SYLLA DEMBA 

**BUILDR**

Titre professionnel Developpeur-se logiciel Niveau III

Remerciements

Je tiens tout d’abord à remercier Sandrine Oriol et Alexandre Denurra de m’avoir permis de suivre cette formation, à Mickaël Noel ainsi que Johanne Badin pour leurs aide, d’avoir répondu a nos questions ainsi

Résumé

Buildr est une plateforme de mise en relation entre un particulier et un maître d’oeuvre pour des travaux de rénovation.

Mon projet a pour but de répondre aux besoins des clients qui ont du mal à trouver des maîtres d’oeuvres de confiance.

Via un questionnaire qui est disponible sur la plateforme, le client devra renseigner ses informations comme le type de bien, la superficie, le prix pour la réalisation des travaux etc. Par la suite, Buildrse chargera de rechercher un maître d’oeuvre disponible pour la réalisation des travaux souhaités.

Grace à Buidr, le client et le maître d’oeuvre pourront échanger des documents qui seront essentiels pour la réalisation des travaux et pourront garder une traçabilité des travaux effectués.

C’est un outil qui a vocation à faciliter la réalisation des travaux, pour les clients qui n’auront plus aucun soucis à se faire car les maîtres d’oeuvre qui font partie du réseau Buildr sont qualifiés et sérieux et aussi pour les entreprises car ça leur permettra de trouver régulièrement des contrats.

Un système de notation y est notamment disponible afin que les maitre d’oeuvre soient classés selon la qualité de leurs réalisations et la satisfaction de leurs clients.

Abstract

Buildr is a platform for connecting a client and a project manager for a house renovation work. My project aims to meet the needs of clients who have difficulty finding trusted project managers. Via a survey that is available on the platform, the client must provide information such as the type of property, area, price for the work, etc. Buildr will then be in charge of finding a project manager available to carry out the work. Through Buildr, the client and the project manager will be able to exchange documents that will be essential for the completion of the work and will be able to keep a traceability of the work performed.  It is a tool that aims to facilitate because the customers who will no longer have any worries to because the contractors who are part of the Buildr network are qualified and serious and also for the companies because it will allow them to regularly find contracts.

Sommaire

|  |  |
| --- | --- |
| Remerciements | 2 |
| Résumé | 3 |
| Abstract | 4 |
| Sommaire | 5 |
| Cahier des charges | 6 |
| Descriptif de la demande | 6 |
| Architecture technique | 6 |
| Spécifications fonctionelles | 6-7 |
| Spécifications technique | 8-9 |
| Compéténces référentiel | 10 |
| Réalisation | 11 |
| 1-1 Maquetter une application |  |
| 1-2 Concevoir une base de donnes |  |
| 1-3 Mettre en place une base de données |  |
| 1-4 Développer une interface utilisateur |  |
| 1-5 Développer des composants d’accès aux données |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Sylla Demba |  |

Cahier des charges

**Descriptif de la demande**

Pour la réalisation des travaux, l’utilisateur doit d’abord, après s’être inscrit réaliser un questionnaire afin de déterminer au mieux quel maitre d’oeuvre est disponible pour la réalisation de ses travaux

**Architecture technique**

Le projet sera développé au travers d’une application web et sera déployer sous LAMP et devra être accessible à tout les navigateur. Le développement sera réaliser en HTML/CSS, JavaScript et PHP ainsi qu’une structure MVC (Modèle, Vue, Controlleur) Les données seront stockées sur une base de données mySQL.

**Spécifications fonctionelles**

***Description***

***Utilisateur (clients) est defini par les points suivants :***

* Email
* Adresse
* Ville
* Nom
* Prenom
* Code Postal
* Numéro de téléphone
* Mot de Passe

***Projet est défini par les point suivants :***

* Metre carré
* Superficie
* Pièce
* Type de Propriété
* Le budget
* Description
* Ville

***Utilisation***

Création d’un compte utilisateur

Le site permet la création d’un compte et ainsi d’enregistrer de nouveaux utilisateurs dans la base de données, ainsi il pourra réaliser le questionnaire et avoir accès au service proposé par la plateforme.

Lecture d’un profil   
 Après que l’utilisateur se soit inscrit, il pourra directement avoir accès à ses informations dans sa page profil.

Modification du profil

L’utilisateur sera libre de modifier ses informations personnelles depuis sa page profil.

Suppression du profil

L’utilisateur pourra s’il le souhaite supprimer son profil.

Création d’un projet   
 Une fois que l’utilisateur sera enregistré dans la base de données, il pourra réaliser le questionnaire et va pouvoir effectuer les démarches nécessaires pour la réalisation de ses travaux

Lecture du projet   
 Une fois le questionnaire rempli, il pourra ainsi voir le récapitulatif de ses réponses et elles seront en même temps soumis au maître d’œuvre.

Spécification techniques

Les langages

|  |  |
| --- | --- |
| Appelation | Description |
| **HTML5** | L’HyperText Markup Language est le format de données conçu pour représenter les pages web. C’est un languages de balisage permettant d’écrure de l’hypertexte |
| **CSS3** | Les feuilles de style en cascade, généralement appelés Cascading StyleSheets, forment un langage informatique qui stylise des documents HTML XML |
| **PHP** | PHP (HyperText Preprocessor) est un langage de scripts généraliste et OpenSource, spécialement conçu pour le développement d’applications web dynamique. Il s’intégre facilement avec de l’HTML |
| **JavaScript** | JavaScript est un langage événementiel de programmation de scripts, principalement employés sur les pages interactives. |
| **MySQL** | Langage de requête fonctionnant avec le système de gestion de bases de données relationnelles, Permet de créer une base données, la lectures et la modifications des tables |

Les bibliothèques et frameworks

|  |  |
| --- | --- |
| JQuery | C’est une bibliothèque de JavaScript qui en simplifie grandement la syntaxes ainsi que l’utilisation |
| Bootstrap | Collection d’outils qui facilite grandement de rendre responsive tout simplement une application grâce à son système de grilles et de colonnes. |

Les outils

|  |  |
| --- | --- |
| Netbeans | Environnement de développement intégré qui prend en charge de nombreux langages dont HTML/CSS, C++, PHP, JavaScript etc |
| Git | Logiciel de versionning fonctionnant avec le service d’hébergement gratuit Github |
| Pencil | Logiciel de conception de maquette |
| PHPMyAdmin | Application Web de gestion de gestion pour les système de gestion de base de données MySQL réalisée en PHP. Permet d’avoir une gestion graphique pour pouvoir manipuler les données. |
| JMerise | Logiciel d’aide à la conception des bases de données(Modèle Conceptuel de Données) qui permet la génération d’un Modèle Logique de Données et d’un script SQL. |

Compéténces référentiel

|  |  |  |
| --- | --- | --- |
|  | Activités types | Compétences professionnel |
| 1 | Développer une application client serveur | |  |  | | --- | --- | | 1 | Maquetter une application | | 2 | Concevoir une base de données | | 3 | Mettre en place une base de données | | 4 | Développer une interface utilisateur | | 5 | Développer des composants d’accès aux données | |
| 2 | Développer une application web | |  |  | | --- | --- | | 6 | Développer des pages web en lien avec une base de données | | 7 | Mettre en oeuvre une solution de gestion de contenu en e-commerce | | 8 | Développer une application simple en mobilité numérique | | 9 | Utiliser un anglais professionnelle en informatique | |

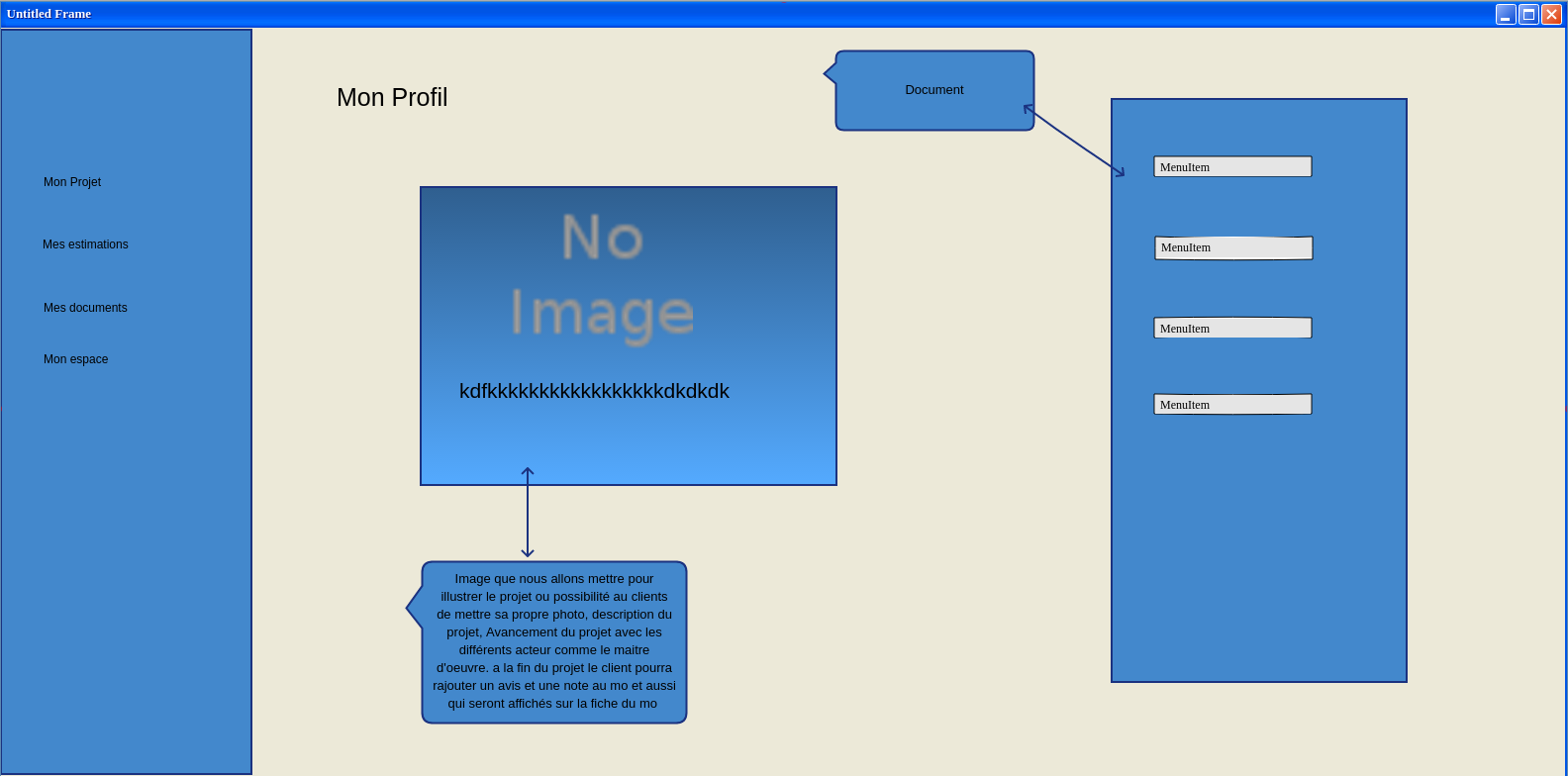
Réalisation

1.1 Maquetter une application

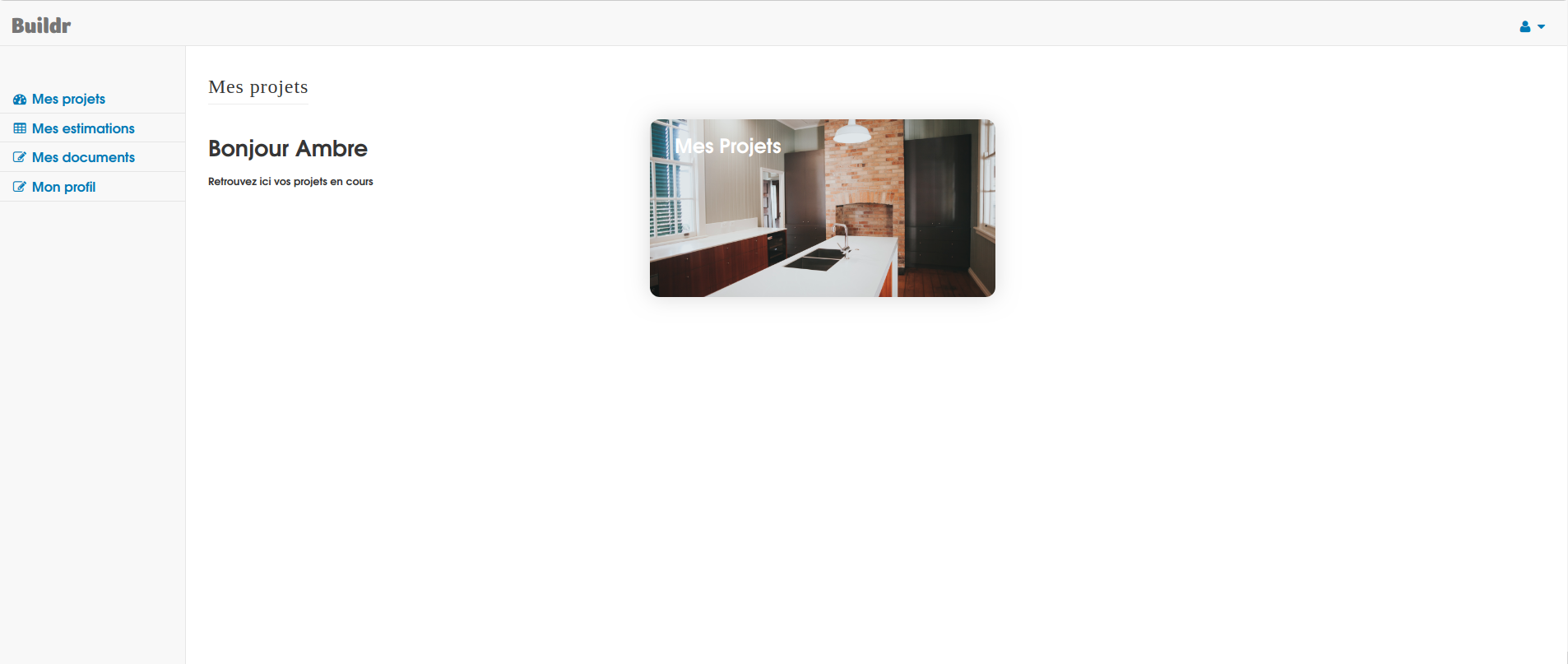
Une fois le cahier des charges défini, j’ai commencé mon projet en réalisant une maquette sur Pencil.  
Il est important de créer une maquette de son projet afin d'avoir une vue globale et tangible de l'application que l'on souhaite développer. La maquette permet ainsi d'anticiper certaines problématiques notamment en terme de design.

Pour facilité le parcours client, je me suis donc décidé a réalisé un dashboard.   
  
Pour le maquettage j'ai utilisé le logiciel Pencil. J'ai maquetté les principales pages du projet tel que l'accueil ou le dashboard.  
  
C’est avec le logiciel Pencil que j’entreprends le maquettage des interfaces utilisateurs et que j'imagine leurs interactions. Cela m’a permis d’avoir une réelle approche sur l’ergonomie ainsi que du parcours utilisateur.

***Version maquétté sous pencil***

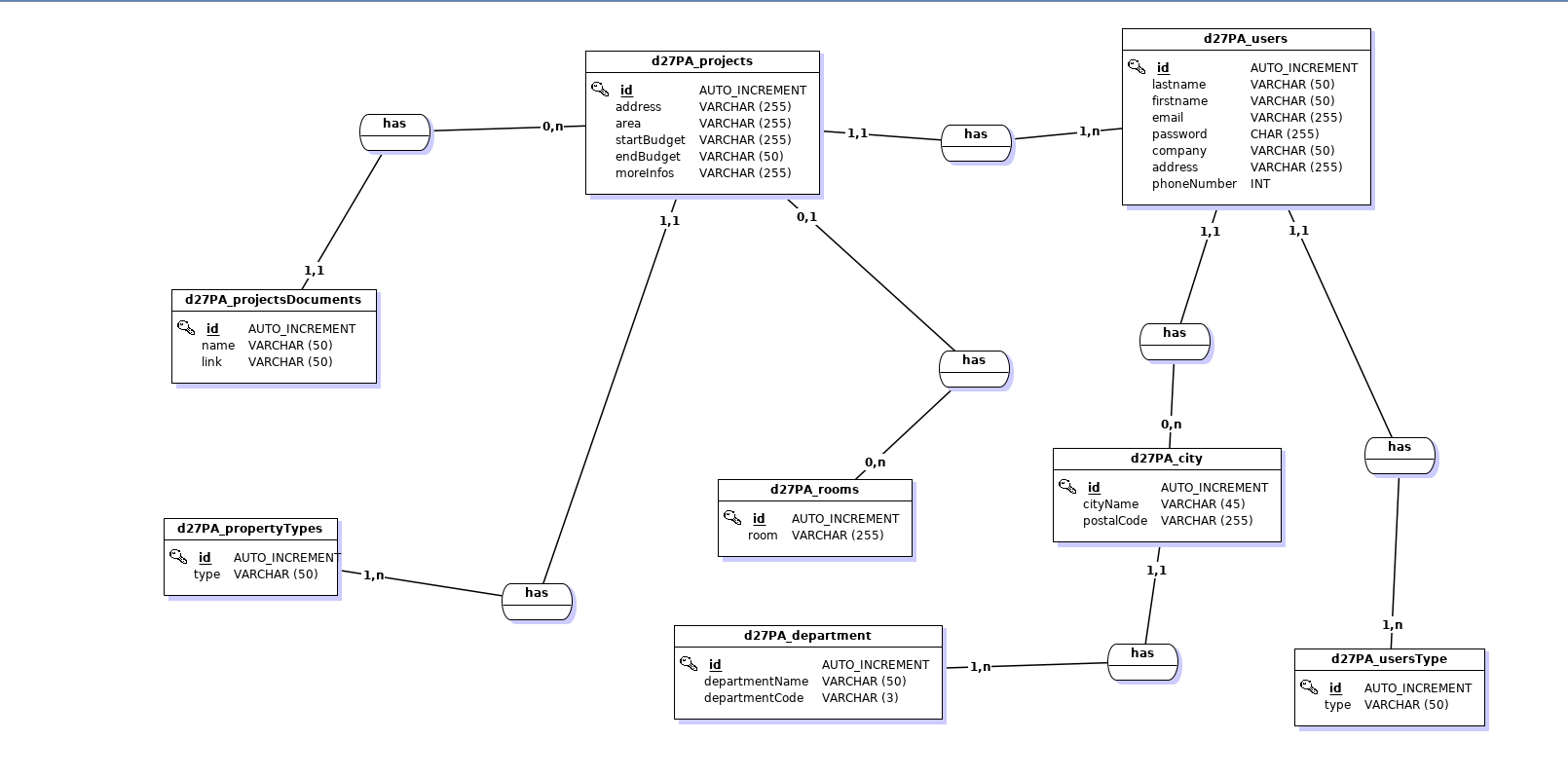


*Voici la version définitif du dashboard*

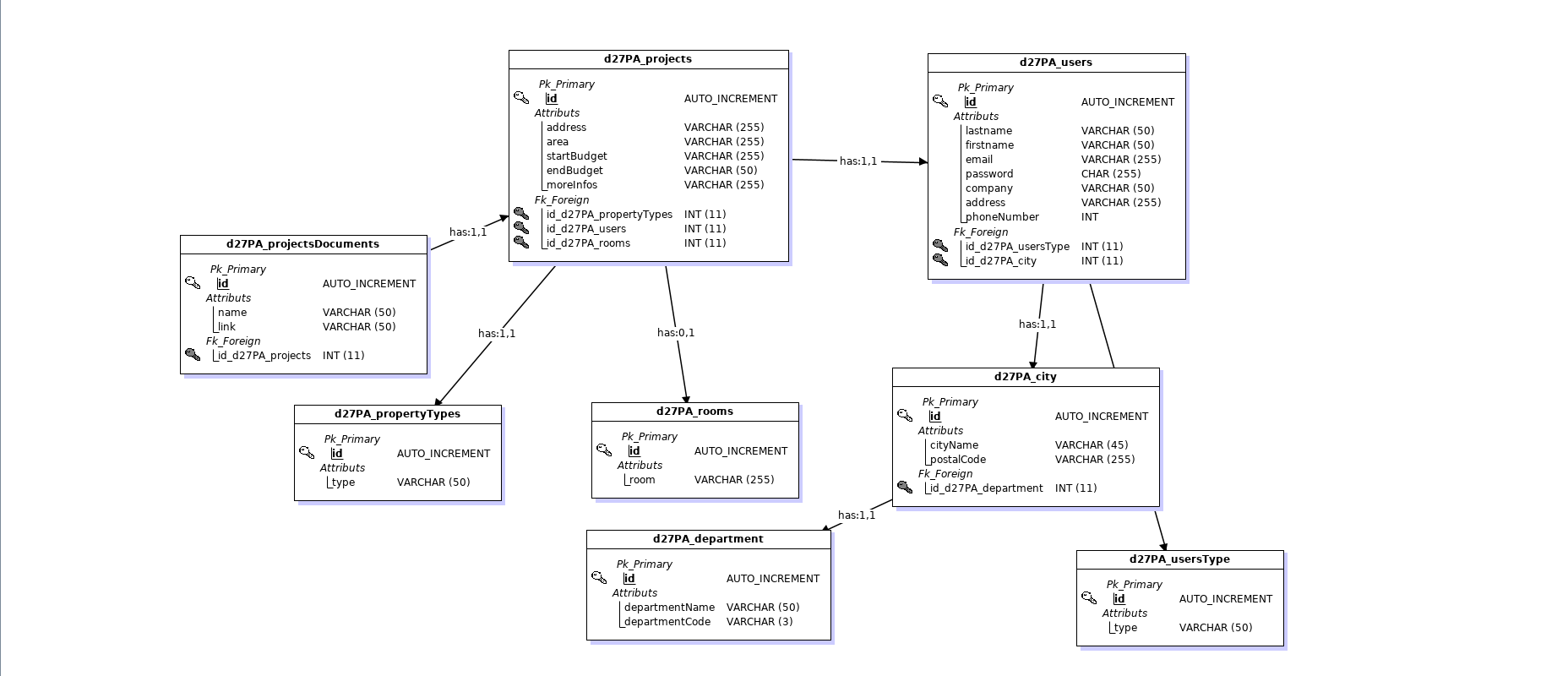


1.2 Concevoir une base de donnes

Suite à cela, j’ai ensuite réalisé la base de données, c’est-à-dire ajouter les éléments et informations que les utilisateurs entreront, ainsi que déterminé quels liens ils auront.  
  
J’ai utilisé JMerise pour la conception de la base de données, car c’est celui la qui me permettra ensuite d’obtenir mon MCD (Modèle Conceptuel de Données).



C’est à ce moment là que je génère le MLD (Modèle Logique des Données) qui réfléchit aux relations entre les tables via des clés primaires et étrangère à l’aide des cardinalités. Les cardinalités définissant les relations entre les tables.



1.3 Mettre en place une base de donnes

Et c’est à partir de ce moment-là que je vais pouvoir générer ma base de données grâce à une fonction de JMerise qui me générera automatiquement mon Script SQL. J’ai dû modifier quelques éléments pour apporter plus de sécurité en y ajoutant des magiques côtes.

#------------------------------------------------------------#

Script MySQL.

#------------------------------------------------------------

#------------------------------------------------------------#

Table: d27PA\_usersType

#------------------------------------------------------------

CREATE TABLE d27PA\_usersType(

id Int Auto\_increment NOT NULL ,

type Varchar (50) NOT NULL

,CONSTRAINT d27PA\_usersType\_PK PRIMARY KEY (id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_propertyTypes

#------------------------------------------------------------#

CREATE TABLE d27PA\_propertyTypes(

id Int Auto\_increment NOT NULL ,

type Varchar (50) NOT NULL

,CONSTRAINT d27PA\_propertyTypes\_PK PRIMARY KEY (id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_department

#------------------------------------------------------------#

CREATE TABLE d27PA\_department(

id Int Auto\_increment NOT NULL ,

departmentName Varchar (50) NOT NULL ,

departmentCode Varchar (3) NOT NULL

,CONSTRAINT d27PA\_department\_PK PRIMARY KEY (id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_city

#------------------------------------------------------------#

CREATE TABLE d27PA\_city(

id Int Auto\_increment NOT NULL ,

cityName Varchar (45) NOT NULL ,

postalCode Varchar (255) NOT NULL ,

id\_d27PA\_department Int NOT NULL

,CONSTRAINT d27PA\_city\_PK PRIMARY KEY (id)

,CONSTRAINT d27PA\_city\_d27PA\_department\_FK FOREIGN KEY (id\_d27PA\_department) REFERENCES d27PA\_department(id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_users

#------------------------------------------------------------

CREATE TABLE d27PA\_users(

id Int Auto\_increment NOT NULL ,

lastname Varchar (50) NOT NULL ,

firstname Varchar (50) NOT NULL ,

email Varchar (255) NOT NULL ,

password Char (255) NOT NULL ,

company Varchar (50) ,

address Varchar (255) NOT NULL ,

phoneNumber Int NOT NULL ,

id\_d27PA\_usersType Int NOT NULL ,

id\_d27PA\_city Int NOT NULL

,CONSTRAINT d27PA\_users\_PK PRIMARY KEY (id)

,CONSTRAINT d27PA\_users\_d27PA\_usersType\_FK FOREIGN KEY (id\_d27PA\_usersType) REFERENCES d27PA\_usersType(id)

,CONSTRAINT d27PA\_users\_d27PA\_city0\_FK FOREIGN KEY (id\_d27PA\_city) REFERENCES d27PA\_city(id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_rooms

#------------------------------------------------------------

CREATE TABLE d27PA\_rooms(

id Int Auto\_increment NOT NULL ,

room Varchar (255) NOT NULL

,CONSTRAINT d27PA\_rooms\_PK PRIMARY KEY (id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_projects

#------------------------------------------------------------

CREATE TABLE d27PA\_projects(

id Int Auto\_increment NOT NULL ,

address Varchar (255) NOT NULL ,

area Varchar (255) ,

startBudget Varchar (255) ,

endBudget Varchar (50) NOT NULL ,

moreInfos Varchar (255) NOT NULL ,

id\_d27PA\_propertyTypes Int NOT NULL ,

id\_d27PA\_users Int NOT NULL ,

id\_d27PA\_rooms Int

,CONSTRAINT d27PA\_projects\_PK PRIMARY KEY (id)

,CONSTRAINT d27PA\_projects\_d27PA\_propertyTypes\_FK FOREIGN KEY (id\_d27PA\_propertyTypes) REFERENCES d27PA\_propertyTypes(id)

,CONSTRAINT d27PA\_projects\_d27PA\_users0\_FK FOREIGN KEY (id\_d27PA\_users) REFERENCES d27PA\_users(id)

,CONSTRAINT d27PA\_projects\_d27PA\_rooms1\_FK FOREIGN KEY (id\_d27PA\_rooms) REFERENCES d27PA\_rooms(id)

)ENGINE=InnoDB;

#------------------------------------------------------------#

Table: d27PA\_projectsDocuments

#------------------------------------------------------------#

CREATE TABLE d27PA\_projectsDocuments(

id Int Auto\_increment NOT NULL ,

name Varchar (50) NOT NULL ,

link Varchar (50) NOT NULL ,

id\_d27PA\_projects Int NOT NULL

,CONSTRAINT d27PA\_projectsDocuments\_PK PRIMARY KEY (id)

,CONSTRAINT d27PA\_projectsDocuments\_d27PA\_projects\_FK FOREIGN KEY (id\_d27PA\_projects) REFERENCES d27PA\_projects(id)

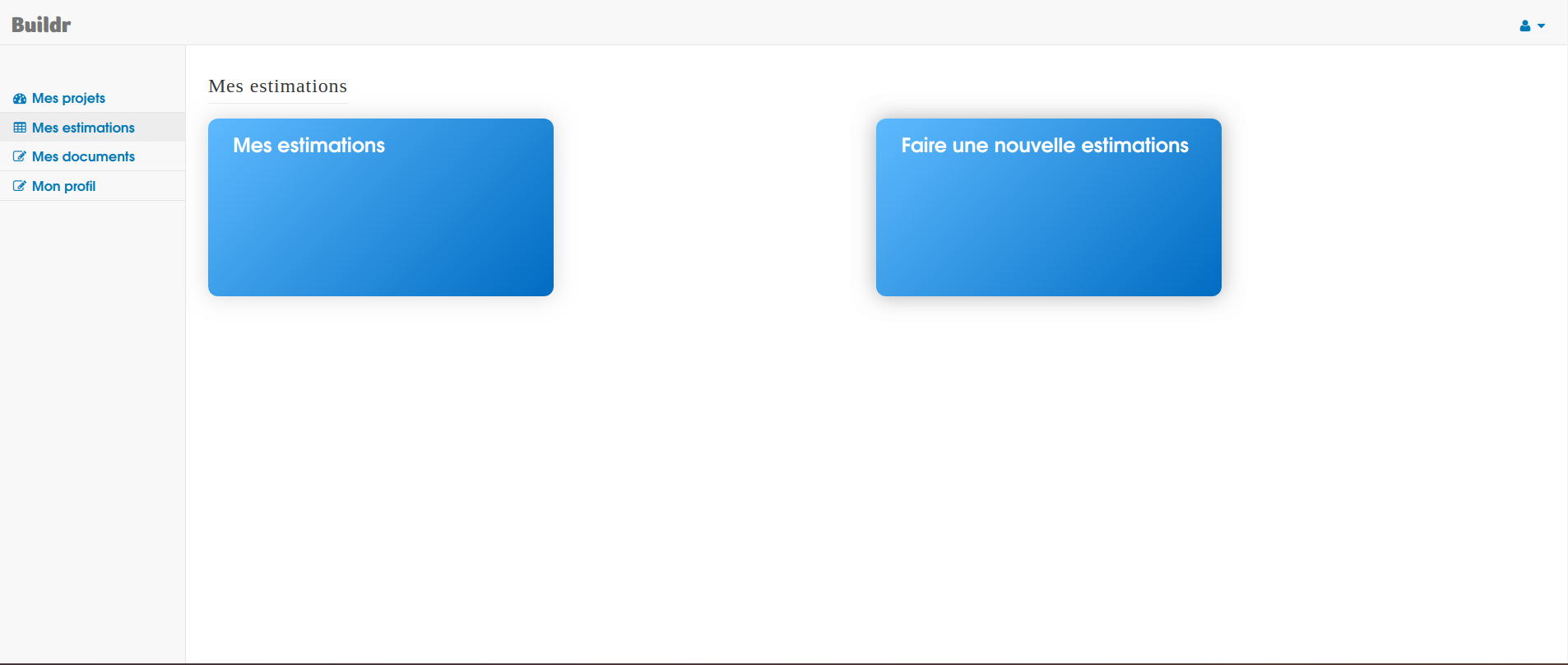
)ENGINE=InnoDB;

 Après correction et compléments, j’utilise phpMyAdmin pour l’importer le serveur mysql et créer ma base de données buildr.



1.4 Développer une interface utilisateur

Ayant choisi l’architecture MVC (Modèle Vue Controller), une interface était indispensable pour afficher les données provenant de mes différents modèles et de mes contrôleurs. Le MVC, permet de faciliter la Programmation Orienté Objet (POO).  
J’ai donc créer un dashboard ou les utilisateurs pourront accéder à leurs récapitulatifs et pourront répondre au questionnaire qui sera ensuite communiqué au maître d’œuvre.



J’ai crée une classe spécifique regroupant les éléments nécessaire a la connexion à la base de données en créant la class database.

\* Classe permettant de se connecter à la base de données MYSQL

\*/**class database** {

// on crée les attribut qui $db, $host, $login, $password, $dbname pour voir seront donc disponible pour celle qui sont private et

**protected** $db;

**private** $host = '';

// attribut correspondant à mon login

**private** $login = '';

// attribut correspondant à mon mot de passe

**private** $password = '';

// attribut correspondant à buildr

**private** $dbname = '';

/\*\*

\* On crée la méthode magique construct permettant de créer l'instance PDO

\*/

**public** **function** \_\_construct() {

**$this**->host = HOST;

**$this**->login = LOGIN;

**$this**->password = PASSWORD;

**$this**->dbname = DBNAME;

}

/\*\*

\* Méthode dbConnect permettant de créer l'instance PDO

\*/

**protected** **function** dbConnect() {

// via le try, on essaye de se connecter en instanciant un nouvel objet PDO

**try** {

**$this**->db = **new** PDO('mysql:host=' . **$this**->host . ';port=3306;dbname=' . **$this**->dbname . ';charset=UTF8;', **$this**->login, **$this**->password);

// si une erreur se produit, on "attrape" l'exception via un catch et on le stock dans une variable $errorMessage puis on arrête le script PHP via un die

} **catch** (**Exception** $errorMessage) {

$errorMessage->getMessage();

}

}

}

1.4 Développer des pages web en lien avec une base de données

Vue : On inclue les Modele et Controlleur via un include\_once

**<?php**

// Création d’une nouvelle session

session\_start();

// Ajout des fichiers nécessaires au fonctionnement du site

**include\_once** '../lang/FR\_FR.php';

**include\_once** '../class/database.php';

**include\_once** '../controllers/loginCtl.php';

**include\_once** '../controllers/profileCtl.php';

**include\_once** '../controllers/createProjectsCtl.php';**?>**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="description" content="">

<meta name="author" content="">

<title>Buildr | Tableau de bord</title>

<!-- Bootstrap Core CSS -->

<link href="../vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->

<link href="../assets/css/style.css" rel="stylesheet">

<!-- Custom Fonts -->

<link href="../vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="wrapper">

<!-- Navigation -->

Controlleur : J’instancie un nouvel objet à l’aide du Modèle respectif et j’appelle les méthode. Dans cette exemple j’appelle la méthode

//appel de l 'ajax// On définit donc que si le PostalCodeSearch est définie

**if** (**isset**($\_POST['postalCodeSearch'])) {

// on va include le fichier configuration qui contient toute nos informations pour pouvoir nous connecter à la base données

**include\_once** '../configuration.php';

// On instancie alors un nouvel objet $city comme la classe city

$city = **new** city();

$city->postalCode = $\_POST['postalCodeSearch'];

// on va donc affiché le résultat de la requếte et ainsi aussi éxécuté la requête getCityPostalCode

**echo** json\_encode($city->getCityByPostalCode());

}

Modèle : Connexion à la base de données et création de la méthode qui va me permettre d’effectuer une requête dans la table city.

/\*\*

\* Création de la classe clients

\*/**class city extends database** {

// Liste des attributs que l'on va utilisé dans la méthode

**public** $id;

**public** $ville\_departement;

**public** $postalCode;

**public** $cityName;

/\*\*

\* Méthode construct

\*/

**public** **function** \_\_construct() {

**parent**::\_\_construct();

**$this**->dbConnect();

}

/\*\*

\* Méthode getCityByPostalCode permettant de récupérer le code postal ainsi que la ville

\* @return type

\*/

**public** **function** getCityByPostalCode() {

// Définition d'une variable ou l'on va donc récupérer un tableau qui contiendra notre résultat

$queryResult = **array**();

// on prépare la requête query qui contient le marqueur nominatif :postalCode pour l'attribut qui se trouve dans la table city

// requête SQL ou l'a va séléctionner les élements que l'on souhaite récupérer

$query = 'SELECT `id`,`cityName`, `postalCode` FROM `d27PJ\_city` '

. 'WHERE `postalCode` LIKE :postalCode';

// $result est la requête que l'on prépare via un prepare() et un bindValue.

$result = **$this**->db->prepare($query);

$result->bindValue(':postalCode', **$this**->postalCode . '%', PDO::PARAM\_STR);

// On teste si la requête s'éxécute correctement, si oui on crée $queryResult qui est un objet tableaux avec les résultats de la requête

**if** ($result->execute()) {

// on réalise un fetchAll car il retournera un tableau

$queryResult = $result->fetchAll(PDO::FETCH\_OBJ);

} **else** {

$queryResult = **false**;

}

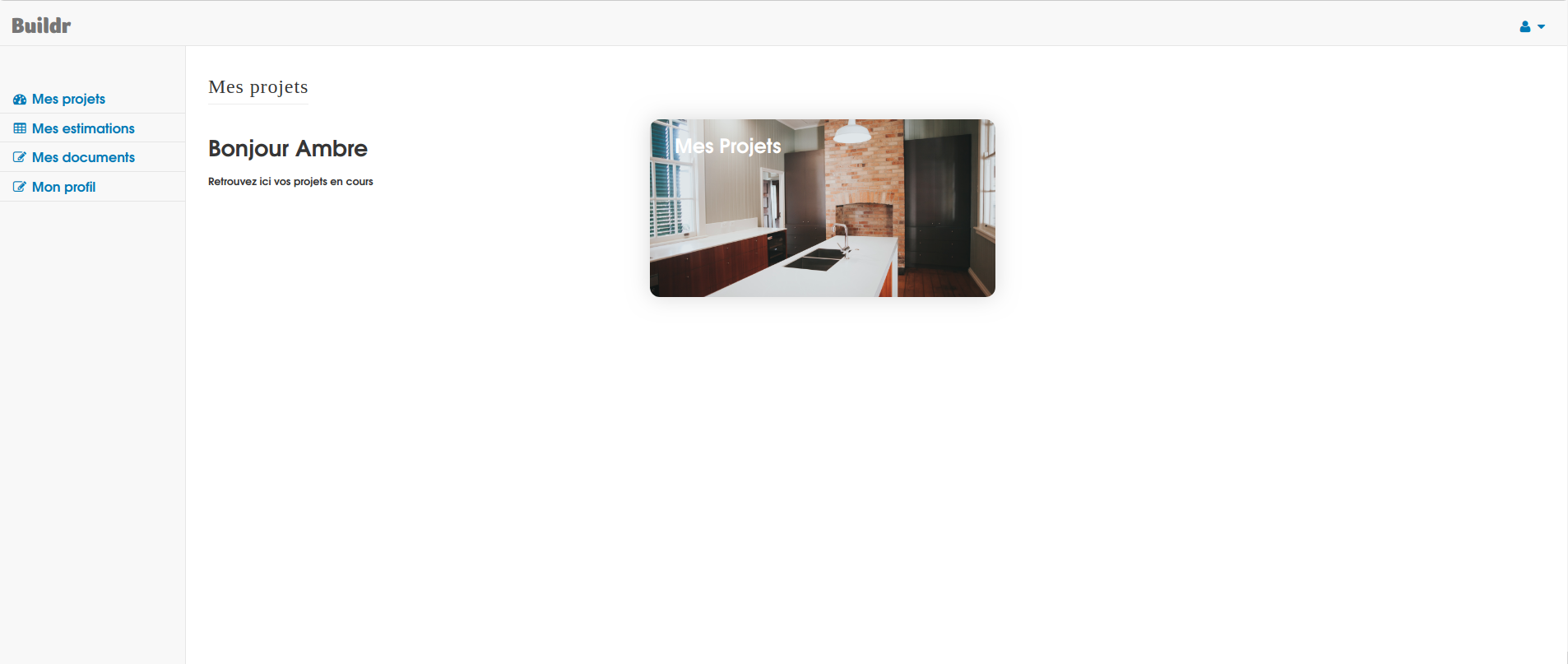
// on retourne le résultat

**return** $queryResult;

}

1.4 Développer une application simple de mobilité numérique

Pour gagner en mobilité, j’ai fait en sorte que mon application soit responsive en m’appuyant sur le framework Bootstrap. Il m’a permis d’utiliser un système de grid pour une meilleure structure ainsi que des classes pour rendre mon application optimale pour différents types d’appareils.





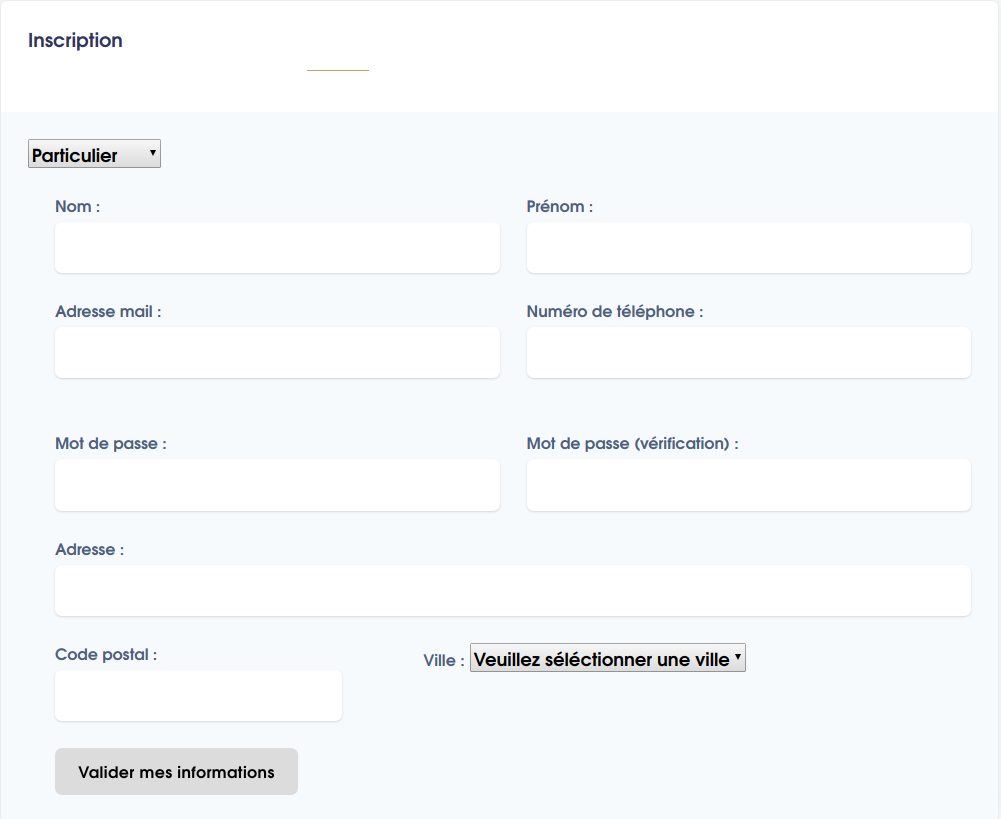
1.4 Utiliser l’anglais dans son activité professionnelle en informatique

Tout au long de mon projet et aussi de ma formation, je me suis efforcé à utiliser au jour le jour l’anglais dans le nommage de mes dossiers, fichiers, quelques commentaires et mes recherches lorsque je fus bloqué.

Réalisation

CREATE (Enregistrement d’un utilisateur pour pouvoir avoir accées au questionnaire)

Vue de la page d’inscription pour pouvoir avoir accés à la plateforme



Controller de la page d’inscription

Création d’un controller pour effectuer l’enregistrement des inputs en effectuant des contrôle de condition ainsi que des regex.

/ Regex des différents inputs

$regexPhoneNumber = '/^[0][1-9][0-9]{8}$/';

$regexzipCode = '/^[0-9]{5}$/';

$regexName = '/^[a-zA-Zàáâãäåçèéêëìíîïðòóôõöùúûüýÿ\-]+$/';

$regexNumberLetter = '/^[0-9-a-zA-Zàáâãäåçèéêëìíîïðòóôõöùúûüýÿ\- ]+$/';

$errorList = **array**();//Condition pour le formulaire de création de projet

**if** (**isset**($\_POST['submitRegister'])) {

// On test si l'input n'est pas vide et

**if** (!**empty**($\_POST['lastname'])) {

// On vérifie avec le preg match si la valeur respecte bien les valeurs que l'ont attends si c'est le cas

**if** (preg\_match($regexName, $\_POST['lastname'])) {

// Insérer dans l'attribut $lastname la valeur de l'input

$lastname = htmlspecialchars($\_POST['lastname']);

} **else** {

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

$errorList['lastname'] = 'La saisie de votre nom est invalide';

}

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

} **else** {

$errorList['lastname'] = 'Veuillez indiquer votre nom';

}

// On test si l'input n'est pas vide et

**if** (!**empty**($\_POST['firstname'])) {

// On vérifie avec le preg match si la valeur respecte bien les valeurs que l'ont attends si c'est le cas

**if** (preg\_match($regexName, $\_POST['firstname'])) {

// Insérer dans l'attribut $firstname la valeur de l'input

$firstname = htmlspecialchars($\_POST['firstname']);

// Sinon

} **else** {

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

$errorList['firstname'] = 'La saisie de votre prénom est invalide';

}

// Sinon

} **else** {

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

$errorList['firstname'] = 'Veuillez indiquer votre prénom';

}

// On instancie une variable user comme la class user

$user->usersRegister();

}

Modele de la page d’inscription

Dans le modèle de la table user, je crée la méthode qui va permettre d’insérer un utilisateur dans la base de données une fois toutes les informations validées.

*/\*\**

*\* Création de la classe users*

*\*/***class** **users** **extends** **database**{

*//Liste des attributs*

**public** $id = '';

**public** $email = '';

**public** $lastname = '';

**public** $firstname = '';

**public** $password = '';

**public** $firm = '';

**public** $address = '';

**public** $postalCode = '';

**public** $phoneNumber = '';

**public** $idUsersType = '';

**public** $idCity = '';

*/\*\**

*\* Méthode construct*

*\*/*

**public** **function** **\_\_construct**()

{

**parent**::\_\_construct();

**$this**->dbConnect();

}

*/\*\**

*\* Méthode userRegister permettant l'enregistement d'un utilisateur*

*\* @return type*

*\*/*

**public** **function** **usersRegister**() {

$query = 'INSERT INTO `d27PG\_users` (`lastname`, `firstname`, `email`, `password`, `firm`, `address`, `phoneNumber`, `idUsersType` , `idCity`) '

. 'VALUES (:lastname, :firstname, :email, :password, :firm , :address, :phoneNumber, :usersType, :city)';

*// Etant donné que les données vont être entrées par l'utilisateur on fait un prepare puis un bindValue avec marqueur nominatif et on finit par un execute*

$result = **$this**->db->prepare($query);

$result->bindValue(':lastname', **$this**->lastname, PDO::PARAM\_STR);

$result->bindValue(':firstname', **$this**->firstname, PDO::PARAM\_STR);

$result->bindValue(':email', **$this**->email, PDO::PARAM\_STR);

$result->bindValue(':password', **$this**->password, PDO::PARAM\_STR);

$result->bindValue(':firm', **$this**->firm, PDO::PARAM\_STR);

$result->bindValue(':address', **$this**->address, PDO::PARAM\_STR);

$result->bindValue(':usersType', **$this**->idUsersType, PDO::PARAM\_INT);

$result->bindValue(':phoneNumber', **$this**->phoneNumber, PDO::PARAM\_STR);

$result->bindValue(':city', **$this**->idCity, PDO::PARAM\_INT);

**return** $result->execute();

}

READ (Affichage des informations d’un utilisateur )

Vue des informations du profil

<form action="#" method="POST">

<div **class**="**pl**-**lg**-4">

<**div** **class**="**row**">

<d**iv** **class**="**col**-**lg**-6">

<**div** **class**="**form**-**group** **has**-**error**">

<**label** **class**="**form**-**control**-**label**" **for**="**lastname**"><?= **REGISTER\_LASTNAME** ?></**label**>

<**input** **type**="**text**" **name**="**lastname**" **id**="**lastname**" **class**="**form**-**control** **form**-**control**-**alternative** **deactivation**" **value**="<?= $**userProfile**->**lastname** ?>">

<**p** **class**="**text**-**danger**"><?= **isset**($**errorList**['**lastname**']) ? $**errorList**['**lastname**'] : ''; ?></**p**>

</**div**>

</**div**>

Controleur

**if** (**isset**($\_POST['modify'])) {

// On test si l'input n'est pas vide et

**if** (!**empty**($\_POST['lastname'])) {

// On vérifie avec le preg match si la valeur respecte bien les valeurs que l'ont attends si c'est le cas

**if** (preg\_match($regexName, $\_POST['lastname'])) {

// Insérer dans l'attribut $lastname la valeur de l'input

$lastname = htmlspecialchars($\_POST['lastname']);

} **else** {

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

$errorList['lastname'] = 'La saisie de votre nom est invalide';

}

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

} **else** {

$errorList['lastname'] = 'Veuillez indiquer votre nom';

}

// Instanciation et éxécution de la méthode

$userProfile = $profile->getUsersInfos();

Modele de la page profil

*/\*\**

*\* Méthode permettant d’afficher les informations de l'utilisateur*

*\* @return boolean*

*\*/*

**public** **function** getUsersInfos() {

$PDOResult = **$this**->db->prepare(**'SELECT `d27PG\_users`.`id`,`d27PG\_users`.`lastname`, `d27PG\_users`.`firstname`, `d27PG\_users`.`email`, `d27PG\_users`.`password`, `d27PG\_users`.`firm`, `d27PG\_users`.`address`, `d27PG\_users`.`phoneNumber`, `d27PJ\_city`.`postalCode`, `d27PG\_users`.`idUsersType` , `d27PJ\_city`.`cityName`'**

. **' FROM `d27PG\_users`'**

. **' INNER JOIN `d27PJ\_city`'**

. **' ON `d27PG\_users`.`idCity` = `d27PJ\_city`.`id`'**

. **' WHERE `d27PG\_users`.`id` = :id'**);

$PDOResult->bindValue(**':id'**, **$this**->id, PDO::PARAM\_INT);

$PDOResult->execute(); *//On vérifie que la requête s'est bien exécutée*

**if** (is\_object($PDOResult)) { *//On vérifie que l'on a bien trouvé un utilisateur*

$result = $PDOResult->fetch(PDO::FETCH\_OBJ);

}

**return** $result;

}

*/\*\**

*\* Méthode permettant de faire la connexion de l'utilisateur*

*\* @return boolean*

*\*/*

UPDATE

Controlleur de la page profil qui permet de modifier les informations de l”utilisateur

**if** (**isset**($\_POST['modify'])) {

// On test si l'input n'est pas vide et

**if** (!**empty**($\_POST['lastname'])) {

// On vérifie avec le preg match si la valeur respecte bien les valeurs que l'ont attends si c'est le cas

**if** (preg\_match($regexName, $\_POST['lastname'])) {

// Insérer dans l'attribut $lastname la valeur de l'input

$lastname = htmlspecialchars($\_POST['lastname']);

} **else** {

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

$errorList['lastname'] = 'La saisie de votre nom est invalide';

}

// On indique qu'il y a eu une erreur et on invite a resaisir une valeur

} **else** {

$errorList['lastname'] = 'Veuillez indiquer votre nom';

}

Methode de la page profil qui permet de modifier les informations de l”utilisateur

*/\*\**

*\* Méthode permettant de faire la modification des informations de l'utilisateur*

*\* @return boolean*

*\*/*

**public** **function** usersModifications()

{

$query = **'UPDATE `d27PG\_users` SET `lastname`= :lastname,`firstname`= :firstname,`address`= :address, `phoneNumber`= :phoneNumber, `email`= :email, `idCity` = :city WHERE `id` = :id'**;

$modifUsers = **$this**->db->prepare($query);

$modifUsers->bindValue(**':id'**, **$this**->id, PDO::PARAM\_INT);

$modifUsers->bindValue(**':lastname'**, **$this**->lastname, PDO::PARAM\_STR);

$modifUsers->bindValue(**':firstname'**, **$this**->firstname, PDO::PARAM\_STR);

$modifUsers->bindValue(**':address'**, **$this**->address, PDO::PARAM\_STR);

$modifUsers->bindValue(**':phoneNumber'**, **$this**->phoneNumber, PDO::PARAM\_STR);

$modifUsers->bindValue(**':email'**, **$this**->email, PDO::PARAM\_STR);

$modifUsers->bindValue(**':city'**, **$this**->idCity, PDO::PARAM\_INT);

**return** $modifUsers->execute();

}

DELETE

Vue du formulaire de profile et du bouton qui va permettre de supprimer les informations

<a type=**"submit"** href=**"profile.php?delete=1"** **class**="btn btn-danger" name="delete" id="idRemove">Supprimer mes informations</a>

Modele du formulaire de profile et du bouton qui va permettre de supprimer les informations

*/\*\**

*\* Méthode permettant de supprimer l'utilisateur*

*\* @return boolean*

*\*/*

**public** **function** removeUsers()

{

$remove = **$this**->connexion->prepare(**'DELETE FROM `d27PG\_users`'**

. **'WHERE `id` = :idRemove '**);

$remove->bindValue(**':idRemove'**, **$this**->id, PDO::PARAM\_INT);

**return** $remove->execute();

}

Controlleur du formulaire de profil et du bouton qui va permettre de supprimer les informations

**if** (**isset**($\_GET[**'delete'**])) {

$profile = **new** users();

$profile->id = $\_SESSION[**'id'**];

*//Utilisation de la méthode de suppression*

**if** ($profile->removeUsers()) {

session\_unset();

session\_destroy();

header(**'Location:http://buildr'**);

**exit**;

}

}

CONCLUSION