "prenom" (value: ethane), "nom" (value: zimmermann), "Identifiant" (value: ethane.zim), "mot de passe" (value: 123456789), and "adresse mail" (value: ethane@hotmail.com). At the bottom center is a button labeled "Valider".

Une fois terminé, cliquez sur Valider : votre nouvel utilisateur apparaîtra dans la liste :



PRENOM	NOM	IDENTIFIANT	PASSWORD	EMAIL
Ethane	Zimmermann	EZ	ez123	zimmermann@gmail...
Mathis	Perot	MP	mp123	perot@gmail.com
Teddy	Xiong	TX	tx123	xiong@gmail.com
Paul	Merguez	PM	pm123	merguez@gmail.com
Ronald	Pina	RP	rp123	pina@gmail.com
Prof1	Duke	PD	pd123	duke@gmail.com
Prof2	Ktm	PK	pk123	ktm@gmail.com
ethane	zimmermann	ethane.zim	123456789	ethane@hotmail.com

## Répartition des tâches

### 6.1. Membres de l'équipe et rôles attribués

Dans ce projet, nous avons réparti les tâches de manière que chacun puisse travailler, progresser et apprendre à utiliser NetBeans.

La répartition des tâches s'est faite comme suit :

- **Matéo Sage** : gestion de la liste des utilisateurs et suppression d'un utilisateur,
- **Stefan Marcelo** : ajout d'un utilisateur et création de la page d'ajout,
- **Ethane Zimmermann** : modification d'un utilisateur et création de la page de modification.

### 6.2. Description des tâches effectuées

(Pour la création des User j'ai commencé par faire mon formulaire

Je me suis ensuite attaqué à la création de la méthode pour envoyer les données du formulaire au contrôleur

Ensuite la formule d'ajoute dans le DAO )

### **Ajout d'un utilisateur**

Pour la création des utilisateurs (User), j'ai commencé par concevoir le formulaire d'ajout via l'interface graphique dans NetBeans.

Ce formulaire permet de saisir toutes les informations nécessaires à la création d'un nouvel utilisateur (ex : nom, prénom, email, etc.).

Une fois le formulaire créé, j'ai développé une méthode permettant de récupérer les données saisies par l'utilisateur et de les transmettre au contrôleur. Cette étape est essentielle pour assurer la liaison entre l'interface utilisateur et la logique métier du projet.

Ensuite, je me suis chargé de la mise en place de la méthode d'insertion dans le DAO (Data Access Object). Cette méthode se connecte à la base de données et exécute une requête SQL pour ajouter un nouvel utilisateur dans la table correspondante.

---

## **Suppression d'un utilisateur**

Pour la suppression d'un utilisateur, j'ai commencé par intégrer un bouton "Supprimer" dans l'interface qui permet à l'utilisateur de sélectionner un utilisateur à retirer. J'ai ensuite développé une méthode qui récupère l'identifiant de l'utilisateur sélectionné et le transmet au contrôleur.

Dans le DAO, j'ai créé une méthode de suppression qui exécute une requête SQL DELETE pour retirer l'utilisateur de la base de données.

Enfin, j'ai mis en place un message de confirmation avant la suppression afin d'éviter toute suppression accidentelle.

---

## **Modification d'un utilisateur**

Pour la modification d'un utilisateur, j'ai d'abord ajouté un bouton "Modifier" dans l'interface.

Lorsque l'on sélectionne un utilisateur et que l'on clique sur ce bouton, un formulaire pré-rempli avec les informations existantes de l'utilisateur s'ouvre.

J'ai ensuite développé la méthode permettant de récupérer les modifications effectuées dans le formulaire, puis de transmettre ces nouvelles données au contrôleur.

Dans le DAO, j'ai implémenté une méthode de mise à jour (UPDATE) qui modifie les informations de l'utilisateur dans la base de données en fonction de son identifiant.

---

## **Affichage de la liste des utilisateurs**

Pour l'affichage de la liste des utilisateurs, j'ai conçu une interface permettant de visualiser tous les utilisateurs enregistrés dans la base de données.

J'ai créé une méthode dans le DAO qui exécute une requête SELECT pour récupérer toutes les entrées de la table utilisateur.

Ensuite, ces données sont envoyées au contrôleur, qui les affiche dans un tableau ou une liste graphique sur l'interface.

J'ai également ajouté une fonctionnalité pour actualiser la liste à chaque ajout, modification ou suppression d'un utilisateur.

# Planification et gestion de projet

## 7.1. Outils de planification utilisés (Gantt)

Nous n'avons pas eu le temps de réaliser un diagramme de Gantt. Cependant, nous avons travaillé sur le projet pendant les séances dédiées, d'une durée de 4 heures chacune. En complément, chacun a également avancé sur sa partie du projet chez soi en dehors des heures de cours.

## Retour sur l'expérience projet

### 8.1. Déroulement global

Le projet s'est déroulé principalement lors des séances dédiées, d'une durée de 4 heures chacune. Durant ces séances, nous avons avancé sur nos différentes parties du projet. En complément, chacun a poursuivi son travail personnellement chez soi. Nous avons réparti les tâches de façon que tout le monde puisse apprendre, progresser et comprendre le fonctionnement de NetBeans, notamment sur la création d'une application Java de type CRUD (Create, Read, Update, Delete) sur une table d'utilisateurs.

### 8.2. Difficultés rencontrées

#### **Ethane Zimmermann**

- **Difficultés rencontrées :**

J'ai rencontré des difficultés lors de la mise en place de la fonctionnalité de modification des utilisateurs, en particulier pour préremplir correctement le formulaire avec les informations existantes.

- **Solutions apportées :**

Grâce à des recherches sur les méthodes de récupération des données en Java et avec l'aide des professeurs, j'ai compris comment charger les informations d'un utilisateur sélectionné et les afficher automatiquement dans le formulaire de modification.

---

## Matéo Sage

- **Difficultés rencontrées :**

Ma principale difficulté a été de comprendre la gestion des événements sur la liste des utilisateurs, notamment pour récupérer l'utilisateur sélectionné avant de procéder à sa suppression.

- **Solutions apportées :**

Avec de l'entraînement et en consultant la documentation Java Swing, ainsi qu'avec les conseils donnés pendant les séances, j'ai réussi à maîtriser la sélection d'éléments dans une liste et à lier cette action aux opérations de suppression dans le DAO.

---

## Stefans marcelo

- **Difficultés rencontrées :**

Pour Ma part mon problème rencontre sont surtout pour la récupération des données du formulaire je ne savais pas comment le faire en java

- **Solution apportée :**

Avec des recherches dès l'aider des professeurs j'ai pu résoudre mon problème

# Conclusion

## 9.1. Résumé de l'expérience de chaque personne du groupe

### **Ethane :**

Ce projet m'a permis d'approfondir mes connaissances sur la création d'un site en Java en utilisant l'IDE NetBeans.

Au début, cela n'a pas été évident pour moi, le temps de réassimiler les bases et de bien comprendre le fonctionnement du modèle MVC (Modèle-Vue-Contrôleur) en Java.

Cette expérience m'a offert une première approche concrète du développement d'applications Java, et peut-être qu'à l'avenir, je serai amené à créer d'autres projets ou sites en Java.

### **Matéo :**

Le projet m'a permis de comprendre et approfondir la méthode d'architecture MVC en langage JAVA qui a une ressemblance avec le C# que j'ai pratiqué pendant mon stage.

Grâce à ce projet j'ai pu avoir une vision approfondie du développement backend et du logiciel lourd.

### **Stephan :**

Le projet m'a fait découvrir netbean le java et m'améliorer sur le MVC la découverte du client lourd ma permit de développer cette compétence et l'approfondir. avec quelle difficulté rencontrée sur la récupération des valeur cela fut un vrais défié pour moi