Table de matières

[**Dédicaces** 4](#_Toc51938768)

[**Remerciement** 5](#_Toc51938769)

[**Introduction générale** 6](#_Toc51938770)

[**Chapitre 1** 7](#_Toc51938771)

[**CONTEXTE GENERAL DU PROJET** 7](#_Toc51938772)

[1 Présentation Du Projet 8](#_Toc51938773)

[1.1 Etude de l’existant: 8](#_Toc51938774)

[1.2 Problématique 8](#_Toc51938775)

[1.3 Solutions proposées : 8](#_Toc51938776)

[1.4 Cahier de charges 9](#_Toc51938777)

[1.5 Analyse des besoins 10](#_Toc51938778)

[1.6 Diagramme Représentant La Planification Du Projet: 10](#_Toc51938779)

[**Chapitre II** 11](#_Toc51938780)

[**LA CONCEPTION** 11](#_Toc51938781)

[**2** **Méthodologie d’analyse** 12](#_Toc51938782)

[2.1 **cycle en V** 12](#_Toc51938783)

[2.2 **Le langage UML:** 14](#_Toc51938784)

[2.3 **Le modèle MVC** 14](#_Toc51938785)

[2.3.1 **Définition:** 14](#_Toc51938786)

[2.3.2 **Pourquoi le modèle MVC?** 16](#_Toc51938787)

[2.4 **Modélisation du contexte:** 16](#_Toc51938788)

[**2.4.1** **Les acteurs et leurs rôles:** 16](#_Toc51938789)

[**Modélisation du contexte:** 19](#_Toc51938790)

[2.4.2 **Diagramme de Package:** 19](#_Toc51938791)

[2.4.3 **Diagrammes des cas** d’ut**ilisation:** 20](#_Toc51938792)

[**2.4.4** **Diagrammes de Séquence:** 21](#_Toc51938793)

[**2.4.5** **Diagrammes de classe :** 24](#_Toc51938794)

[**Chapitre III** 27](#_Toc51938795)

[**PRESENTATION DE** L’APPLICATION 27](#_Toc51938796)

[3 **Outils de développement** 29](#_Toc51938797)

[3.1 **HTML 5** 29](#_Toc51938798)

[3.2 **CSS 3** 29](#_Toc51938799)

[3.3 **PHP** 29](#_Toc51938800)

[3.4 **JavaScript** 29](#_Toc51938801)

[3.5 **Ajax** 30](#_Toc51938803)

[3.6 **MySQL** 30](#_Toc51938804)

[3.7 **PHPQrCode** 30](#_Toc51938805)

[4.1 **Présentation de la phase authentification :** 30](#_Toc51938807)

[**Conclusion** 32](#_Toc51938808)

[**Conclusion générale:** 33](#_Toc51938809)

# **Introduction générale**

Dans le cadre de l'obtention de notre diplôme de MASTER en …, nous avons  
été appelés à réaliser un projet de fin d'études afin de clôturer notre formation du second cycle universitaire. C'est ainsi que j'ai eu l'occasion d'approfondir mes connaissances théoriques par la conception et la réalisation d'une application de gestion des membres et des espaces verts. Cette application a été conçue dans le dessein de gérer les membres, les espaces verts et les abonnements de ces derniers afin de faciliter la tâche à FORMA-GREEN. Pour atteindre cet objectif, nous avons créé une application web, modélisée à partir du langage UML (langage de modélisation unifié). Le langage de programmation choisi est le  
langage PHP et le système de gestion de base de données (SGBD)  
est MySQL. Il faut noter que l'outil Star UML nous a été utile pour dessiner et gérer les différents diagrammes UML. Ce travail a été effectué sous le tutorat du docteur …. Dans ce rapport nous allons présenter dans un premier temps la société un cahier de charges pour expliciter les objectifs de l’application. Dans une autre partie nous allons exposer la méthodologie et les outils utilisés dans la réalisation du projet. Enfin nous allons passer à la conception et la réalisation de l’application.

# **Chapitre 1**

# **CONTEXTE GENERAL DU PROJET**

# Présentation Du Projet

## Etude de l’existant:

Forma-green reçoit chaque jour un grand nombre de personnes qui souhaitent s’abonner pour bénéficier des soins d’entretien des espaces verts. Ces derniers sont enregistrés via un système de cartes papier / carton et un fichier Excel et leur suivi se realise via la consultation quotidienne de ce fichier

## Problématique

FORMA-GREEN ne dispose d’aucun logiciel informatique permettant la gestion des espaces verts. Les membres de la structure sont enregistrés via un système de cartes papier / carton et un fichier Excel. Ceci rend leurs tâches fastidieuses et difficiles, et le travail compliqué, lent et mal structuré.

Parmi ces problèmes:

* **Problème de gestion** l’ajout et la suppression des membres,espaces verts,abonnement se fait avec Excel chaque filière est dans un fichier Excel appart ce qui mène à une perte de temps et une mauvaise organisation de travail.
* **Difficulté** d’**accès à l**’i**nformation** chaque membre est stockée avec ses espaces verts et ses differentes periodes d’abonnement, ce qui engendre le parcours de plusieurs lignes pour consulter les membres et leur espaces verts
* **Problème de gestion de l**’**information** il n’y a pas de base de données pour stocker les listes des membres et leurs espaces verts ce qui peut mener à une perte d’information.
* **Problème de sécurité** n’importe quelle personne peut accéder aux informations et les modifier.

## Solutions proposées :

Afin de remédier aux problèmes présentés dans la section précédente, Forma-Green nous a contacté pour que nous prenions en charge leur projet de numérisation du système de gestion des membres et des espaces verts.L’application doit nous permettre de :

* Faciliter l’enregistrement des membres.
* Faciliter le suivi des abonnements.
* diminuer les risques d’erreur.
* Faciliter la génération des cartes d’abonnement.
* Garder trace sur toutes les activités effectuées par l’administrateur du service.
* Garder l’archive des soucriptions aux abonnements.
* Garantir la sécurité (Authentification).
* Rendre le travail plus structuré et dynamique.

## Cahier de charges

Module 1 : Authentification

* Inscription
* Login

Module 2 : Gestion des membres

* Ajouter un membre
* Supprimer un membre
* Modifier un membre
* Afficher la liste des membres

Module 3 : Gestion des espaces verts

* Localiser les espaces verts de chaque membre sur une carte
* Ajouter un espace vert
* Supprimer un espace vert
* Afficher la liste des membres ayant un espace vert
* Afficher la liste des membres n’ayant pas un espace vert

Module 4 : Gestion de L’abonnement

L’abonnement ici concerne la durée pendant laquelle le membre souhaite travailler avec forma-green

Notifier le membre deux mois avant l’expiration de son abonnement après cela :

* Le membre peut renouveler
* Le membre mettre fin

Module 5 : Gestion du QR code ou Bar code sur la carte forma-green du membre

Lorsque le QR code ou Bar code d’un membre sera scanner celui-ci retournera toutes les données du membre, les espaces verts que le membre possède, et la date de validité de l’abonnement

Module 6 : Gestion des cadeaux

Au vu des partenariats de forma-green et ses différents partenaires, forma-green recevra des cadeaux, et à leur tour ils vont offrir ses différents cadeaux aux membres en fonction du nombre d’espace vert qu’ils auront.

Module 7

* Gestion de la sécurité des données et des intrusions externes
* Gestion de sauvegarde des données

## Analyse des besoins

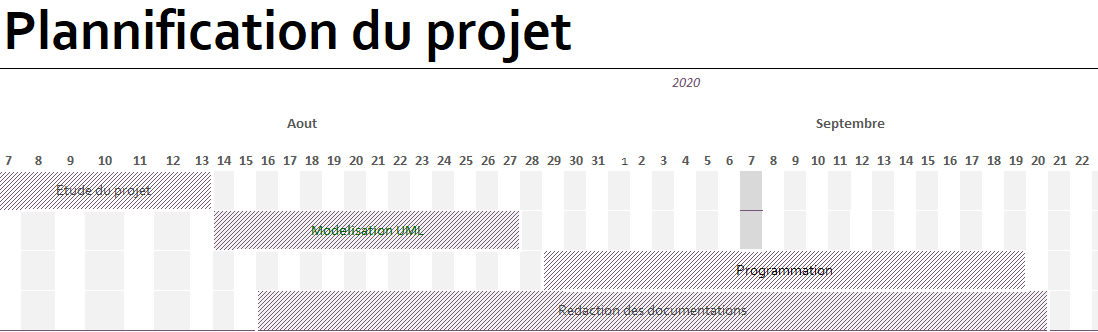
Dans cette partie nous allons clarifier les besoins du système à réaliser pour pouvoir clarifier les besoins des utilisateurs de l’application.

* **Authentification :** pour pouvoir accéder à l’interface de gestion des espaces verts.
* **Gestion des membres :** il s’agit des différentes opérations de gestion des membres telles que l’ajout et la consultation de la liste des membres.
* **Gestion des espaces verts :**il s’agit des différentes opérations de gestion des espaces verts telles que l’ajout et la consultation de la liste des espaces verts.
* **Gestion des abonnements :** il s’agit des différentes opérations de gestion des abonnements telles que l’ajout et la consultation de la liste des abonnements.
* **Generation des cartes d’abonnement** : Suite à la souscription à un abonnement, l’application doit pouvoir generer des cartes sur lesquelles doivent figuré le QR code ou le code barre qui crypte les infos concernant le membre et son abonnement
* Gestion des cadeaux :il s’agit des différentes opérations de gestion des cadeaux telles que l’ajout d’un nouveau don de cadeaux et la distribution des cadeaux aux membres.
* **Consultation** d’h**istorique** consultation des activités effectuées par l’administrateur.

## Diagramme Représentant La Planification Du Projet:

Chaque projet nécessite un planning représentant visuellement l'état d'avancement des différentes activités qui constituent un projet. Mon projet de fin d’études **G**estion des espaces verts est réalisé selon le planning représenté sous forme de diagramme de Gant réalisé à l’aide de Tom’sPlanner.

Le diagramme de Gantt est un outil utilisé en ordonnancement et en gestion de projet et permettant de visualiser dans le temps les diverses tâches composant un projet



**Figure 3 : planning du projet**

# **Chapitre II**

# **LA CONCEPTION**

**Introduction:**

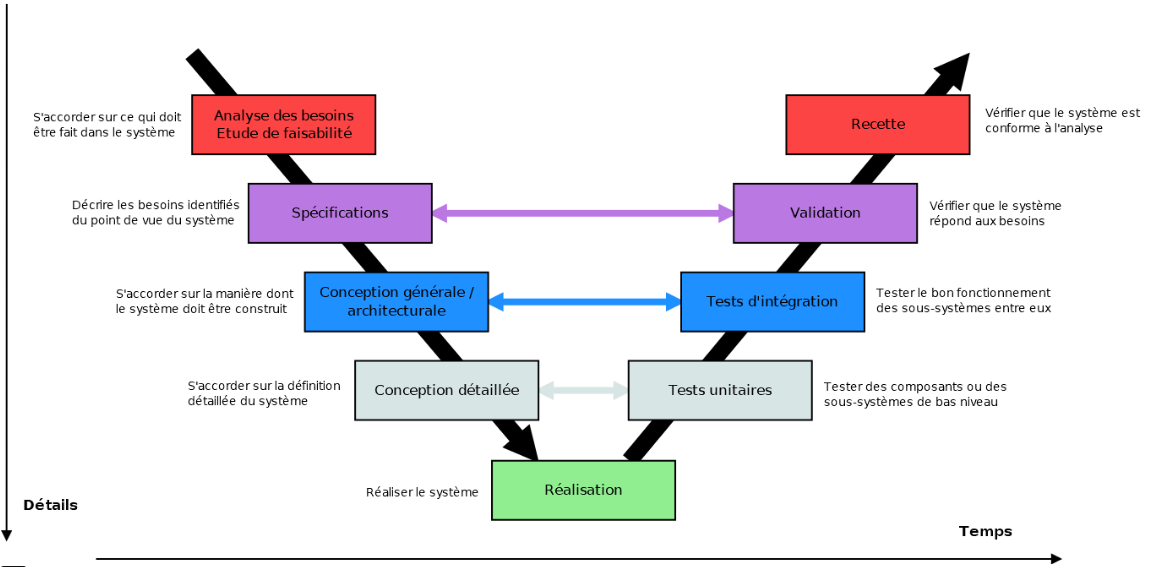
La conception est une phase importante avant la réalisation de tout projet, Cette phase nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer. C’est à dire créer une représentation similaire à la réalité de telle façon à faire ressortir les points auxquels on s'intéresse.

Nous allons commencer ce chapitre par la présentation de la méthodologie d’analyse, en justifiant la raison du choix de cette méthodologie pour notre application, nous allons montrer ensuite les acteurs de l’application et leurs rôles, ainsi que les différents diagrammes UML.

# **Méthodologie d’analyse**

## **cycle en V**

Pour développer mon application « gestion des espaces verts » j’ai utilisé la methode de cycle en V. C’est un modèle d'organisation des activités d'un projet qui se caractérise par un flux d'activité descendant qui détaille le produit jusqu'à sa réalisation, et un flux ascendant, qui assemble le produit en vérifiant sa qualité.



**Figure 4 : Processus de développement en V**

**Le** flux d'activité descendant **comporte:**

* **l’analyse des besoins(Exigences) et l’etude de la faisabilité** : les exigences font l'objet d'une expression des besoins. Le cas échéant, une étude de faisabilité peut être conduite avant d'engager les travaux
* **Analyse** : il s'agit à partir de l'expression de besoin d'établir le cahier des charges fonctionnel ou les spécification fonctionnelle
* **Conception générale, aussi appelé conception architecturale ou conception préliminaire**: il s'agit de concevoir le système qui doit répondre aux exigences et de définir son architecture, et en particulier les différents composants nécessaires;
* **Conception détaillée**: il s'agit de concevoir chaque composant, et la manière dont ils contribuent à la réponse aux besoins;
* **Réalisation** : il s'agit de réaliser chaque composant nécessaire. Pour les composants et systèmes logiciels, l'activité est essentiellement le codage;

**Le flux d'activité ascendant comporte:**

* **Test unitaire**: il s'agit de vérifier le bon fonctionnement et la conformité de chaque composant à sa conception détaillée;
* **Intégration** et test d'intégration: il s'agit d'assembler le système à partir de tous ses composants, et de vérifier que le système dans son ensemble fonctionne conformément à sa conception générale;
* **Validation (anciennement « tests fonctionnels »)** : vérification que le système est conforme aux exigences;
* **Recette** (également appelés « Test d’acceptation » dans le contexte de la sous-traitance) : validation du système par rapport à sa conformité aux besoins exprimés

## **Le langage UML:**

**Définition:**

**Figure 5 : Le langage de modélisation UML**

**UML**, c’est l’acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un langage visuel constitué d’un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer: son fonctionnement, sa mise en route, les actions susceptibles d’être effectuées par le logiciel, etc.

**Pourquoi UML?**

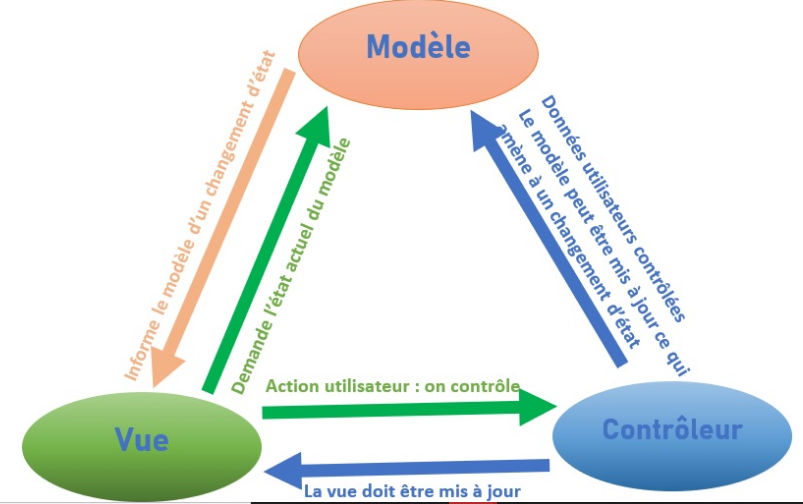
* Pour obtenir une modélisation de très haut niveau indépendante des langages et des environnements.
* Pour faire collaborer des participants de tous horizons autour d'un même document de synthèse.
* Pour faire des simulations avant de construire un système.
* Pour exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications, etc....
* Pour documenter un projet.
* Pour générer automatiquement la partie logicielle d'un système.

## **Le modèle MVC**

### **Définition:**

L’architecture **MVC** (modèle, vue et contrôleur) est un concept très puissant qui intervient dans la réalisation d’une application. Son principal intérêt est la séparation des données (modèle), de l’affichage (vue) et des actions (contrôleur), ce qui assure la clarté de l’architecture et simplifie la tâche du développeur responsable de la maintenance et de l’amélioration du projet.

Les différentes interactions entre le modèle, la vue et le contrôleur sont résumées par le schéma de la figure:



**Figure 6: L'architecture MVC**

**Modèle:** Le modèle représente le cœur de l’application: traitements des données, interactions avec la base de données. Il décrit les données manipulées par l’application. Il regroupe la gestion de ces données et, il est responsable de leur intégrité. La base de données sera l’un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle ne s’occupent pas de la présentation, Le modèle ne contient aucun lien direct vers la vue.

**Vue:** C’est avec quoi l’utilisateur interagit se nomme précisément la vue. Sa première tâche est de présenter les résultats renvoyés par le modèle, sa seconde tâche est de recevoir toute action de l’utilisateur (clic de souris, sélection d’un

bouton radio, coche d’une case, entrée de texte, de mouvements, de voix, etc..). Ces différents événements sont envoyés au contrôleur. La vue n’effectue pas de traitement, elle se contente d’afficher les résultats des traitements effectués par le modèle et d’interagir avec l’utilisateur.

**Contrôleur:** Le contrôleur prend en charge la gestion des événements de

synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l’utilisateur et déclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ce dernier notifie la vue que les données ont changée pour qu’elle se mette à jour.

### **Pourquoi le modèle MVC?**

 La clarté de l'architecture qu'il impose.

 La simplification des tâches au développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet.

## **Modélisation du contexte:**

### **Les acteurs et leurs rôles:**

|  |  |
| --- | --- |
| **Acteur** | Rôles |
| Membres | * S’inscrire * S’authentifier * Modifier son profil * Gestion des abonnements   + Souscrire à un abonnement   + Renouveler un abonnement   + Mettre fin à un abonnement   + Lister ses abonnements * Gestion des espaces verts   + Ajouter un espace vert   + Modifier un espace vert   + Supprimer un espace vert   + Lister ses espaces verts * Imprimer la carte d’abonnement * Faire des dons |
| Administrateur | * S’authentifier * Gestion des membres   + Ajouter un membre   + Modifier les infos d’un membre   + Supprimer un membre   + Afficher la liste des membres * Gestion des abonnements   + Ajouter un abonnement   + Renouveler un abonnement   + Mettre fin à un abonnement   + Lister les abonnements   + Supprimer un abonnement * Gestion des espaces verts   + Ajouter un espace vert   + Modifier un espace vert   + Supprimer un espace vert   + Afficher la liste des membres ayant un espace vert   + Afficher la liste des membres n’ayant pas un espace vert * Imprimer la carte d’abonnement * Faire des dons |

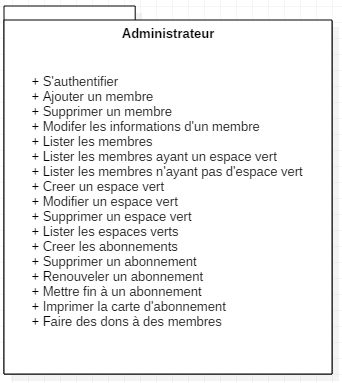
Un acteur est une entité externe qui interagit avec le système (opérateur, centre distant, autre système...). En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Les acteurs peuvent être classés (hiérarchie). Pour notre projet les acteurs sont les suivants : les membres et l’administrateur

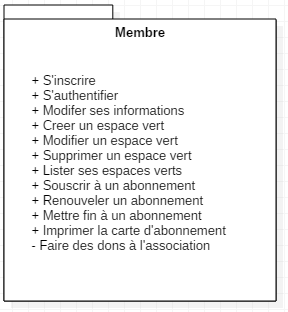
**Tableau 2: acteur et rôle**

## **Modélisation du contexte:**

### **Diagramme de Package:**

Le diagramme de packages permet de décomposer le système en catégories ou parties plus facilement observables, appelés « packages ». Cela permet également d’indiquer les acteurs qui interviennent dans chacun des packages





**Figure 7: Diagramme de package**

### **Diagrammes des cas** d’ut**ilisation:**

Le diagramme de cas d’utilisation représente les fonctionnalités (ou dit cas d’utilisation) nécessaires aux utilisateurs. On peut faire un diagramme de cas d’utilisation pour le logiciel entier ou pour chaque

package. Nous avons donc fait le choix de produire deux dianrammes pour nos deux packages

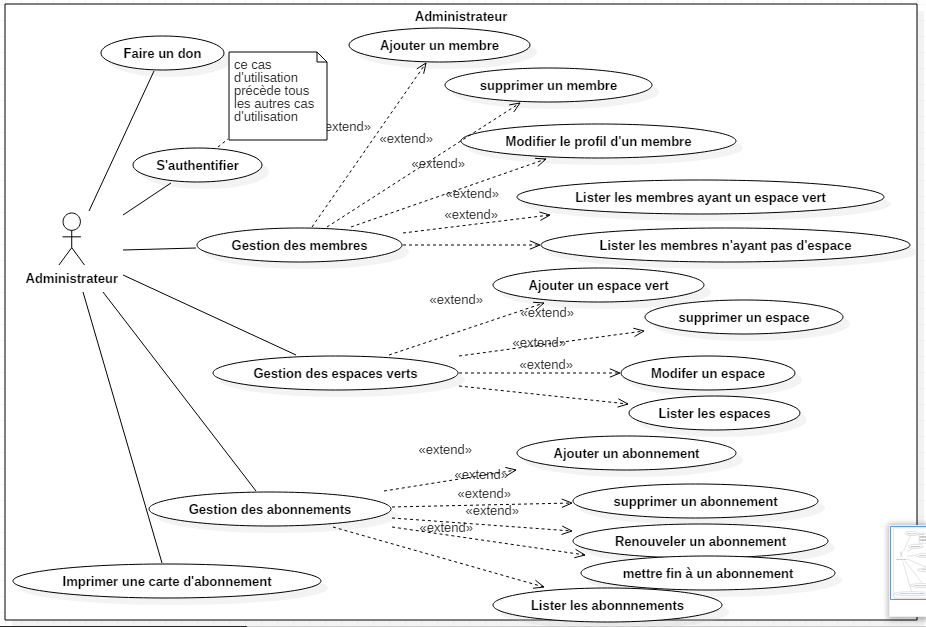


Figure : Cas d’utilisation du package administrateur

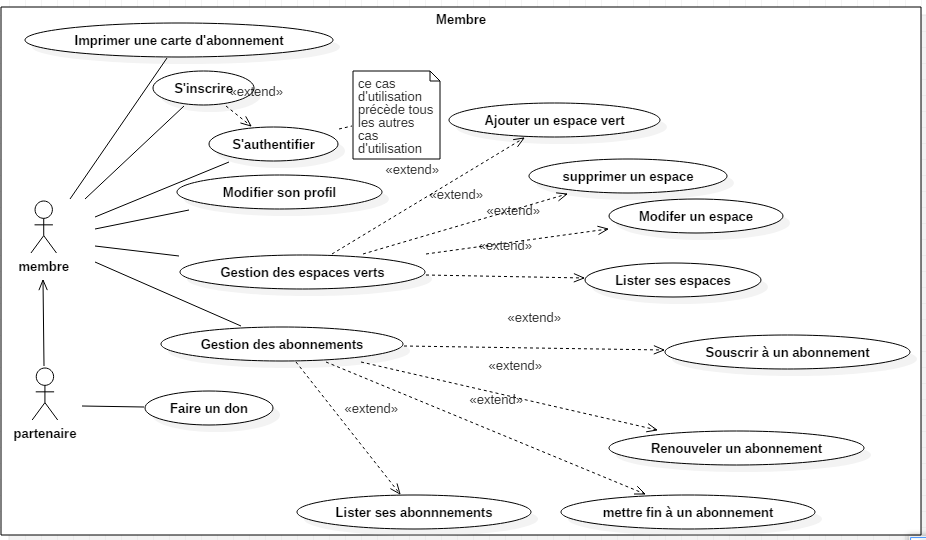


Figure : Cas d’utilisation du package administrateur

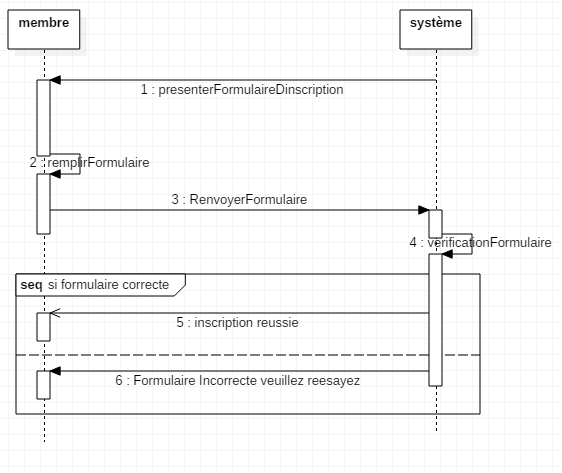
### **Diagrammes de Séquence:**

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met

l'accent sur la chronologie des envois de messages. En ce qui suit-on présentera quelques diagrammes de séquences relatifs aux cas d’utilisations présentées:

* Inscription

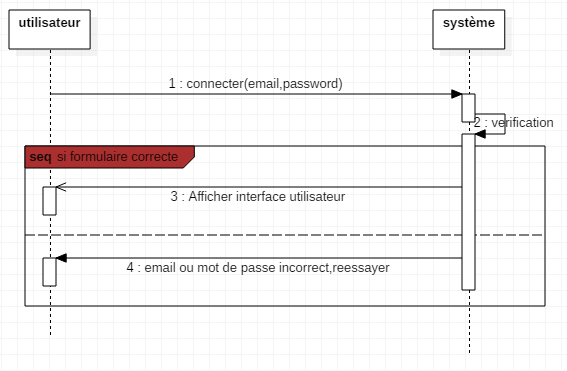
L’inscription permet à un utilisateur de créer son compte sur la plateforme de FORMA-GREEN



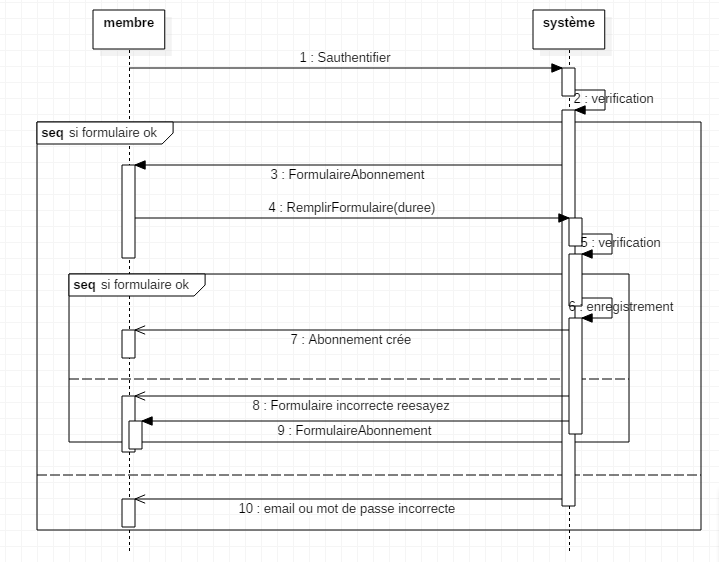
**Figure 9 : Diagramme de séquences (Inscription)**

* Authentification

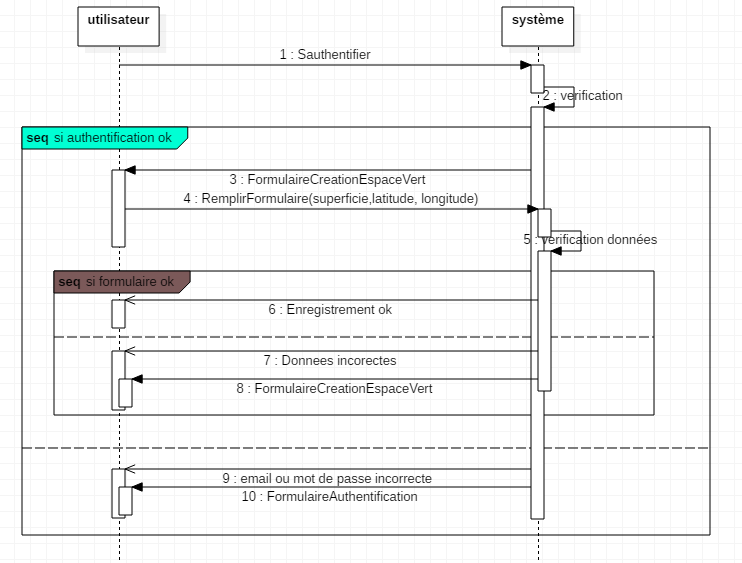
L’authentification permet à l’utilisateur d’accéder à l’application en saisissant le login(Email) et le mot de passe, si tous les champs sont bien remplis et les informations saisies sont correcte elle se redirige vers la page d’accueil, sinon il est redirigé vers la page d’authentification.



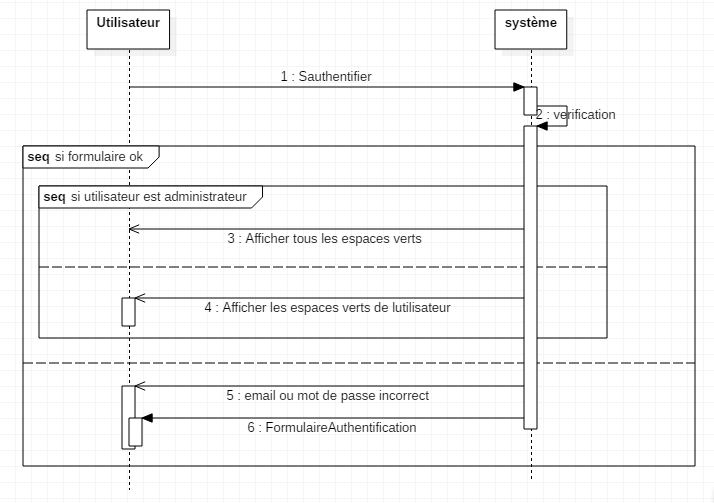
* S’abonner



* Créer un espace vert

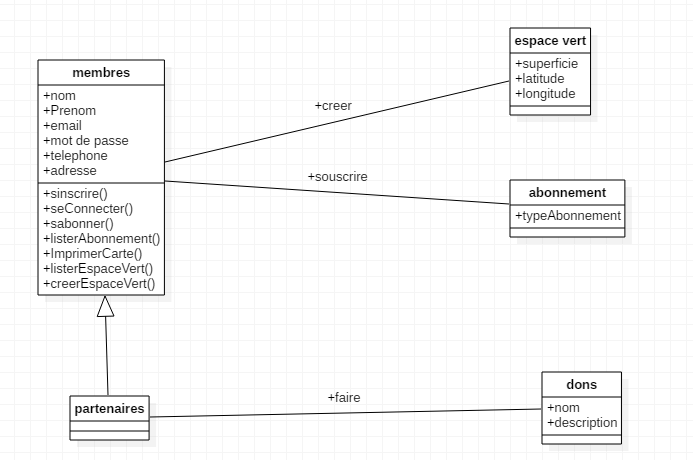


* Lister les espaces verts



### **Diagrammes de classe :**

**le diagramme de classes:** dans la phase d’analyse, ce diagramme représente les entités (des informations) manipulées par les utilisateurs. Dans la phase de conception, il représente la structure objet d’un développement orienté objet.



**Conclusion:**

L’application utilise un seul acteur principal qui est l’administrateur du service d’absence, ce dernier possède plusieurs tâches.

 L’utilisateur doit s’authentifier pour accéder a son espace de travail.

 La gestion des comptes d’utilisateurs est gérée par l’administrateur.

 Un étudiant peut n’avoir aucune absence, comme il peut avoir plusieurs.

 Les absences peuvent être modifiées.

 Les liste d’étudiants sont sauvegardées en archive par année universitaire.

# **Chapitre III**

# **PRESENTATION DE** L’APPLICATION

**Introduction:**

Dans ce chapitre on va parler des outils et des langages de développement, les étapes de la réalisation et nous allons montrer quelques captures d’écran pour avoir une idée sur l’application et son fonctionnement.

# **Outils de développement**

## **HTML 5**

L'HTML est un langage informatique utilisé sur l'internet. Ce langage est utilisé pour créer des pages web. L'acronyme signifie HyperText Markup Language, ce qui signifie en français "langage de balisage d'hypertexte". Cette signification porte bien son nom puisqu'effectivement ce langage permet de réaliser de l'hypertexte à base d'une structure de balisage.

## **CSS 3**

Le terme **CSS** est l'acronyme anglais de *Cascading Style Sheets* qui peut se traduire par "feuilles de style en cascade". Le CSS est un langage informatique

utilisé sur l'internet pour mettre en forme les fichiers

feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet

## **PHP**

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Exécuté du côté serveur (l'endroit où est hébergé le site) il n'y a pas besoin aux visiteurs d'avoir des logiciels ou plugins particulier. Néanmoins, les webmasters qui souhaitent développer un site en Python doivent s'assurer que l'hébergeur prend en compte ce langage.

## **JavaScript**

JavaScript (souvent abrégé JS) est un langage de programmation de scripts principalement utilisé dans les pages web interactives mais aussi côté serveur1. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés

## **PHPQrcode**

PhpQrcode es une librerie php permetant de de cree et dinamise les code QR

Elle es facile a utilise et a empliment

## **Ajax**

AJAX est l'acronyme d'*Asynchronous JavaScript and XML*, ce qui, transcrit en français, signifie « JavaScript et XML asynchrones ».

Derrière ce nom se cache un ensemble de technologies destinées à réaliser de rapides mises à jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web. Les technologies employées sont diverses et dépendent du type de requêtes que l'on souhaite utiliser, mais d'une manière générale le JavaScript est constamment présent.

## **MySQL**

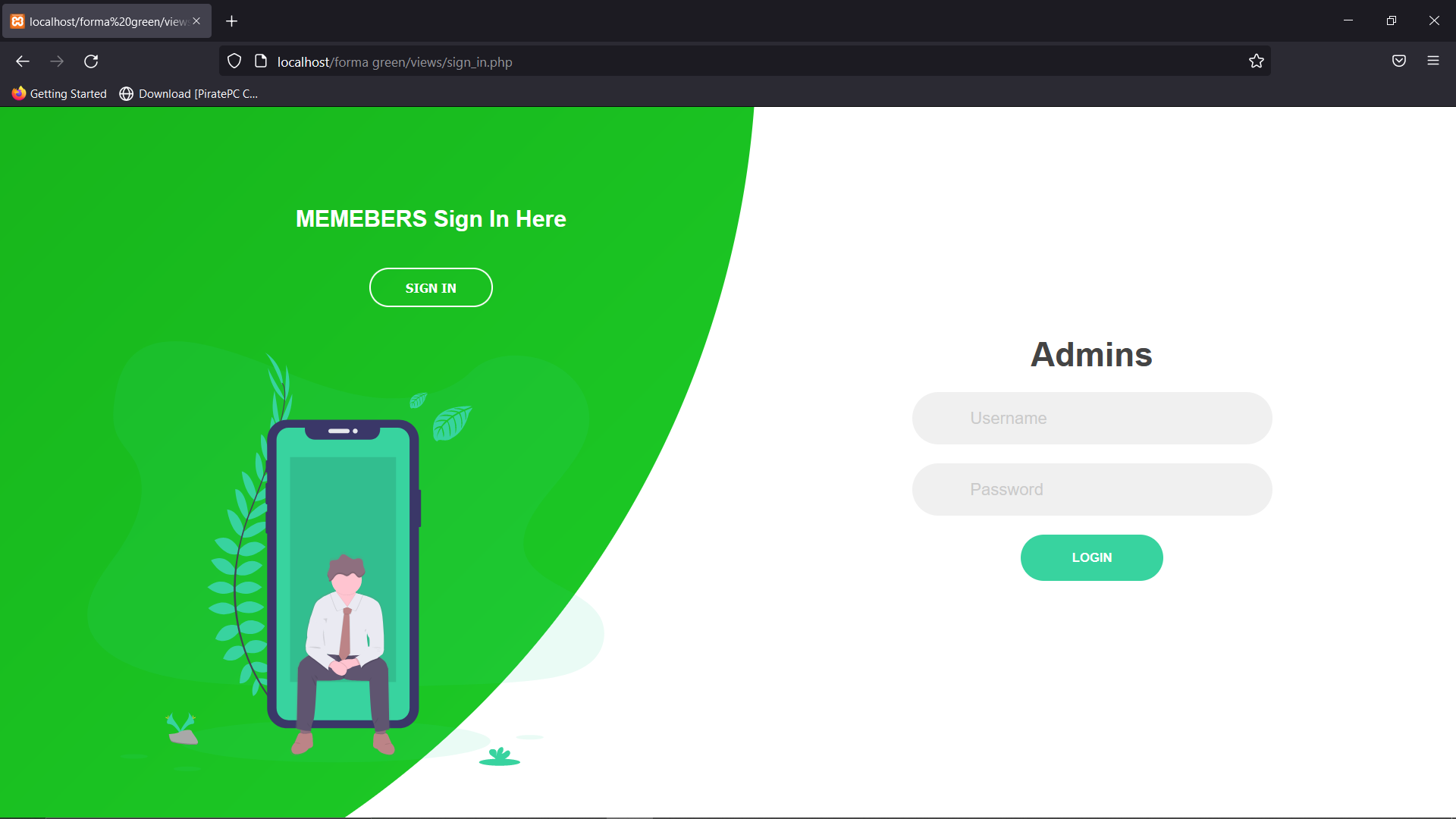
**MySQL** est un système de gestion de base de données relationnelle, puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes. MySQL fonctionne indifféremment sur tous les systèmes d'exploitation (Windows, Linux, Mac OS notamment).Le principe d'une base de données relationnelle est d'enregistrer les informations dans des tables, qui représentent des regroupements de données par sujets (table des clients, table des fournisseurs, table des produits, par exemple). Les tables sont reliées entre elles par des relations.

# **Présentation de** l’appli**cation**

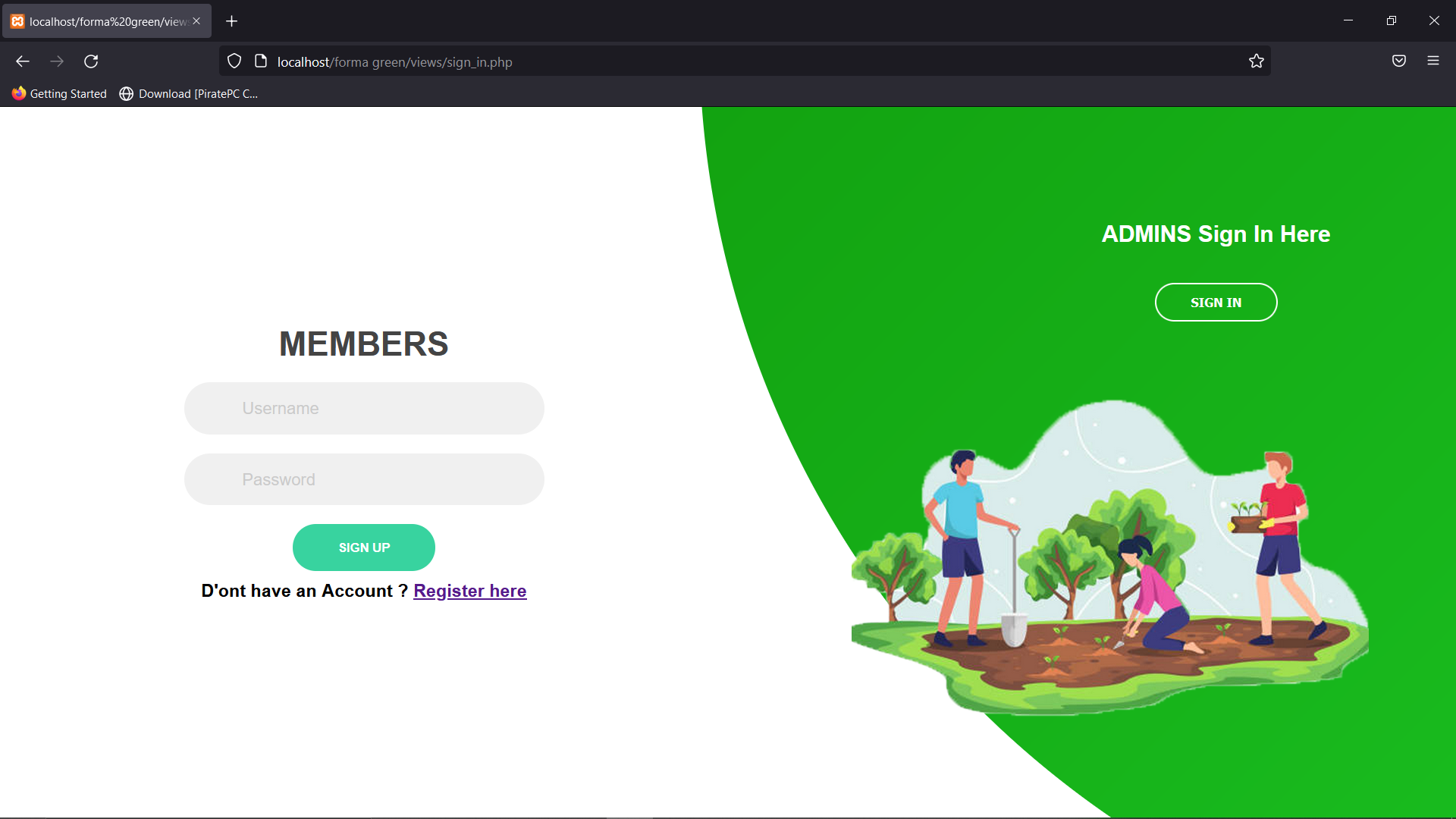
## **Présentation de la phase authentification :**

C’est la page qui permet l’authentification des utilisateurs. Si les coordonnés saisies sont correctes, alors l’utilisateur est dirigé vers son espace, sinon il est redirigé une autre fois vers la page d’authentification.

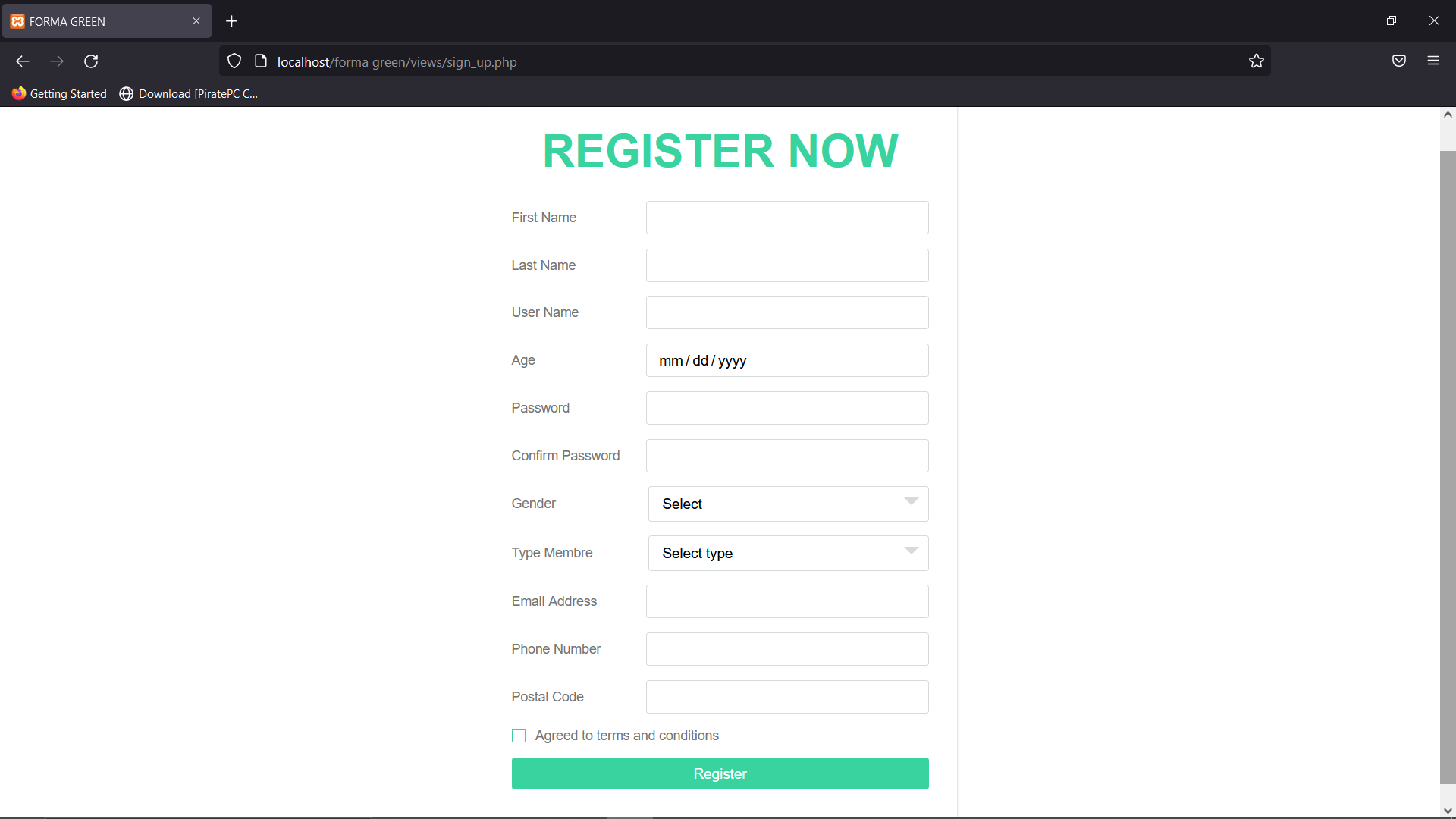
* Interface de connexion de l’administrature



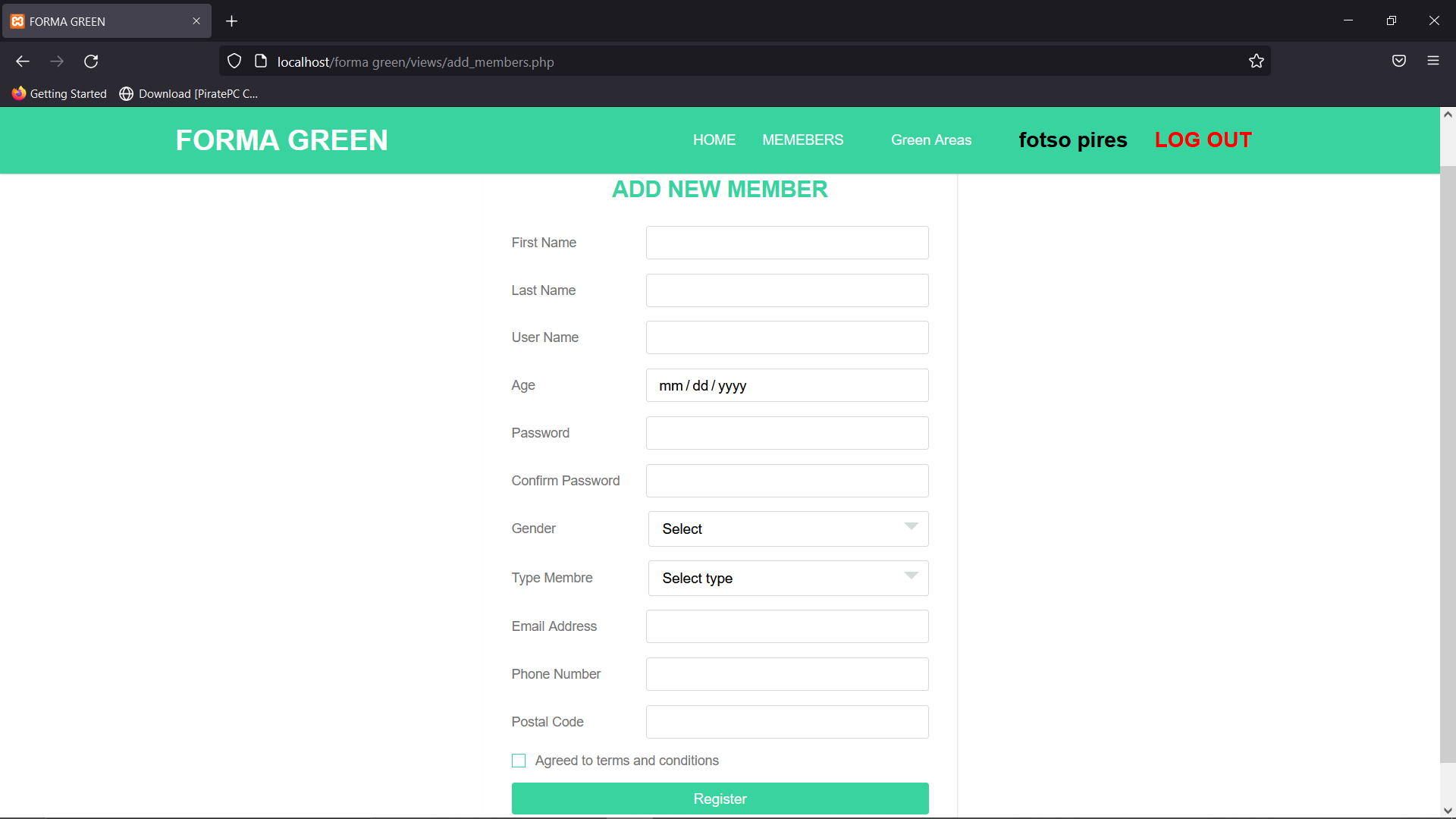
* Interface de connexion des membre



* Interface d’inscription d’un membre pas lui-même

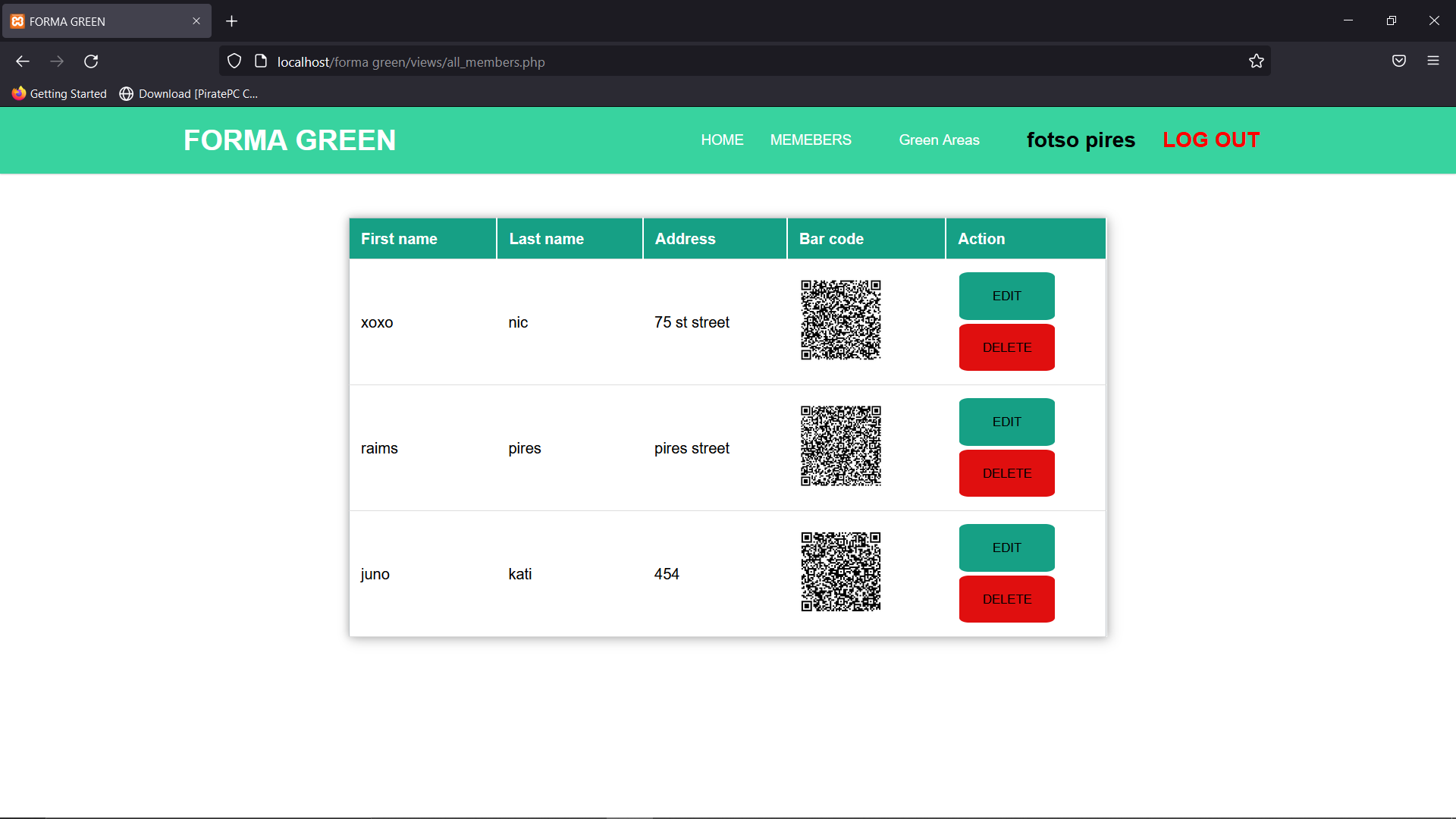


* Interface d’inscription d’un membre pas administrature

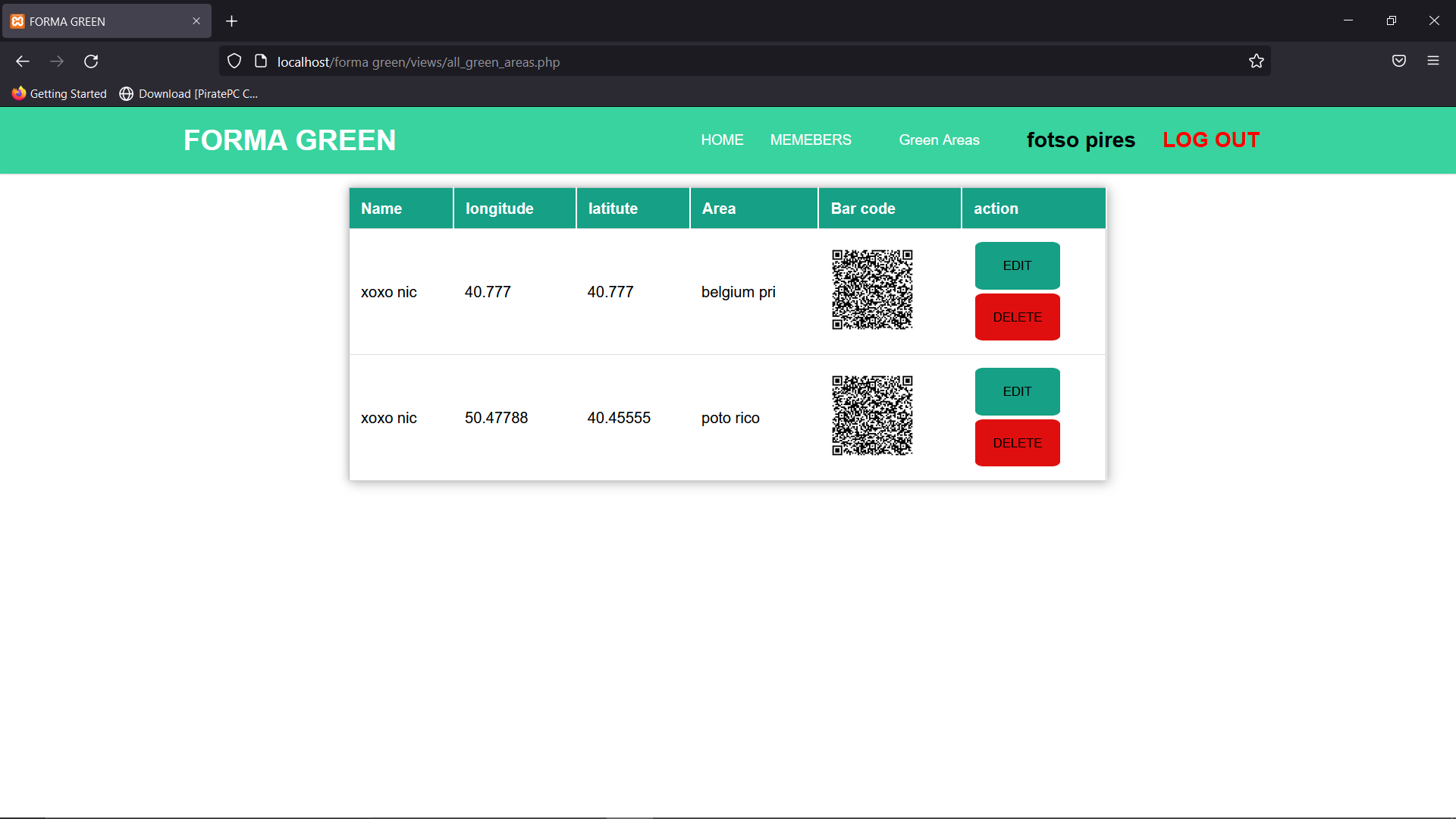


# 

* Interface liste de membre avec ou sans espace vert



* Interface liste des espace vert



# **Conclusion**

A travers ce chapitre, nous avons présenté la réalisation de l'application en justifiant nos choix technologiques, en représentant quelques interfaces graphiques que nous avons jugé les plus importantes.

# **Conclusion générale:**

L'objectif de notre projet de fin d'études était de concevoir et implémenter une application web de gestion des membres et des espaces verts d’une structure, FORMA-GREEN. Cette phase pratique de notre formation consacrée à l'analyse, à la conception et à la programmation nous a conduits à partir des problèmes constatés, à satisfaire les besoins réels en adéquation avec les objectifs prescrits et les  
contraintes rencontrées.  
j’ai appliqué au maximum les règles de base permettant d'avoir une application performante. J'ai utilisé le langage UML pour modéliser le système en suivant les différentes étapes, PostgreSQL comme SGBD et le langage Python pour implémenter notre application. Du fait que Python est un langage flexible, l'application sera portable et opérationnelle indépendamment de toute plateforme.  
La réalisation d'un tel projet, m'a permis d'apprendre et de toucher du doigt divers aspects du métier de développeur et de celui de concepteur. Nous n'avons pas la prétention d'avoir réalisé un travail parfait. C'est pourquoi, nous voudrions au-delà des imperfections et insuffisances que vous constaterez et rencontrerez, tenir compte de vos remarques, critiques, et suggestions afin de rendre possible laperfection de ce projet.

**Webographie:**

 http://fr.wikipedia.org

 https://openclassrooms.com/

 http://www.w3schools.com

 http://stackoverflow.com