Sylvain Torrenti

Dossier projet

« A completer»

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 confrontées mais aussi les formateurs 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. Mais aussi l’entreprise DSI qui m’a accueilli durant mon stage de fin de formation.

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

Table des matières

[Remerciements 2](#_Toc164775480)

[Projet effectuer durant la période de stage 4](#_Toc164775481)

[Liste des compétences couvertes par le projet 4](#_Toc164775482)

[Résumé 5](#_Toc164775483)

[Recherches en langue anglaise 6](#_Toc164775484)

[Introduction 7](#_Toc164775485)

[Prérequis du projet 7](#_Toc164775486)

[Mise en place du projet 8](#_Toc164775487)

[Développement du projet 8](#_Toc164775488)

[Création de la Base De Données 8](#_Toc164775489)

[CodeIgniter 4 9](#_Toc164775490)

[Les routes 10](#_Toc164775491)

[Les vues 11](#_Toc164775492)

[Le controller 15](#_Toc164775493)

[Le Model 16](#_Toc164775494)

[Les utilisateurs 16](#_Toc164775495)

[La création 16](#_Toc164775496)

[Les Messages 20](#_Toc164775497)

[La création 20](#_Toc164775498)

[Upload et affichages des fichiers joints 26](#_Toc164775499)

[Affichage d’un ticket unique 27](#_Toc164775500)

[Tests effectués 30](#_Toc164775501)

[Future évolution 31](#_Toc164775502)

[Conclusion 31](#_Toc164775503)

# Projet effectuer durant la période de stage

## 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.
* Concevoir une Base De Données.

## Résumé

Le projet consiste à créer une plateforme de discussion instantanée.

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

Dans le cas présent, je suis seul sur le projet : je n’ai donc pas fait de **gestion de projet**. 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.**

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

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**.

Viens 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, MVC, Controller, Model,MYSQL, vues, TWIG, routes, Bootstrap, template.

## Recherches en langue anglaise

Lors de la réalisation de mon projet, j’ai effectué différentes recherches dans la documentation de CodeIgniter4 qui est seulement en anglais.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 1 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 effectuée en français.

## Introduction

Durant ma période de stage, qui s’est déroulé du 19 février 2024 au 07 mai 2024 j’ai effectué un projet définit par le Responsable Développement Informatique Sky GUNNING en utilisant des technologies et des outils qu’ils maitrisent.

Pour ce projet, j’ai dû mettre en œuvre une fenêtre de chat instantanée avec une possibilité de discussion privée. Pour débuter ce projet, 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.

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 toutes personnes possédant le niveau Bac aux métiers du numérique.

## Prérequis du projet

Les prérequis techniques ont été vue durant la formation et si des zones restes floues il m’était possible de demander des informations à mes collègues. Ce projet a permis de mettre en pratique des notions que nous avions abordées pendant 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.

## Mise en place du projet

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. Durant mon stage j’ai dû mettre en place 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 projet.

Tout au long du projet, Sky GUNNING met en place des tickets concernant les étapes à faire avec une description pour que je sache précisément ce que je dois faire.

Une foi la démarche à suivre expliquer et préciser je me lance dans le développement du 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 2).

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

Description générée automatiquement

Figure 2 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 3).

Une image contenant capture d’écran

Description générée automatiquement

Figure 3 Interface graphique table

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

### CodeIgniter 4

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

Grace à Twig, nous pouvons créer des Template que nous pouvons utiliser dans nos pages pour les rendre homogènes.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 4 Twig head

La figure 4 représente le Template que j’ai utilisé sur l’ensemble de mes pages. Toutes mes pages auront cet élément comme Head.

Nous pouvons voir que les liens qui mettent en forme, grâce au framework CSS Bootstrap, seront présent sur toutes les pages HTML et permet ainsi d’avoir une cohérence de mise en forme dans tout le projet.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Figure 5 Partial

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

### Faire que le chat soit instantané

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

Description générée automatiquement

Figure 6 SetInterval

Sur la figure 6, nous pouvons voir différentes fonctions qui s’exécute dès le chargement de la page. Des événements sont ajoutés aux clicks de la souris ou de la pression de la touche entré.

La fonction d’affichage des messages est exécuté toutes les secondes ce qui permet que les messages soit affiché en temps réel sans avoir à rafraichir la page par l’utilisateur.

### 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 (que nous verrons par la suite).



Figure 7 Route

La route (figure 7) utilise la fonction « check » présente dans « AuthController » qui permet l’authentification.

Il existe aussi la possibilité de créer des routes paramétrées (figure 8).



Figure 8 route avec paramètre

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

### Les vues

#### La Page d’accueil

La page d’accueil du site créé pour le projet est présentée dans la figure 9.

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

Description générée automatiquement

Figure 9 page d'accueil

Sur la page d’accueil vous avez la possibilité de vous identifier, si vous avez déjà un compte ou de le créer le cas échéant.

#### Création de compte

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

Description générée automatiquement

Figure 10 Création de compte

J’ai indiqué par une astérisque « \* », les champs obligatoires. Dans le code j’ai également indiqué qu’il devait obligatoirement être renseigné :

* L’Email et le Mot de Passe car ils permettent l’authentification et sans eux vous ne pouvez avoir accès à votre 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 vous possédez déjà un compte et que vous êtes arrivé sur cette page par mégarde.

#### Le chat général

Lorsque vous êtes authentifié vous accéder à la page de la discussion instantané général. A partir de cette page vous avez la liste des utilisateurs enregistrer permettant d’accéder à la discussion instantanée privée. (Figure 11)

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

Description générée automatiquement

Figure 11 Chat général

Sur cette fenêtre nous pouvons voir que les messages sont disposés en fonction de l’utilisateur connecté. Les messages envoyés par l’utilisateur connecté sont disposés sur le 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.

Également il y a la présence de la liste des utilisateurs enregistrés permettant l’accès à la fenêtre de discussion privé entre l’utilisateur connecter et l’utilisateur choisit dans la liste.

Au sommet de la fenêtre de discussion, il y a la présence du nom de l’utilisateur connecté.

La figure 12 nous 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, Icône d’ordinateur

Description générée automatiquement

Figure 12 Chat privé

Il y a la présence d’un bouton permettant de revenir sur le chat général.

#### L’envoie de message



Figure 13 Envoie de message

Ce formulaire (Figure 13) permet d’envoyer le message souhaité avec la possibilité de joindre une image. Le message et le lien avec l’image potentiel est enregistré en BDD.

### Le controller

En suivant la méthode MCD il faut obligatoirement passer par un Controller. Celui-ci est appelé par la route qui utilise également la fonction créée dans le Controller (figure 14).

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

Description générée automatiquement

Figure 14 ChatController

### Le Model

Pour finir, il y a aussi la présence du Model qui sert de lien entre le Controller et la Base De Données (figure 15).

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 15 Message Model

## Les 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 permet d’avoir accès à la vue de création et la seconde de sauvegarder les informations fournit en BDD.

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

Description générée automatiquement

Figure 16 Route register

#### Le controller

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 17 Create account

Nous pouvons voir (Figure 17) la présence des différentes fonctions nécessaire à la création d’un compte. Les différentes règles de validation sont également présentes.

#### Le model

Grace à CodeIgniter, il suffit de renseigner seulement les informations essentielles qui pourront être utilisé plus tard en effectuant toutes modifications ou actions voulues.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 18 Model

#### Les vues

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

Description générée automatiquement

Figure 19 Vue Register

La figure 19 montre la vue de la création de compte. Celle-ci est faites via Twig avec la possibilité d’afficher les erreurs s’il y en a présente.

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

Description générée automatiquement

Figure 20 Vue LogIn

La figure 20 montre la vue qui permet de s’authentifier au chat.

## Les Messages

### 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 21 montre la route utilisée quand l’utilisateur crée message.

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

Description générée automatiquement

Figure 21 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 par notre route, elle utilise la fonction « postMessage » se trouvant dans le controller (figure 22).

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Figure 22 fonction postMessage

Grace à cette seule fonction, nous pouvons enregistrer en BDD les messages postés sur le chat général, le chat privé et également si une image est jointe ou non. Ces 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. Nous l’avons déjà vu auparavant (figure 15). Grâce à CodeIgniter 4 nous pouvons directement utiliser le Model dans la Controller en y créant une instance et en utilisant les fonctions présentent dans CodeIgniter4. Grâce aux champs indiquer dans le Model l’utilisation de ceux-ci est permise.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 23 Fonction postMessage

Cette fonction, 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 renseigner dans le Model avec les valeurs souhaitées.

#### La vue

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 24 Vue PostMessage

Pour la création de message, la vue (figure 24), indique que le champs « texte » ne peux accepter que 500 caractères maximum et que le champ « file » n’accepte que les extensions « .jpg » « .jpeg » et « .png ».

#### L’envoie du formulaire

Le formulaire est envoyé via le script JavaScript. (Figure 25)

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

Description générée automatiquement

Figure 25 Script ajoutMessage

Dans cette portion du script nous pouvons voir comment le message est envoyé 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) {». Nous désactivons l’action d’envoie du formulaire et l’activons de nouveau pour qu’elle exécute la fonction définit ensuite.

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

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 lié 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 26)

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

Description générée automatiquement

Figure 26 requête ajax

Cette fonction utilise le 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 affichages des fichiers joints

Le choix à était fait que les fichiers doivent être enregistrer dans le dossier writable/chatFiles/mm/yyyy.

#### La fonction d’upload

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 27 Fonction Upload

Cette fonction 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 28 fonction display

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

Cette fonction (figure 28) utilise une instance de FileModel() qui peut utiliser les fonction définit dans le Model. L’adresse où les fichiers sont affichés sera utilisé pour l’affichage des message avec un fichier joint.

### Affichage des messages

#### Les routes

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

Description générée automatiquement

Figure 29 Routes de l'affichage des messages

Ces routes (figure 29), sont utilisés dans les diverses scenarii.

#### Le controller

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

Description générée automatiquement

Figure 30 Fonction GetMessage

Cette fonction (figure 30) est également appelée via une requête ajax. (figure 31)

Une image contenant capture d’écran, texte

Description générée automatiquement

Figure 31 Requête ajax

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

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

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

#### Le model

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

Description générée automatiquement

Figure 32 Fonction getMessage

Cette fonction (figure 32) permet de récupérer les messages présents en BDD.

Les clause where() sont différentes selon ou se situe l’utilisateur et des allias sont utilisé pour différencier les champs qui ont un nom similaire sur différentes tables utilisées.

## Tests effectués

Pour effectuer les différents tests j’ai fait appel à des personnes extérieures que je souhaite grandement remercier, **BRUNDU Roseline, TORRENTI Natacha et ROBERT 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’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 qui m’avaient échappé car je connaissais le code.

## Future évolution

Ce projet continuera d’évoluer durant la deuxième partie du Fil Rouge avec l’implémentation de nouvelles fonctionnalités, qui me sont, pour l’heure, inconnues.

Si ce n’est pas prévu, je souhaite mettre en place les services liés aux e-mails (mot de passe oublié, notification etc…).

## Conclusion

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

Il m’a aussi permis de mieux appréhender les différentes problématiques que l’on pouvait rencontrer. Il me permettra de mieux appréhender la seconde partie du projet fil rouge en ayant des bases solides.

Il est également possible que des modifications futures y soient apportées. Ces modifications peuvent porter sur la technique mais aussi sur le design.