**Mémoire Nadège LACAN**

Formation Développeur Web / Web mobile : session FADDEV 31-21-01

**SOMMAIRE**

Introduction

Présentation Projet et contexte

I. Analyse du Projet / Cahier des charges

II. Spécifications Fonctionnelles

1. Cas d'utilisation
2. Diagramme d'activité
3. Diagramme de séquence
4. Maquettage

III. Conception d'un Système d'Information

1. MCD
2. MLD
3. Requêtes SQL

IV. Arborescence

V. Outils techniques

VI. Fonctionnalités

1. Site Web : connexion à la BDD
2. Site Web : création compte utilisateur
3. Site Web : connexion utilisateur
4. Site Web : liste des entrainements dans les 15 jours à venir
5. Site Web : Responsive Design
6. Application mobile : POO Serveur : Beans
7. Application mobile : Serveur : WebService
8. Application mobile : Client : Adapter liste entrainements
9. Recherche Anglophone
10. Tests Jeu d’essai

Conclusion

Annexe : Cahier des charges

**Introduction**

My name is Nadege and I am a student at the digital pole of adrar in Toulouse. Before, I was a computer developer, I obtained a BTS in 2003. After this I worked at the Banque Populaire, I was a developer until 2006. I then changed jobs and worked as a customer advisor in this bank.

I did this training to update my knowledge, to refresh my skills as a developer. For this review I have made a project, which I will present to you now.

My project consists in developing a website and a mobile app for motocross riders. This project is done for the French Motorbike Federation (FFM), and will be used by its members, clubs and pilots. My son is a motocross rider and explained to me how difficult it is to find open tracks for training. The FFM provides a website with a list of competitions, but there is nothing in place for riders to know which tracks are avalaible for training.

Currently, each motorbike circuit posts on its Facebook page to announce a track opening for training. So riders have to check the pages of each club to find a training track.

My project is linked to this issue. It is a website and a mobile application allowing :

- clubs to report the opening of tracks for training

- and for pilots to register for training

This will allow to access all the information in one place only. So that, its more convenient.

For this project, the website is developed in HTML, CSS and PHP. The mobile application is developed in JAVA for back-end and front-end.

This project isn’t finisched because I didn’t have enough time. The WebSite is finished with a responive design, but not the mobile application. This project helped me to practice and apply everything I learn, but I want to improve and optimize the website to propose it to the FFM.

**Présentation Projet et contexte**

L’objectif est de créer un site en ligne [www.TrainingMotoCross.fr](http://www.TrainingMotoCross.fr) permettant d’aider les pilotes et les clubs à l’organisation de leurs entraînements. Les clubs pourront positionner leurs jours d’ouverture de la piste à l'entraînement, avec le nombre de pilotes et le type de pilote admis en piste (adultes et / ou kids). Les pilotes pourront consulter la liste des pistes ouvertes et s’enregistrer à une session d'entraînement.

Actuellement, il n’existe aucun outil à disposition pour organiser ces entrainements. Les clubs réalisent des posts sur leurs pages Facebook pour informer les pilotes. Les pilotes sont contraints de consulter toutes les pages afin de trouver sur quelle piste s’entrainer.

**I. Analyse du Projet / Cahier des charges**

Le but de ce projet est de créer un outil simple d’utilisation, ergonomique et pratique pour la gestion des entrainements de motocross.

L’outil doit présenter des fonctionnalités spécifiques en fonction du type d’utilisateur club ou pilote.

Le cahier des charges complet de ce projet est fourni en annexe de ce mémoire.

**II. Spécifications Fonctionnelles**

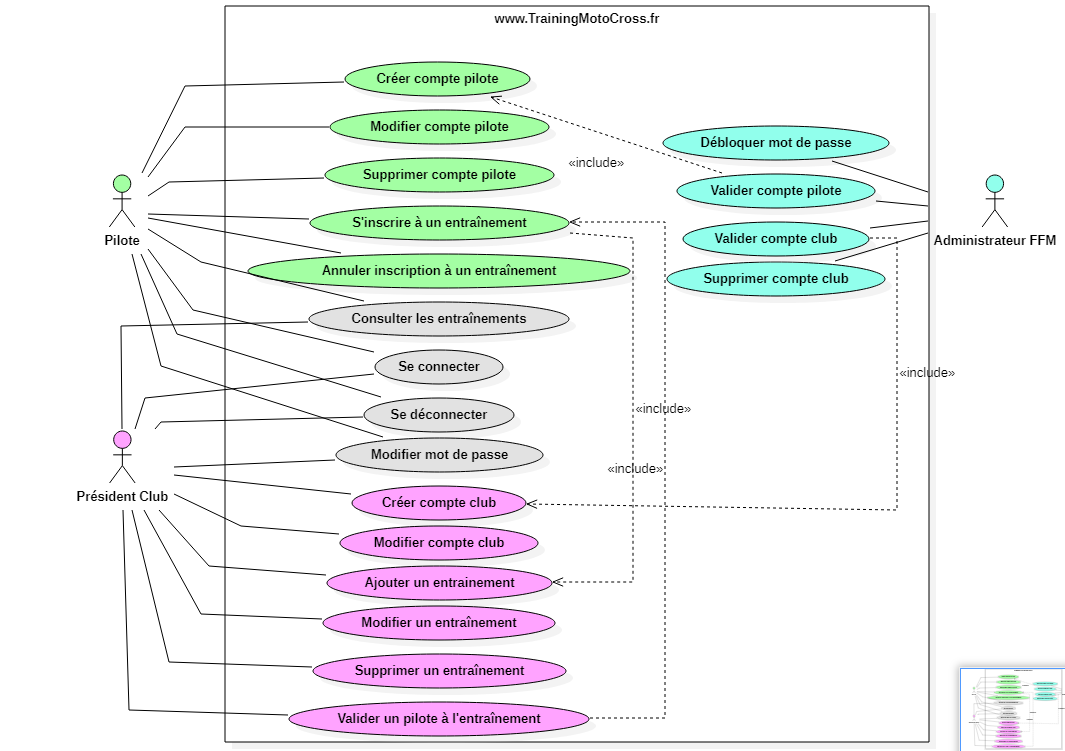
Pour mettre en œuvre ce projet et répondre aux attentes des utilisateurs, une analyse est réalisée et chaque étape d’analyse est validée afin d’obtenir un résultat final aux plus près des attentes utilisateurs.

Nous allons voir ci-dessous, les différents diagramme UML qui ont été réalisé avec l’outil StarUML, ainsi que le maquettage réalisé avec l’outil AdobeXD.

1. **Cas d'utilisation**

Dans un 1er temps, j’ai réalisé un diagramme des cas d’utilisation aussi appelé UseCase. Ce diagramme permet de définir tous les cas d’utilisation de l’application. Il permet de ne pas omettre de cas d’utilisation essentiel pour notre client et également de définir les priorités entre les différentes fonctionnalités accessibles.

Voici le UseCase réalisé pour le site [www.TrainingMotoCross.fr](http://www.TrainingMotoCross.fr) :



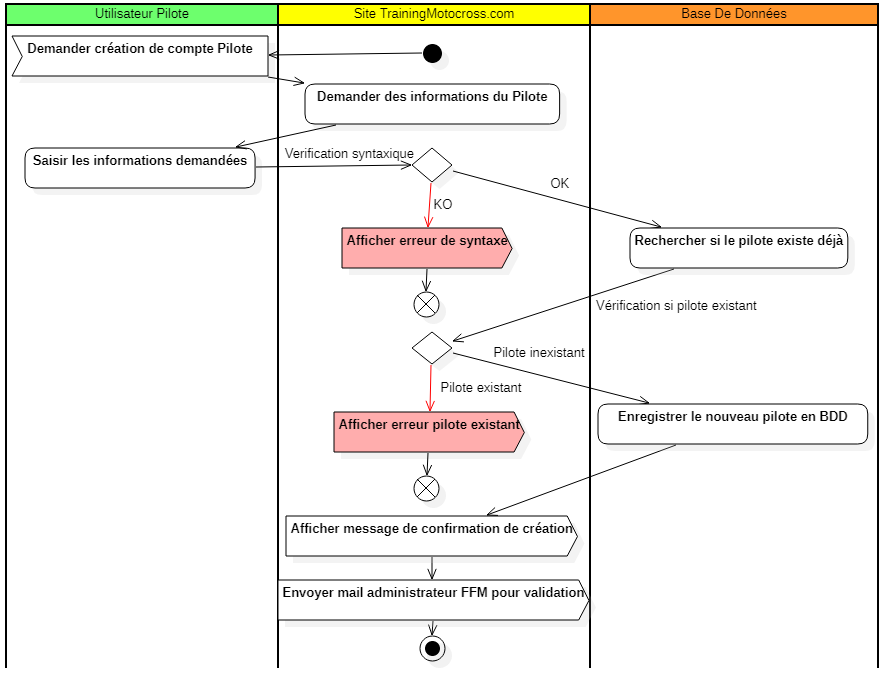
Sur ce diagramme, les différents utilisateurs sont symbolisés d’une couleur différente. En fonction du profil de l’utilisateur les cas d’utilisation sont différents.

Ici, :

* Un utilisateur « ***club*** » pourra « ***ajouter un entrainement*** »
* Un utilisateur « ***pilote*** » pourra «***s’inscrire à un entrainement*** » : à condition que celui-ci soit crée au préalable par un « ***club*** ». La condition est représentée par le « ***include*** ».

1. **Diagramme d’activité**

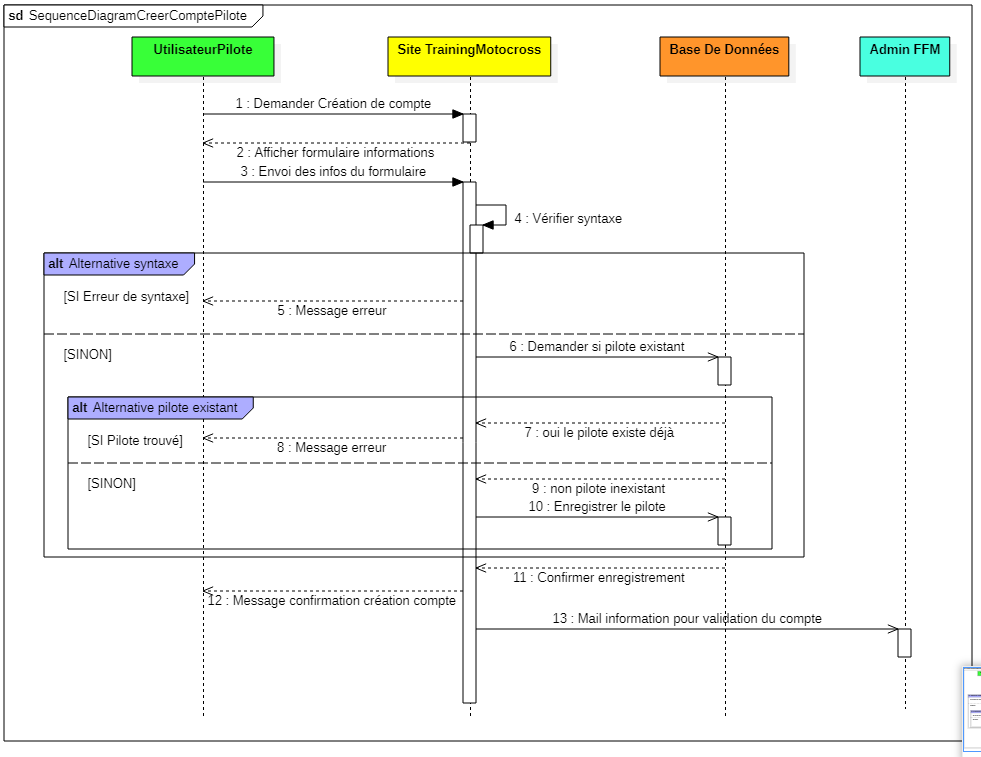
Afin de se rapprocher pas à pas du développement de cette application, chaque cas d’utilisation est détaillé sous forme de de diagramme d’activité afin de détailler le déroulement d’un cas d’utilisation. Ici vous trouverez ci-dessous, le diagramme d’activité correspondant au cas d’utilisation « ***création de compte pilote*** ».



Ce diagramme permet de visualiser le déroulement d’un cas d’utilisation et de connaître les accès nécessaires aux différentes ressources. Les différents appels à la base de données ainsi que les messages d’erreur sont représentés à ce stade.

1. **Diagramme de séquence**

Chaque diagramme d’activité peut être détaillé en diagramme de séquence, celui-ci nous apporte en plus l’information sur la temporalité, on peut y visualiser les taches qui demandent un temps de traitement. Vous trouverez ci-dessous, le diagramme de séquence correspondant au cas d’utilisation « ***créer un compte pilote*** ».



1. **Maquettage**

Avant d’effectuer le maquettage, une charte graphique a été mise en place. Elle précise notamment les couleurs et le logo qui seront utilisées pour cette application.

Les maquettes zoning, wireframe et mockup ont été réalisés avec **Adobe XD**.

* 1. ***Le logo***

Le logo utilisé pour cette application est déjà fourni par la fédération de Moto, le voici :

Une image contenant texte, clipart

Description générée automatiquement

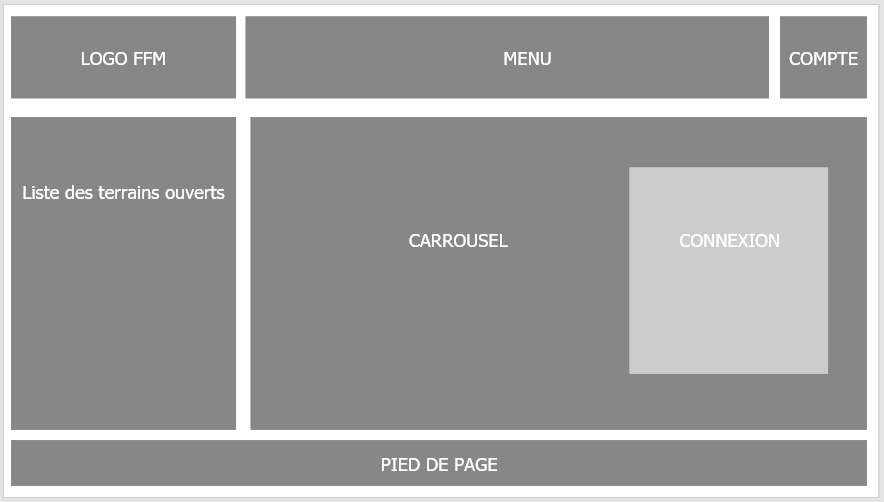
* 1. ***Les couleurs***

Les couleurs de thème choisies sont le bleu et blanc en accord avec le logo de la FFM. La palette de couleur bleu a été prélevée dans les nuances de bleus du logo. Voici les codes couleurs :



* 1. ***Zoning***

Page d’accueil site Web :



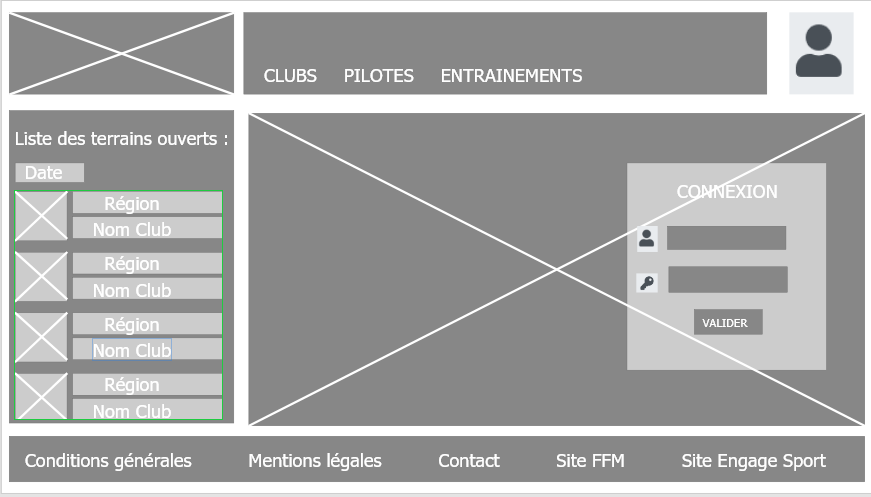
Page consulter les entraînements :

Une image contenant table

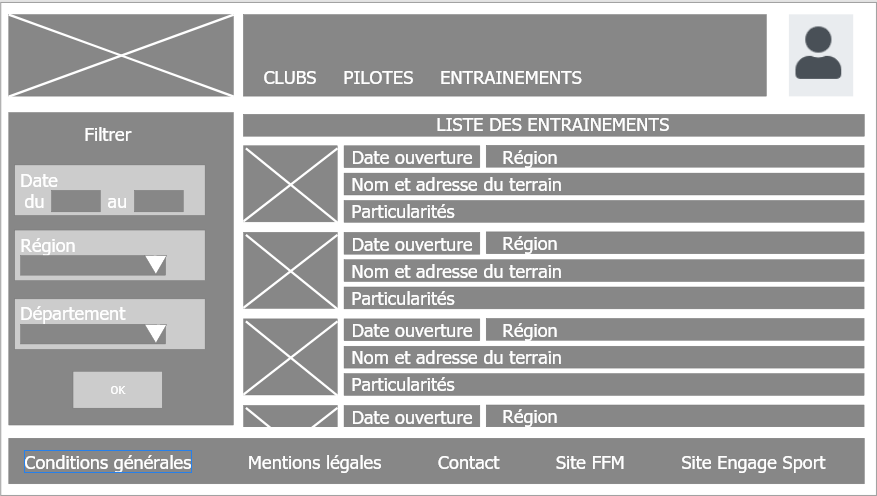
Description générée automatiquement

* 1. ***Wireframe***

Page d’accueil site Web :

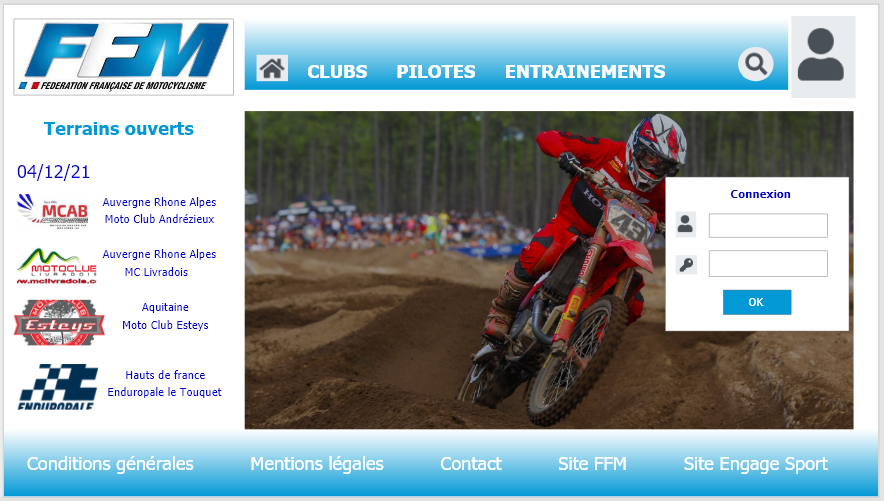


Page consulter les entraînements :



* 1. ***Mockup***

Page d’accueil site Web :



Page consulter les entraînements :



**III. Conception d'un Système d'Information**

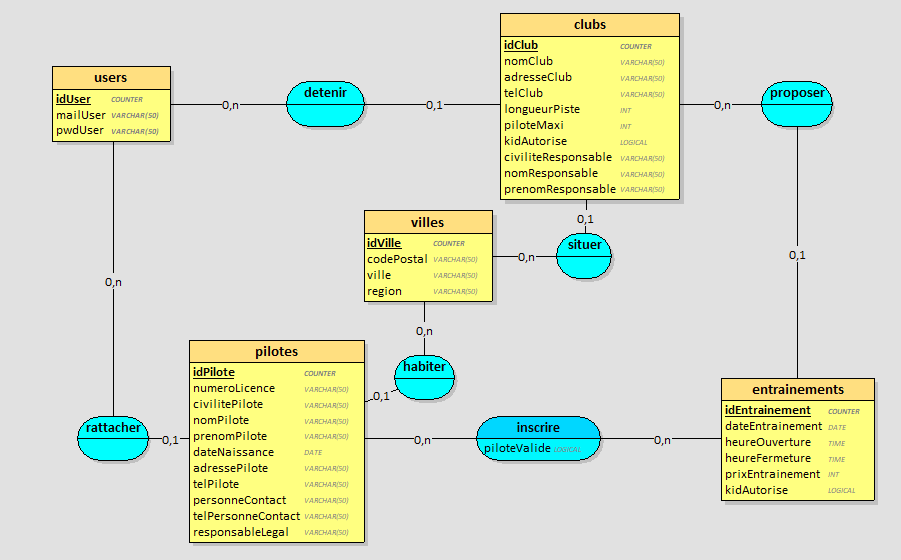
Les diagrammes et les maquettes ont permis d’obtenir la liste des données qui doivent être stockées. Ces données ont été ordonnés afin de les enregistrer de façon logique et ainsi optimiser les accès futurs.

1. **MCD Modèle Conceptuel des Données**

Le MCD ci-dessous présente 5 entités (users, clubs, villes, pilotes, entrainements) chacune de ces entités dispose d’une clé primaire en identifiant unique.

Les liens entre elles sont représentés par des associations (détenir, rattacher, situer, habiter, inscrire, proposer).

Chaque association est liée à une entité par des cardinalités de type 0.1 ou 0.n

****

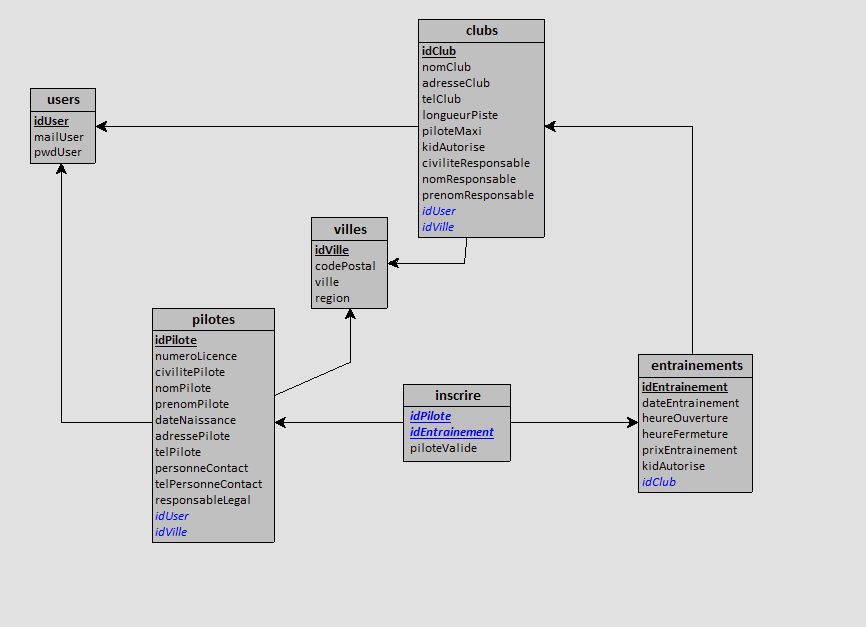
1. **MLD Modèle Logique des Données**

Le MCD a été traduit en MLD, ici les entités sont devenues des tables et les associations ont été transformés en fonction des cardinalités.

On peut voir que :

* L’association **habiter** entre les entités **pilotes** et **villes** a été transformé. La table **pilotes** a absorbée la clé primaire de la table **villes** qui est devenu ici une clé étrangère.
* L’association **inscrire** entre les entités **pilotes** et **entrainements** est devenue la table d’association **inscrire**, elle absorbe les clés primaires **idPilote** et **idEntrainement** des tables **pilotes** et **entrainements** qui sont ici clés étrangères. Et sa clé primaire est constitué du couple **idPilote** et **idEntrainement**.

A ce stade, on peut établir le script de création de la base de données.

****

1. **Requêtes SQL**
   1. ***Requête création de la base de données***

Voici un extrait du script de création de ma Base de Données :

***Une image contenant texte

Description générée automatiquement***

***Une image contenant texte

Description générée automatiquement***

… création des tables clubs et entrainements ...

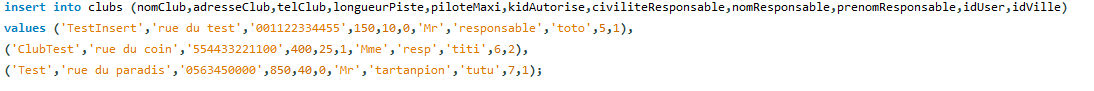
***Une image contenant texte

Description générée automatiquement***

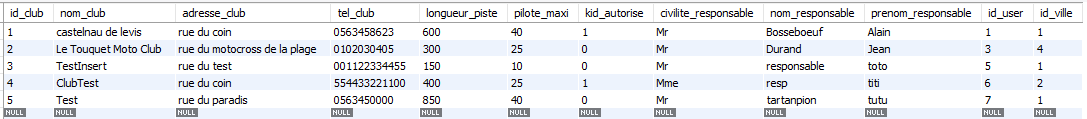
Les tables de la BDD sont créées à partir du MLD en respectant un ordre de création, ici la table **USERS** et **VILLES** doit être crée avant les tables **CLUBS** et **PILOTES**. Et la table **INSCRIRE** est crée après les tables **PILOTES** et **ENTRAINEMENTS.**

Les clés primaires des tables sont autos incrémentées.

Voici un jeu de test d’insertion des données dans la table **CLUBS** :



Voici les 3 enregistrements crées avec les id\_club 3 4 et 5 :





* 1. ***Requête SELECT***

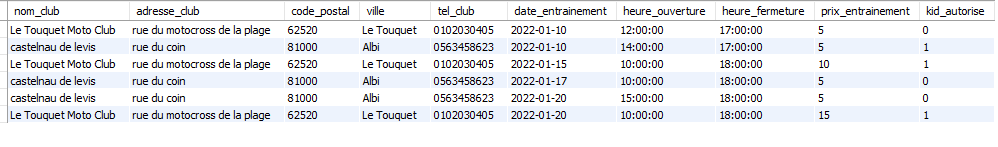
Pour l’affichage des entrainements disponible sur chaque page, sur le volet gauche : requête qui récupère la liste des entrainements enregistrés en BDD dont la date est supérieure à la date du jour et dans les 15 jours qui suivente, trié par date d’ouverture.



Une image contenant texte

Description générée automatiquement

Voici le résultat obtenu au 06/01/2022 :



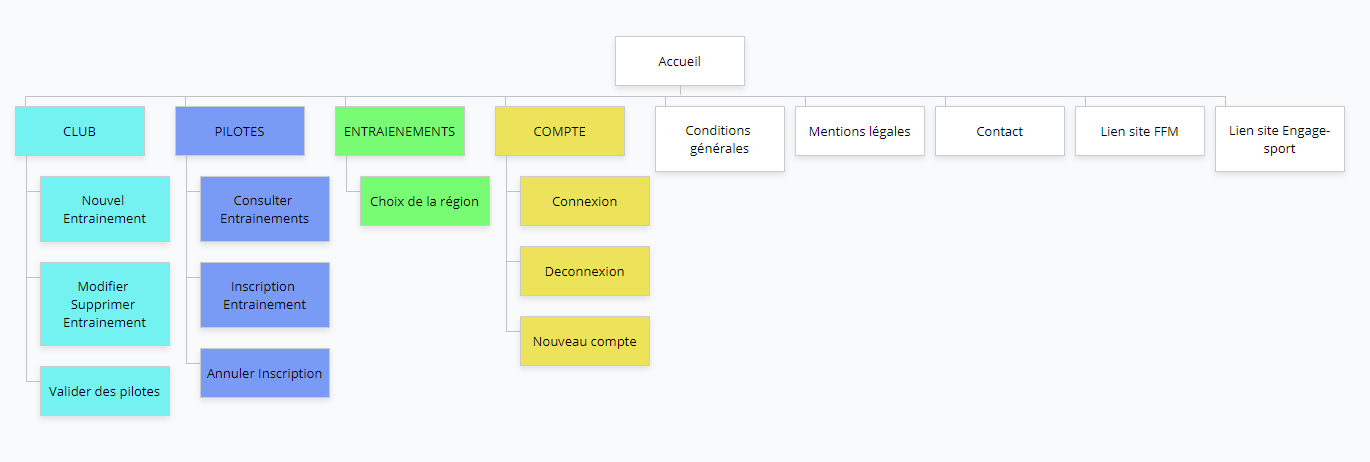
Pour permettre l’affichage des pilotes à valider pour un club : requête qui récupère la liste des pilotes inscrits à un entrainement futur (date entrainement > à la date du jour), qui ne sont pas validés et uniquement pour les entrainements sur la piste du club concerné. Ici le **?** sera remplacé par l’**idUser** du club connecté au site.

Une image contenant texte

Description générée automatiquement

**IV. Arborescence**

L’arborescence du site internet a été établie avec **gloomaps**, le but étant d’accéder à toutes les pages en 2 clics maximum.

****

**V. Outils techniques**

**Pour la base de données :**

* Le choix a été d’utiliser MySQL afin d’avoir une base de données stable qui peut être utilisée en simultanée par le site Web et l’application mobile

**Pour le site Web :**

* Développement avec les langages HTML, CSS et PHP. Il y a également un peu de JS.
* IntelliJ a été utilisé comme IDE
* WAMP a été utilisé pour effectuer les tests avec un serveur local

**Pour l’application mobile, côté serveur, back-end :**

* Développement en Programmation Orientée Objet en JAVA
* IntelliJ a été utilisé comme IDE
* Utilisation des Framework SpringBoot, JPA et OkHttp
* Les tests ont été effectué avec le navigateur et l’outil Advanced Rest Cient

**Pour l’application mobile, côté client, front-end :**

* Développement en JAVA avec Android Studio

**Pour la sauvegarde de mon travail :**

* J’ai utilisé GIT pour versionner et segmenter mon travail

**VI. Fonctionnalités**

1. **Site Web : connexion à la BDD**

Une image contenant texte

Description générée automatiquement

La connexion à la BDD est enregistrée dans le fichier connectBDD.php. Il sera appelé dès lors qu’une page nécessite un accès à la BDD.

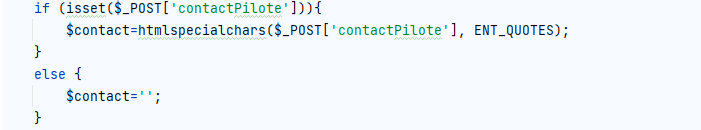
1. **Site Web : création compte utilisateur**

Une image contenant texte

Description générée automatiquement

… Ici l’utilisateur à créer est un Pilote. On retire tous les caractères spéciaux des champs obligatoires récupérés depuis le formulaire pour des raisons de sécurité et éviter notamment les injections SQL …

… Pour les champs non obligatoires, on retire les caractères spéciaux s’ils sont renseignés sinon on mais les champs à blanc …



… INSERT des données dans les différentes tables …

… Dans la table USERS on enregistre le mot de passe crypté à l’aide de BCRYPT …

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

… Procédure semblable si l’utilisateur à créer est un club …

… Si l’utilisateur n’est ni un club ni un pilote, affichage d’un message d’erreur …

… Si une exception est levée lors de l’exécution des requêtes, affichage d’un message d’erreur …

… Si tout est OK, on redirige vers une page de confirmation de création …

Une image contenant texte

Description générée automatiquement

1. **Site Web : connexion utilisateur**

Ici on enregistre l’identifiant mail saisi par l’utilisateur en Id de Session, on va rechercher en BDD l’utilisateur concerné et on effectue un contrôle du mot de passe saisie avec celui crypté en BDD grâce à **password\_verify**

Une image contenant texte

Description générée automatiquement

1. **Site Web : liste des entrainements dans les 15 jours à venir**

Sur cet extrait de code, on effectue une requête sur notre BDD pour récupérer la liste des entrainements à venir dans les 15 jours qui suivent.

Ensuite on boucle sur le résultat obtenu afin d’effectuer l’affichage des données en HTML dans notre page Web.

Une image contenant texte

Description générée automatiquement

Ce code, a été enregistré dans un fichier **training\_15jours.php**. Ce fichier sera appelé par les pages qui nécessite l’affichage de cette liste dans le Frame Gauche.

1. **Site Web : Responsive Design**

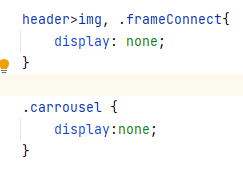
Comme tout site internet aujourd’hui et pour répondre aux normes de la W3C, le site doit être adapté aux écrans de téléphone mobile.

Afin que les mises en page soient différentes sur les écrans mobiles j’ai utilisé les MediasQueries du langage CSS pour la mise en forme.

Voici l’écran d’accueil du site internet en mode mobile et une partie du code CSS :

On agit sur les écrans dont la largeur est inférieure à 900 pixels

On change les couleurs et le positionnement de l’entête

Une image contenant texte

Description générée automatiquement

On masque le logo et le pavé de connexion On masque le carrousel

1. **Application mobile : POO Serveur : Beans**

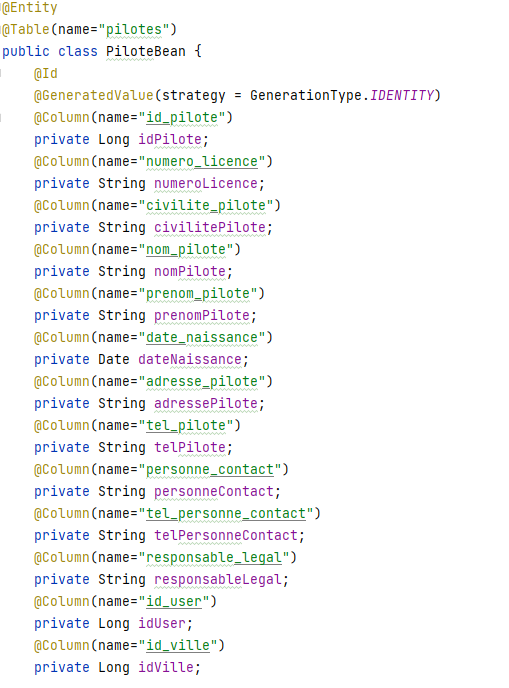
Afin de pouvoir relier mon serveur SpringBoot et JPA à ma base de données, j’ai réalisé la connexion, puis j’ai réalisé des beans qui correspondent aux tables de ma BDD pour pouvoir relier ma BDD et j’ai créé les DAO correspondantes pour pouvoir accéder aux données.

Voici le script de connexion présent dans le fichier **application.properties** :

Une image contenant texte

Description générée automatiquement

Voici **PiloteBean**, qui correspond et qui est rattaché à notre table **pilotes** :



Dans **PiloteBean**, j’ai été contrainte de nommer les attributs différemment dans car le caractère \_ n’était pas supporté.

Voici l’interface **PiloteDAO**, qui nous permet d’accéder est de manipuler les données de notre BDD :

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

1. **Application mobile : Serveur : WebService**

Maintenant que les beans et les DAO ont été créés, nous pouvons utiliser et manipuler les données de notre BDD.

Ici le rôle du serveur est de mettre à disposition des WebService pour notre application cliente.

J’’ai donc crée une classe WebServices de type RestController, dans laquelle j’ai déclaré les différentes DAO pour permettre l’accés aux données :

Une image contenant texte

Description générée automatiquement

* 1. ***Connexion d’un utilisateur***

Ici il s’agit d’une méthode de type POST, pour laquelle l’application cliente doit nous transmettre des données au format JSON.

Les données reçues doivent correspondre à un **UserBean**. La méthode retourne **l’idUser**, 0 ou -1 en cas d’erreur d’identification.

Le contrôle du mail et mot de passe s’effectue dans la méthode **verifUser** implémentée dans la classe **OkHttpUtils**

Une image contenant texte

Description générée automatiquement

Voici la méthode **verifUser**, qui prend en paramètre login et mot de passe fourni par l’utilisateur et le login et mot de passe présent en BDD via notre **userDAO**.

On vérifie l’égalité des logins et des mots de passe par l’intermédiaire de BCRYPT.

La méthode retourne true si les données saisies et celles en BDD sont identiques, false sinon.

Une image contenant texte

Description générée automatiquement

* 1. ***Liste des entrainements à venir***

Ici nous allons faire un zoom sur la méthode **GetTraining** qui retourne la liste des entrainements à venir sous forme de fichier JSON.

Pour le retour toutes les données utiles à l’affichage par notre client sont présentes dans le JSON.

Le fichier JSON retourné sera donc de type ArrayList de **TrainingBean**.

Voici la déclaration de la classe **TrainingBean** :

Une image contenant texte, table

Description générée automatiquement

On peut voir que les champs aux formats date et heure ont été formatés et transformés en String pour pallier aux différences de formats entre notre BDD rattachée au serveur et notre application cliente.

Voici la méthode qui peut être appelé par notre application cliente. Cette méthode est de type GetMapping.

Son nom et url : /**GetTraining**

Elle ne nécessite aucun paramètre en entrée.

Ici, on crée une instance de type **Calendar** afin de récupérer la date du jour. A l’aide de **entrainementDAO** on récupère la liste des entrainements. Ensuite on boucle sur la liste des entrainements récupérés pour créer une liste de type **TrainingBean** que l’on alimente à partir de la liste récupérée et des données sur le club et la ville (à l’aide de **clubDAO** et **villeDAO**)

Cette méthode retourne une liste de **TrainingBean** au format json.

Une image contenant texte

Description générée automatiquement

1. **Application mobile : Client : Adapter liste entrainements**

Pour l’application mobile Android, développée en JAVA avec l’outil Android Studio, nous allons faire un zoom sur la page d’accueil de l’application.

Cette page affiche la liste des entrainements à venir classé par date.

Notre application cliente va récupérer la liste des entrainements à venir sur notre serveur à l’aide de la méthode **GetTraining** que nous avons vu plus haut.

Pour mémoire, cette méthode nous retourne un fichier JSON de **TrainingBean**, de ce fait **TrainingBean** a été créé du côté client.

Ainsi dans notre MainActivity on déclare une ArrayList de **TrainingBean** et un adapter qui est basé sur cette ArrayList.

Une image contenant texte

Description générée automatiquement

Dans la méthode OnCreate de MainActivity on rattache notre adapter à notre élément graphique RecyclerView qui est **rvTraining**.

Une image contenant texte

Description générée automatiquement

Ensuite dans un nouveau Thread on appelle la méthode **loadTraining()** afin d’actualiser notre variable **listTraining**. L’adapter est mis à jour par l’appel à **showList()** :

Une image contenant texte

Description générée automatiquement

*Voici la méthode showList :*

Une image contenant texte

Description générée automatiquement

*Voici la méthode loadTraining :*

Une image contenant texte

Description générée automatiquement

*Voici la méthode sendGetOkHttpRequest :*

Une image contenant texte

Description générée automatiquement

Ces extraits de code permettent de visualiser comment depuis l’application cliente nous pouvons récupérer les données de notre serveur et les utiliser pour les afficher sur un élément graphique de l’application cliente.

1. **Recherche Anglophone**

Dans le cadre de ce projet j’ai été amené à effectuer de nombreuses recherches sur internet. La plupart de la documentation est en anglais.

Dans mon projet les formats Date pour l’application mobile m’ont posé pas mal de tracas. De ce fait, je vous partage ici un extrait de la documentation en anglais sur SimpleDateFormat que j’ai été amené à comprendre et à utiliser :

Une image contenant texte

Description générée automatiquement

**Voici la traduction que je vous propose :**

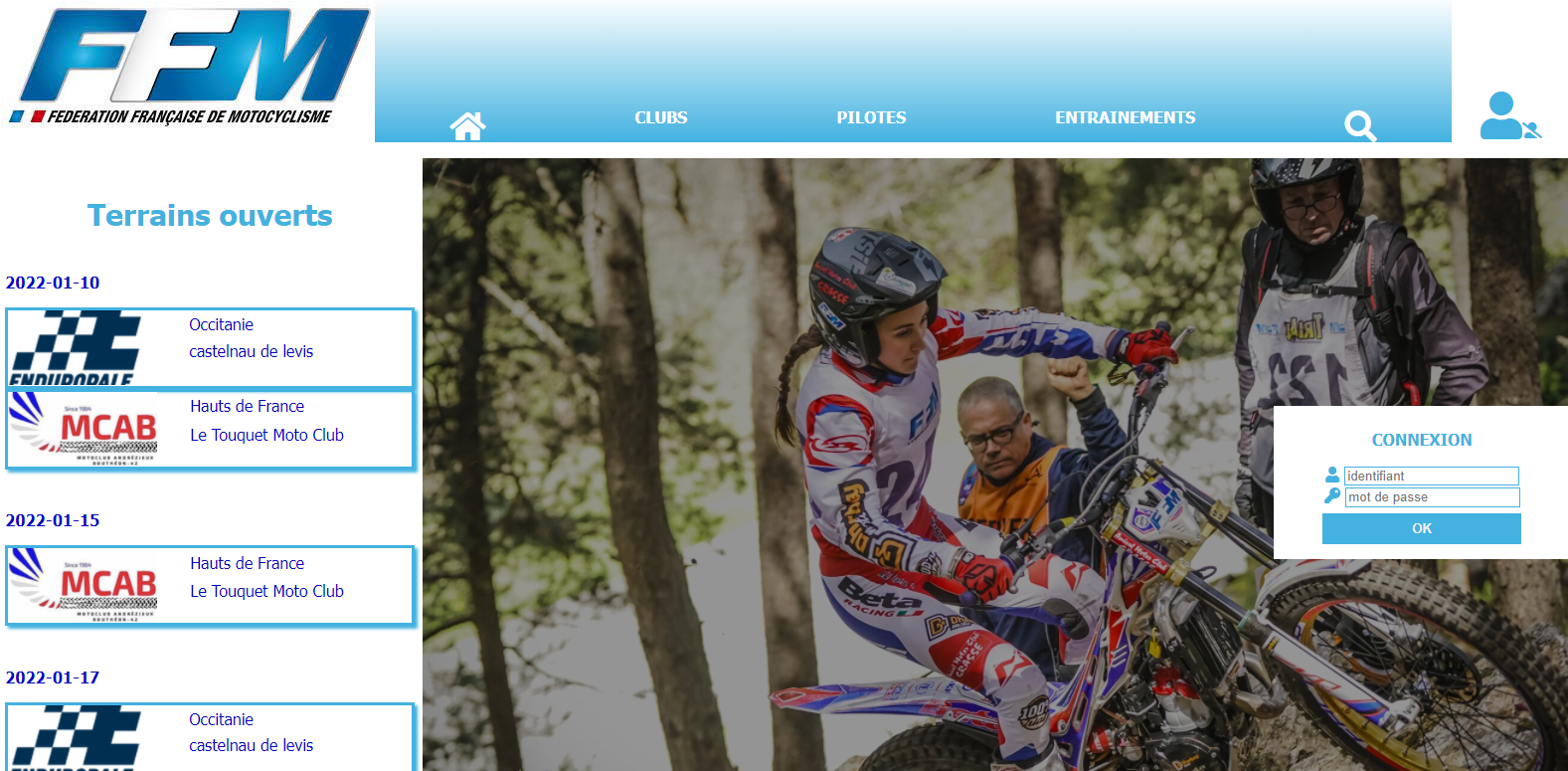
SimpleDateFormat est une classe concrète pour formater et parser les dates à sa manière (de manière locale). Cela permet de formater des dates en texte, de parser du texte en date et la normalisation.

SimpleDateFormat vous permet dès le départ de choisir un modèle défini par l’utilisateur pour le formatage des dates et heures. Cependant, nous vous encourageons à créer un modèle de format date/heure avec les méthodes getTimeInstance, getDateInstance ou getDateTimeInstance de DateFormat. Chacune de ces méthodes de classe peuvent retourner un modèle de format date/heure initialisé à partir d’un modèle de format par défaut. Vous pouvez modifier le modèle de format en utilisant les méthodes applyPattern si désiré. Pour plus d’information sur l’utilisation de ces méthodes, voir DateFormat.

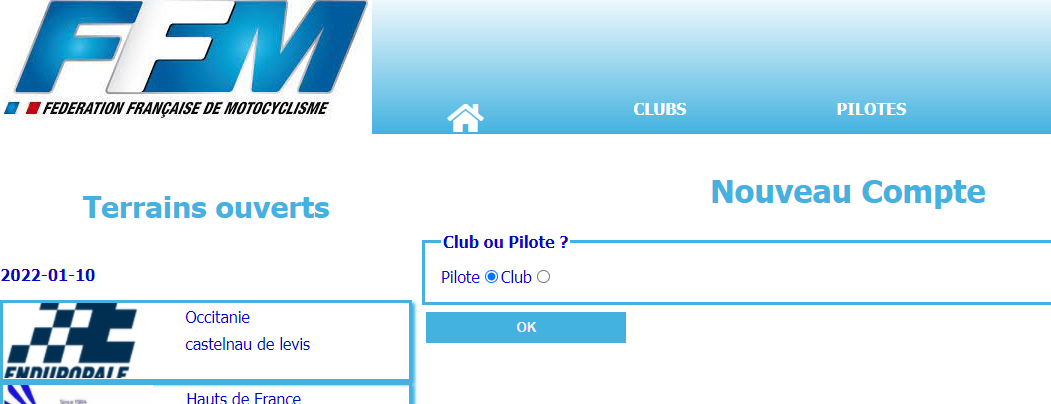
1. **Tests Jeu d’essai**

Je vous présente ici un jeu d’essai pour la création d’un compte pilote suivi d’une inscription à un entrainement.

Page accueil du site :



Sur clic, page création de compte :



Sur clic, affichage du formulaire sélectionné club ou pilote, ici pilote :



Sur clic de validation de ce formulaire de création, page de confirmation :

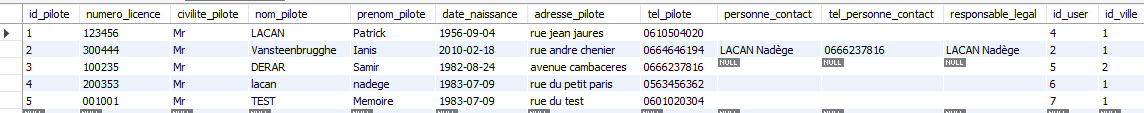


On peut vérifier en base de données les enregistrements crées dans les tables **users** et **pilotes** :

Une image contenant texte

Description générée automatiquement





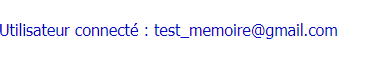


Connexion de ce nouvel utilisateur :

Une image contenant texte, extérieur, personne, course

Description générée automatiquement

Sur clic, on peut voir le login de l’utilisateur connecté :



Sur clic dans le menu pilotes, liste des entrainements pour inscription :

Une image contenant texte

Description générée automatiquement

Dans ce test, j’ai inscrit le pilote sur le 1er entrainement de la liste, voici la confirmation obtenue sur clic :

Une image contenant texte

Description générée automatiquement

On peut également vérifier en BDD, que le pilote est bien enregistré sur cet entrainement :

Une image contenant table

Description générée automatiquement



**Conclusion**

Ce projet n’est actuellement pas terminé, de nombreux éléments sont encore absents. La partie site internet est plus avancé que l’application mobile. Ici le site web, qui se doit d’être responsive sera bien plus utilisé qu’une application mobile par les utilisateurs, c’est pour cela que cette partie a été priorisée.

Sur le site internet, il manque des éléments dont :

* L’envoi de mail pour la validation des comptes utilisateurs par l’administrateur FFM
* Le contrôle de cohérence entre le nombre de pilote validé et le nombre de pilote maximum autorisé
* Le contrôle de cohérence entre l’âge des pilotes qui s’inscrivent et l’autorisation des kids sur la piste
* L’envoi de mail automatique aux clubs pour la validation des pilotes inscrits

Ce projet a été réalisé dans le cadre de ma formation mais à l’issue de celle-ci je vais proposer le projet du site internet à la FFM.

Ce projet fil rouge qui a été réalisé en plusieurs étapes tout au long de ma formation, il m’a permis de mettre en pratique les différents cours suivis et de relier mes différentes compétences entre elles afin d’aboutir à un projet global.

Également, une partie du site internet est réalisée en PHP, j’ai appris ce langage en autodidacte lors de mon stage en entreprise. J’ai réutilisé ce langage et ces nouvelles compétences dans ce projet. Ceci m’a demandé d’effectuer des recherches de comprendre la documentation, de regarder des tutoriels afin de comprendre, d’apprendre et de me lancer.

Ce projet ainsi que l’apprentissage de nouveau langages m’ont beaucoup apporté à titre personnel.

Cela m’a permis de reconnaître mes compétences, ma capacité d’adaptation et d’auto-formation sur de nouveaux langages ou de nouveaux outils.