**Cédric CHIMOT**

****

Dossier de Stage

Création d’un site E-commerce, mise en place d’un système de notation client avec javascript, montée en compétence sur Symfony 6 et insertion d’une fonction de pagination

Développeur Web & Web Mobile Session 2022

Table des matières

[**Présentation Personnelle** 4](#_Toc123247739)

[Remerciements 4](#_Toc123247740)

[Je tiens à remercier Tahar AZZAOUI, pour la qualité de la formation, ainsi que son soutien durant celle-ci. 4](#_Toc123247741)

[Introducing yourself & Why this formation 5](#_Toc123247742)

[Se Présenter & Pourquoi cette formation 6](#_Toc123247743)

[**Présentation de la Société** 7](#_Toc123247744)

[Présentation INSY2S 7](#_Toc123247745)

[Présentation de l’équipe 8](#_Toc123247746)

[**Présentation du Projet fil rouge, ‘Villagegreen’** 9](#_Toc123247747)

[*Expression des besoins* 9](#_Toc123247748)

[*Mise en place du cahier des charges* 9](#_Toc123247749)

[*Fonctionnalités* 10](#_Toc123247750)

[**Montée en compétence Symfony 6** 11](#_Toc123247751)

[*Expression des besoins* 11](#_Toc123247752)

[*Méthode de travail* 11](#_Toc123247753)

[*Fonctionnalités* 11](#_Toc123247754)

[**Présentation du Projet ‘REPART’** 12](#_Toc123247755)

[*Expression des besoins* 12](#_Toc123247756)

[*Fonctionnalité* 12](#_Toc123247757)

[*Méthode de travail* 12](#_Toc123247758)

[**Environnement Technique** 13](#_Toc123247759)

[*Choix des technologies* 13](#_Toc123247760)

[- Looping 13](#_Toc123247761)

[- GitHub 13](#_Toc123247762)

[**Réalisation du projet ‘Villagegreen’** 14](#_Toc123247763)

[*Création du MCD* 14](#_Toc123247764)

[*Script de la BDD* 16](#_Toc123247765)

[Élaboration du dictionnaire de Données 18](#_Toc123247766)

[*Création du projet* 19](#_Toc123247767)

[*Création de la base de Données* 20](#_Toc123247768)

[*Création du Dashboard* 20](#_Toc123247769)

[*Création de la Page d’accueil* 24](#_Toc123247770)

[**Fonctionnalités du projet ‘Villagegreen’** 25](#_Toc123247771)

[*Gestion des commandes (CRUD)* 25](#_Toc123247772)

[*Commentaires clients sur les commandes* 28](#_Toc123247773)

[**Symfony 6 et projet REPART** 32](#_Toc123247774)

[*Les formulaires* 32](#_Toc123247775)

[*Les ‘Datas Fixtures’* 33](#_Toc123247776)

[*Fonction de pagination sur les tickets* 34](#_Toc123247777)

[**Bilan** 37](#_Toc123247780)

# **Présentation Personnelle**

## Remerciements

## Je tiens à remercier Tahar AZZAOUI, pour la qualité de la formation, ainsi que son soutien durant celle-ci.

Je remercie également Youssef CHAREF de m’avoir accepté en tant que stagiaire dans son entreprise.

Je tiens enfin à remercier mon groupe de formation pour leur soutien, l’entraide, la bonne ambiance et l’esprit d’équipe qui régnait.

Enfin, je tiens à remercier les membres du Jury qui m’ont écouté lors de mon passage pour ma soutenance.

## Introducing yourself & Why this formation

Having followed administrative studies and after a long period without activity, I decided on a professional retraining. So, I took the Web and Mobile Web Developer training course at the AFPA center in Rousies.

During my training, I had to do an internship at the company Insy2s in Lille. During this internship, I was able to carry out the design of my personal thread project, I was able to improve my skills on the Symfony Framework and by integrating a group of trainees following the same training as me on Roubaix, I participated in the REPART project.

My personal thread project is an e-commerce site selling music equipment. I implemented a shopping cart there; orders can be placed.

The REPART project is a repair management application for computer & telephone equipment. The purpose of this is to facilitate the work of the employee by having a follow-up of the requests for interventions by means of tickets.

For the personal thread project, according to the specifications, I made it possible to follow orders according to their status, I also created a rating system with a JavaScript function so that customers can give their opinion. I made sure that this system is only accessible if the order has a certain status.

In order to better understand how the REPART project was designed, I chose to increase my skills on the Symfony 6 Framework by following the official documentation as well as a series of explanatory videos on the different functionalities. Finally using the Bootstrap library, I implemented a ticket-level pagination system on the REPART project.

This internship allowed me to apply the skills I acquired during my training but also to better understand the functioning of a Framework like Symfony.

## Se Présenter & Pourquoi cette formation

D’une formation administrative et après une longue période sans activité, j’ai décidé d’une reconversion professionnelle. J’ai donc suivi la formation de Développeur Web et Web mobile au centre AFPA de Rousies.

Durant ma formation, j’ai été amené à effectuer un stage au sein de la société Insy2s à Lille. Au cours de ce stage, j’ai pu mener à bien la conception de mon projet fil rouge personnel, j’ai pu monter en compétence sur le Framework Symfony et en intégrant un groupe de stagiaire suivant la même formation que moi à Roubaix, j’ai participé au projet REPART.

Mon fil rouge est un site E-commerce de vente de matériel de musique. J’y ai implémenté un panier, des commandes peuvent être passées.

Le projet REPART est une application de gestion de réparation de matériels informatiques & téléphoniques. Celle-ci a pour but de faciliter le travail du salarié en ayant un suivi des demandes d’interventions par le biais de tickets.

Pour le projet fil rouge, suivant le cahier des charges, j’ai fait en sorte que l’on puisse suivre les commandes selon leur statut, j’ai également créé un système de notation avec une fonction javascript afin que les clients donnent leur avis. J’ai fait en sorte que ce système ne soit accessible que si la commande possède un certain statut.

Afin de mieux comprendre comment le projet REPART a été conçu, j’ai choisi de monter en compétence sur le Framework Symfony 6 en suivant la documentation officielle ainsi qu’une série de vidéos explicatives sur les différentes fonctionnalités. Enfin à l’aide de la bibliothèque Bootstrap, j’ai implémenté un système de pagination au niveau des tickets sur le projet REPART.

Ce stage m’a permis de mettre en application les compétences que j’ai acquises durant ma formation mais aussi de mieux appréhender le fonctionnement d’un Framework comme Symfony.

# **Présentation de la Société**

## Présentation INSY2S

INSY2S est une entreprise spécialisée dans le secteur d’activité du conseil en système et logiciels informatiques. La société est active depuis 2014, elle a été créée pour répondre aux besoins des entreprises dans le domaine du digital.



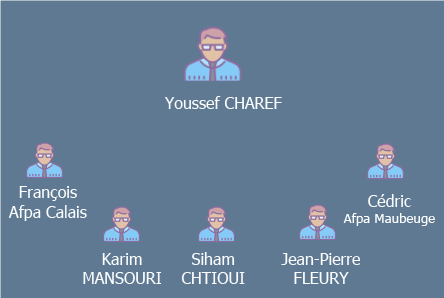
Les produits proposés par INSY2S sont regroupés en 3 catégories :

* Workstation : Permet une gestion globale de l’activité, un traitement rapide des applications métier en limitant les erreurs.
* Portail d’entreprise : Permet de centraliser et de partager en temps réel le flux d’informations dans un contexte multi site et pour une meilleure gestion des ressources sur le réseau.
* Application mobile : Permet de gérer l’activité à distance, de rester connecté et de gagner en mobilité et en agilité. Quelques entreprises qui ont fait confiance à INSY2S :



## Présentation de l’équipe

Pour le projet REPART, l’équipe était formée de 5 personnes, 3 venant de l’AFPA de Roubaix puis une autre personne de l’AFPA de Calais et moi-même de Maubeuge sont arrivées en cours de projet, tout cela sous la direction du chef de projet Mr CHAREF Youssef.



# **Présentation du Projet fil rouge, ‘Villagegreen’**

## *Expression des besoins*

Le projet est un site e-commerce de vente de matériel de musique (instruments, accessoires etc…). Il doit comporter un panier avec la possibilité de passer des commandes.

L’administrateur devra pouvoir gérer le site complètement, aussi bien la partie client que la partie commandes.

Le client aura la possibilité de pouvoir trouver des produits selon les différentes catégories proposées.

## *Mise en place du cahier des charges*

Selon le cahier des charges, un administrateur doit pouvoir accéder à toutes les commandes selon leur statut (en retard, annulée etc…) et en gérer la modification. Il doit aussi avoir la possibilité d’ajouter de nouveaux produits, de modifier ou de supprimer ceux existants.

Le client pourra ajouter des produits dans le panier et passer des commandes. Il pourra également donner une note à son expérience sur le site par l’intermédiaire d’un formulaire seulement si la commande possède un statut particulier.

Seul un client possédant un compte et étant connecté aura accès à toutes les fonctionnalités du site, comme ajouter des produits au panier ou voir le récapitulatif des ses commandes effectuées par exemple.

## *Fonctionnalités*

Concernant la gestion du site, l’objectif du projet est de créer un tableau de bord qui pourra permettre aux administrateurs de gérer les produits, les commandes, les comptes d’utilisateurs et son propre compte admin.

*Droits Admin/Clients :*

**Côté Admin :**

* Gestion des clients : L’administrateur pourra supprimer des clients (D).
* Gestion des Admins : L’administrateur pourra créer, lire, modifier et supprimer son compte d’amin (CRUD).
* Gestion des Produits : L’administrateur pourra créer, lire, modifier et supprimer des produits (CRUD).
* Gestion des Commandes : L’administrateur pourra créer, lire, modifier et supprimer des commandes (CRUD).

**Côté Client :**

* Panier : Le client pourra ajouter des produits au panier, supprimer tout ou partie de celui-ci, modifier la quantité des produits déjà ajoutés (CRUD).
* Evaluation : Si une commande possède le statut ‘terminée’, le client aura la possibilité de laisser une évaluation sous forme d’étoiles à attribuer de 0 à 5 (CRD).
* Gestion de son compte : Le client pourra créer et modifier son compte client (CRU).
* Message : Le client aura la possibilité d’envoyer un message (C).

# **Montée en compétence Symfony 6**

## *Expression des besoins*

Pour pouvoir comprendre le fonctionnement du projet REPART conçu par les stagiaires de Roubaix, j’ai dû commencer par monter en compétences sur le Framework Symfony 6 afin d’en apprendre plus sur son fonctionnement et sur ce qu’il pouvait me proposer.

## *Méthode de travail*

Pour en apprendre d’avantage, je me suis appuyé sur une série de vidéos proposant une revue complète des différentes fonctionnalités proposées par Symfony 6.

J’ai aussi gardé en soutient la documentation officielle puisque les vidéos en question se basaient également sur celle-ci, ainsi que l’utilisation de Template Bootstrap.

## *Fonctionnalités*

Durant cette formation, j’ai pu manipuler Symfony assez largement en apprenant à me servir de ses diverses fonctionnalités.

J’ai appris à créer des controlers, des entités, des formulaires, j’ai vu la gestion des rôles sur les questions de droits entre administrateurs et simples utilisateurs ainsi que la liaison avec une base de données.

# **Présentation du Projet ‘REPART’**

## *Expression des besoins*

Le client souhaite une application de gestion de réparation de matériel informatique. Elle lui permettra d’une part de gérer son matériel informatique et téléphonique dans un premier temps et d’autre part la possibilité de gérer les pannes sous la formation d’un outil de ticketing.

L’application doit aussi lui permettre d’administrer la partie « employés » avec la possibilité d’attribuer à chaque salarié un rôle, soit administrateur soit utilisateur. Le client souhaite une gestion des Droits d’accès à l’application en fonction du profil.

Cette application ciblera aussi bien les clients professionnels que les clients particuliers. Elle sera amenée à évoluer sur la nature des réparations proposées (par ex : Réparation de consoles de jeux vidéo).

## *Fonctionnalité*

Ayant rejoint le projet en cours alors que les autres membres du groupe finissaient leur stage, je me suis concentré sur une fonctionnalité à développer pour améliorer la présentation des tickets.

J’ai donc réalisé un système de pagination à l’aide de Bootstrap afin d’afficher un minimum de tickets par page lorsqu’il y en a un certain nombre en cours de traitement.

## *Méthode de travail*

J’ai utilisé la pagination de Bootstrap que j’ai paramétrée à l’aide d’une fonction dans le controller des tickets.

# **Environnement Technique**

## *Choix des technologies*

Pour des contraintes de temps, le choix s’est porté sur des technologies sur lesquelles j’ai travaillé pendant la formation.

* Pour la partie Frontend :
  + Visual Studio Code (Éditeur de code)
  + HTML5 & CSS3
  + Framework Bootstrap
  + JavaScript
* Pour la partie Backend & Base de Données :
  + PHP 8
  + PhpMyAdmin (Via Xampp)
* Pour les librairies :
  + Font Awesome
  + Bootstrap

## 

J’ai également utilisé d’autres outils afin de gérer au mieux la conception de mon projet :

### Looping

Looping est un logiciel de modélisation conceptuelle de données entièrement gratuit et libre d'utilisation.

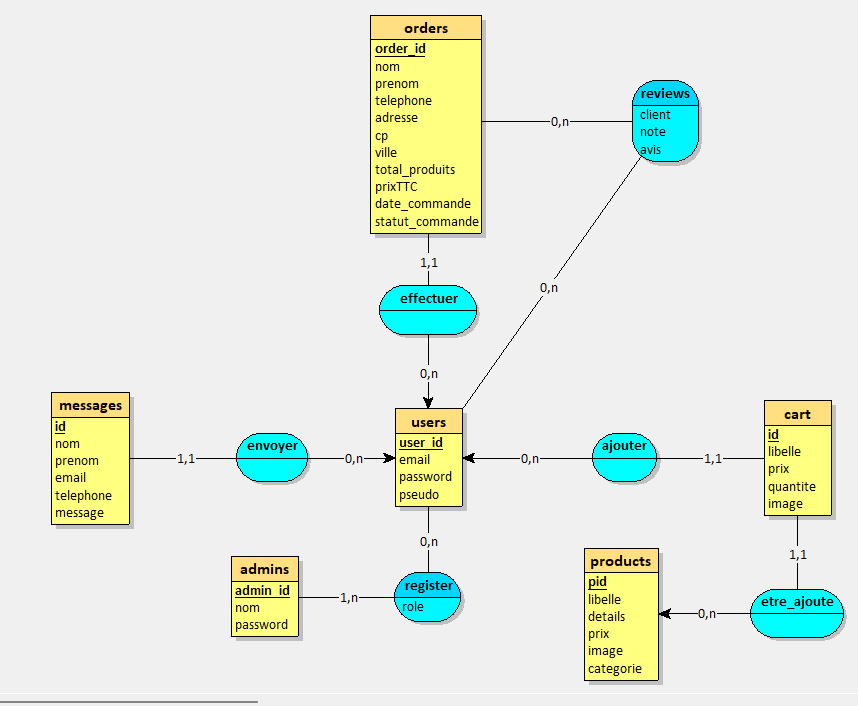
### GitHub

Service Web d’hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

# **Réalisation du projet ‘Villagegreen’**

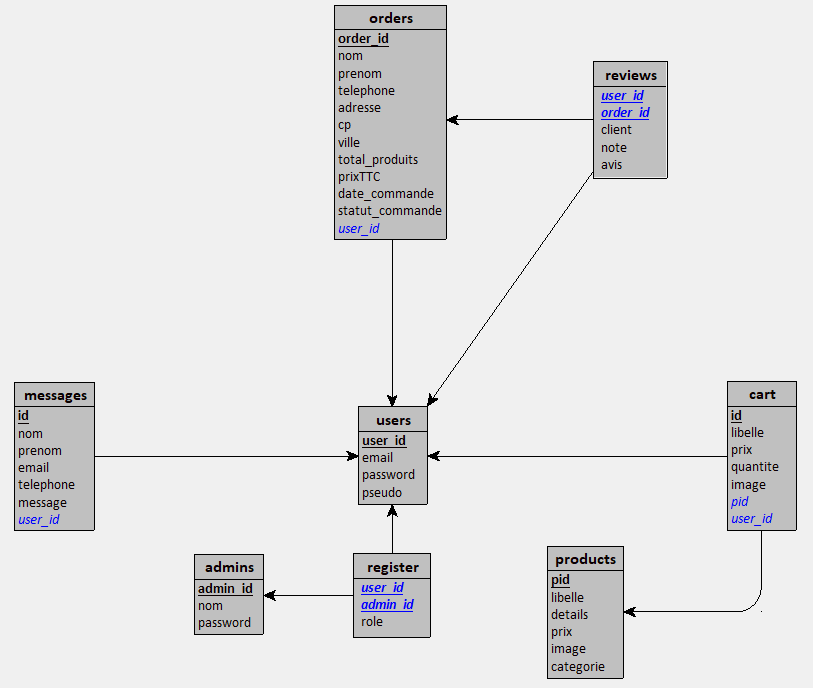
Je vais maintenant développer les différentes étapes de la conception de mon projet fil rouge.

## *Création du MCD*

A l’aide du logiciel de modélisation conceptuelle de données ‘Looping’, j’ai commencé par réaliser le MCD (Modèle Conceptuel de Données) en suivant les demandes du cahier des charges.

Dans ce MCD, on peut voir des Entités contenant chacune des attributs. Entre chaque Entité, on retrouve des relations ou associations (Verbe) qui expliquent et précisent comment les Entités sont reliées entre elles (Les ovales avec leurs « pattes » qui se rattachent aux Entités). Enfin, des cardinalités qui sont les petits chiffres au-dessus des « pattes ». Le MCD a été modifié plusieurs fois afin d’intégrer au mieux les différentes fonctionnalités du site, notamment les reviews.

Du MCD, toujours grâce à Looping, j’ai pu élaborer le MLD, qui est le Modèle Logique de Données. Le MLD transforme les entités en tables avec ses attributs et transformes les associations en clés primaire et clés étrangères. Comme dans l’exemple ci-dessous.



Pour terminer avec Looping, j’ai transformé le MLD en MPD (Modèle Physique de Données). Ce modèle est la représentation linéaire du MLD. On peut à partir du MPD générer un script qui permettra d’importer la base de Données.

## *Script de la BDD*

CREATE TABLE users (

user\_id INT,

email VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

pseudo VARCHAR(20) NOT NULL,

PRIMARY KEY(user\_id)

);

CREATE TABLE products(

pid INT,

libelle VARCHAR(100) NOT NULL,

details VARCHAR(500) NOT NULL,

prix INT NOT NULL,

image VARCHAR(100) NOT NULL,

categorie VARCHAR(100) NOT NULL,

PRIMARY KEY(pid)

);

CREATE TABLE orders(

order\_id INT,

nom VARCHAR(20) NOT NULL,

prenom VARCHAR(50) NOT NULL,

telephone VARCHAR(10) NOT NULL,

adresse VARCHAR(50) NOT NULL,

cp VARCHAR(50) NOT NULL,

ville VARCHAR(50) NOT NULL,

total\_produits VARCHAR(1000) NOT NULL,

prixTTC INT NOT NULL,

date\_commande DATE NOT NULL,

statut\_commande VARCHAR(20) NOT NULL en attente,

user\_id INT NOT NULL,

PRIMARY KEY(order\_id),

FOREIGN KEY(user\_id) REFERENCES users(user\_id)

);

CREATE TABLE admins(

admin\_id INT,

nom VARCHAR(20) NOT NULL,

password VARCHAR(50) NOT NULL,

PRIMARY KEY(admin\_id)

);

CREATE TABLE cart(

id INT,

libelle VARCHAR(50) NOT NULL,

prix INT NOT NULL,

quantite VARCHAR(50) NOT NULL,

image VARCHAR(1000) NOT NULL,

pid INT NOT NULL,

user\_id INT NOT NULL,

PRIMARY KEY(id),

FOREIGN KEY(pid) REFERENCES products(pid),

FOREIGN KEY(user\_id) REFERENCES users(user\_id) );

CREATE TABLE messages(

id INT,

nom VARCHAR(100) NOT NULL,

prenom VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

telephone VARCHAR(12) NOT NULL,

message VARCHAR(500) NOT NULL,

user\_id INT DEFAULT NULL,

PRIMARY KEY(id),

FOREIGN KEY(user\_id) REFERENCES users(user\_id)

);

CREATE TABLE register(

user\_id INT,

admin\_id INT,

role VARCHAR(50) NOT NULL,

PRIMARY KEY(user\_id, admin\_id),

FOREIGN KEY(user\_id) REFERENCES users(user\_id),

FOREIGN KEY(admin\_id) REFERENCES admins(admin\_id)

);

CREATE TABLE reviews(

user\_id INT,

order\_id INT,

client VARCHAR(50) NOT NULL,

note INT NOT NULL,

avis VARCHAR(50) NOT NULL,

PRIMARY KEY(user\_id, order\_id),

FOREIGN KEY(user\_id) REFERENCES users(user\_id),

FOREIGN KEY(order\_id) REFERENCES orders(order\_id)

);

## Élaboration du dictionnaire de Données

Une image contenant table

Description générée automatiquementPour rester dans un vocabulaire conceptuel il a fallu créer un dictionnaire de données, celui-ci regroupe toutes les données et le vocabulaire commun qui figureront dans le projet et le MCD.

## *Création du projet*

Pour la création du projet, j’ai utilisé l’éditeur de code Visual Studio Code. J’ai créé un dossier au nom du projet et j’ai commencé à mettre en place son arborescence.

Une image contenant texte, écran, capture d’écran

Description générée automatiquementUne image contenant texte

Description générée automatiquement

* Le dossier ‘Admin’ contiendra tous les fichiers liés au compte d’administrateur ;
* Le dossier ‘Assets’ contiendra les fichiers de support, images, CSS, Javascript ;
* Le dossier ‘Components’ contiendra les fichiers concernant les composants : le header et le footer implémentés sur les pages du site ou le Dashboard, le logout, le fichier de connexion à la base de données ;
* Les fichiers concernant les utilisateurs sont ajoutés à la racine du dossier de projet.

## *Création de la base de Données*

Pour créer la base de données dans phpMyAdmin, je me suis servi du script que j’ai généré à partir du MLD conçu sur Looping.

Pour connecter mes fichiers à la BDD, j’ai créé un fichier ‘connect.php’ qui contient toutes les informations de connexion, comme le nom d’utilisateur et le mot de passe, en voici une image :

Une image contenant texte

Description générée automatiquement

On commence par créer une variable pour rechercher la BDD, puis un nom d’utilisateur et un mot de passe créés au préalable sur phpMyAdmin. On termine par la variable de connexion en créant un nouveau PDO(PHP Data Object) qui liera la BDD à l’ensemble des fichiers du dossier.

## *Création du Dashboard*

Ensuite, vient la préparation du Dashboard administrateur sur lequel seront affichés les liens vers les différentes pages de gestion du site. Dans le cahier des charges il est par exemple demandé de pouvoir gérer les commandes selon leur statut.

Pour commencer, j’ai créé un header qui sera inclus sur toutes les pages liées au tableau de bord. Il donnera la possibilité d’accéder à toutes les informations du site (commandes, messages, avis etc…), il permettra aussi d’afficher le nom de l’admin connecté.

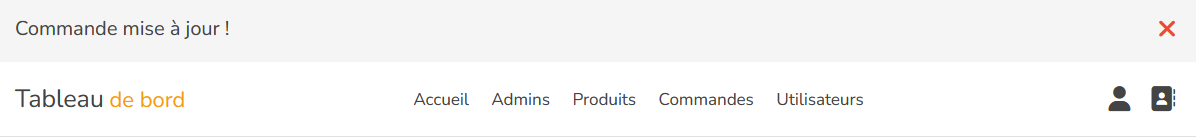
On commence par paramétrer une variable ‘message ‘ qui affichera les différentes confirmations liées aux actions effectuer par l’admin (suppression, modification etc…). Voici une image de la variable en question :

Une image contenant texte

Description générée automatiquement

La variable est incluse dans une boucle précisant qu’à chaque appel de celle-ci le message s’affiche en haut de page et donne la possibilité de l’effacer en cliquant sur un bouton paramétrer avec une option ‘onClick’. L’élément, lui, provient de la librairie FontAwesome.

Une image contenant texte

Description générée automatiquementUne fois le message paramétrer, on peut créer le header. Ci-dessous, une partie du code et une image du rendu du header avec le message de confirmation :

La petite croix rouge en haut à droite du message est l’élément qui permet de supprimer le message une fois lu.

Dans la navbar du tableau de bord, on retrouve des liens cliquables qui renvoient vers les différentes pages. Le design du header est réalisé avec du CSS3.

Une fois le header terminé, on peut passer à la création de notre Dashboard. Celui-ci devra regrouper toutes les informations accessibles à l’administrateur connecté, le cahier des charges précisait que l’on devait pouvoir gérer les commandes et les afficher selon divers statuts.

Pour commencer, on inclut le lien de connexion à la BDD et on lance la session. Celle-ci est paramétrée de manière que seul un admin connecté puisse accéder au Dashboard. En outre pour une question de sécurité, on fera en sorte qu’une fois logout, on ne pourra pas revenir sur le site ou le tableau de bord comme ci l’on était encore connecté.

Ci-dessous le logout :

Une image contenant texte

Description générée automatiquement

Il y aura également un formulaire de login et d’inscription accessible à tous les admins, un admin par défaut est également enregistré :



Un extrait du code du Dashboard, avec au début la condition ‘IF’ signifiant que seul l’admin connecté peut y accéder sinon il est envoyé vers le login.

Une image contenant texte

Description générée automatiquement

Enfin les options d’affichage des commandes. Lorsque l’on va chercher la tale ‘orders’ on précise avec une requête que l’on veut que le lien nous renvoie seulement sur les commandes ‘en attente’.

Une image contenant texte

Description générée automatiquement

Le rendu du Dashboard dans sa version finale :



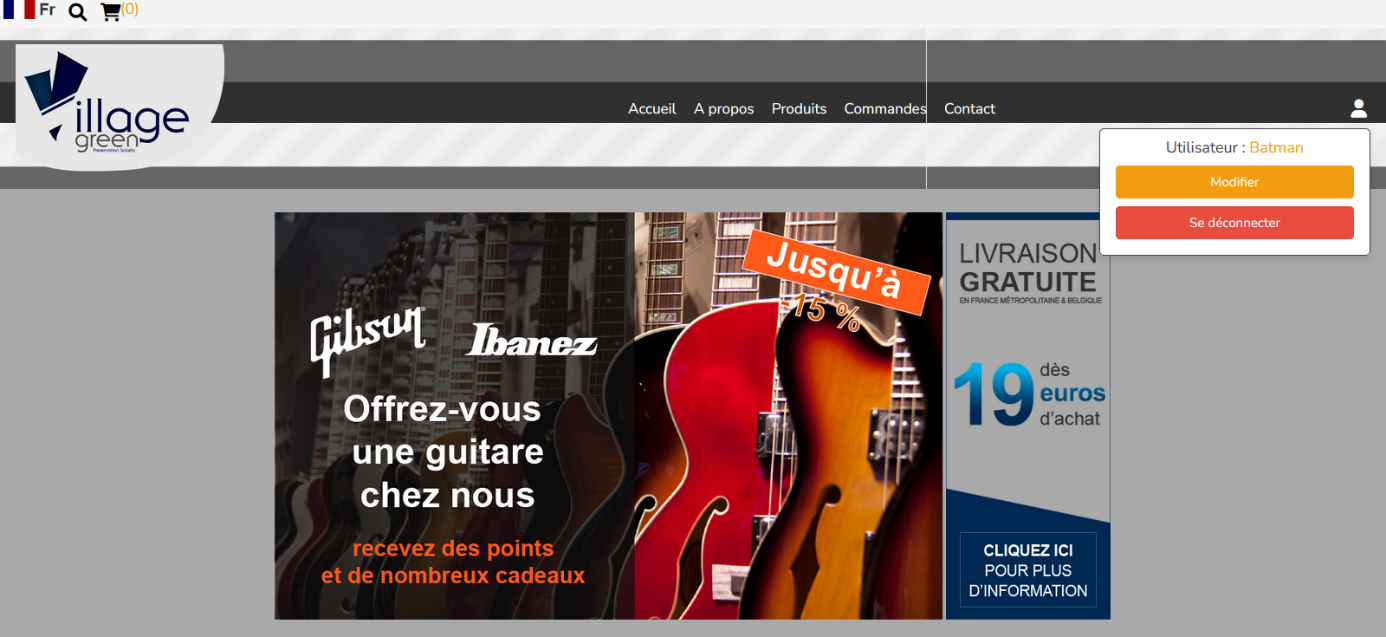
## *Création de la Page d’accueil*

Une fois les pages administrateur créées, on peut s’attaquer à la création du site lui-même. De la même manière que le Dashboard, on va d’abord créer un header commun qui sera inclus sur chaque page du site. Il comprendra une barre de navigation ainsi qu’un panier et une barre de recherche.

Le site reste accessible dans sa globalité cependant, seul un utilisateur connecté et/ou inscrit aura la possibilité d’afficher certaines informations (commandes passées…) ou encore d’ajouter des produits au panier.

Cette fonction est paramétrée de la même façon que le Dashboard, avec les sessions, on vérifie si l’utilisateur est connecté, s’il ne l’est pas on renvoie sur le login.

Ci-dessous, la version finale de la page d’accueil :





Chaque image représentant les catégories de produits renverra sur une page ne comprenant que des produits de la catégorie concernée. L’icône d’utilisateur affichera le nom du User connecté ainsi que la possibilité de modifier son mot de passe et de se déconnecter du site.

# **Fonctionnalités du projet ‘Villagegreen’**

## *Gestion des commandes (CRUD)*

Suivant le cahier des charges, il était demandé de pouvoir assurer une gestion des commandes suivant leur statut. Pour ce faire, j’ai décidé de créer un CRUD (Create, Read, Update, Delete).

La partie Create se fait du côté du client lorsque celui-ci valide une commande. Le Read consistera à afficher les données des commandes sur une page, la partie modification Update ainsi que la suppression Delete seront liés à l’ID d’une commande en particulier pour que ne soit mis à jour que la commande concernée.

Dans le Dashboard, on pourra voir les commandes selon leur statut, ci-dessous par exemple la page recensant les commandes en attente :

Une image contenant texte

Description générée automatiquement

On commence par sélectionner la table concernant les commandes dans la BDD. A l’aide d’une requête, on se concentre sur le statut ‘en attente’.

Ensuite, on va gérer l’affichage des commandes.

Une image contenant texte

Description générée automatiquement

Associer à la variable créée précédemment, le ‘rowCount’ va aller chercher le nombre de lignes dans la table ‘orders’ où le statut de la commande est ‘en attente’.

Avec la fonction ‘while’, on dit que tant qu’il y a une commande au statut ‘en attente’ dans la table, on affichera les données correspondantes.

Le ‘fetch’ va créer une association avec la table grâce au PDO. Par ce moyen, on va afficher les données contenues dans les colonnes. ‘$fetch\_orders’ est la variable d’association, ‘[‘date\_commande’]’ par exemple représente une colonne de la table ‘orders’.

Dans le paragraphe ‘statut\_commande’, j’ai créé une condition sur l’affichage du statut, en précisant la couleur correspondante. Les commandes ‘en attente’ par exemple sont affichées en bleu.

Une image contenant texte

Description générée automatiquement

On ajoute enfin un formulaire qui nous servira plus tard avec la fonction Update. Dans celui-ci, on insère une liste déroulante pour afficher le statut de la commande.

Il y a également deux boutons pour gérer la mise à jour mais aussi pour supprimer la commande.

L’input caché sera là pour utiliser les options du CRUD afin de cibler l’ID d’une commande en particulier.

Pour la fonction ‘Update’, on commence par aller chercher le bouton dans le formulaire créé précédemment.

Une image contenant texte

Description générée automatiquement

Avec une condition, on applique la fonction au moment de l’appui du bouton ‘Modifier’, ‘$\_POST[‘update\_statut’]’ va chercher le ‘Name’ du bouton. On utilise ensuite l’ID et le statut de la commande pour créer une requête.

On prend la table ‘orders’ et on précise que l’on veut modifier le statut de la commande correspondant à un ID précis.

Lorsque la requête est exécutée correctement un message de confirmation s’affiche en haut de l’écran.

On paramètre également une fonction ‘delete’ liée elle au bouton ‘Supprimer’ du formulaire en allant également chercher l’ID de la commande correspondante.

Une image contenant texte

Description générée automatiquement

Un fois la commande supprimée, on renvoie sur la page où sont affichées toutes les commandes.

Ci-dessous un exemple d’affichage d’une commande :



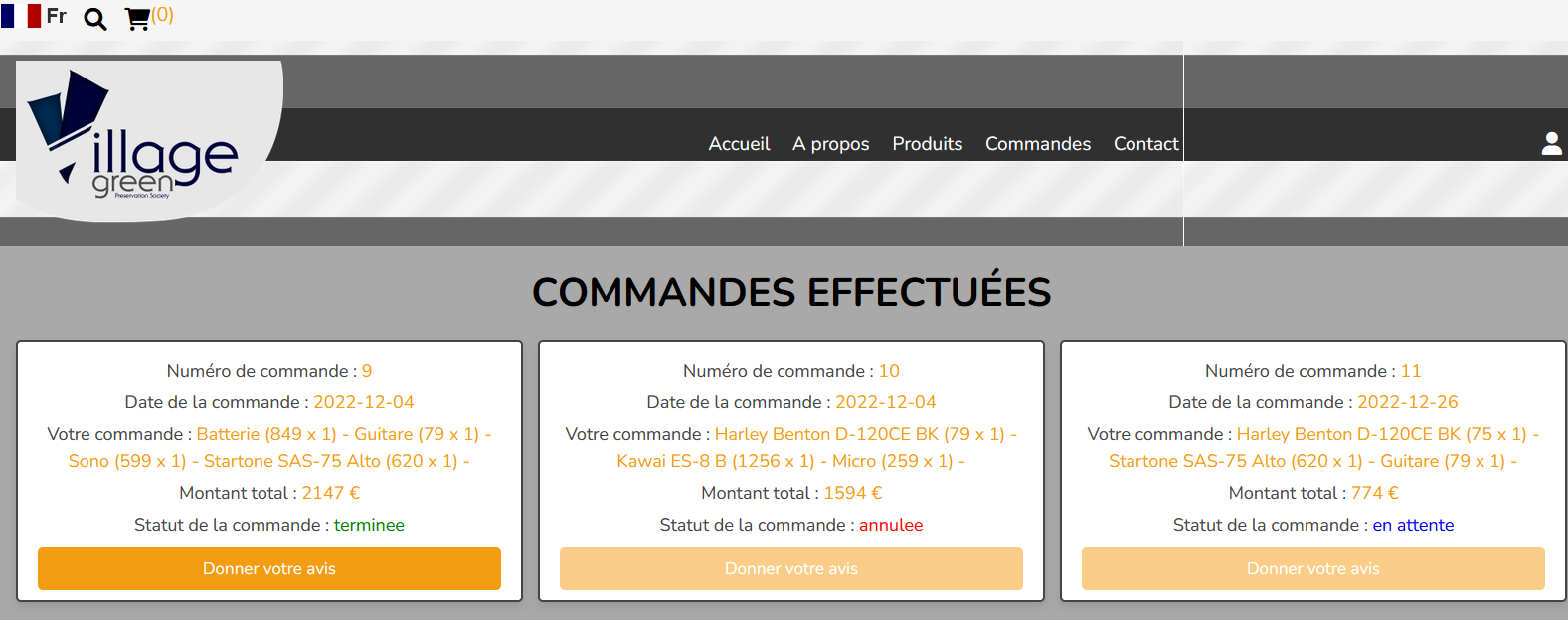
La mise en page est là aussi gérée par le CSS. Le header est inclus comme sur chacune des pages du Dashboard.

## *Commentaires clients sur les commandes*

L’idée avec cette fonction était de permettre aux clients de laisser leur avis sur leur expérience sur le site à la suite d’une commande.

L’utilisateur devra bien entendu être connecté au site s’il veut accéder au récapitulatif de ses commandes et laisser une appréciation.

Ci-dessous l’affichage du récapitulatif des commandes :



On commencera donc par créer un formulaire. Il est accessible depuis le récapitulatif des commandes en appuyant sur le bouton ‘Donner votre avis’. Ce bouton est paramétré de façon qu’un commentaire ne puisse être donné que si la commande est complètement terminée.

On peut remarquer que le bouton d’accès au formulaire est bloqué si le statut de la commande est autre que terminé.

Le code pour l’affichage des commandes est sensiblement le même que celui dans le Dashboard, à l’exception de l’option pour modifier le statut de la commande.

L’option ‘mute’ du bouton est paramétrée de cette manière :

Une image contenant texte

Description générée automatiquement

Grâce à l’association à la table ‘orders’ dans la BDD par la variable ‘$fetch\_orders’, on vient dire, à l’aide des attributs ‘disabled’ et ‘enabled’, que suivant le statut de la commande on active ou désactive le bouton.

Si l’utilisateur possède une commande avec un statut ‘terminée’, il peut alors laisser son commentaire grâce au formulaire prévu à cet effet.

Ce formulaire comportera plusieurs inputs liés aux informations du client. La partie ‘note’ qui affiche des icônes en forme d’étoile provenant de la librairie FontAwesome, chacune liées à une valeur, est associée à une fonction Javascript que nous verrons ensuite. De même l’input caché servira également au paramétrage de la fonction.

Le code du formulaire et le rendu sur le site :

Une image contenant texte

Description générée automatiquement Une image contenant texte

Description générée automatiquement

Au moment de l’insertion en BDD, grâce à l’input ‘numéro de commande’, on vérifiera si le client n’a pas déjà ajouté un commentaire à propos de la commande en question. Si c’est le cas, un message d’erreur s’affichera.

Je vais maintenant abordé la fonction Javascript liée aux étoiles permettant d’attribuer la note.

L’idée est qu’au survol des étoiles, celles-ci changent de couleur mais aussi qu’au ‘click’ cette couleur soit fixe. Si le client appuie sur la cinquième étoile alors cinq étoiles deviendront dorées. Il pourra s’il le désire changer sa note, en appuyant sur la quatrième étoile par exemple cela désactivera l’étoile suivante.

La totalité de la fonction réagira au chargement de la page.

On va commencer par créer une constante sur les icônes et sur l’input caché à l’aide d’un ‘querySelector’. Le premier prendra en compte toutes les étoiles l’autre la valeur de l’input paramétrée à 0 par défaut :

Une image contenant texte

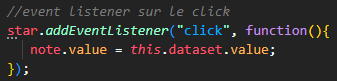
Description générée automatiquement

Il faut ensuite faire une boucle ‘for’ afin d’ajouter un écouteur sur le survol des étoiles. De base, elles seront de couleur noire et vide, au survol on fera apparaitre une étoile pleine, noire d’abord puis dorée si l’on passe à l’étoile suivante.

Une image contenant texte

Description générée automatiquement

On paramètre aussi un écouteur sur le ‘click’ pour l’attribution de la note.



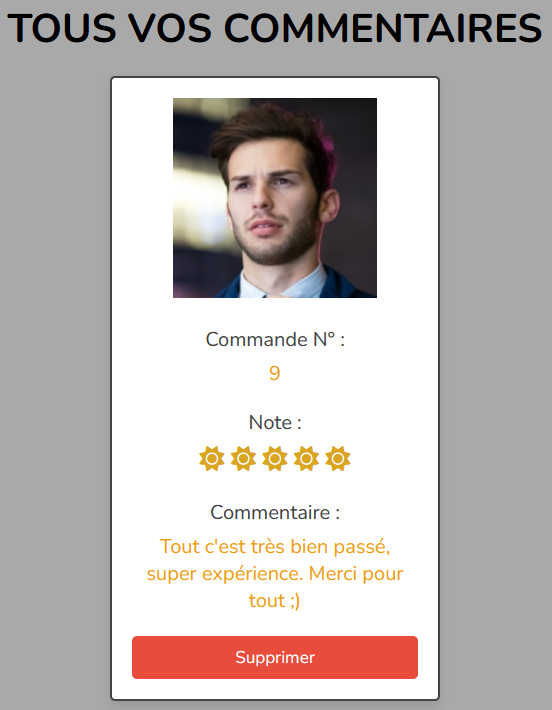
Enfin arrive la fonction ‘resetStars’. C’est elle qui permettra d’annuler les différentes actions de la fonction principale, comme remettre la couleur d’origine des étoiles lorsque la souris ne les survole et ce suivant la variation du paramètre ‘note’.

Une image contenant texte

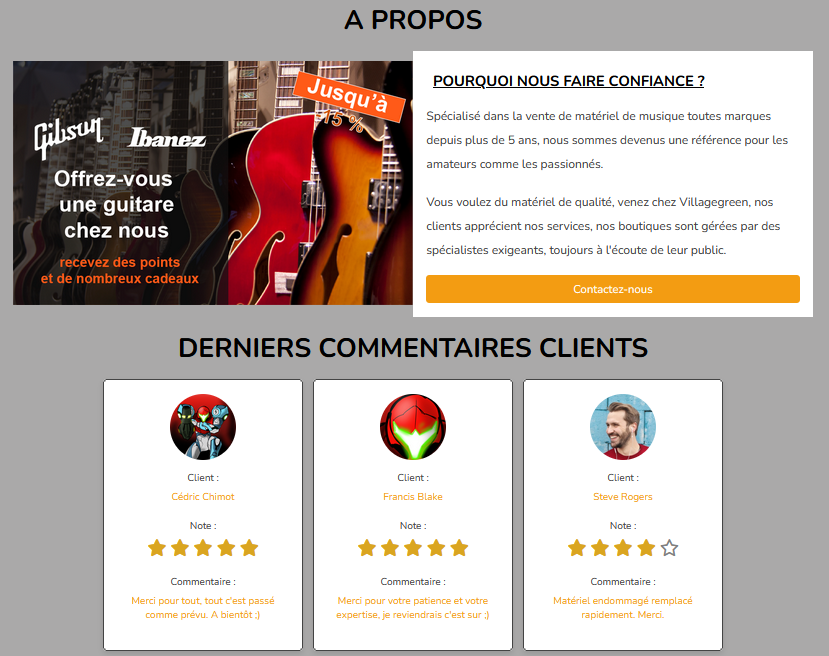
Description générée automatiquement

De base la note est égale à ‘0’, suivant l’icône sur laquelle on clique sa valeur change. Cette fonction active le changement de couleur de la note en rapport aux étoiles dans le formulaire qui ont elles aussi une valeur de base. C’est cette valeur qui détermine la note finale.

Une fois que le client à valider le formulaire, le commentaire s’affiche en dessous, il trouvera également tous les avis qu’il a déjà posté.

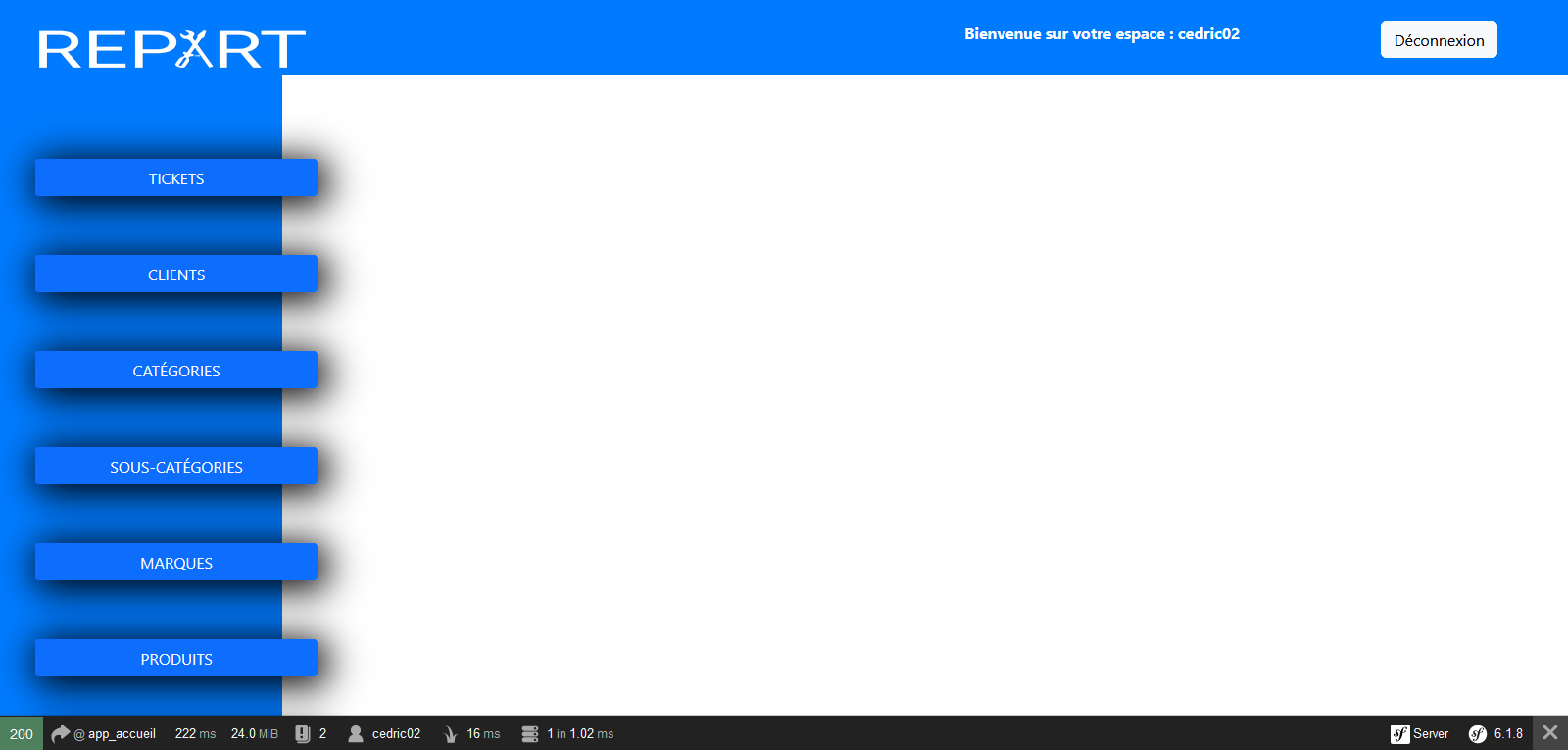


J’ai aussi ajouté une requête dans la section ‘à propos’ afin d’afficher sur la page dédiée les derniers commentaires postés par les clients.



# **Symfony 6 et projet REPART**

Le projet REPART a été conçu entièrement par le groupe de stagiaire de Roubaix que j’ai rejoint lors de mon stage.



Pour en comprendre le fonctionnement, j’ai suivi une série de vidéos en m’appuyant sur la documentation officielle pour découvrir le Framework Symfony 6. Ce tutoriel m’a permis de manipuler un certain nombre de ses fonctionnalités.

Le Template utilisé pour la mise en page du site est un Template Bootstrap gratuit que j’ai téléchargé au préalable et modifié au fur et à mesure de la montée en compétences.

## *Les formulaires*

J’ai pu voir la création des formulaires notamment le login ou l’ajout de nouveaux éléments en BDD.

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, moniteur, intérieur

Description générée automatiquement

## *Les ‘Datas Fixtures’*

Les ‘Fixtures’ sont utilisées pour charger un faux ensemble de données dans une base de données qui peut ensuite être utilisé pour des tests ou pour fournir des données intéressantes pendant que l’on développe une application. Je les ai utilisés pour afficher une liste de personne.

Ci-dessous, un exemple de fixtures pour créer des administrateurs et des utilisateurs ainsi que des mots de passe. On utilisera les données créées pour se connecter au site et pour gérer les options d’affichage avec entre autres l’attribution des rôles.

Une image contenant texte

Description générée automatiquement

Une fois paramétrer, on envoie les données en BDD en écrivant la ligne de commande suivante dans le terminal de VS Code : symfony console doctrine:fixtures:load.

## *Fonction de pagination sur les tickets*

Pour apporter ma contribution au projet ‘REPART’, j’ai créé un système de pagination au niveau des tickets afin d’améliorer la présentation des pages s’il y a de nombreux tickets générés en BDD.

J’ai commencé par créer une route dans le controller des Tickets. Je lui ai donné un nom (‘tickets.list.all’) et attribué deux paramètres qui seront important dans la fonction : la page (portant ici le numéro ‘1’) et un nombre correspondant au nombre de tickets que l’ont veut afficher sur chaque page (ici on précise 5 tickets par page maximum).

Une image contenant texte

Description générée automatiquement

Je vais à présent détailler la fonction :

* La variable $doctrine ira chercher la classe ‘Tickets’, $page correspond au numéro de la page, $nbre le nombre de tickets par page ;
* Avec $nbTicket on va compter le nombre d’élément dans l’entité Tickets ;
* $nbPage servira à calculer le nombre de page que nécessitera la présentation des tickets, on prend le nombre total de tickets que l’on divise par le nombre d’éléments voulus par page ;
* Dans la variable $ticket, on va trouver les éléments dans l’entité, récupérer le nombre et intégrer les pages ;
* La fonction ‘render’ va nous donner le rendu de chaque page avec les tickets, la pagination paramétrée sur ‘true’, le nombre de page et la page actuellement sélectionnée

De base, la pagination utilisée est un composant de la librairie Bootstrap que voici :

Une image contenant texte

Description générée automatiquement

Un bloc de liens connecté est utilisé ici pour la pagination, ils sont donc immanquables et peuvent évoluer facilement. La pagination est construite avec des éléments HTML de liste afin d’afficher le nombre de liens disponibles. On utilise un élément ‘nav’ pour qu’elle soit identifiée comme une section de navigation.

Il ne reste plus à présent qu’à afficher la pagination sur l’index des tickets. Pour cela, une condition sera utilisée pour vérifier que la pagination est définie.

Une image contenant texte

Description générée automatiquement

On implémente une autre condition de façon à faire apparaitre ou non les boutons ‘précédent’ et ‘suivant’ par rapport à la page sur laquelle on se trouve (première ou dernière page).

Chaque élément renverra sur une page sur laquelle sont affichés le nombre maximum de tickets paramétré dans le controller.

Ci-dessous le rendu final de la pagination :

# Une image contenant texte Description générée automatiquement

# On peut voir s’afficher les pages et le bouton suivant puisqu’il s’agit de la première page. Il n’y a que deux liens puisque le nombre de tickets total dans la BDD ne dépasse pas 10.

# **Bilan**

Cette formation m’a permis de découvrir un nouveau métier et m’a permis également d’apprécier cette découverte.

Le stage, lui, m’a permis de développer mes capacités assimilées durant cette formation. Que ce soit concernant l’autonomie, le travail en équipe, l’apprentissage de divers langages et la montée en compétences, lente mais régulière.

J’ai pu découvrir de nouvelles technologiques comme SYMFONY (en stage) et renforcer des connaissances apprises lors de ma formation.

J’ai eu la chance de développer un projet depuis le début en mode From Scratch, c’est-à-dire, de la découverte du cahier des charges à la conception en passant par le développement de celui-ci. Cela a été très formateur.