Dossier projet

Examen CDA

Sylvain Torrenti

CDA 07

A picture containing text, circle, diagram, design

Description automatically generated

# Remerciements

Pour commencer, je souhaite remercier la Maison Départementale des Personnes Handicapées des Bouches du Rhône, Cap Emploi, CT conseil et le centre 2ISA pour m’avoir permis cette reconversion professionnelle.

Je remercie vivement l’équipe pédagogique. Que ce soient les formateurs techniques, **Serge BOISSEAU**, **Théo OLLIVIER-TRIQUET**, **Fabien BELUGOU**, **Philippe PALAU** et **Hakim LASSOUED** qui m’ont permis de mieux appréhender les différentes problématiques auxquelles j’ai été confronté mais aussi les formateurs qui sont intervenus pour les compétences transverses, **Sarah KINSLEY, Annelies AANTJES, Dylan PEREZ** et **Alexandra TESTART.**

Je souhaite également remercier **Muriel ANDREO** et **Carole GALTIER** pour leur accueil au sein de l’établissement ainsi que tous mes collègues de la formation, qui ont permis un cadre propice à l’apprentissage, sans oublier **Sky GUNNING** ainsi que l’entreprise DSI qui m’ont accueilli durant mon stage de fin de formation.

Et enfin, merci à ma famille qui m’a soutenu et encouragé durant cette dernière étape de ma formation.

Table des matières

[Remerciements 2](#_Toc166319037)

[Projet effectué durant la période de stage 5](#_Toc166319038)

[Présentation de l’entreprise 5](#_Toc166319039)

[Résumé 5](#_Toc166319040)

[Présentation personnelle et de la formation 6](#_Toc166319041)

[Liste des compétences couvertes par le projet 7](#_Toc166319042)

[Expressions des besoins 7](#_Toc166319043)

[Spécifications fonctionnelles 8](#_Toc166319044)

[Prérequis du projet 8](#_Toc166319045)

[Présentation du MVC 8](#_Toc166319046)

[Mise en place de l’environnement de travail 8](#_Toc166319047)

[Développement du projet 9](#_Toc166319048)

[Création de la Base De Données 9](#_Toc166319049)

[La migration 10](#_Toc166319050)

[Les routes 11](#_Toc166319051)

[Les vues 12](#_Toc166319052)

[La Page d’accueil 13](#_Toc166319053)

[Création de compte 14](#_Toc166319054)

[Le chat général 15](#_Toc166319055)

[L’envoi de message 16](#_Toc166319056)

[Le Controller 17](#_Toc166319057)

[Le Model 18](#_Toc166319058)

[Gestion des utilisateurs 18](#_Toc166319059)

[La création 18](#_Toc166319060)

[Les Messages de la discussion 23](#_Toc166319061)

[La création 23](#_Toc166319062)

[Upload et affichage des fichiers joints 28](#_Toc166319063)

[Affichage des messages 29](#_Toc166319064)

[Utilisation de Javascript pour éviter un maximum les rechargements de pages 32](#_Toc166319065)

[Docker 33](#_Toc166319066)

[Dockerfile 33](#_Toc166319067)

[Docker-compose 33](#_Toc166319068)

[Tests effectués 35](#_Toc166319069)

[Recherches en langue anglaise 36](#_Toc166319070)

[Conclusion 37](#_Toc166319071)

[Projet fil rouge 2 : 37](#_Toc166319072)

[Maintenance du parc informatique 37](#_Toc166319073)

[Liste des compétences couvertes par le projet 37](#_Toc166319074)

[Résumé 38](#_Toc166319075)

[Prérequis du projet 39](#_Toc166319076)

[Mise en place du projet 39](#_Toc166319077)

[Conception de la Base de Données 40](#_Toc166319078)

[Côté Client 43](#_Toc166319079)

[Windows Form 43](#_Toc166319080)

[Operations possibles 43](#_Toc166319081)

[Côté Serveur 47](#_Toc166319082)

[La structure 47](#_Toc166319083)

[L’API 47](#_Toc166319084)

[BLL 48](#_Toc166319085)

[DAL 48](#_Toc166319086)

[Les Tests 49](#_Toc166319087)

[Conclusion du dossier 57](#_Toc166319088)

Dans ce dossier je présenterai deux projets différents.

En premier lieu, je présenterai le projet effectué pendant ma période de stage puis le second projet qui s’est déroulé pendant ma formation pour aborder les compétences qui n’auront pas été utilisées pendant la période de stage.

# Projet effectué durant la période de stage

## Présentation de l’entreprise

L’entreprise DSI a été fondée en 1994. C’est une SARL avec les agréments d’états « Entreprise adaptée » et « Entreprise Solidaire d’Utilité Sociale ». L’entreprise compte plus de 800 salariés partout en France.

Elle évolue dans 6 corps de métiers différents, qui vont de travail intérimaire à la production agricole. Chaque corps de métiers est situé sur différents sites en France.

Durant ma période de stage j’ai intégré le service informatique qui se situe à Toulouse. Ce service s’occupe, entre autres, de développement de logiciels, de support et assistance aux utilisateurs.

## Résumé

Le projet consiste à créer une plateforme de discussion instantanée.

Dans cette optique, je commence par analyser le **cahier des charges**. Je mets en avant les fonctionnalités qui seront nécessaires à l’accomplissement du projet.

Dans le cas présent, je suis seul sur le projet. j’ai mis en place une **gestion de projet** en utilisant des tickets. Tout au long de ce projet, les différentes étapes répertoriées sont définies par des **ticket** via l’outil **GitLab**.

Le travail que j’effectue est partagé sur la plateforme GitLab grâce à l’outil **Fork** (Voir annexe 1) qui est un outil permettant la gestion de différentes branches d’un projet sur GitLab.

J’entreprends la **conception** du projet. J’utilise **VS Code** comme **IDE** et **CodeIgniter 4** comme **Framework PHP**.

Ensuite, j’utilise l’architecture **MVC** (Model, Vue, Controller). Les **Controllers** encapsuleront la logique métier. En cas de besoin de données persistantes, j’utilise un **Model** qui récupère les données dans le SGBD **MYSQL** et les retourne au Controller. Les **vues**, quant à elles, seront créées par le moteur de Template **TWIG**. Le tout sera lié par des **routes**.

Vient ensuite la mise en forme avec l’aide de **Bootstrap**. Pour ce faire, j’utilise les conseils prodigués par mes collègues, je trouve un **Template** qui me convient et le modifie pour qu’il corresponde à mon projet.

Mots Clés :cahier des charges, gestion de projet, ticket, GitLab, Fork, conception, VS Code, IDE, CodeIgniter 4, Framework PHP, MVC, Controller, Model,MYSQL, vues, TWIG, routes, Bootstrap, Template.

# Présentation personnelle et de la formation

Je m’appelle Sylvain TORRENTI, j’ai 33 ans. J’étais, auparavant, employé polyvalent dans l’hôtellerie restauration et j’ai décidé de me reconvertir dans l’informatique. Ce secteur m’attire depuis toujours.

L’association 2ISA a été créé le 23 juin 2010. Elle propose des formations pour les personnes en situation de handicap mais possède également un organisme qui forme toute personne possédant le niveau Bac aux métiers du numérique.

Le Concepteur Développeur d’Applications conçoit et développe des services numériques à destination des utilisateurs. Il intervient dans des projets visant à automatiser des processus de l'entreprise. Ces projets font suite à des demandes formulées directement par un client, par une maîtrise d’ouvrage ou par l’intermédiaire d’un chef de projet.

Durant ma période de stage, qui s’est déroulée du 19 février 2024 au 07 mai 2024 j’ai mis en place un projet défini par le Responsable Développement Informatique Sky GUNNING en utilisant des technologies et des outils qui sont maitrisés dans l’entreprise qui m’accueille.

Pour ce projet, j’ai dû mettre en œuvre une fenêtre de chat instantanée avec une possibilité de discussion privée. Pour le débuter, j’ai dû permettre la création d’utilisateurs qui, eux, peuvent s’authentifier et poster des messages et ont la possibilité de choisir quelle fenêtre de discussion ils souhaitent afficher.

## Liste des compétences couvertes par le projet

* Développer des composants d’accès aux données.
* Développer la partie front-end d’une interface utilisateur web.
* Développer la partie back-end d’une interface utilisateur web.
* Contribuer à la mise en production dans une démarche DevOps

# Expressions des besoins

Pour ce projet, il m’était demandé de mettre en place un chat de discussion instantané en PHP avec l’aide de CodeIgniter 4, Twig et Bootstrap. Cette technologie et ces outils ont étaient sélectionnés car ils sont habituellement utilisés et maitrisés par le personnel de l’entreprise où j’effectue mon stage.

# Spécifications fonctionnelles

Les spécifications étaient de réaliser un salon de discussion, de rendre ensuite cette communication instantanée puis de donner la possibilité de création de salon de discussion privé.

## Prérequis du projet

Il m’a fallu approfondir mes compétences avec le langage JavaScript et l’utilisation de requête Ajax. Mes collègues et des recherches internet m’ont permis de résoudre les problèmes rencontrés.

J’ai également appris l’utilisation du framework CodeIgniter 4. L’apprentissage a été assez fluide car il est semblable à Laravel que j’avais utilisé pour effectuer le premier projet fil rouge de la formation.

Ce framework facilite grandement le développement web en mettant à disposition de nombreuses librairies pour effectuer toutes sortes de tâches.

Pour ce projet, et pour toute la formation, les éléments présents sur le site AMIO-FIT permettent de trouver les informations recherchées qu’elles soient transverses ou techniques.

## Présentation du MVC

Le MVC (Model-Vue-Controller) est un motif d’architecture. L’utilisateur fait une demande qui est exécutée par le Controller. Si des données présentes en BDD sont nécessaires, le Controller fait appel au Model approprié et renvoie la réponse via une Vue à l’utilisateur. (Voir annexe 2)

## Mise en place de l’environnement de travail

A mon arrivé, au sein de l’entreprise DSI où j’ai effectué mon stage, un compte avec l’accès aux projets de l’entreprise m’a été fourni. J’ai mis en place l’IDE Visual Studio Code, le logiciel Fork ainsi que le logiciel Docker Dekstop.

Durant mon stage, il m’a été donné comme mission de créer un chat de discussion instantanée.

Grâce à cet accès, un repository comprenant le projet sur lequel j’ai travaillé a été créé. Sky GUNNING et moi-même, étions membres de ce repository ce qui nous permettaient de faire les modifications nécessaires tout au long du parcours.

Pour superviser mon travail, Sky GUNNING a mis en place des tickets concernant les étapes et leur description afin de me guider précisément.

Après cette présentation, voici le développement de ce projet :

# Développement du projet

## Création de la Base De Données

Dans un premier temps, j’ai créé la Base De Données sur laquelle je devais effectuer mon projet. J’ai utilisé phpMyAdmin et son interface graphique pour créer la BDD (figure 1).

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure 1 Interface graphique

Pour créer les tables, j’ai également utilisé l’interface graphique de phpMyAdmin. Cette interface permet de choisir toutes les caractéristiques des tables désirées (figure 2).

Une image contenant capture d’écran

Description générée automatiquement

Figure 2 Interface graphique table

Par la suite, quand j’avais des erreurs dans mes requêtes, je les effectuais d’abord grâce à phpMyAdmin pour vérifier leur exactitude, ensuite je les incorporais dans mon code en y effectuant les modifications nécessaires pour son bon fonctionnement.

## La migration

La migration est un moyen pratique de garder sa BDD de manière structurée et organisée. Elle permet de partager les changements effectués sur la BDD auprès de tous les participants au projet. Elle est également utile lors de la montée en version chez les clients.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 3 Migration CreateUsersTable

Comme le montre la Figure 3, ce script est utilisé pour la migration qui concerne la table users. La partie qui se situe dans la fonction « up() » est exécutée lors de la migration, celle qui se situe dans la fonction « down() » est exécutée lors des rollback effectués. Pour chaque table de la BDD, un script de migration est créé.

## Les routes

Elles sont essentielles au bon fonctionnement de CodeIgniter 4. Elles permettent de faire le lien entre les vues et d’utiliser les fonctions présentes dans un Controller (qui seront présentées par la suite) (Annexe 3).



Figure 4 Route

La route (figure 4) utilise la fonction « check » présente dans « AuthController » qui permet l’authentification.

Il existe aussi la possibilité de créer des routes paramétrées (figure 5).



Figure 5 route avec paramètre

Cette route permet de retourner la vue de la fenêtre de discussion privée sélectionnée par l’utilisateur grâce à la fonction « chatPerso/$1 » présente dans ChatController. Cette fonction permet d’afficher la page de discussion entre l’utilisateur authentifié et un utilisateur sélectionné par celui-ci.

## Les vues

Pour ce projet, j’ai utilisé le Framework CodeIgniter 4. Ce Framework est open-source et respecte le MVC (Model Vue Controller). Il peut être utilisé en association avec Twig qui est un moteur de Template.

Twig, permet la création des Template qui pourront être utilisés dans les pages pour les rendre homogènes.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 6 Twig head

La figure 6 représente le Template que j’ai utilisé sur l’ensemble de mes pages. Toutes mes pages ont cet élément comme Head.

Il est à noter que les liens de mise en forme, grâce au framework CSS Bootstrap, sont présents sur toutes les pages HTML et permettent ainsi d’avoir une cohérence de mise en forme dans tout le projet.

La Figure 7 montre tous les Template créés.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Figure 7 Partial

L’utilisation d’un layout qui regroupe et assemble tous les éléments créés permet d’avoir une cohérence sur l’ensemble du projet.

## La Page d’accueil

La page d’accueil du site créée pour le projet est présentée dans la figure 8.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 8 page d'accueil

Sur la page d’accueil il y a la possibilité de s’identifier, si le compte existe ou de le créer le cas échéant.

### Création de compte

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 9 Création de compte

La figure 9 montre le formulaire de création de compte. J’ai indiqué par une astérisque « \* », les champs obligatoires. Dans le code, j’ai également indiqué que devaient obligatoirement être renseignés :

* L’Email et le Mot de Passe car ils permettent l’authentification et sans eux il n’est pas possible d’avoir accès au compte.
* Le Nom car il est nécessaire au bon fonctionnement du projet dans les cas que nous verrons plus tard.
* Un lien vers la page de Login est également présent si l’utilisateur possède déjà un compte et qu’il est arrivé sur cette page par mégarde.

## Le chat général

Lorsque l’authentification est réussie, l’accès à la page de la discussion instantanée général est ouverte. A partir de cette page, l’utilisateur a accès à la liste des utilisateurs enregistrés qui lui permettra d’accéder à la discussion instantanée privée. (Figure 10)

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Figure 10 Chat général

Dans cette fenêtre, les messages sont disposés en fonction de l’utilisateur connecté. Les messages envoyés par celui-ci sont disposés sur la gauche et les autres sur la droite. Le nom de l’utilisateur qui a envoyé le message, est seulement indiqué si le message n’a pas été envoyé par l’utilisateur connecté.

Il y a également la présence de la liste des utilisateurs enregistrés permettant l’accès à la fenêtre de discussion privée entre l’utilisateur connecté et l’utilisateur choisi dans la liste.

Au sommet de la fenêtre de discussion, on retrouve le nom de l’utilisateur connecté.

La figure 11 montre la discussion privée entre l’utilisateur Sylvain et Jose. Cette discussion n’est visible que par ces deux utilisateurs.

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 11 Chat privé

Un bouton permet de revenir sur le chat général.

### L’envoi de message



Figure 12 Envoi de message

Ce formulaire (Figure 12) permet d’envoyer le message souhaité avec la possibilité de joindre une image. Le message et le lien avec l’image potentielle sont enregistrés en BDD.

### Le Controller

En suivant la méthode MVC, il faut obligatoirement passer par un Controller. Celui-ci est appelé par la route qui utilise également la fonction créée dans le Controller (figure 13).

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 13 ChatController

Sur la figure 13, est présentée la fonction qui récupère les messages dans la BDD avec l’aide du Model. Elle vérifie où se situe l’utilisateur (chat général ou salon privé) et récupère les messages en conséquence.

### Le Model

Il est aussi nécessaire de mettre en place un Model qui sert de lien entre le Controller et la Base De Données (figure 14).

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 14 Message Model

## Gestion des utilisateurs

### La création

Pour créer des utilisateurs j’ai utilisé la méthode MVC (Model, Vues, Controller).

#### La route

Ces deux routes permettent la création d’un compte utilisateur. La première donne accès à la vue de création et la seconde sauvegarde les informations fournit en BDD. (Figure 15)

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Figure 15 Route register

#### Le Controller

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 16 Create account

La Figure 16 détaille la présence des différentes fonctions nécessaires à la création d’un compte. Les différentes règles de validation sont également présentes.

Voici ce qui est mis en place pour les emails :

* La présence d’un email est obligatoire grâce à la règle « required »
* L’email doit être dans le format conforme grâce à la règle « valid\_email »
* L’email ne doit pas être déjà présent dans la BDD grâce à « is\_unique[users.email] » cette règle vérifie la présence de l’email dans la table users.

Ainsi, des messages d’erreur personnalisés sont définis pour chaque règle de validation.

#### Le Model

Grâce à CodeIgniter, il suffit de renseigner seulement les informations essentielles qui pourront être utilisées plus tard en effectuant toute modification ou action voulues. (Figure 17)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 17 Model

#### Les vues

Une image contenant texte, capture d’écran, menu, Police

Description générée automatiquement

Figure 18 Vue Register

La figure 18 montre la vue de la création de compte. Celle-ci est faite via Twig avec la possibilité d’afficher les erreurs le cas échéant.

Une image contenant texte, capture d’écran, Police, menu

Description générée automatiquement

Figure 19 Vue LogIn

La figure 19 montre la vue qui permet de s’authentifier au chat.

## Les Messages de la discussion

### La création

Pour créer des messages en suivant la méthode MVC, il faut trois fichiers différents : le Model, la Vue et le Controller.

#### La route

La figure 20 montre la route utilisée quand l’utilisateur crée des messages.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 20 route création message

Cette route indique que le formulaire doit être utilisé avec une méthode POST. Elle utilise la fonction « postMessage » qui se trouve dans « ChatController ».

#### Le Controller

Comme indiqué, la route utilise la fonction « postMessage » se trouvant dans le Controller (figure 21).

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 21 fonction postMessage

Grace à cette seule fonction, il est possible d’enregistrer en BDD les messages postés sur le chat général, le chat privé et également si une image est jointe ou non. Ces différents cas seront importants en ce qui concerne la restitution des messages.

#### Le Model

Pour finir, il y a le « Model » : il sert de lien entre la Base De Données et le Controller. Il a déjà été vu auparavant (figure 15). Grâce à CodeIgniter 4, il est possible d’utiliser directement le Model dans le Controller en y créant une instance du Model et en utilisant les fonctions présentent dans CodeIgniter4. Grâce aux champs indiqués dans le Model, l’utilisation de ceux-ci est permise.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 22 Fonction postMessage

Cette fonction (figure 22), présente dans le Controller, utilise une instance du Model des messages et du Model des fichiers si elle est nécessaire.

Utiliser la fonction « save » permet d’enregistrer en BDD en indiquant les champs préalablement renseignés dans le Model avec les valeurs souhaitées.

#### La vue

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 23 Vue PostMessage

Pour la création de message, la vue (figure 23), indique que le champ « texte » ne peut accepter que 500 caractères maximum et que le champ « file » n’accepte que les extensions « .jpg » « .jpeg » et « .png ».

#### L’envoi du formulaire

Le formulaire est envoyé via le script JavaScript. (Figure 24)

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 24 Script ajoutMessage

Dans cette portion du script, est présentée la procédure d’envoi du message au Controller permettant ainsi la sauvegarde en BDD.

La première condition « if (document.querySelector('#texte') != null) » permet de savoir que nous sommes situés sur la page de chat général car un élément html avec l’id « texte » est présent sur cette page.

Les différentes variables nécessaires sont définies pour correspondre à la demande. Ensuite, il nous faut changer ce que la validation du formulaire exécute, cela est fait grâce à

« $(‘#chat’).off(‘submit’).on(‘submit’, function(e) {».

La désactivation et la réactivation d’envoi du formulaire sont nécessaires pour qu’elle exécute la fonction définie ensuite.

La création d’un « FormData() » permet de définir et d’ajouter les informations nécessaires.

La deuxième condition « if (fileInput.files.length != 0) » vérifie si un fichier est joint au message. Si c’est le cas, le nom du fichier est récupéré pour l’enregistrer en BDD et le lier au message. Les fichiers ne sont pas sauvegardés en BDD mais dans le dossier « writable/chatFiles » pour une meilleure sécurité et un accès limité aux fichiers.

Une fois toutes les informations récupérées, une requête Ajax effectue l’appel au Controller. (Figure 25) »

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Figure 25 requête ajax

Cette fonction utilise la route postMessage qui mène au ChatController et exécute la fonction postMessage. « data » correspond à l’ensemble des données que nous voulons enregistrer en BDD.

### Upload et affichage des fichiers joints

J’ai choisi d’enregistrer les fichiers joints dans le dossier writable/chatFiles/mm/yyyy.

#### La fonction d’upload

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 26 Fonction Upload

Cette fonction (Figure 26) permet l’upload du fichier joint avec la création du sous-dossier avec le nom correspondant s’il n’existe pas. Le nom du fichier est également modifié pour correspondre au format demandé.

Cette fonction est appelée à chaque fois qu’un message avec un fichier joint est envoyé.

#### L’affichage du fichier

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 27 fonction display

Les fichiers étant enregistrés dans le dossier writable, ils ne sont pas accessibles car le dossier est protégé et interdit la lecture. Il faut donc créer une fonction permettant de les récupérer et de les afficher.

Cette fonction (figure 27) utilise une instance de FileModel() qui peut utiliser la fonction définie dans le Model. L’adresse où les fichiers sont affichés sera utilisée pour l’affichage des messages avec un fichier joint.

### Affichage des messages

#### Les routes

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Figure 28 Routes de l'affichage des messages

Ces routes (figure 28), sont utilisées dans les divers scenarii.

#### Le Controller

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Figure 29 Fonction GetMessage

Cette fonction (figure 29) est également appelée via une requête Ajax. (Figure 30)

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 30 Requête ajax

Dans cette requête, les messages sont récupérés sous format JSON, analysés et ajoutés au fil de discussion.

Si le message est lié à un fichier, celui-ci est obtenu avec l’adresse retrouvée grâce à la fonction vue plus haut.

Pour définir la disposition des messages (receveur ou envoyeur), un élément html « hidden », qui a pour valeur l’id de la personne authentifiée, est placé dans le chat général et dans le chat privé.

#### Le Model

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 31 Fonction getMessage

Cette fonction (figure 31) permet de récupérer les messages présents en BDD.

Les clauses where() sont différentes selon où se situe l’utilisateur et des alias sont utilisés pour différencier les champs qui ont un nom similaire sur différentes tables utilisées.

### Utilisation de Javascript pour éviter un maximum les rechargements de pages

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure 32 Fonction JS

Sur la figure 31, sont présentées les différentes fonctions qui s’exécutent dès le chargement de la page. Des événements sont ajoutés aux clicks de la souris ou de la pression de la touche « entrée ».

La fonction verifEntree est utilisée pour s’assurer que la touche pressée est bien la touche « entrée » et effectue alors la fonction ajoutMessage().

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 33 Fonction verifEntree

La fonction d’affichage des messages est exécutée toutes les secondes ce qui permet que les messages soient affichés en temps réel sans avoir à rafraichir la page par l’utilisateur. Cette fonctionnalité est présentée dans la Figure 25 plus haut dans ce dossier.

## **Docker**

Docker est un outil permettant d’empaqueter une application et ses dépendances dans un conteneur qui pourra, par la suite, être exécuté sur n’importe quel serveur. Cette pratique permet, entre autres, de faire fonctionner de manière stable une application quelle que soit la machine hôte.

Pour créer un conteneur, il faut, au préalable, construire une image de l’application. Les images peuvent être facilement partagées et contiennent tout ce qui est nécessaire pour le bon fonctionnement d’une application.

### Dockerfile

Le Dockerfile est un fichier texte qui contient les instructions permettant de construire une image Docker.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 34 Dockerfile

Sur la figure 34, sont détaillés les instructions de création de l’image de l’application :

* **FROM :** Indique l’image de base qui est utilisée pour construire l’image.
* **COPY :** Copie les fichiers et les répertoires dans le conteneur.
* **RUN :** Exécute une commande dans le conteneur.
* **WORKDIR :** Permet de définir le répertoire de travail.

### Docker-compose

Docker-compose est un outil permettant de définir et d’exécuter des applications Docker multi-conteneurs. Il permet de définir tous les services nécessaires à l’application.

Une image contenant texte, capture d’écran, menu, Police

Description générée automatiquement

Figure 35 Docker-compose

Dans ce Docker-compose, trois conteneurs sont créés.

Le premier est le conteneur de l’application qui utilisera le fichier Dockerfile, expliqué plus haut, pour créer l’image qui sera dans le conteneur avec les variables d’environnement nécessaires pour se connecter à la base de données.

Le deuxième contiendra la base de données qui sera basée sur l’image officielle mysql. Le mot de passe et la base de données sont définis par des variables d’environnement.

Enfin, le troisième n’est pas obligatoire, il permet de mettre en place Adminer, qui est un outil d’administration de base de données, semblable à PhpMyAdmin et permettre ainsi d’avoir une accessibilité facilitée à la base de données.

Un volume est également créé pour permettre la persistance des données car quand un conteneur est arrêté, les données ne sont pas enregistrées.

Lors de la création des conteneurs via un docker-compose, un réseau est créé dans lequel les différents conteneurs créés sont présents et permet qu’ils soient connectés entre eux.

## Tests effectués

Durant ce projet je n’ai mis en place que des tests d’acceptations. Pour les effectuer, j’ai fait appel à des personnes extérieures que je souhaite grandement remercier, **GUNNING Sky, BAYONNOVE Robin et NOEL Jean Christophe.**

Ils m’ont permis de mettre en évidence des problématiques que je n’avais pas envisagées et m’ont aussi prodigué des conseils au niveau du design et de l’ergonomie.

Pour que les tests soient le plus fidèles à la réalité, je ne leur ai fait qu’un rapide résumé de la plateforme sans donner d’indication sur la marche à suivre (mis à part qu’ils devaient créer un compte). Grâce à cela, j’ai pu mettre en lumière des dysfonctionnements de quelques fonctionnalités sous certaines conditions et qui m’avaient échappé car je connaissais le code.

## Recherches en langue anglaise

Lors de la réalisation de mon projet, j’ai effectué différentes recherches dans la documentation de CodeIgniter4 disponible uniquement en anglais.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 36 Documentation CI4

Dans la figure 1, nous pouvons voir comment utiliser le Query Builder qui permet facilement l’utilisation de la BDD. Si toutefois les informations trouvées via la documentation ne sont pas suffisantes, effectuer une recherche annexe avec les termes anglais permet de trouver des résultats plus pertinents que si la recherche est faite en français.

# Conclusion

En conclusion, ce projet m’a permis de connaitre et d’apprendre à utiliser le Framework CodeIgniter 4 même si je suis conscient que je n’ai fait qu’en effleurer la surface.

J’ai pu approfondir et améliorer mes compétences et ma compréhension du langage PHP que je n’avais pas eu l’occasion de beaucoup pratiquer durant la formation.

J’ai également pu améliorer mes compétences en ce qui concerne le JavaScript et apprendre à travailler à l’aide de tickets liés à différentes problématiques sur lesquelles je devais travailler.

# Projet fil rouge 2 :

# Maintenance du parc informatique

## Liste des compétences couvertes par le projet

* Maquetter une application.
* Concevoir une application organisée en couche.
* Préparer et exécuter les plans de tests d’une application
* Concevoir une Base De Données.

## 

## Résumé

Le projet consiste à créer une application Windows Form pour gérer la gestion du parc informatique.

Dans cette optique, je commence par analyser le **cahier des charges**. Je mets avant les fonctionnalités qui seront nécessaires à l’accomplissement du projet.

J’effectue ma **gestion de projet** grâce à l’outil en ligne [www.monday.com](http://www.monday.com) qui me permet de mieux gérer mon temps et de planifier les étapes qu’il me faut mettre en place pour ce projet.

Une fois ces fonctionnalités listées, je produis une **maquette** en utilisant l’outil **FIGMA.** Cet outil permet de créer l’enchainement des écrans disponibles en indiquant les liens à suivre et peut se partager en ligne sans contrainte.

Une fois satisfait de la maquette, j’entreprends la **conception** du projet. J’utilise **Visual Studio** comme **IDE** et **.Net** comme **Framework**.

Je crée une **bibliothèque de classe** où je réunis les **DTO** (Data Transfert Object), les **entités** et les **exceptions** pour bien séparer les différents éléments composant l’ensemble du projet.

Ensuite, j’utilise une structure **multicouche** coté serveur pour récupérer les données dans le SGBD **MYSQL** créé à l’aide de **DBeaver** et les retourne à l’API.

L’utilisateur utilise le logiciel créé grâce à la maquette pour interroger l’API et obtenir les données voulues ou effectuer les opérations souhaitées selon son rôle.

Mots Clés : cahier des charges, gestion de projet, maquette, FIGMA, conception, Visual Studio, IDE, .Net, Framework, bibliothèque de classe, DTO, entités, exceptions, multicouches, MYSQL, DBeaver

## Prérequis du projet

Les prérequis techniques présents dans le projet ont été étudiés avec les différents formateurs techniques. Ils ont été présentés de façon à pouvoir effectuer le travail en continu. Ce projet a permis de mettre en pratique toutes les notions que nous avions abordées depuis le début de la formation.

Pour ce projet, et pour toute la formation, les éléments présents sur le site AMIO-FIT permettent de trouver les informations recherchées qu’elles soient transverses ou techniques.

Certaines informations ont nécessité des recherches internet que j’ai effectuées en anglais car les résultats sont plus nombreux et mieux documentés.

## 

## Mise en place du projet

Pour commencer, j’ai utilisé le langage UML pour dégager les comportements nécessaires. J’ai tout d’abord fait un Diagramme de cas d’utilisation comme le montre la figure 37.

Une image contenant texte, diagramme, capture d’écran, conception

Description générée automatiquement

Figure 37 Use Case Diagram

J’ai également produit un Diagramme de Classe (figure38).

Une image contenant texte, diagramme, ligne, Police

Description générée automatiquement

Figure 38 Class Diagram

## Conception de la Base de Données

Des scripts pour la création des différentes tables sont produits comme le montre la Figure 39. Le nom des champs, leur type et leur contraintes (s’il y en a) sont indiqués. La clé primaire est également renseignée.

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Figure 39 Script création de table Materiel

Pour faciliter la création de la BDD, les clés étrangères sont ajoutées par la suite par l’utilisation d’un script d’altération de la table. (Figure 40)



Figure 40 Alter Table

Grace à DBeaver il est également possible d’avoir une vision globale de la BDD : une partie est affichée dans la figure 41.

Une image contenant texte, capture d’écran, diagramme, logiciel

Description générée automatiquement

Figure 41 Diagramme BDD

J’ai ensuite utilisé l’outil de maquettage en ligne FIGMA comme le montre la figure 42.

Une image contenant capture d’écran, texte, logiciel, conception

Description générée automatiquement

Figure 42 Illustration des onglets sous FIGMA

Le maquettage m’a permis d’avoir une vision globale de mon projet et ainsi de réfléchir en amont aux problématiques possibles.

## Côté Client

### Windows Form

J’ai utilisé une IHM réalisée grâce à Windows Form. Cette technologie permet de créer des fenêtres qui permettent d’afficher les données souhaitées.

Il y avait plusieurs possibilités et j’ai opté pour une approche avec une seule fenêtre contenant les différents onglets nécessaires (figure43).

Une image contenant texte, capture d’écran, logiciel, ordinateur

Description générée automatiquement

Figure 43 Fenêtre avec onglets

### Operations possibles

#### Consulter

Une image contenant texte, nombre, logiciel, Page web

Description générée automatiquement

Figure 44 Onglet Consulter

Cet onglet permet d’afficher les différents matériels présents sur la plateforme mais aussi à l’utilisateur de filtrer les résultats qu’il souhaite afficher en fonction d’une catégorie ou d’un utilisateur demandé (figure 44).

#### Création

Une image contenant texte, logiciel, nombre, capture d’écran

Description générée automatiquement

Figure 45 onglet Création

Cet onglet permet de renseigner toutes les informations nécessaires à l’enregistrement d’un nouveau matériel. Si la/les catégories n’existent pas il est possible d’en créer de nouvelles (figure 45).

#### Modification

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Figure 46 Onglet Modification

Ainsi que présenté dans la figure 46, l’administrateur peut modifier toutes les informations concernant un matériel. Il peut également, pour faciliter sa recherche, les trier selon une catégorie spécifique ou un utilisateur.

En ce qui concerne la date de mise en service, j’ai mis en place un DateTimePicker avec une CheckBox qui permet d’indiquer si le matériel est mis en service ou non.

J’ai également rajouté la possibilité de ne pas allouer de propriétaire à un matériel en sélectionnant « Aucun propriétaire » dans la sélection du propriétaire.

#### Suppression

Une image contenant texte, logiciel, nombre, Page web

Description générée automatiquement

Figure 47 Onglet Suppression

Dans la figure 47, nous pouvons voir que l’administrateur peut également trier les matériels comme il le souhaite et ainsi trouver plus facilement ce qu’il cherche. Quand il décide de supprimer un matériel les relations présentent dans la BDD qui lui sont liées sont également supprimées.

## Côté Serveur

### La structure

Du côté serveur, j’ai mis en place plusieurs couches pour faciliter l’organisation, la séparation des responsabilités et la maintenabilité.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 48 Structure côté serveur

Comme le montre la figure 48, plusieurs projets sont présents dans la solution. L’utilisateur ne communique qu’avec l’API. C’est elle qui fait les requêtes permettant de récupérer les données dans la BDD.

* L’utilisateur via l’IHM fait sa demande à l’API.
* L’API fait suivre cette demande à la BLL (procède aux vérifications qui lui sont allouées).
* La BLL interroge la DAL qui va faire les requêtes auprès de la BDD pour récupérer les données qui sont demandées.
* Les données font le chemin inverse pour arriver à l’utilisateur.

Durant toute la procédure des vérifications sont faites en fonction de chaque couche et de leurs fonctions.

### L’API

L’API est la porte d’entrée pour permettre à l’utilisateur d’accéder aux données. A chaque fois que l’utilisateur fait une demande, celle-ci passe par le controller adéquat qui fait suivre cette demande à la BLL.(figure 49)

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 49 Structure de l'API

#### Les Controller

Les Controller servent de ponts entre l’API et les couches suivantes.

### BLL

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Figure 50 Structure de la BLL

La BLL sert d’intermédiaire entre l’API et la DAL (Data Acess Layer) et compile les règles métier de l’application. Comme le montre la figure 50, la présence d’interface pour chaque service permet de faciliter de futures modifications.

### DAL

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 51 Structure de la DAL

La DAL (figure 51) permet d’avoir accès à la BDD. Dans toute l’application, c’est le seul module qui a accès directement à la BDD.

### Les Tests

#### Les tests Unitaire

Les tests unitaires permettent de vérifier le bon fonctionnement d’une partie précise d’un logiciel. Ils permettent également de mieux comprendre comment utiliser une méthode et permettent de passer outre une potentielle obsolescence de la documentation.

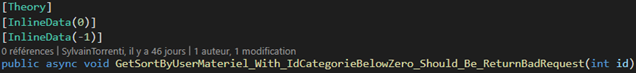


Figure 52 Test Unitaire nom

Le nom de chaque test doit reprendre le nom de la méthode testée « **GetSortByUserMateriel** », les conditions « **with\_IdCategorieBelowZero** » et enfin les résultats attendus « **Should\_Be\_ReurnBadRequest** ». Chaque élément doit être séparé par des « **\_** ».

Il existe différents tags à apposer aux tests. Dans la figure 52, il s’agit d’un test avec le tag **[Theory]** qui permet de rajouter des paramètres dans la signature du test et qui est suivi des décorateurs **[InlineData(0)]**et **[InlineData(-1)]** qui indiquent que les valeurs 0 et -1 seront prisent comme paramètres.

Les tests unitaires sont séparés en trois parties. Ces trois parties sont :

* **Arrange** : Lors de cette étape, le programmeur doit faire en sorte que l’environnement obtienne les caractéristiques voulues par le test. Pour cela des Mock (ce sont des simulacres qui reproduisent le comportement d’objets réels de manière contrôlée) des éléments externes à la méthode testée sont créés et configurés pour correspondre au résultat souhaité.
* **Act**: Fait appel à la méthode testée.
* **Assert**: Lors de cette étape, les résultats obtenus sont évalués. Cette ultime étape définit si le test réussit ou échoue.

Pour exécuter mes tests, j’ai analysé mon code pour détecter les résultats possibles issus de la fonction.

Comme le montre la figure 53, je suis donc parti de la méthode que j’avais déjà codée pour savoir quels tests je devais pratiquer. Il est également possible de faire l’inverse avec le TDD (Test Driven Développement) qui, lui, part des tests pour coder la fonction.

##### Test GetSortByCategorieMaterielAsync

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 53 Analyse du test

Une fois les tests dégagés, il faut créer du code pour chaque situation. Nous continuons avec l’exemple de notre méthode de filtrage par les catégories.

##### Test 1

Lors de ce premier test, j’utilise le décorateur **[Theory]** car le but de ce test est de vérifier le comportement de la fonction si l’Id de la catégorie donnée est 0 ou -1.

Pour la partie **Arrange,** il faut créer un **Mock** du service présent dans la BLL car nous testons la méthode et non ce qui est externe à celle-ci.

Pour la partie **Act,** j’exécute la fonction. Les paramètres sont définis par **[InlineData(0)]** et **[InlineData(-1)]** ce qui signifie que deux tests seront faits. Le premier avec l’Id qui sera égal à 0 et le suivant l’Id sera égal à -1.

La dernière partie **Assert** permet de vérifier l’exactitude des résultats. Dans notre cas le résultat doit être une **BadRequest**.

##### Test 2

Pour ce deuxième test (figure 43), le décorateur **[Fact]** est utilisé.

Pour la partie **Arrange,** un **Mock** est créé et paramétré pour s’accorder avec les demandes du test. Il est donc configuré pour que quelle que soit l’Id fournie rien ne soit retourné.

Pour la partie **Act,** la fonction est exécutée.

La partie **Assert** vérifie l’exactitude du résultat.

##### Test 3

Le troisième test permet de vérifier le bon fonctionnement de la fonction.

Dans la partie **Arrange,** il y a toujours la création d’un **Mock** mais également de deux **Matériels** car il faut paramétrer le **Mock** pour qu’il retourne ces deux **Materiels**. Il faut également le paramétrer pour que n’importe quel Id soit donné.

La partie **Act** exécute la fonction.

Enfin la partie **Assert** vérifie l’exactitude des résultats.

##### Résultat des tests

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 54 Résultat des tests

Comme le montre la figure 54, toutes les fonctions se trouvant dans le Controller sont testées. Tous les tests sont valides. Nous pouvons également remarquer que chaque fonction n’a pas le même nombre de tests.

##### Test d’Intégration

Les tests d’intégration permettent de vérifier le bon fonctionnement entre les différentes couches de l’application. Il faut fixer l’environnement de tests pour simuler les appels de requêtes grâce à une fixture. (Figure 55)

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 55 Fixture

Il faut également créer une base de données de tests pour éviter que la base de données ne se replisse de productions avec les différents tests. Pour cela, j’ai créé un script qui copie la base de données utilisée par l’application. J’ai également récupéré quelques données présentes dans la base de données grâce au SGBDR DBeaver qui m’a permis de directement récupérer les données avec les commandes SQL nécessaires pour les insérer dans la base de données de tests. (Figure 56 et 57)

Une image contenant texte, capture d’écran, Police, menu

Description générée automatiquement

Figure 56 Script Création de Table

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 57 Script jeu de données

L’ordre d’exécution des différentes requêtes est important car il faut prendre en compte les contraintes de la Base de Données. Ce script sera lancé lors de l’exécution de chaque test.

Le fichier contenant ce script est lu à chaque exécution d’un test ce qui permet d’avoir une BDD dans un état similaire lors de chaque test.

J’ai également créé une méthode qui permet de se loguer avec les renseignements d’un Bearer Token d’un compte administrateur pour avoir accès à toutes les fonctionnalités. (Figure 58)

Une image contenant capture d’écran, Police, texte

Description générée automatiquement

Figure 58 Test d'intégration Login

Cette méthode sera appelée lors de chaque test.

Je n’ai malheureusement pas eu le temps de faire tous les tests d’intégration. Voici une capture d’écran des tests réalisés. (Figure 59)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 59 Test d'intégration résultats

##### Tests d’Acceptation

Pour effectuer les différents tests d’acceptation, j’ai fait appel à des personnes extérieures que je souhaite grandement remercier, **BRUNDU Roseline et BRUNDU Nicolas.**

Ils m’ont permis de mettre en évidence des problématiques que je n’avais pas envisagées et m’ont aussi prodigué des conseils au niveau du design et de l’ergonomie.

Pour que les tests soient le plus fidèles à la réalité, je ne leur ai fait qu’un rapide résumé de la plateforme sans donner d’indications sur la marche à suivre. Ils ont ainsi utilisé leurs comptes créés lors de la première étape du projet et je leur ai également fournit un compte administrateur pour qu’il puisse tester toutes les fonctionnalités. Grâce à cela, j’ai pu mettre en lumière des dysfonctionnements ~~de quelques fonctionnalités~~ sous certaines conditions et qui m’avaient échappées car je connaissais le code.

#### Future évolution

Ce projet pourra, dans le futur, être relié au premier projet réalisé lors du premier EPCF. Par la suite, il est possible de rajouter une gestion de contrat de maintenance et une gestion des entreprises externes qui gèreront ces contrats de maintenances. Pour cela le choix de travailler avec des onglets, facilitera l’implémentation de futures fonctionnalités.

#### Conclusion

En conclusion, ce projet m’a permis de connaitre et d’apprendre à utiliser la conception en multicouche et comment bien séparer les différentes problématiques.

# Conclusion du dossier

Pendant ma formation Concepteur Développeur d’Application j’ai pu améliorer mes compétences dans plusieurs langages de programmation. Elles ont été améliorées avec l’aide des formateurs et des ressources présentent sur la plateforme AMIO-FIT mais également grâce à différentes recherches effectuées principalement en Anglais.

Le stage pratiqué en fin de formation m’a également permis de mieux appréhender le fonctionnement du travail dans une entreprise.