Luigi JOUBERT

Mémoire pour le titre de Développeur Web & Web Mobile

Site vitrine NAVA

1. Introduction

Je m'appelle Luigi JOUBERT. Après avoir passé un Baccalauréat Économique et Social en 2019, j'ai déménagé à Toulouse afin de rejoindre l'université Toulouse II Jean-Jaurès en filière LLCER Japonais, car c'est une langue qui avait éveillé ma curiosité à la suite d'un voyage au Japon fin 2019.

Après 2 années passées à l'université, j'ai décidé de me réorienter car la pédagogie universitaire ne me semblait pas adaptée à ce que je recherchais, et le peu de débouchés possibles avec un diplôme en langue japonaise ne me faisaient pas rêver.

Ayant depuis très jeune eu une attirance pour l'informatique de façon générale, ainsi que pour le code, et grâce à l'aide de deux conseillères qui m'ont suivi à la Mission Locale de Bagatelle, j'ai eu la chance de pouvoir passer avec succès les examens au sein de l'Adrar afin de rejoindre la formation qui pourrait me permettre d'obtenir le titre de Développeur Web & Web Mobile.

Ayant un très bon ami ingénieur du son qui a lancé son propre studio d'enregistrement ainsi que son label musical au cours de ces 3 dernières années, j'ai choisi, en tant que Projet Fil Rouge, de lui créer un site vitrine qui pourrait lui permettre de donner de la visibilité à ses artistes ainsi qu'à ses projets.

Son projet ayant nécessité la création d'une entreprise, c'est également chez cet ami que j'ai effectué mon stage de 2 mois afin de pouvoir avancer sur ce projet qui me tenait à cœur.

Ayant déjà eu différentes expériences professionnelles dans ma vie et connaissant le monde du travail en entreprise, le fait de me retrouver dans un stage sans développeur confirmé pour m'épauler était également un challenge que je recherchais pour m'essayer au mode de vie que pourrait avoir un freelance afin de voir si cela pouvait me correspondre.

2. Abstract

My name is Luigi JOUBERT. After obtaining a Baccalauréat in Economics and Social Sciences in 2019, I moved to Toulouse to join Toulouse II Jean-Jaurès University in the Japanese LLCER program, as it is a language that sparked my curiosity following a trip to Japan at the end of 2019.

After spending two years at the university, I decided to change direction as the university pedagogy didn't seem suitable for what I was looking for, and the limited job prospects with a degree in the Japanese language didn't inspire me.

Having had a strong interest in computer science in general, as well as coding, from a young age, and with the help of two advisors who supported me at the Mission Locale Bagatelle, I had the opportunity to successfully pass the exams at Adrar to join a training program that would allow me to obtain the title of Web & Mobile Web Developer.

Having a very close friend who is a sound engineer and has started his own recording studio and music label in the past three years, I chose, as my main project, to create a showcase website for him, which would give visibility to his artists and projects.

Since his project required the creation of a company, I also completed a two-month internship at his place to make progress on this project that was dear to me.

Having already had various professional experiences in my life and being familiar with the corporate work environment, the challenge of being in an internship without an experienced developer to support me was also something I sought to try out the freelance lifestyle and see if it could be a good fit for me.

3. Remerciements

Avant de continuer ce mémoire, je souhaitais remercier l'Adrar et en particulier la coordinatrice du pôle numérique Sophie POULAKOS pour m'avoir fait confiance lors de la phase d'admission.

Je remercie tous les formateurs m'ayant transmis leurs connaissances et ayant pu m'apporter leur aide aussi bien lors de nos journées à l'Adrar que durant leur temps libre sur discord.

Je remercie Arman ZARANDI, mon tuteur de stage et ami de longue date m'ayant fait confiance pour porter un projet nouveau dans son entreprise.

Et pour terminer, je souhaite remercier tous mes camarades de formation pour les échanges et les bons moments partagés au cours de ces 9 mois de formation.

4. Analyse du besoin

4.1) Présentation du projet

Afin de valider notre titre, il nous a été demandé en début de formation de choisir un Projet Fil Rouge, projet personnel sur lequel nous travaillerons tout au long de la formation afin d'avoir du contenu duquel parlé dans ce mémoire ainsi que lors de nos oraux.

J'ai personnellement choisi de créer un site vitrine pour NAVA, un collectif d'artistes en tout genre gravitant autour du monde de la musique (chanteurs, beatmakers, artistes 3D, caméramans, ingénieurs du son…). Ce collectif a été fondé par 3 amis ingénieurs du son (dont mon tuteur de stage Arman ZARANDI) ayant été diplômés en 2019, suite à quoi ils ont fondé leur propre studio d'enregistrement en 2020 suivi par leur propre label musical en juin 2022.

4.2) Contexte et besoins

Ce projet est donc un site vitrine sur lequel seront exposés tous les artistes ainsi que les projets réalisés par le collectif NAVA.

Chaque artiste aura une fiche personnelle avec les informations suivantes :

- Nom d'artiste
- Nom / Prénom
- Âge
- Ville
- Liens de ses réseaux sociaux

Chaque projet aura également une fiche personnelle avec les informations suivantes :

- Nom du projet
- Image de couverture du projet
- Date de sortie
- Artistes ayant participé au projet

- Producteur du projet
- Personne ayant mixé le projet
- Liens sur les plateformes de musique (Spotify, Deezer, Apple Music, …)

Il est à noter que certaines de ces informations ne sont pas obligatoires (comme le nom, prénom et âge des artistes, par exemple) et que cette liste est susceptible d'évoluer avec le temps.

Le site comporte plusieurs fonctionnalités, notamment :

1) Un système de création de compte permettant de créer des comptes d'administration (un système de rôles est déjà codé et fonctionnel pour à l'avenir pouvoir créer également des comptes clients lorsqu'un eshop de merch sera mis en place)
2) Un système de connexion et de déconnexion
3) La consultation des fiches des artistes et des projets
4) L'ajout, édition, suppression des fiches des artistes et des projets (pour les personnes connectées avec un compte d'administration)
5) L'ajout, édition, suppression de nouveaux rôles (types de compte)
6) L'ajout, édition, suppression de nouveaux Producteurs, Mixeurs (ceux-ci n'ont pas de fiches personnelles mais permettront par exemple de créer des filtres afin d'afficher tous les projets ayant le même producteur)

Langages utilisés :

Pour le Front-end, j'ai utilisé HTML, CSS, et JavaScript

Pour Back-end, j'ai utilisé le langage PHP

A noter qu'une grande partie du CSS a été faite en utilisant le framework Bootstrap, car le CSS n'est pas à titre personnel la partie du code qui me passionne le plus. J'ai cependant, lorsque c'était nécessaire, utilisé du CSS moi-même afin de modifier des éléments Bootstrap qui ne me convenaient pas.

4.3) Contraintes Techniques

Le site devra être responsive, c'est-à-dire que l'affichage du site devra s'adapter à différentes tailles d'écran afin que l'utilisateur puisse l'utiliser convenablement sur ordinateur, tablette ou smartphone.

5. Outils Utilisés

5.1) Logiciels

5.1.1) Éditeur de code :

VScode : VSCode est un éditeur de code léger et polyvalent, largement utilisé par les développeurs pour écrire et travailler sur des projets de développement.

5.1.2) Maquettage :

Balsamiq Wireframes : Balsamiq Wireframe est un logiciel de création de maquettes (wireframes) d'interfaces utilisateur, apprécié des concepteurs pour sa simplicité et sa convivialité. Il permet de visualiser et de partager rapidement des idées de conception pour des projets web et mobiles.

Figma : Figma est un logiciel de conception d'interface utilisateur (UI) et de prototypage, largement utilisé par les concepteurs pour créer, collaborer et itérer sur des projets d'interface utilisateur de manière efficace et intuitive.

5.1.3) Conception :

Gloomaps : Gloomaps est un outil en ligne utilisé pour créer des cartes mentales visuelles et interactives. Il permet aux utilisateurs de structurer et d'organiser des idées, des concepts et des informations de manière visuelle, favorisant ainsi la compréhension et la collaboration. Gloomaps est largement utilisé pour la planification de projets, la prise de notes, la gestion des connaissances et la création de présentations.

StarUML : StarUML est un logiciel de modélisation UML (Unified Modeling Language) utilisé par les développeurs et les architectes logiciels pour concevoir et représenter graphiquement des modèles de systèmes et de logiciels. Il offre des fonctionnalités avancées de

modélisation et de génération de code pour faciliter le développement logiciel.

5.1.4) Serveur local et base de données

Laragon : Laragon est un environnement de développement local (Local Development Environment - LDE) convivial et léger, utilisé par les développeurs pour configurer facilement et rapidement des serveurs web locaux, notamment pour les projets basés sur PHP et MySQL.

HeidiSQL : HeidiSQL est un logiciel de gestion de base de données graphique, largement utilisé par les développeurs et les administrateurs de base de données pour interagir avec des bases de données MySQL, MariaDB, PostgreSQL et SQL Server. Il offre une interface conviviale et des fonctionnalités avancées pour faciliter la gestion et la manipulation des données.

5.2) Langages :

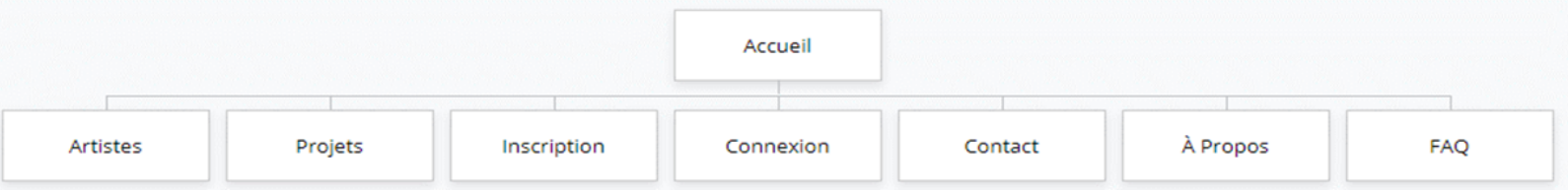
HTML : HTML (Hypertext Markup Language) est un langage de balisage utilisé pour structurer le contenu des pages web. Il permet de définir la structure, le style et les éléments interactifs d'une page web en utilisant des balises et des attributs spécifiques, facilitant ainsi la création et la présentation du contenu sur Internet.

CSS : CSS (Cascading Style Sheets) est un langage de feuilles de style utilisé pour styliser et mettre en forme les éléments HTML d'une page web. Il permet de contrôler l'apparence visuelle, telle que les couleurs, les polices, les marges et les dispositions, offrant ainsi un contrôle précis sur l'aspect esthétique des sites web.

JavaScript : JavaScript est un langage de programmation de script utilisé principalement pour rendre les pages web interactives. Il permet d'ajouter des fonctionnalités dynamiques aux sites web, de manipuler le contenu de la page, de répondre aux actions de l'utilisateur et de communiquer avec les serveurs, offrant ainsi une expérience utilisateur plus immersive et interactive.

PHP : PHP est un langage de programmation côté serveur utilisé pour développer des applications web dynamiques. Il permet de traiter des formulaires, d'accéder à des bases de données, de générer du contenu dynamique et de gérer des sessions utilisateur. PHP est largement utilisé et offre une grande flexibilité pour la création de sites web interactifs et fonctionnels.

6. Arborescence du projet

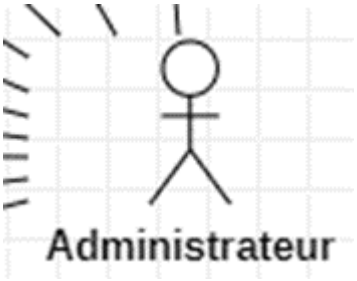


L'arborescence d'un site web désigne la structure hiérarchique et organisée des pages et des contenus qui composent le site. Elle représente la manière dont les différentes pages sont organisées et reliées entre elles, formant une navigation cohérente pour les utilisateurs. L'arborescence peut inclure des niveaux de navigation tels que les pages principales, les sous-pages, les catégories et les sous-catégories, permettant aux utilisateurs de parcourir et de trouver facilement l'information qu'ils recherchent.

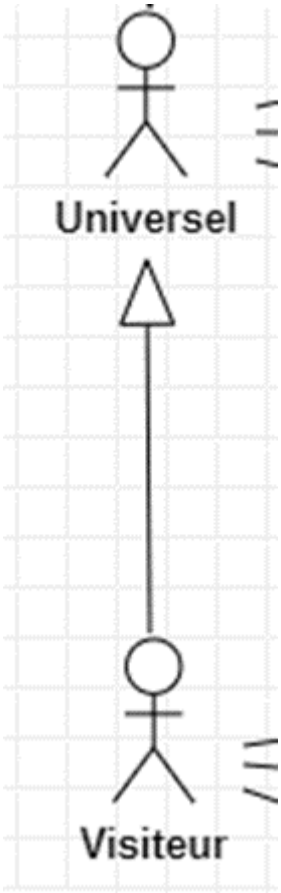
7. Spécifications Fonctionnelles

7.1) Use case

Un use case est une description textuelle ou graphique d'une interaction spécifique entre un utilisateur (**acteur**) et un **système d'information**. Il décrit les actions et les interactions qui se produisent lorsqu'un utilisateur utilise le système pour atteindre un objectif particulier. Un use case identifie les acteurs impliqués, les scénarios d'utilisation, les préconditions et les postconditions, offrant ainsi une vue détaillée des fonctionnalités et des comportements attendus d'un système.

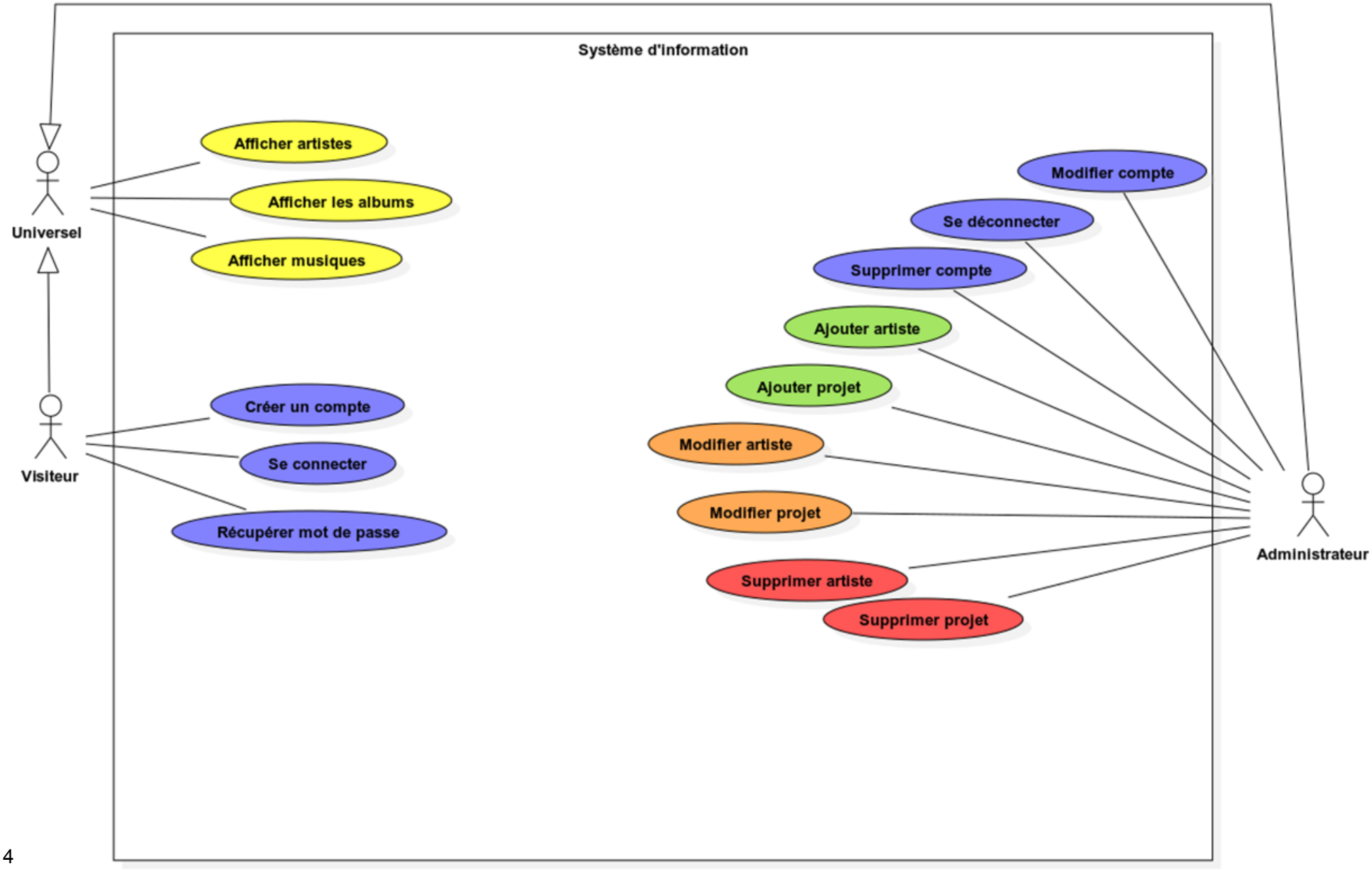


Nous avons ici l'Administrateur qui est un exemple d'**acteur**



Nous avons ici un **héritage**. Dans le contexte des cas d'utilisation, l'héritage se réfère à la relation entre les cas d'utilisation où un cas d'utilisation plus spécifique (**enfant**) peut hériter des caractéristiques et des comportements d'un cas d'utilisation plus général (**parent**). L'héritage permet de réutiliser les fonctionnalités communes et d'éviter la redondance dans la modélisation des cas d'utilisation. Les cas d'utilisation enfants héritent des attributs, des actions et des relations du cas d'utilisation parent, tout en ajoutant ou en spécialisant des fonctionnalités spécifiques à leur contexte particulier. Cela permet de capturer la relation de généralisation-spécialisation entre les cas d'utilisation.

Dans le cas présent, l'acteur « Visiteur » (ainsi que l'Administrateur qui n'est pas présent sur la capture d'écran) héritent de l'**acteur virtuel** « Universel »

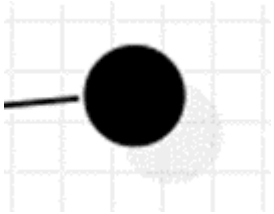

Système d'information
Universel
Visiteur
Administrateur
Afficher artistes
Afficher les albums
Afficher musiques
Créer un compte
Se connecter
Récupérer mot de passe
Modifier compte
Se déconnecter
Supprimer compte
Ajouter artiste
Ajouter projet
Modifier artiste
Modifier projet
Supprimer artiste
Supprimer projet

7.2) Diagramme d’activité

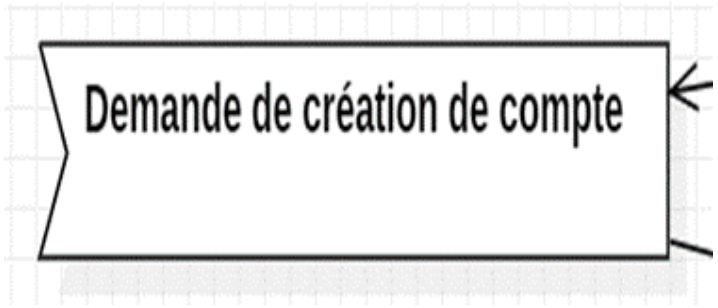
Un diagramme d'activité est un type de diagramme **UML** utilisé pour modéliser le flux d'activités, les décisions et les actions d'un processus ou d'un système. Il représente graphiquement les différentes étapes, les actions et les transitions entre les activités, offrant une vue visuelle du déroulement du processus. Les diagrammes d'activité utilisent des symboles facilitant la compréhension et l'analyse des processus.

Pour ce mémoire j’ai choisi de vous présenter la fonctionnalité de **création de compte**, qui même si pour le moment sera désactivée pour les utilisateurs lambdas sera une fonction primordiale dans le futur.

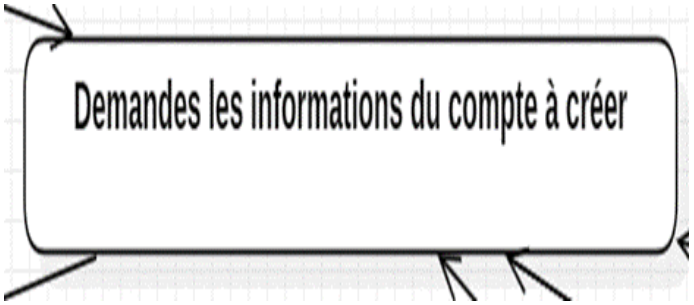
Symboles d’un diagramme d’activités :



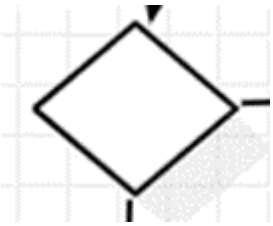
L’**”état initial”** est le point de départ d'un diagramme d'activité, représenté par un cercle rempli avec une flèche entrante. Il marque le début du flux d'activités.



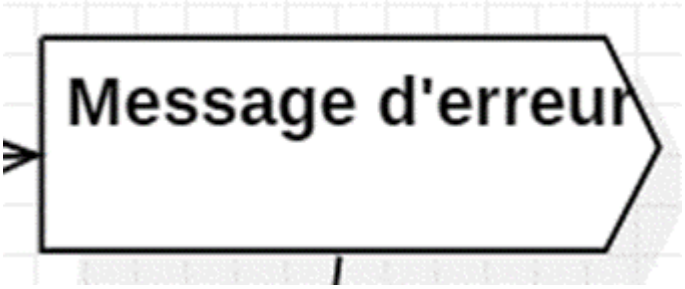
Dans un diagramme d'activité, un **"accept signal"** représente la réception d'un signal ou d'un événement par un élément, indiquant qu'il est prêt à réagir ou à effectuer une action spécifique en réponse.



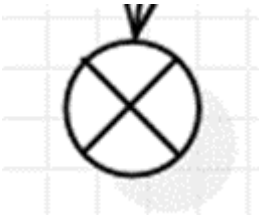
Dans un diagramme d'activité, une **“action”** représente une étape spécifique ou une unité d'activité dans le flux d'exécution.



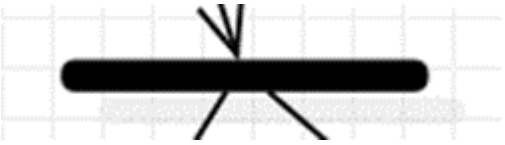
Dans un diagramme d'activité, une **“décision”** est un point où une condition est évaluée pour déterminer le chemin à suivre dans le flux d'activités



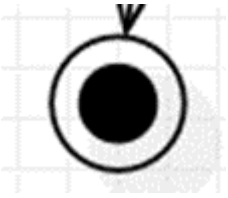
Dans un diagramme d'activité, un **"send signal"** représente l'envoi d'un signal ou d'un événement par un élément à un autre élément du système.



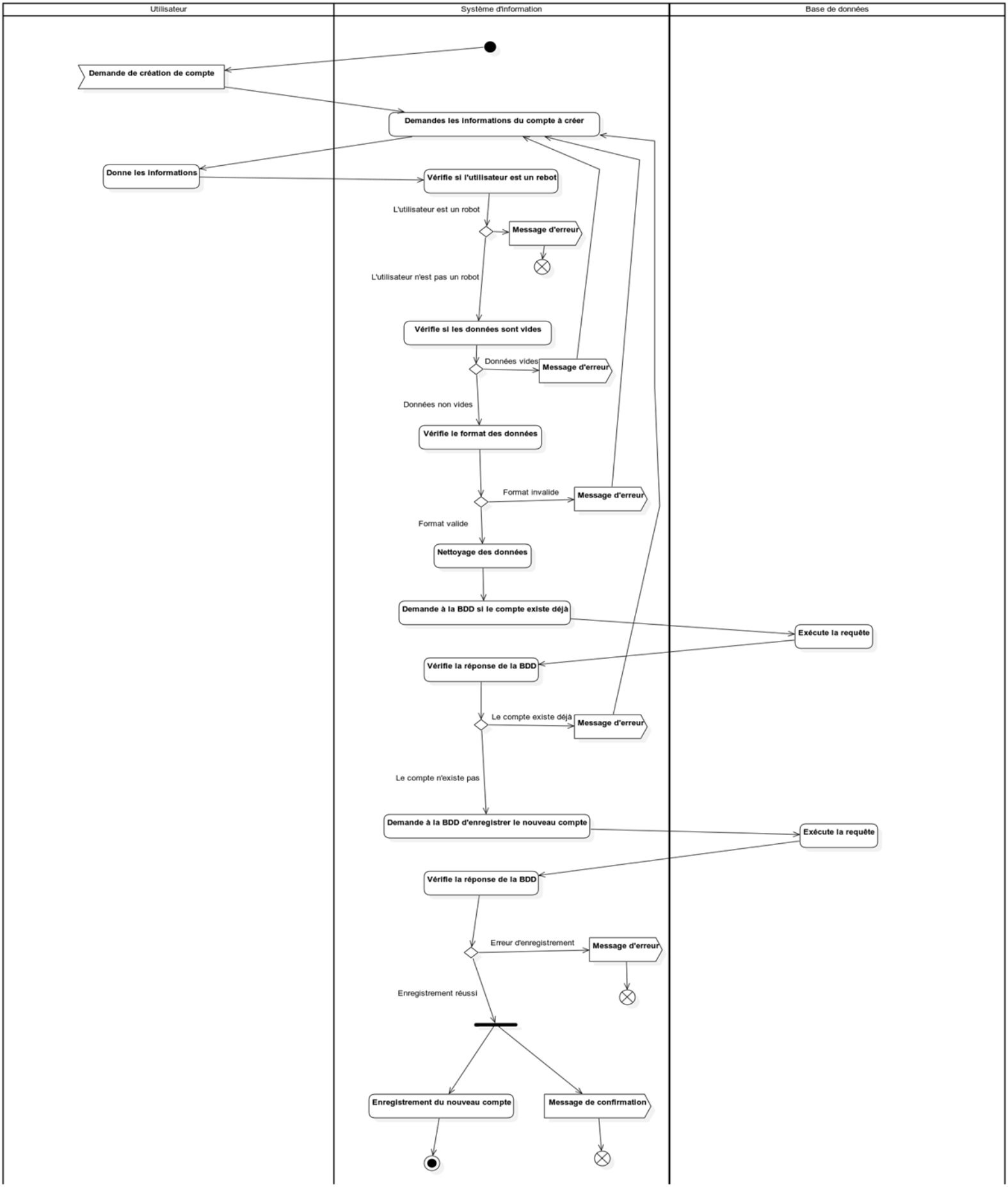
Dans un diagramme d'activité, un **"flow final"** représente la fin du processus ou du flux d'activités modélisé. Il indique la terminaison du processus sans aucune autre activité ultérieure.



Dans un diagramme d'activité, un **"fork”** représente la division d'un flux d'activités en plusieurs chemins parallèles.



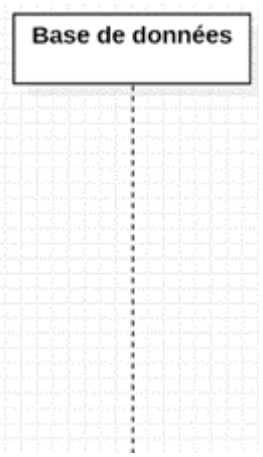
Dans un diagramme d'activité, un **"état final"** représente la fin complète du flux d'activités ou du processus modélisé. Il marque la terminaison et la conclusion du diagramme.


Utilisateur
Système d'information
Base de données
Demande de création de compte
Demandes les informations du compte à créer
Donne les informations
Vérifie si l'utilisateur est un rebot
L'utilisateur est un robot
Message d'erreur
L'utilisateur n'est pas un robot
Vérifie si les données sont vides
Données vides
Message d'erreur
Données non vides
Vérifie le format des données
Format invalide
Message d'erreur
Format valide
Nettoyage des données
Demande à la BDD si le compte existe déjà
Exécute la requête
Vérifie la réponse de la BDD
Le compte existe déjà
Message d'erreur
Le compte n'existe pas
Demande à la BDD d'enregistrer le nouveau compte
Exécute la requête
Vérifie la réponse de la BDD
Erreur d'enregistrement
Message d'erreur
Enregistrement réussi
Enregistrement du nouveau compte
Message de confirmation

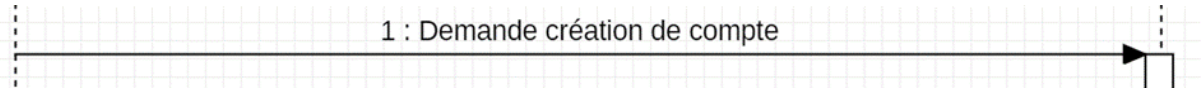
7.3) Diagramme de séquence

Un diagramme de séquence en **UML** est un type de diagramme utilisé pour représenter la séquence chronologique des interactions entre les objets ou les acteurs dans un système. Il montre comment les messages sont échangés entre les différents éléments du système au fil du temps. Les objets ou acteurs sont représentés par des rectangles verticaux, et les messages échangés entre eux sont représentés par des flèches avec des numéros d'ordre pour indiquer la séquence. Les diagrammes de séquence aident à visualiser et à analyser le déroulement des scénarios d'utilisation ou des fonctionnalités du système, en mettant l'accent sur la chronologie des interactions entre les éléments.

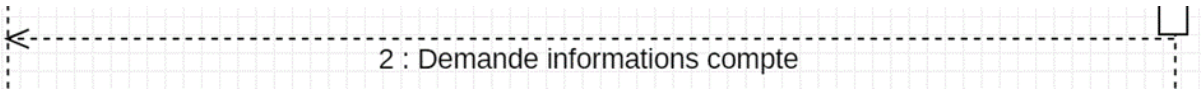
Le diagramme de séquence peut présenter plusieurs types d’éléments :



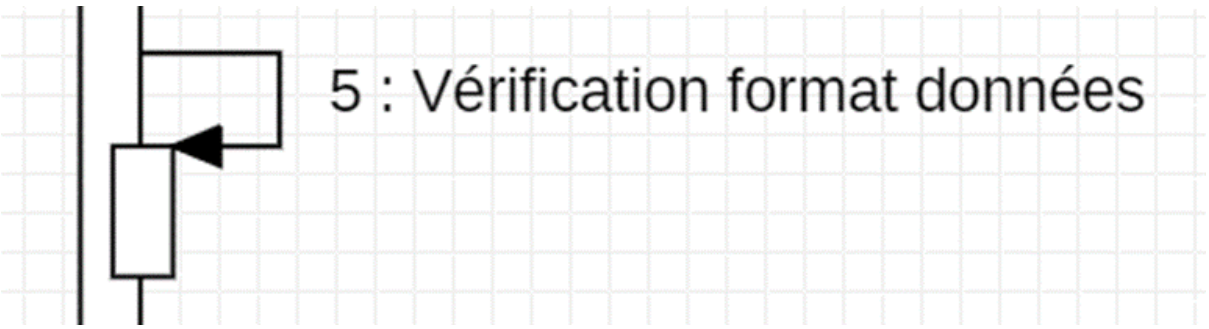
La **lifeline** : une "lifeline" représente la présence d'un objet ou d'un acteur et permet de montrer ses interactions et les messages échangés au fil du temps.



- Le **message synchrone** : un "message synchrone" est un type de message où l'émetteur attend une réponse avant de continuer son exécution

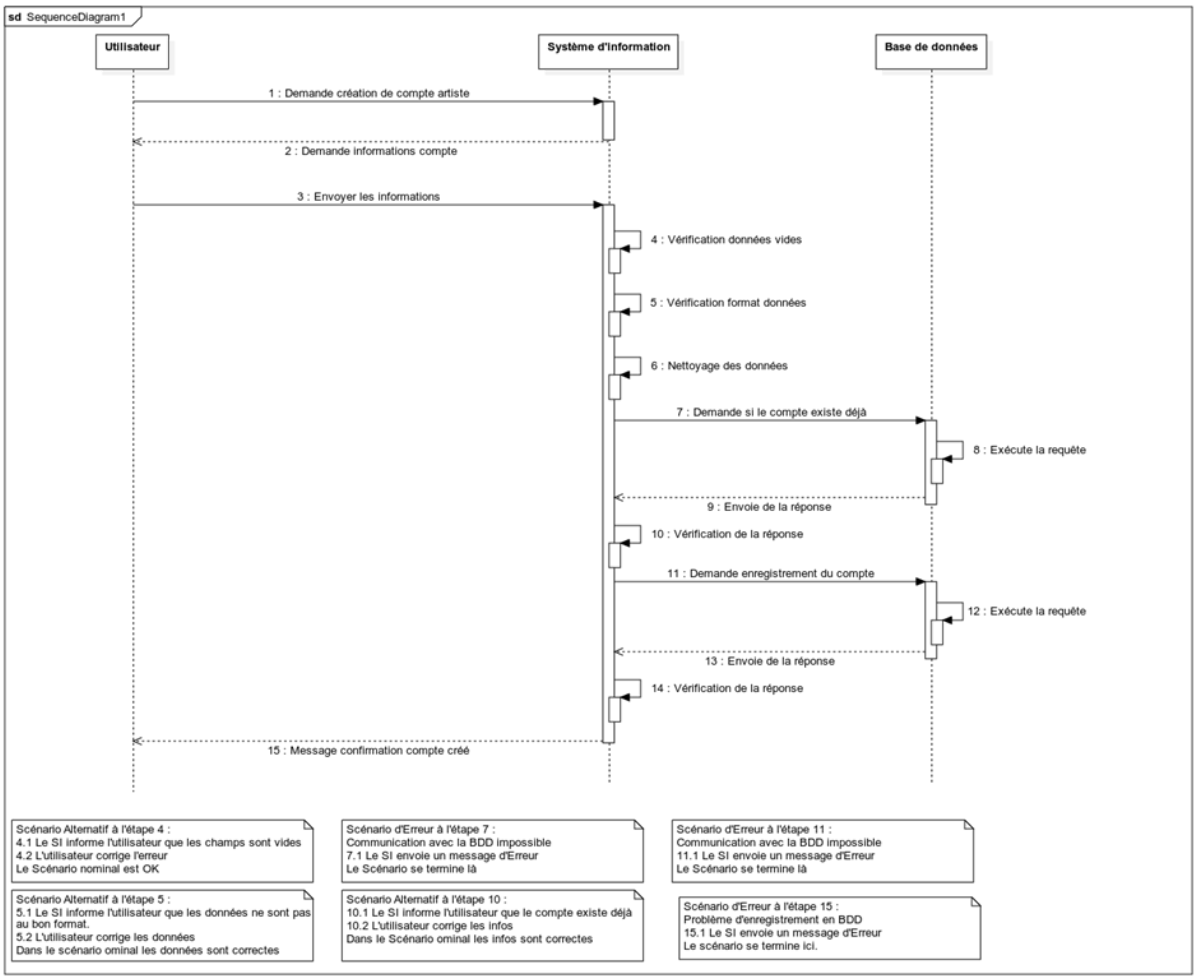


- Le **reply message** : un "reply message" est un message de réponse envoyé en retour à un message précédent, représentant la communication bidirectionnelle entre les éléments du système.



- Le **self message** : un "self message" est un type de message dans lequel un objet s'envoie un message à lui-même, représentant une action ou un événement interne à cet objet.

- Le **message asynchrone** : un "message asynchrone" est un type de message où l'émetteur n'attend pas de réponse immédiate du destinataire.



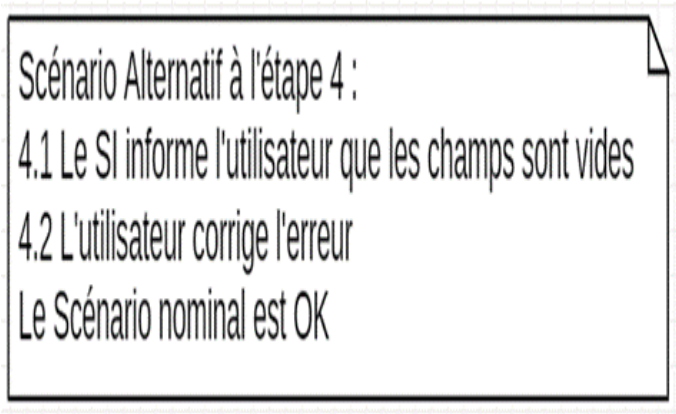
Ce diagramme de séquence présente le **scénario nominal**. On parle de scénario nominal lorsque le système suit son cheminement normal, c’est-à-dire qu’il ne rencontre ni **scénario alternatif** ni **scénario d’erreur**.

7.4) Les scénarios alternatifs et scénarios d’erreurs

Les **scénarios alternatifs** : Un scénario alternatif dans un diagramme de séquence représente un cheminement différent du scénario principal, décrivant une séquence d'événements et d'interactions alternatives en réponse à des conditions spécifiques.

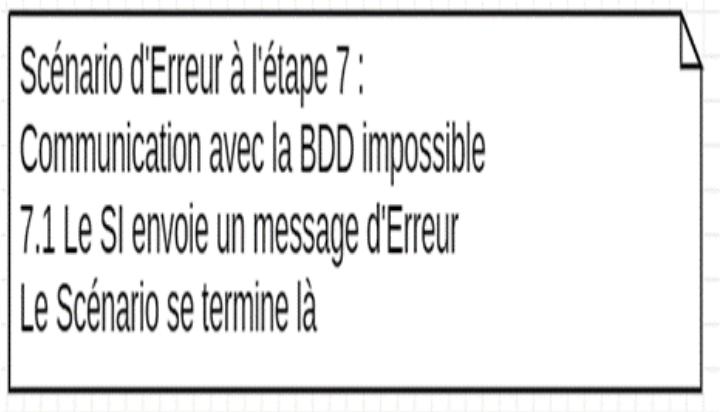
Les **scénarios d’erreurs** : Un scénario d'erreur dans un diagramme de séquence représente la séquence d'événements et d'actions qui se produit lorsqu'une erreur ou une exception survient pendant l'exécution du système.

Exemple de scénario alternatif :



A l’étape 4 du scénario, lors de la vérification de si les données sont vides, il peut y avoir 2 scénarios alternatifs. Le scénario 4.1, où le système d’information informe l’utilisateur que les champs sont vides, suivi du scénario 4.2 où l’utilisateur corrige le problème en remplissant correctement les champs de données. Le scénario nominal reprend alors son cours.

Exemple de scénario d’erreur :



A l’étape 7, lors de la connexion à la BDD afin de vérifier si le compte existe déjà, le système d’information ne parvient pas à s’y connecter.

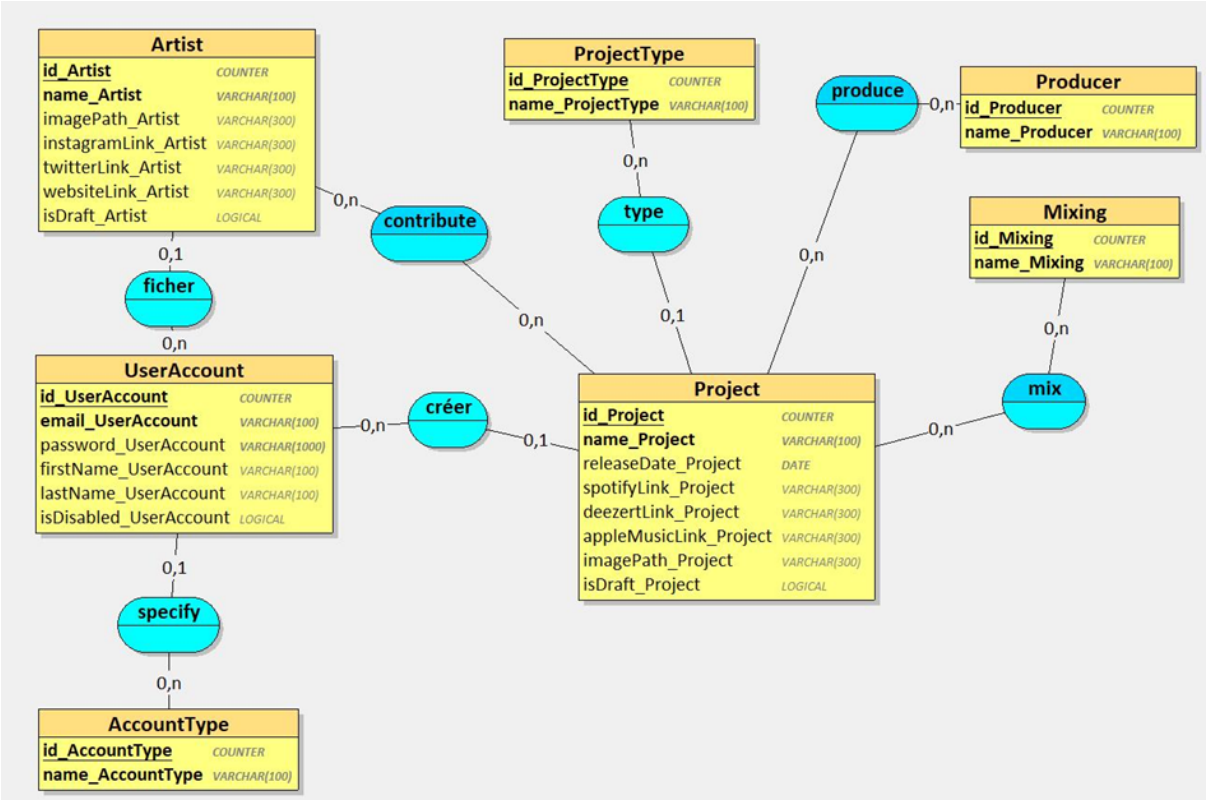
On renvoie alors un message d’erreur et le scénario s’arrête ici car l’utilisateur ne peut rien faire pour corriger ce problème.

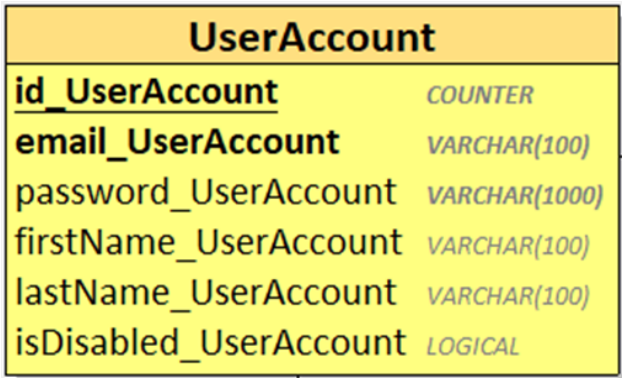
8. Conception

MCD : Un "MCD" (**Modèle Conceptuel de Données**) est un modèle utilisé dans le domaine de la gestion de bases de données pour représenter les concepts et les relations entre les données d'un système. Il fournit une vision des entités, des attributs et des associations, en mettant l'accent sur la structure logique des données. Le MCD utilise des diagrammes entité-association pour illustrer les entités, leurs attributs et les relations entre eux. Il aide à définir et à organiser les données d'un système, facilitant ainsi la conception et la compréhension de la structure des bases de données.

MLD : Un "MLD" (**Modèle Logique de Données**) est un modèle utilisé dans le domaine de la gestion de bases de données pour représenter la structure logique des données d'un système. Contrairement au MCD qui se concentre sur les concepts et les relations, le MLD se concentre sur les tables, les colonnes, les contraintes et les clés de la base de données. Il fournit une représentation plus détaillée et concrète des données, en spécifiant les types de données, les relations entre les tables et les règles de gestion. Le MLD facilite la traduction du modèle conceptuel en une structure logique prête à être mise en œuvre dans une base de données.

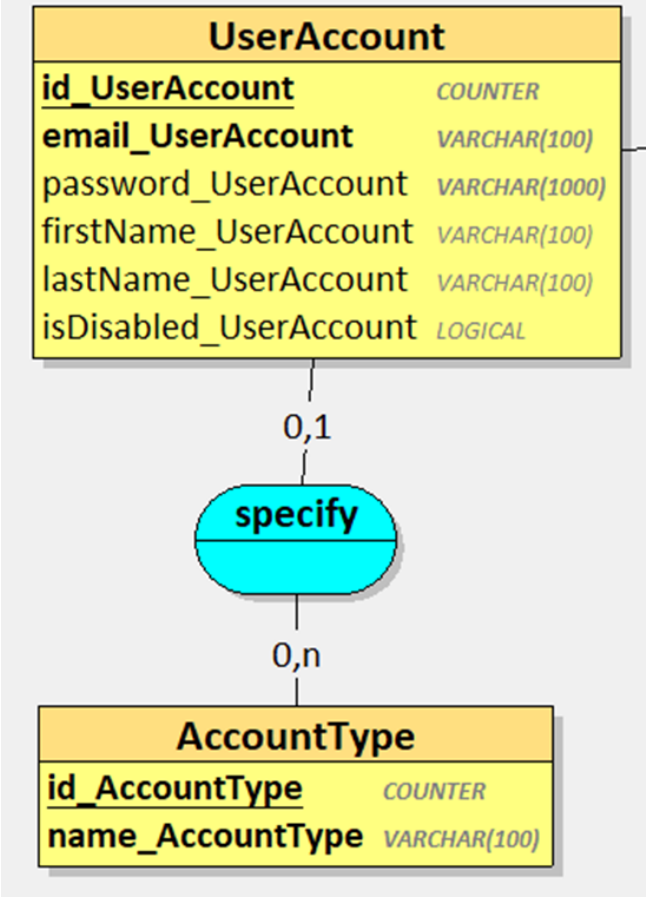
8.1) MCD





Entité: Dans un MCD, une "entité" représente un objet ou un concept du monde réel qui est important pour le système, avec des attributs qui décrivent ses caractéristiques.

L'entité UserAccount comporte une clé primaire (id_UserAccount) sera unique et qui permettra au système de différencier chaque compte utilisateur présent dans la BDD. Les autres attributs sont définis par différents types de données : VARCHAR (suite de caractères), INT (nombre entier), LOGICAL (boolean / tinyint), …

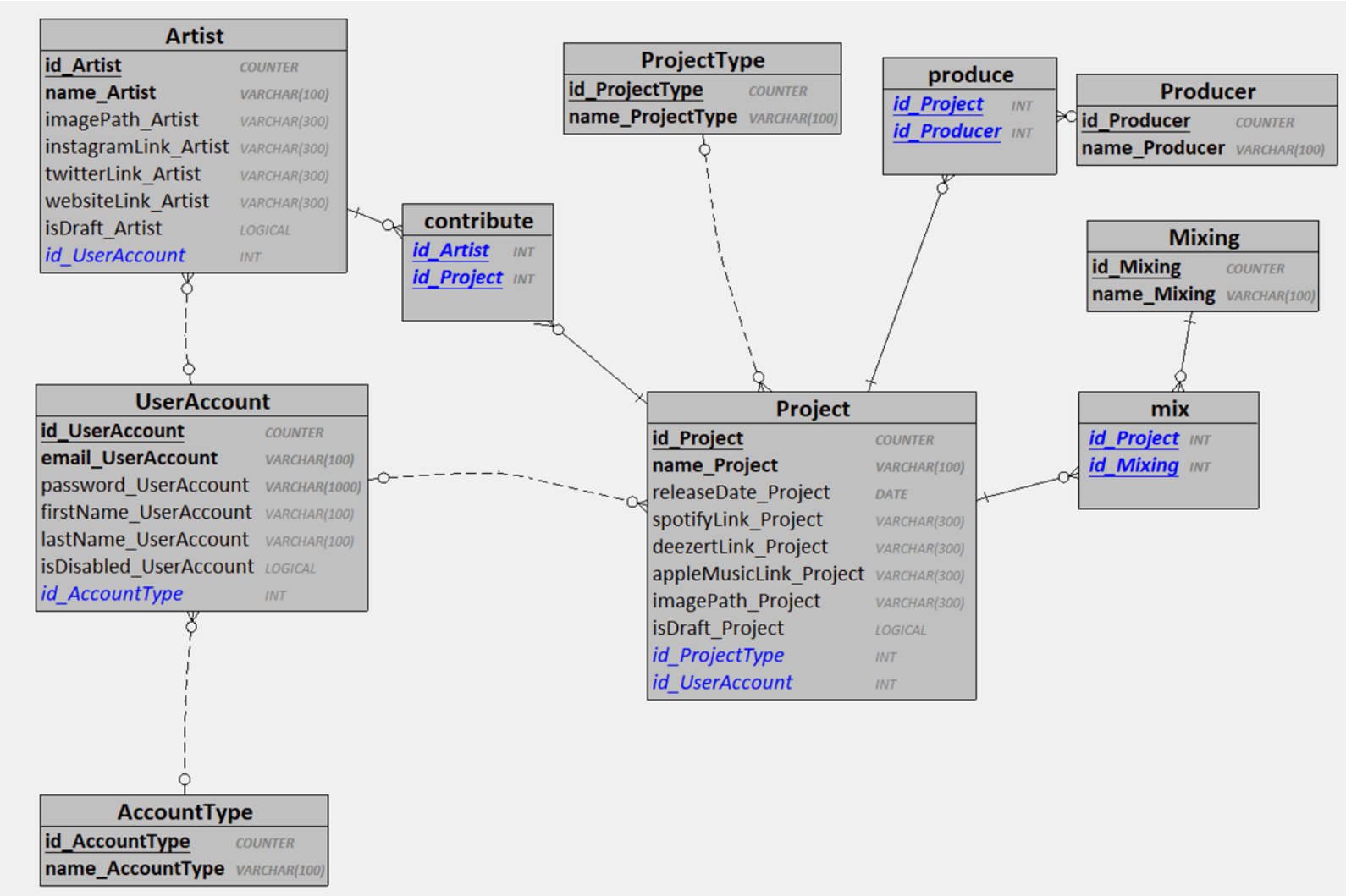


Les **cardinalités** : Dans un MCD, la « cardinalité » représente le nombre de relations possibles entre les entités, telles que « un-à-un » (0,1) ou « un-à-plusieurs » (0,n)

Dans cet exemple, un compte utilisateur ne peut être que d'un seul type (rôle), mais un même type peut être attribué à plusieurs comptes.

Les **associations** : Dans un MCD, une "association" représente la relation logique entre deux entités, indiquant qu'elles sont liées d'une certaine manière.

8.2) MLD



Le MLD est construit à partir du MCD.

Les entités du MCD deviennent des **tables** en MLD. Chaque table contient un nombre de lignes qui seront des champs dans lesquels la valeur des attributs sera stockée en BDD.

Les tables contiennent une **clé primaire** qui sert à identifier de façon unique chaque enregistrement.

Les tables peuvent également contenir des **clés étrangères**, qui permettent de faire le lien entre différentes tables. La clé étrangère d'une table est toujours la clé primaire d'une autre table.

9. Maquettage

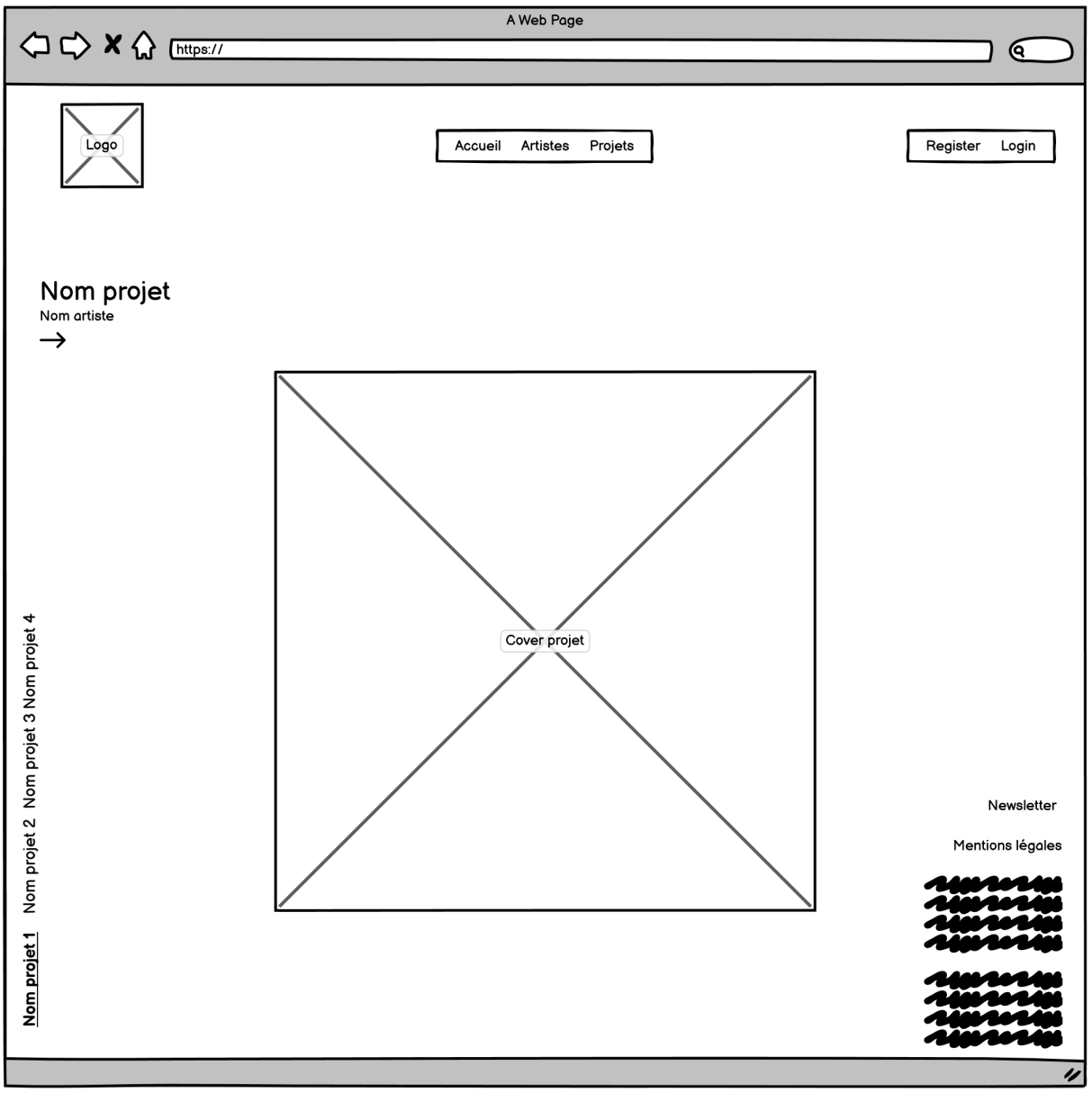
9.1) Wireframes

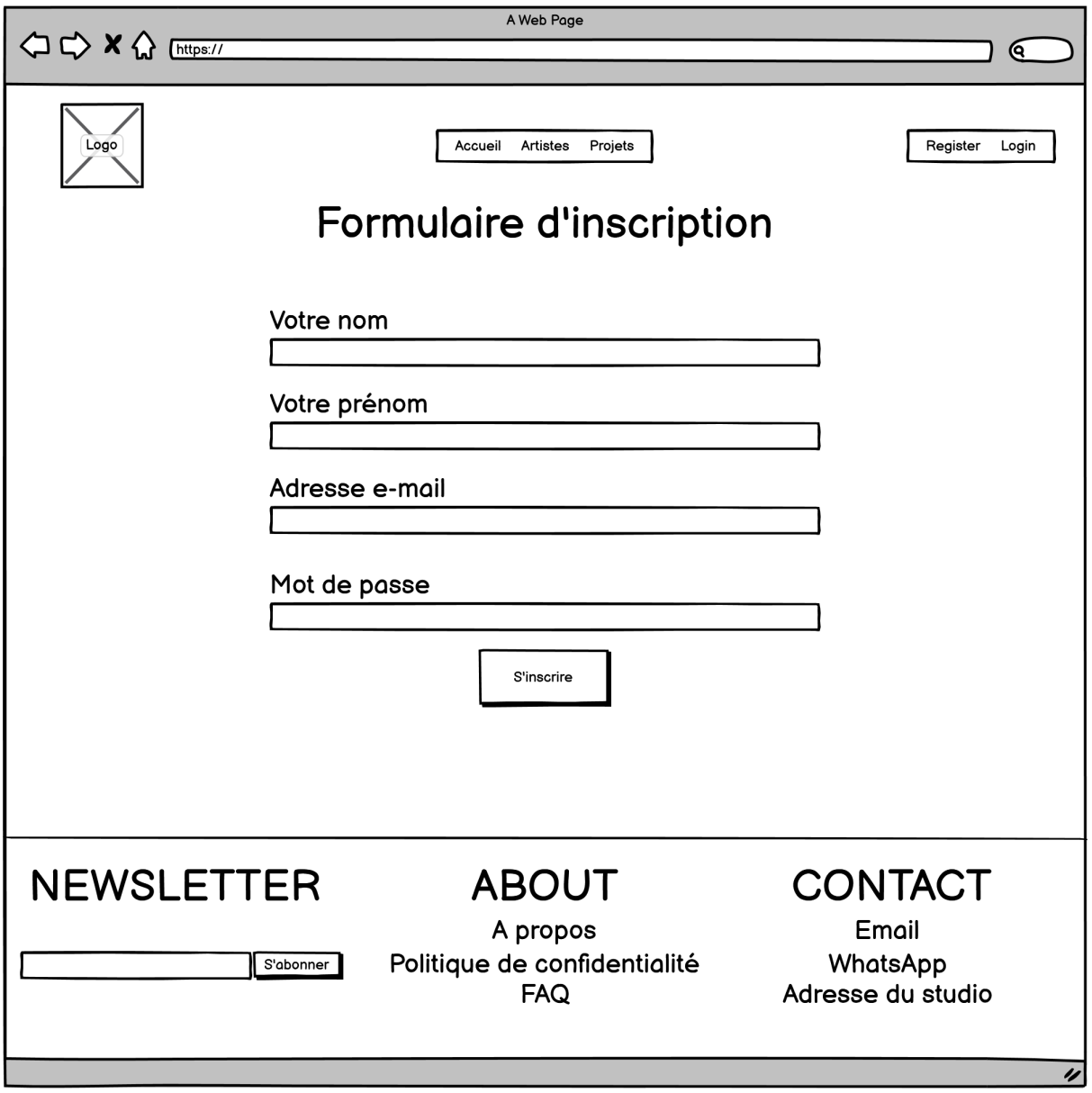
Un wireframe, dans le domaine du maquettage, désigne une représentation visuelle rudimentaire et simplifiée d'une interface utilisateur, d'un site web ou d'une application. Il s'agit d'une étape préliminaire dans la conception d'une interface, qui permet de définir la structure, l'agencement des éléments et la navigation générale de manière conceptuelle, sans se soucier des aspects esthétiques et du design détaillé.

L'objectif principal d'un wireframe est de visualiser et de communiquer la structure et l'organisation de l'interface avant d'entrer dans les détails du design visuel. Il permet aux concepteurs, aux développeurs et aux parties prenantes de collaborer efficacement, d'identifier les problèmes potentiels et d'explorer différentes options de conception. Les wireframes servent également de guide lors de la phase de développement, en fournissant un aperçu clair des fonctionnalités et des interactions prévues.

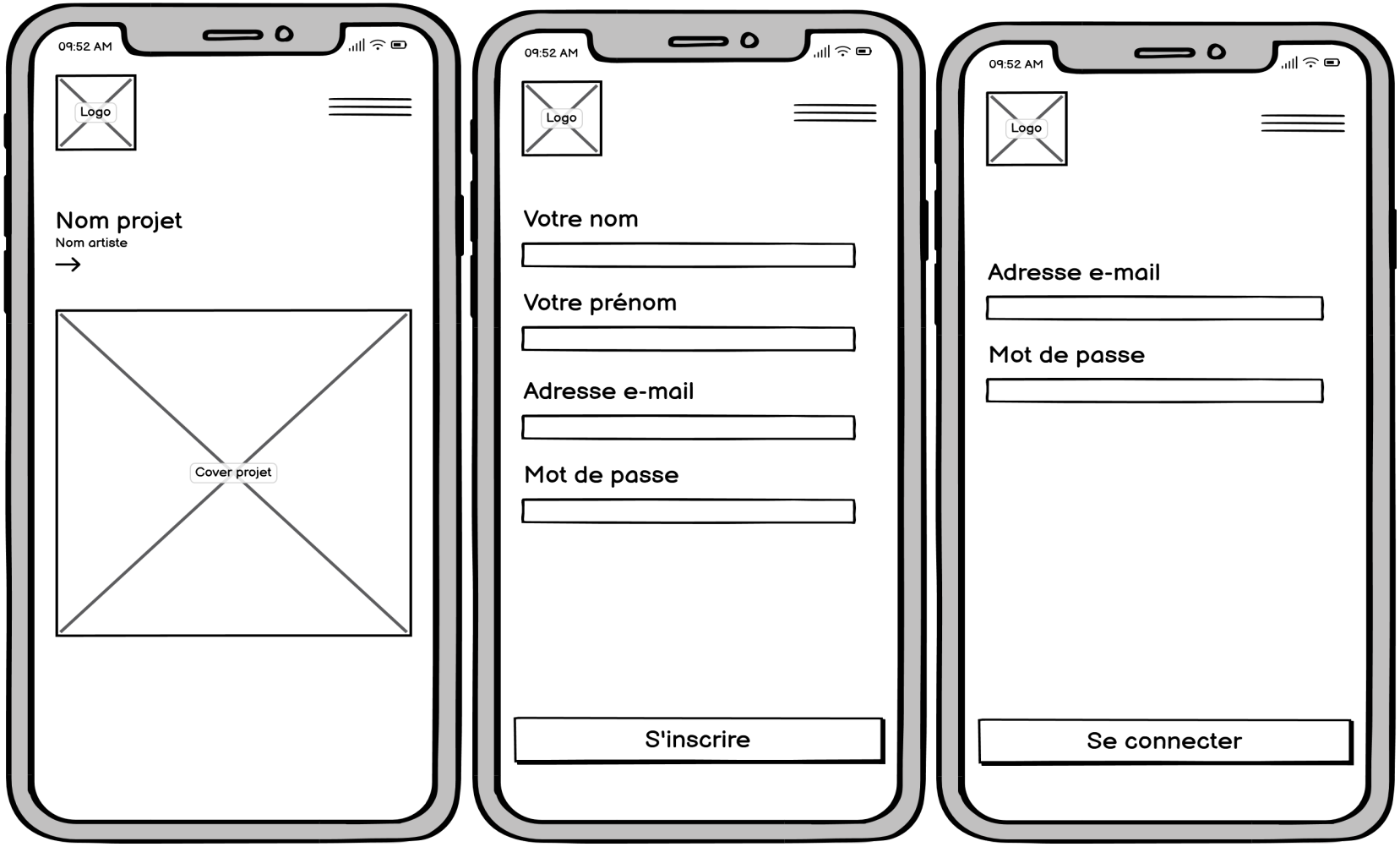
En résumé, un wireframe est une maquette simplifiée et schématique utilisée dans le processus de conception d'interfaces pour représenter la structure, l'agencement et la navigation d'un site web ou d'une application, en mettant l'accent sur la disposition des éléments plutôt que sur leur esthétique.

Ci-dessous seront présentés les wireframes, au format desktop puis mobile, de la page d’accueil, la page d’inscription, et de la page de connexion.


A Web Page
https://
Logo
Accueil Artistes Projets
Register Login
Nom projet
Nom artiste
Cover projet
Nom projet 1 Nom projet 2 Nom projet 3 Nom projet 4
Newsletter
Mentions légales


A Web Page
https://
Logo
Accueil Artistes Projets
Register Login
Formulaire d'inscription
Votre nom
Votre prénom
Adresse e-mail
Mot de passe
S'inscrire
NEWSLETTER
S'abonner
ABOUT
A propos
Politique de confidentialité
FAQ
CONTACT
Email
WhatsApp
Adresse du studio


A Web Page
https://
Logo
Accueil Artistes Projets
Register Login
Formulaire de connexion
Adresse e-mail
Mot de passe
Se connecter
NEWSLETTER
S'abonner
ABOUT
A propos
Politique de confidentialité
FAQ
CONTACT
Email
WhatsApp
Adresse du studio


09:52 AM
Logo
Nom projet
Nom artiste
Cover projet
09:52 AM
Logo
Votre nom
Votre prénom
Adresse e-mail
Mot de passe
S'inscrire
09:52 AM
Logo
Adresse e-mail
Mot de passe
Se connecter

10. Code SQL

Afin de gérer toute la partie SQL lors de mon projet, j'ai décidé d'utiliser **HeidiSQL**, car celui-ci était intégré à **Laragon** et je m'y suis très vite familiarisé du fait de sa ressemblance avec un logiciel que j'avais l'habitude d'utiliser il y a plusieurs années (Navicat).

```sql
CREATE DATABASE nava;
USE nava;
```

La commande **CREATE DATABASE** permet de créer une BDD nommée comme voulu, "nava" en l'occurrence pour ma BDD. Celle-ci sera utilisée pour stocker les données de mon projet.

La commande **USE** quant-à-elle, sert à sélectionner une BDD à laquelle se connecter afin que les requêtes suivantes soient exécutées dans celle-ci.

```sql
CREATE TABLE UserAccount (
    id_UserAccount INT NOT NULL AUTO_INCREMENT,
    email_UserAccount VARCHAR(100) NOT NULL,
    password_UserAccount VARCHAR(1000) NOT NULL,
    firstName_UserAccount VARCHAR(100),
    lastName_UserAccount VARCHAR(100),
    isDisabled_UserAccount BOOLEAN DEFAULT 0,
    id_AccountType INT,
    PRIMARY KEY (id_UserAccount),
    UNIQUE (email_UserAccount),
        CONSTRAINT fk_AccountType_id_AccountType
            FOREIGN KEY (id_AccountType)
            REFERENCES AccountType(id_AccountType)
);
```

La commande **CREATE TABLE** permet de créer une table "UserAccount" dans ma BDD. Elle sera composée des colonnes suivantes :

id_UserAccount, une colonne **INT** (nombre entier) qui sert de clé primaire pour la table. Elle est marquée comme **NOT NULL**, ce qui signifie qu'une valeur doit être fournie pour cette colonne lors de l'insertion d'une nouvelle ligne, et **AUTO_INCREMENT**, ce qui signifie que le système de base de données générera automatiquement une valeur unique pour cette colonne à chaque insertion d'une nouvelle ligne.

email_UserAccount est une colonne **VARCHAR(100)** qui stocke l'adresse e-mail de l'utilisateur. Elle est marquée comme **NOT NULL**.

password_UserAccount est une colonne **VARCHAR(1000)** qui stocke le mot de passe de l'utilisateur. Elle est marquée comme NOT NULL. La limite du VARCHAR a été fixée à 1000 car **l'encryptage du mot de passe** rend celui-ci plus long, je prends donc une marge en sachant que lors du formulaire d'inscription PHP limitera le longueur du mot de passe à un nombre de caractères moindre.

firstName_UserAccount est une colonne VARCHAR(100) qui stocke le prénom de l'utilisateur. Elle permet un maximum de 100 caractères et peut être laissée vide (NULL) si aucun prénom n'est fourni.

lastName_UserAccount est une colonne VARCHAR(100) qui stocke le nom de famille de l'utilisateur. De manière similaire à la colonne firstName_UserAccount, elle permet un maximum de 100 caractères et peut être laissée vide (NULL).

isDisabled_UserAccount est une colonne BOOLEAN qui indique si le compte utilisateur est désactivé ou non. Elle a une valeur par défaut de 0 (false) si aucune valeur n'est fournie. Elle permettra de mettre en place un système de désactivation de compte si son propriétaire le souhaite.

id_AccountType est une colonne INT qui représente la clé étrangère faisant référence à la colonne id_AccountType de la table AccountType. Cette colonne établit une relation entre les tables UserAccount et AccountType en fonction des valeurs de id_AccountType.

La table possède également les contraintes suivantes :

PRIMARY KEY (id_UserAccount): Indique que la colonne id_UserAccount est la clé primaire de la table.

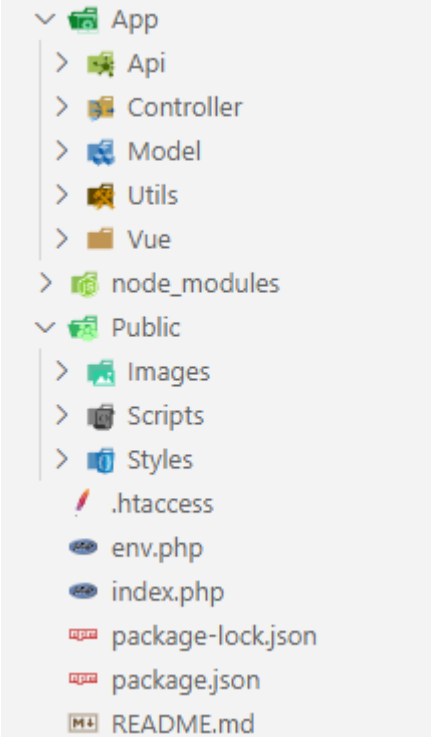
UNIQUE (email_UserAccount): Garantit que chaque adresse e-mail de la colonne email_UserAccount est unique, empêchant de créer plusieurs comptes avec la même adresse e-mail.

CONSTRAINT fk_AccountType_id_AccountType **FOREIGN KEY** (id_AccountType) **REFERENCES** AccountType(id_AccountType): Définit une contrainte de clé étrangère qui établit une relation entre la colonne id_AccountType de la table UserAccount et la colonne id_AccountType de la table AccountType. Cette contrainte garantit l'intégrité référentielle en vérifiant que les valeurs de la colonne id_AccountType de UserAccount doivent exister dans la colonne id_AccountType de AccountType.

En plus de ce gestionnaire de requêtes intégré, HeidiSQL présente une interface graphique agréable et facile à prendre en main afin de gérer les BDD. Il permet entre

autres, via son interface graphique, sans avoir à effectuer des requêtes SQL comme présenté plus haut, créer, supprimer modifier des BDD, des tables, des données, ainsi que de les exporter dans un fichier format sql, qui pourrait ensuite être importé dans une autre BDD.

11. Code du projet



Le projet a été construit avec un code en architecture **MVC (Model, View, Controller)**, modèle de conception couramment utilisé dans le développement logiciel pour organiser et structurer une application. Voici les définitions des composants de l'architecture MVC :

App est le dossier global contenant tout le projet.

Api est le dossier contenant les fichiers gérant l'API qui sera mise en place dans le futur.

Controller est le dossier contenant tous les contrôleurs.Le contrôleur reçoit les actions et les événements de l'utilisateur, interagit avec le modèle pour obtenir les données nécessaires et communique avec la vue pour mettre à jour l'interface utilisateur en conséquence. Il orchestre le flux de contrôle entre le modèle et la vue, traitant les entrées utilisateur, effectuant des opérations de mise à jour et coordonnant les interactions globales de l'application.

Model est le dossier contenant tous les fichiers model: Le modèle représente les données et la logique de l'application. Il gère l'accès aux données, effectue les opérations de lecture/écriture et contient les règles de validation et de traitement des données. Le modèle est généralement indépendant de l'interface utilisateur ou des contraintes de présentation.

Il contient notamment les classes PHP ainsi que les requêtes SQL effectuées avec les données entrées dans les vues.

Vue est le dossier contenant tous les fichiers vue: La vue est responsable de l'affichage des données et de l'interface utilisateur. Elle présente les informations au format approprié pour l'utilisateur et interagit avec le contrôleur pour récupérer les données nécessaires à l'affichage. Dans certains cas, la vue peut également gérer les interactions utilisateur de base.

Les **Utils** regroupent des fonctionnalités réutilisables qui ne sont pas spécifiques à un modèle, une vue ou un contrôleur particulier. Dans mon cas, le dossier Utils contient un fichier contenant la fonction de connexion à la BDD, ainsi qu'un fichier contenant la fonction qui sert à nettoyer les données entrées par les utilisateurs dans les formulaires (inscription et connexion notamment).

```php
<?php
    namespace App\Utils;
    ...
    class Functions{
        public static function cleanInput($value){
            return htmlspecialchars(strip_tags(trim($value)));
        }
    }
?>
```

Cette **fonction**, nommée "**cleanInputs**", va récupérer la valeur d'un champ, puis passer cette valeur à travers plusieurs "filtres":

La **fonction trim($value)** est utilisée pour supprimer les espaces en début et en fin de la valeur. Cela permet de supprimer les éventuels espaces indésirables qui pourraient être présents dans la valeur.

La **fonction strip_tags($value)** est utilisée pour supprimer toutes les balises HTML et PHP de la valeur. Cela permet de s'assurer que la valeur ne contient pas de code potentiellement dangereux ou indésirable.

La **fonction htmlspecialchars($value)** est utilisée pour convertir certains caractères spéciaux en entités HTML équivalentes. Cela aide à éviter les problèmes de sécurité tels que les attaques par injection de code.

En combinant ces trois fonctions, la méthode **cleanInput** effectue un nettoyage basique de la valeur fournie en entrée, en supprimant les espaces indésirables, les balises et les caractères spéciaux.

Le dossier **node_modules** contient les fichiers de packages que j'ai installé via npm grâce au terminal. Dans mon cas nous avons Bootstrap 5.3.0 ainsi que Bootstrap-icons 1.10.5

Le dossier **Public** regroupe tous les fichiers servant de ressources accessibles depuis le navigateur comme les fichiers CSS, JavaScript, Images, …

Le fichier **.htaccess** est un fichier de configuration de serveur Apache.

Le fichier **index.php** est le routeur du site. C'est à dire que c'est ce fichier qui s'occupera de faire que en sorte que les URLs du site renvoient au bon endroit.

L'architecture **MVC** vise à séparer les préoccupations et à favoriser la maintenabilité et la réutilisabilité du code. En divisant l'application en ces composants distincts, chaque partie peut évoluer indépendamment des autres, ce qui facilite la modification ou l'extension d'une fonctionnalité spécifique sans affecter les autres parties de l'application.

11.1) Code HTML (formulaire d'inscription)

App > Vue > Register.php > ...

```php
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Créer une compte</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef048
    <link rel="stylesheet" href="../../node_modules/bootstrap-icons/font/bootstrap-icons.css">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/
    <script src="../../Public/Scripts/register_error_message.js" defer></script>
</head>

<body>
    <?php include './App/Vue/Navbar.php'; ?>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <h1 class="text-center">Formulaire d'inscription</h1>
                <form action="" method="POST" enctype="multipart/form-data">
                    <div class="mb-3">
                        <label for="firstname" class="form-label">Votre nom</label>
                        <input type="text" class="form-control" name="firstname">
                    </div>
                    <div class="mb-3">
                        <label for="lastname" class="form-label">Votre prénom</label>
                        <input type="text" class="form-control" name="lastname">
                    </div>
                    <div class="mb-3">
                        <label for="email" class="form-label">Adresse e-mail</label>
                        <input type="email" class="form-control" name="email">
                    </div>
                    <div class="mb-3">
                        <label for="password" class="form-label">Mot de passe</label>
                        <input type="password" class="form-control" name="password">
                    </div>
                    <div class="text-center">
                        <button type="submit" name="submit" class="btn btn-primary">S'inscrire</button>
                    </div>
                </form>
                <div class="text-center response-message"><?php echo $message; ?></div>
            </div>
        </div>
    </div>
    <?php include './App/Vue/Footer.php'; ?>
</body>
</html>
```

Dans ce code, je commence par importer les ressources dans ma balise <head>, à savoir **Boostrap** (CSS et JavaScript), **Bootstrap-icons**, ainsi qu'un petit script **JavaScript** personnel dont nous verrons l'utilité plus tard.

C'est un formulaire très simple utilisant la méthode **POST** avec 4 champs à remplir. La méthode **POST** est plus sécurisée que la méthode **GET** car les informations entrées par l'utilisateur ne circulent pas par l'URL dans la page.

Il y a plusieurs balises qui servent à diviser différentes parties de code, et chaque possède des classes propres à Bootstrap afin d'y ajouter du CSS sans avoir à l'écrire manuellement

La balise <h1> est utilisée pour afficher le titre principal de la page.

Les balises <label> sont utilisées en conjonction avec un élément de formulaire <input>, le label servant à décrire l'élément de mon formulaire, et l'input étant un champ de formulaire que l'utilisateur doit remplir avec une donnée.
L'attribut for de la balise <label> est utilisé pour spécifier l'ID de l'élément de formulaire auquel il est associé. Cela permet de lier le libellé à l'élément de formulaire de manière explicite.
L'attribut type de la balise <Input> est utilisée pour spécifier le type de donnée attendue.

11.2) Code CSS (formulaire d'inscription)

Le formulaire de création de compte ayant été entièrement stylisé avec des classes Bootstrap, je n'ai pas de code CSS à présenter pour celui-ci, mais du code CSS sera présenté à la fin de ce mémoire, en guise d'Annexe.

11.3) Code JavaScript

Public > Scripts > JS register_error_message.js > ...

```
1   document.addEventListener('DOMContentLoaded', function () {
2       const button = document.querySelector('button[name="submit"]');
3       button.addEventListener('mouseenter', function () {
4           button.classList.remove('btn-primary');
5           button.classList.add('btn-success');
6       });
7       button.addEventListener('mouseleave', function () {
8           button.classList.remove('btn-success');
9           button.classList.add('btn-primary');
10      });
11  });
```

Ce script JavaScript écoute l'événement "**DOMContentLoaded**", qui se déclenche lorsque le document HTML a été complètement chargé par le navigateur.

Une fois que l'événement est déclenché, la **fonction** est exécutée. À l'intérieur de cette fonction, le script sélectionne le premier bouton trouvé dans le document HTML avec l'attribut name ayant la valeur "submit". Il le récupère en utilisant la méthode document.querySelector().

Ensuite, deux **event listerner** sont ajoutés à ce bouton. Le premier est pour l'événement "mouseenter", qui se déclenche lorsque le curseur de la souris entre dans la zone du bouton. La fonction associée à cet événement est exécutée lorsque l'événement se produit.

À l'intérieur de la fonction pour l'événement "**mouseenter**", la classe CSS "btn-primary" est supprimée de l'élément du bouton à l'aide de la méthode **classList.remove()**. Ensuite, la classe CSS "btn-success" est ajoutée à l'élément du bouton avec la méthode **classList.add().** Cela modifie l'apparence du bouton, passant du bleu au vert.

Le deuxième event listener est pour l'événement "**mouseleave**", qui se déclenche lorsque le curseur de la souris quitte la zone du bouton. La fonction associée à cet événement est exécutée lorsque l'événement se produit.

À l'intérieur de la fonction pour l'événement "mouseleave", la classe CSS "btn-success" est supprimée de l'élément du bouton à l'aide de **classList.remove().** Ensuite, la classe CSS "btn-primary" est ajoutée à l'élément du bouton avec **classList.add().** Cela rétablit l'apparence initiale du bouton en remplaçant la classe "btn-success" par la classe "btn-primary".

11.3) Code PHP

Pour la fonctionnalité de création de compte, j'ai utilisé deux models :

- **User.php**, qui s'occupe d'enregistrer les comptes en base de données.
- **AccountType.php**, qui s'occupe d'attribuer un rôle aux comptes créés.

```php
<?php

namespace App\Model;

use App\Utils\BddConnect;
use App\Model\AccountType;

You, hier | 1 author (You)
class User extends BddConnect
{
    /*------------------------
            Attributs
    ------------------------*/
    private ?int $id_UserAccount;
    private ?string $email_UserAccount;
    private ?string $password_UserAccount;
    private ?string $firstName_UserAccount;
    private ?string $lastName_UserAccount;
    private ?bool $isDisabled_UserAccount;
    private ?AccountType $accountType;
```

Le **namespace** correspond au chemin auquel se situe le fichier, en l'occurence "App\Model".

"**use**" sert à indiquer au fichier que nous aurons besoin d'utiliser les classes et fonctions présentes dans d'autres fichiers.

La **classe "User"** étend la classe "**BddConnect**", qui est une classe permettant la connexion à la BDD. La classe user aura donc accès à la BDD pour y enregistrer les comptes.

Les **attributs de ma classe User sont déclarés en private.** En déclarant les attributs d'une classe PHP en privé, on favorise l'encapsulation des données, le contrôle d'accès, et la sécurité du code.

```php
    /*------------------------
            Constructeur
    ------------------------*/
    public function __construct()
    {
        $this->accountType = new AccountType();
        $this->accountType->setAccountTypeId(1);
    }
```

Ici, je configure le **constructeur** de la classe User afin qu'à la création d'un nouveau compte, il y attribue le rôle possédant l'id "1" dans ma table AccountType

```php
    /*-----------------------
        Getters et Setters
    -----------------------*/
    public function getUserId(): ?int
    {
        return $this->id_UserAccount;
    }
    public function setUserId(?int $id): void
    {
        $this->id_UserAccount = $id;
    }
    public function getUserEmail(): ?string
    {
        return $this->email_UserAccount;
    }
    public function setUserEmail(?string $email): void
    {
        $this->email_UserAccount = $email;
    }
    public function getUserPassword(): ?string
    {
        return $this->password_UserAccount;
    }
    public function setUserPassword(?string $pwd): void
    {
        $this->password_UserAccount = $pwd;
    }
    public function getUserFirstName(): ?string
    {
        return $this->firstName_UserAccount;
    }
    public function setUserFirstName(?string $firstName): void
    {
        $this->firstName_UserAccount = $firstName;
    }
    public function getUserLastName(): ?string
    {
        return $this->lastName_UserAccount;
    }
    public function setUserLastName(?string $lastName): void
    {
        $this->lastName_UserAccount = $lastName;
    }
```

Afin de pouvoir accéder à mes attributs en dehors de la classe User, j'ai mis un place un système de **Getters & Setters**, des **fonctions en public** pouvant être utilisées depuis n'importe quel fichier de mon projet et permettant de récupérer ou modifier la valeur des attributs de ma classe User.

```php
/*-----------------------
        Méthodes
------------------------*/
//Méthode pour ajouter un utilisateur en BDD
public function addUserAccount(): void
{
    try {
        $firstName = $this->lastName_UserAccount;
        $lastName = $this->firstName_UserAccount;
        $email = $this->email_UserAccount;
        $password = $this->password_UserAccount;
        $id_AccountType = $this->accountType->getAccountTypeId();
        $req = $this->connexion()->prepare('INSERT INTO
        useraccount(firstName_UserAccount, lastName_UserAccount, email_UserAccount, password_UserAccount, id_AccountType)
        VALUES(?,?,?,?,?)');
        $req->bindParam(1, $firstName, \PDO::PARAM_STR);
        $req->bindParam(2, $lastName, \PDO::PARAM_STR);
        $req->bindParam(3, $email, \PDO::PARAM_STR);
        $req->bindParam(4, $password, \PDO::PARAM_STR);
        $req->bindParam(5, $id_AccountType, \PDO::PARAM_INT);
        $req->execute();
    } catch (\Exception $e) {
        die('Erreur : ' . $e->getMessage());
    }
}
```

La **méthode addUserAccount** de ma classe User permet d'ajouter les données d'un utilisateur en base de données

La méthode est composée d'un "**try**" et d'un "**catch**", le "**try**" étant les instructions à exécuter si tout se passe bien et le "**catch**" servant à gérant les exceptions en me retournant un message d'erreur avec un "**die**".

Lors du "**try**", chaque attribut de l'objet est attribué à une variable locale unique.
Je prépare et stocke ensuite une requête SQL dans la variable $req à l'aide de la fonction **prepare**, qui suit la **fonction connexion** me permettant de me connecter à ma BDD.
J'utilise la fonction **bindParam** afin de lier la valeur de chaque variable locale à la bonne colonne de la table dans laquelle elle sera envoyée en BDD.

Si un problème se produit lors du "**try**", celui-ci est attrapé par le "**catch**" et le message d'erreur est stocké dans une variable via "**die**".

Fichier **UserController.php** :

App > Controller > UserController.php > UserController

```php
You, il y a 1 minute | 1 author (You)
<?php

namespace App\Controller;

use App\Utils\Functions;
use App\Model\User;


You, il y a 1 minute | 1 author (You)
class UserController extends User
{

    public function insertUser()
    {
        $message = "";

        if (isset($_POST['submit'])) {;
            $firstname = Functions::cleanInput($_POST['firstname']);
            $lastname = Functions::cleanInput($_POST['lastname']);
            $email = Functions::cleanInput($_POST['email']);
            $password = Functions::cleanInput($_POST['password']);

            if (!empty($firstname) and !empty($lastname) and !empty($email) and !empty($password)) {
                $this->setUserEmail($email);

                if ($this->getUserByEmail()) {
                    $message = "Le compte existe déjà";
                } else {
                    $password = password_hash($password, PASSWORD_DEFAULT);
                    $this->setUserFirstName($firstname);
                    $this->setUserLastName($lastname);
                    $this->setUserPassword($password);
                    $this->addUserAccount();
                }
                $message = "Vous avez bien été enregistré.";
            } else {
                $message = "Veuillez remplir tous les champs";
            }
        }
        include './App/Vue/Register.php';
    }
```

Ma fonction “**insertUser**” effectuer plusieurs étapes :

Elle initialise une variable **$message** vide. Cette variable sera utilisée pour stocker les messages de retour destinés à l'utilisateur.

Elle vérifie si le formulaire a été soumis en vérifiant si la variable **$_POST['submit']** est définie.

Si le formulaire a été soumis, elle récupère les valeurs des champs du formulaire (**$_POST['firstname'], $_POST['lastname'], $_POST['email'],**

$_POST['password']) en utilisant la fonction **Functions::cleanInput()** dont il a été question plus tôt dans ce mémoire.

Ensuite, elle vérifie si tous les champs (**$firstname, $lastname, $email, $password**) sont non vides.

Si tous les champs sont remplis, elle appelle les méthodes appropriées pour configurer les valeurs de l'utilisateur dans l'objet courant.

Elle vérifie si un compte utilisateur avec l'email fourni existe déjà en appelant la méthode **$this->getUserByEmail()**. Si un compte existe déjà, elle assigne le message "Le compte existe déjà" à la variable $message.

Si aucun compte utilisateur avec cet email n'existe, elle utilise la fonction **password_hash()** pour hacher le mot de passe fourni.

Elle appelle les méthodes appropriées pour configurer les valeurs du prénom, du nom et du mot de passe de l'utilisateur dans l'objet courant.

Elle appelle la méthode **$this->addUserAccount()** pour ajouter le compte utilisateur à la base de données.

Elle assigne le message "Vous avez bien été enregistré" à la variable $message.

Si un ou plusieurs champs du formulaire sont vides, elle assigne le message "Veuillez remplir tous les champs" à la variable **$message.**

Enfin, elle inclut le fichier de vue "Register.php" qui sera utilisé pour afficher le formulaire d'inscription à l'utilisateur.

En résumé, cette fonction vérifie et traite les données soumises via un formulaire d'inscription, effectue des vérifications et des opérations liées à la création d'un nouvel utilisateur, et renvoie des messages de retour à l'utilisateur via la variable $message.

12. Conclusion

Cette formation au sein du pôle numérique de l'Adrar m'a permis d'explorer dans un contexte plus sérieux et encadré le code, qui est quelque chose qui a été pour moi un passe-temps auquel j'aimais m'essayer lorsque l'occasion se présentait. Cela m'a permis de me rendre compte des différents aspects que ce corps de métier avait à proposer et de me rendre compte de ce qui m'attirait ou non (attirance prononcée pour le back-end, très peu pour le front et toute la partie graphique de façon générale).

La stage pour le collectif NAVA m'a permis de me rendre compte des difficultés du style de travail freelance, statut qui semble être très prisé dans le secteur du développement.

Suite à cette formation, j'ai bien entendu l'intention de continuer à approfondir mes connaissances, tout d'abord en continuant de travailler sur le projet NAVA, en lançant un nouveau projet à titre personnel, dans l'espoir de pouvoir intégrer une entreprise ou pourquoi pas une formation à niveau plus élevé si l'occasion se présente.

13. Annexes

```php
    //Méthode pour récupérer un utilisateur avec son email
    public function getUserByEmail(): ?array
    {
        try {
            $email = $this->email_UserAccount;
            $req = $this->connexion()->prepare('SELECT id_UserAccount, firstName_UserAccount, lastName_UserAccount,
                email_UserAccount, password_UserAccount, isDisabled_UserAccount, id_AccountType
                FROM UserAccount WHERE email_UserAccount = ?');
            $req->bindParam(1, $email, \PDO::PARAM_STR);
            $req->execute();
            $data = $req->fetchAll(\PDO::FETCH_OBJ);
            return $data;
        }
        catch (\Exception $e) {
            die('Erreur : ' . $e->getMessage());
        }
    }
```

```php
    public function connectUser()
    {
        $message = "";

        if (isset($_POST['submit'])) {
            $email = Functions::cleanInput($_POST['email']);
            $password = Functions::cleanInput($_POST['password']);

            if (!empty($email) and !empty($password)) {
                $this->setUserEmail($email);
                $this->setUserPassword($password);

                if ($this->getUserByEmail()) {
                    $data = $this->getUserByEmail();

                    if ($data) {
                        if (password_verify($password, $data[0]->password_UserAccount)) {
                            $_SESSION['connected'] = true;
                            $_SESSION['id'] = $data[0]->id_UserAccount;
                            $_SESSION['email'] = $data[0]->email_UserAccount;
                            $_SESSION['firstname'] = $data[0]->firstName_UserAccount;
                            $_SESSION['lastname'] = $data[0]->lastName_UserAccount;
                            $_SESSION['is_Disabled'] = $data[0]->isDisabled_UserAccount;
                            $_SESSION['AccountType'] = $data[0]->id_AccountType;


                            $message = "Vous êtes connecté";
                            // include '.App/Vue/viewAccount.php';
                        } else {
                            $message = "Le mail ou le mot de passe est incorrect";
                        }
                    } else {
                        $message = "Le mail ou le mot de passe est incorrect";
                    }
                } else {
                    $message = "Le compte n'existe pas, veuillez vous inscrire";
                }
            } else {
                $message = "Veuillez remplir les champs";
            }
        }
        include './App/Vue/Login.php';
    }
```

```
86      public function disconnectUser()
87      {
88          session_destroy();
89          header('Location: ./');
90      }
```

Footer stylisé avec Bootstrap, puis ajout de CSS custom :

App > Vue > Footer.php > style

```
You, il y a 7 heures | 1 author (You)
<footer class="footer bg-dark text-white">
    <div class="container">
        <div class="row">
            <div class="col-md-6 mb-4">
                <h5 class="text-uppercase">Newsletter</h5>
                <p class="mb-2">Pour rester à jour sur nos projets !</p>
                <div class="input-group">
                    <input type="email" class="form-control rounded-0 custom-input-width-small" placeholder="Votre adresse mail">
                    <button class="btn btn-primary rounded-0 mt-2 mt-md-0" type="button">S'abonner</button>
                </div>
            </div>
            <div class="col-md-3 mb-4">
                <h5 class="text-uppercase">About</h5>
                <ul class="list-unstyled">
                    <li><a href="#" class="text-white">À propos</a></li>
                    <li><a href="#" class="text-white">FAQ</a></li>
                    <li><a href="#" class="text-white">Politique de confidentialité</a></li>
                </ul>
            </div>
            <div class="col-md-3 mb-4">
                <h5 class="text-uppercase">Contact</h5>
                <ul class="list-unstyled">
                    <li><a href="#" class="text-white">Email</a></li>
                    <li><a href="#" class="text-white">WhatsApp</a></li>
                    <li><a href="#" class="text-white">Adresse du studio</a></li>
                </ul>
            </div>
        </div>

        <div class="row">
            <div class="col-12">
                <hr class="mb-3">
                <ul class="list-inline text-center">
                    <li class="list-inline-item"><a href="https://twitter.com"><i class="bi bi-twitter"></i></a></li>
                    <li class="list-inline-item"><a href="https://instagram.com"><i class="bi bi-instagram"></i></a></li>
                </ul>
                <p class="text-center mb-0">&copy; 2023 NAVA. All rights reserved.</p>
            </div>
        </div>
    </div>
</footer>

<style>
    You, hier | 1 author (You)
    .list-unstyled a {
        text-decoration: none;
    }

    You, il y a 7 heures | 1 author (You)
    html,
    You, il y a 7 heures | 1 author (You)
    body {
        height: 100%;
        margin: 0;
        padding: 0;
    }

    You, il y a 7 heures | 1 author (You)
    .container {
        min-height: 100%;
        display: flex;
        flex-direction: column;
    }

    You, il y a 7 heures | 1 author (You)
    .footer {
        margin-top: auto;
        padding-top: 20px;
        padding-bottom: 20px;
    }
</style>
```

13.1) Documentation

13.1.1) Introduction :

I used the README documentation on Laragon's GitHub to find out what this software had to offer in helping me establish a reliable local development environment after being disappointed by other software like XAMPP

13.1.2) “What is Laragon ?” & “Features”

What is Laragon?

Laragon is a portable, isolated, fast & powerful universal development environment for PHP, Node.js, Python, Java, Go, Ruby. It is fast, lightweight, easy-to-use and easy-to-extend.

Laragon is great for building and managing modern web applications. It is focused on performance - designed around stability, simplicity, flexibility and freedom.

Laragon is very lightweight and will stay as lean as possible. The core binary itself is less than 2MB and uses less than 4MB RAM when running.

Laragon doesn't use Windows services. It has its own service orchestration which manages services asynchronously and non-blocking so you'll find things run fast & smoothly with Laragon.

Enjoy!

Features
Pretty URLs Use app.test instead of localhost/app.
Portable You can move Laragon folder around (to another disks, to another laptops, sync to Cloud,...) without any worries.
Isolated Laragon has an isolated environment with your OS - it will keep your system clean.
Easy Operation Unlike others which pre-config for you, Laragon auto-configs all the complicated things. That way you can add another versions of PHP, Python, Ruby, Java, Go, Apache, Nginx, MySQL, PostgreSQL, MongoDB,... effortlessly.
Modern & Powerful Laragon comes with a modern architecture which is suitable to build modern web apps. You can work with both Apache & Nginx as they are fully-managed. Also, Laragon makes things a lot easier:
Wanna have a Wordpress CMS? Just 1 click.

Wanna show your local project to customers? Just 1 click.
Wanna enable/disable a PHP extension? Just 1 click.
Laragon is trully isolated & portable. However, you may need to use the installer as it will detect and install missing run-time components that are required to run C++ applications built using Visual Studio such as PHP, Apache for you.

13.1.3) Documentation traduire

Qu'est-ce que Laragon ?

Laragon est un environnement de développement universel portable, isolé, rapide et puissant pour PHP, Node.js, Python, Java, Go, Ruby. Il est rapide, léger, facile à utiliser.

Laragon est idéal pour la création et la gestion d'applications web modernes. Il est axé sur les performances - conçu autour de la stabilité, de la simplicité, de la flexibilité et de la liberté.

Laragon est très léger et le restera autant que possible. Le binaire principal lui-même fait moins de 2 Mo et utilise moins de 4 Mo de RAM lorsqu'il est en cours d'exécution.

Laragon n'utilise pas les services Windows. Il dispose de sa propre orchestration de services qui gère les services de manière asynchrone et non bloquante, vous constaterez donc que les choses se déroulent rapidement et en douceur avec Laragon.

Profitez-en !

Fonctionnalités
URLs conviviales : Utilisez app.test au lieu de localhost/app.
Portable : Vous pouvez déplacer le dossier Laragon (vers un autre disque, un autre ordinateur portable, le synchroniser avec le Cloud, etc.) sans soucis.
Isolé : Laragon dispose d'un environnement isolé avec votre système d'exploitation - il gardera votre système propre.
Facilité d'utilisation : Contrairement à d'autres qui préconfigureront pour vous, Laragon configure automatiquement toutes les choses compliquées. De cette façon, vous pouvez ajouter d'autres versions de PHP, Python, Ruby, Java, Go, Apache, Nginx, MySQL, PostgreSQL, MongoDB,... sans effort.
Moderne et puissant : Laragon est doté d'une architecture moderne qui convient à la création d'applications web modernes. Vous pouvez travailler à la fois avec Apache et Nginx car ils sont entièrement gérés. De plus, Laragon facilite grandement les choses :

Vous voulez avoir un CMS Wordpress ? Juste 1 clic.
Vous voulez montrer votre projet local aux clients ? Juste 1 clic.
Vous voulez activer/désactiver une extension PHP ? Juste 1 clic.
Laragon est vraiment isolé et portable. Cependant, vous devrez peut-être utiliser l'installateur car il détectera et installera les composants d'exécution manquants nécessaires pour exécuter des applications C++ créées à l'aide de Visual Studio, tels que PHP et Apache, pour vous.