

**Solution Leonn**

# Rapport De Stage



BUT informatique

Réalisé par Xavier TROUCHE

Année universitaire 2023-2024

# Sommaire

Solution Leonn	1
<b>Contexte du stage :</b>	<b>7</b>
Présentation de l'entreprise :	7
Présentation du service informatique	7
Présentation du projet	8
Contexte	9
Participants du projet :	9
Délai du projet :	9
Cible du projet :	9
Pour le Service Informatique :	9
Pour les Salariés de l'Entreprise :	9
Les objectifs :	10
Analyse du besoin :	11
Gestion d'inventaire	11
Gestion de matériel	11
Gestion de prêts	11
Gestion des tickets :	11
Fonctionnalités du projet :	12
<b>Rapport Technique</b>	<b>14</b>
<b>Choix de l'entreprise</b>	<b>15</b>
Environnement de développement	15
Système d'exploitation	15
Outils de développement	15
IDE Vscode	15
La suite office o365	15
Microsoft Teams	15
Planner :	16
VirtualBox	16
Méthodologie et pratiques	16
Hiérarchie et prise de décision	16
<b>Technologies utilisées :</b>	<b>16</b>
Symfony 6.4 avec PHP 8.2 :	17
Tailwind CSS :	17
Symfony UX :	17

Typesense :	18
Twig Component :	18
Twig Live Components :	18
API Platform :	18
Conception et choix de la BDD	19
Conception du prêt	22
Réalisation de la base de données	22
Sécurité	25
Mise en application de la sécurité	25
Système de token pour la connexion utilisateur	25
Contrôle d'accès avec le Bundle Security	27
Api Platform	29
Sécurisation de l'API avec JWT	29
JWT Refresh Token	30
Problèmes rencontrés	30
Intégration de la page d'accueil de Leonn	31
Écran d'accueil	31
Partie Header :	31
Partie Sidebar :	32
Réalisation de la page d'accueil	32
Les DQL de la page d'accueil	32
Processus de Scan Gestion de prêt	35
Conception Message Flash	36
Avantages des messages flash	36
Réalisation Message flash	37
Types de messages flash	37
Légèreté de l'interface graphique	38
Logo de navbar :	38
Logo SideBar :	39
Conception d'autocomplétion	40
Réalisation d'autocomplétion avec Symfony UX	40
Problèmes rencontrés	41
Conception Subscribers	42
Réalisation du Subscriber	43
Marque Subscriber	43
Conception Twig components	45
Réalisations avec Twig components	45
Composant Tableau	46

Conception Live Components	49
Réalisation Twig Live Components	49
Spécifications :	51
Étape d'installation du projet :	51
1.0 Cloner le projet	51
1.1 Installer les dépendances PHP avec Composer :	51
2. Installer les dépendances JavaScript :	51
3. Créer un fichier .env.local avec les informations nécessaires au fonctionnement de la base de données et à typesense	52
4. Lancez la création du serveur typesense	52
Spécification	52
5. Lancez l'import des données de votre base de données vers la base de données typesense	52
Spécification	52
6. Lancez le server symfony, puis yarn	53
<b>Méthodologie et organisation du projet</b>	<b>54</b>
Organisation du travail dans l'entreprise	54
Méthodes de développement et de travail	55
Autonomie et proactivité	55
Prise d'initiative et proposition de solutions	55
Démarche pour trouver des informations	55
Travail effectué.	56
Tâches secondaires	56
<b>Annexe</b>	<b>57</b>
Configuration VirtualBox	57
Système d'exploitation :	57
Configuration réseau :	57
Configuration hardware :	57
Configuration SSH	58
Partie Clé SSH :	58
Une fois votre clé SSH créée, je vous invite à suivre l'étape ci-dessus.	58
Source : <a href="https://chrisjhart.com/Windows-10-ssh-copy-id/">https://chrisjhart.com/Windows-10-ssh-copy-id/</a>	58
Partie Open SSH :	58
Script de configuration :	59
Script de suppression de la configuration	60
Configuration Vscode	61
Les extensions vscode :	61
Configuration Docker	63
Configuration des réseaux et des volumes	66

Partie networks :	66
Partie volumes :	66
<b>Organigramme</b>	<b>67</b>
<b>Documentation Api-Platform</b>	<b>69</b>
<b>Documentation Php Flasher</b>	<b>74</b>
Snippets vscode	76

# Remerciements

Je tiens à remercier toutes les personnes qui m'ont soutenu tout au long de ce projet. Tout d'abord, simplement merci à Guillaume Carrio, alternant dans l'entreprise et mon binôme sur ce projet.

Je souhaite également remercier Gilles Gout, développeur de l'entreprise et mon maître de stage, ainsi que Xavier Bonnabel, administrateur système, pour leur soutien. Leur aide a été précieuse, que ce soit pour mes recherches, mes premières interventions techniques, ou pour comprendre les rouages du fonctionnement en entreprise.

Grâce à eux, j'ai pu développer et appliquer de nombreuses compétences techniques, notamment en réseau, paramétrage de Switch, fonctionnement d'un réseau, adressage IP et bien d'autres encore. Bien que ces notions aient été abordées en cours, un bon rafraîchissement ne fait jamais de mal.

Ensuite, j'ai acquis des compétences en développement web. J'ai appris à comprendre et à adapter le code d'autres développeurs, à découvrir et à implémenter de nouvelles technologies et à les utiliser de manière efficace.

Par ailleurs, j'ai également développé de nombreuses compétences organisationnelles. J'ai appris à participer aux réunions quotidiennes (daily meetings), à m'exprimer de manière professionnelle en entreprise et à adapter mon vocabulaire à mon interlocuteur. Trouver les mots justes pour transmettre ma pensée n'était pas mon point fort au début, je l'avoue, mais grâce à cette expérience, j'ai beaucoup progressé.

Mais également des compétences relationnelles telles que l'adaptabilité, la collaboration, l'esprit d'équipe, la gestion des conflits et le sens du service client.

Merci à tous ceux qui m'ont suivi et soutenu tout au long de ce stage.

# **Contexte du stage :**

## **Présentation de l'entreprise :**

L'ECIR (École de la construction des infrastructures et des réseaux) a été fondée en 1972 par Émile Pico. Depuis sa création, l'institution formatrice des métiers du secteur des travaux publics a considérablement évolué, tant au niveau du personnel que des équipements.

Actuellement, l'ECIR inclut deux pôles distincts :

- **ECIR Apprentissage** : CFA dédié à la formation des travaux publics en alternance.
- **ECIR Formations** : centre de formation continue sur les travaux publics qui propose des diplômes tels que l'obtention/renouvellement du CACES (certificat d'aptitude à la conduite d'engins en sécurité).

**L'ECIR** représente l'organisme principal, la direction générale qui supervise les deux autres pôles administrativement.

**L'ECIR** emploie 73 salariés et fait appel à de nombreux vacataires qui transmettent chaque jour leurs savoirs afin que les apprentis et stagiaires puissent accéder à un emploi, acquérir de nouvelles compétences ou évoluer dans le domaine des travaux publics.

Les principales sources de revenus de l'ECIR proviennent de ses formations axées sur les travaux publics, qui peuvent être financées par l'État, des entreprises ou des particuliers. L'institution génère également des revenus secondaires grâce à l'intégration d'un service de cantine et d'hébergement sur place.

Pour plus d'informations sur le personnel de l'entreprise, je vous invite à consulter [l'organigramme](#).

## **Présentation du service informatique**

J'ai été intégré au service informatique de l'ECIR, sous la supervision de mon maître de stage, Gilles Gout, qui est par ailleurs le responsable site Internet et communication de l'entreprise. L'équipe du service informatique se compose de trois membres :

Gilles Gout, l'administrateur système, Xavier Bonnabel et Guillaume Carrio en apprentissage sur les technologies du web. Ensemble, nous collaborons pour assurer le bon fonctionnement et l'innovation des systèmes informatiques de l'entreprise.

La présence d'un si grand nombre de membres dans une entreprise de près de quatre-vingts personnes s'explique par la politique informatique du directeur de l'ECIR, Jean-François Chabaud. En effet, il a une vision numérique ambitieuse. Pour lui, les progrès technologiques représentent un enjeu crucial.

Il prend ce sujet très au sérieux et le met en avant depuis déjà sept ans.

Cet investissement vise à ce que le service informatique utilise ses compétences pour améliorer la qualité des formations et la vie au sein de l'entreprise en proposant des outils innovants.

## **Présentation du projet**

Le service informatique est confronté à un problème : l'absence de visibilité en temps réel sur notre inventaire de prêt de matériel. Cette situation engendre des incertitudes lors des requêtes des salariés/vacataires pour du matériel, car nous ne pouvons pas toujours confirmer sa disponibilité immédiate.

De plus, une organisation déficiente du stockage entraîne une accumulation et souvent la perte d'objets. Face à cette situation, nous reconnaissons la nécessité d'une solution qui nous permettrait de suivre précisément et facilement nos prêts en temps réel.

En parallèle, bien que notre équipe mette tout en œuvre pour résoudre afin de pallier les problèmes techniques rencontrés par le personnel, l'absence d'un système formel pour enregistrer et gérer les tickets d'incidents limite notre capacité à offrir une assistance optimale. Cette problématique met en évidence le besoin d'une solution qui simplifierait l'ouverture et le suivi des tickets d'incidents, permettant ainsi une réponse plus réactive et un meilleur service aux salariés.

Ainsi, une solution globale s'impose pour adresser ces points critiques : améliorer la gestion de notre inventaire de prêts, optimiser le traitement des incidents techniques et gérer de manière proactive le matériel informatique alloué à nos salariés. La mise en place de cette solution serait une amélioration extrêmement bénéfique vers une meilleure organisation et une augmentation de l'efficacité de notre support.

# Cahier des charges

## Contexte

À mon arrivée dans l'entreprise, l'idée d'une application destinée à améliorer la gestion du matériel de prêt du service informatique de l'ECIR m'avait été transmise lors de mon entretien. Cependant, aucune conceptualisation n'avait encore été réalisée. Guillaume Carrio, apprenti en développement des technologies Web, et moi-même avons été chargés de la réalisation du projet à partir de zéro, en commençant par rédiger ce cahier des charges.

### ***Participants du projet :***

Gilles Gout pour valider les spécifications techniques, les versions de démonstration du produit afin de donner son avis et enfin la version définitive livrée.

Les développeurs Xavier Trouche et Guillaume Carrio auront la charge du développement du projet ainsi que du respect des dates de livraisons des versions.

### ***Délai du projet :***

Guillaume et moi serons affectés à ce projet pour une durée de douze semaines à compter du lundi 8 avril. La livraison du produit fini devra donc se faire le vendredi 5 juillet 2024, dernier délai.

### ***Cible du projet :***

Il est important de noter que les fonctionnalités de gestion détaillée de l'équipement resteront exclusives au service informatique. Les employés auront uniquement accès à la fonctionnalité de soumission des tickets.

### **Pour le Service Informatique :**

La plateforme fournira un système complet de gestion de prêt de matériel informatique. Ce système va permettre au personnel du service informatique de suivre avec précision l'inventaire, de mettre à jour le statut des équipements et de gérer efficacement le cycle de vie du matériel de l'entreprise.

### **Pour les Salariés de l'Entreprise :**

Les employés auront la visibilité sur leur prêt en cours via l'Intranet de l'entreprise, ce qui leur donnera une visibilité sur leur propre équipement informatique. Ce portail permettra de consulter des informations spécifiques à leur matériel prêté.

De plus, les employés pourront facilement déclarer des incidents liés à leur matériel en soumettant des tickets d'incident directement au service informatique.

## **Les objectifs :**

Nous devrons répondre à plusieurs objectifs :

1. Gérer l'inventaire de notre matériel de prêt.
2. Suivre les prêts de matériel aux salariés/vacataires afin de savoir qui a emprunté quel matériel et quand il doit nous le ramener.
3. Un rappel visuel (tableau) depuis l'intranet avec les prêts en cours du salarié.
4. Mise en place d'une option de mise au rebus sur les produits avec possibilité de mise en place de « commentaires de rebus ».
5. Gestion de tickets afin de mettre en place et de suivre les interventions en temps réel. Un salarié ou un vacataire qui aurait déclaré un ticket pourrait avoir un visuel sur celui-ci afin de connaître l'état d'avancement de sa demande.

## **Analyse du besoin :**

Le but de la solution Leonn est de pouvoir suivre l'inventaire des prêts. Savoir ce que l'on prête, à qui et quand on doit nous rendre le ou les matériels empruntés dans le but d'avoir une gestion plus économique et environnementale.

### **Gestion d'inventaire**

- Enregistrement de nouveaux matériels : les membres du service informatique pourront facilement ajouter des produits non répertoriés à l'inventaire actuel, assurant une mise à jour dynamique.
- Suppression de matériels : Leonn offrira la possibilité de retirer les produits qui n'ont jamais été prêtés.
- Mise au rebut : pour créer un prêt, nous avons besoin d'un utilisateur (à qui) et d'une liste de produits (quoi). Si un produit est déjà affecté à un prêt, il ne pourra pas être supprimé, car nous souhaitons conserver un historique des produits prêtés. C'est pourquoi la fonctionnalité de mise au rebut a été introduite.
- Suivi des prêts : L'outil de gestion suivra et enregistrera les prêts de produits pour garantir une visibilité complète sur les échanges et les mouvements de stock.

### **Gestion de matériel**

Nous disposons de matériel pour faciliter l'utilisation de Leonn, à savoir une douchette pour scanner nos produits, un lecteur de cartes NFC accompagné d'un stock de cartes NFC, ainsi qu'une imprimante pour pouvoir imprimer nos propres codes-barres.

### **Gestion de prêts**

- Nous devons pouvoir initier un prêt à partir d'une carte NFC.
- Nous devons pouvoir ajouter un prêt.
- Nous devons pouvoir modifier un prêt.
- Nous devons pouvoir supprimer un prêt.

### **Gestion des tickets :**

- Création de demandes de service : les salariés pourront initier des demandes d'assistance et signaler des tickets incidents en quelques clics, facilitant l'intervention rapide du service informatique.

# **Fonctionnalités du projet :**

## **1. L'inventaire**

- a. Nous devons pouvoir accéder à une liste avec les différentes informations essentielles d'un matériel.
- b. Nous devons intégrer un champ de recherche général pour faciliter la navigation parmi les matériels.
- c. Nous devons pouvoir filtrer les matériels afin d'apporter une meilleure visibilité à l'utilisateur.
- d. Nous devons créer une alerte par e-mail en cas de stock faible d'un matériel. Chaque matériel aura une limite propre.
- e. Nous devons pouvoir ajouter un matériel rapidement.
- f. Nous devons pouvoir modifier un matériel rapidement.
- g. Nous devons pouvoir mettre au rebut un matériel.
- h. Nous devons pouvoir supprimer un matériel jamais prêté.
- i. un dashboard avec les informations essentielles, la liste des prêts en cours, le nombre d'utilisateurs, le nombre de prêts, l'évolution des prêts par rapport au mois dernier.

## **2. Suivi des prêts**

- a. Nous devons accéder à une liste des prêts en cours depuis la page d'accueil.
- b. Nous devons avoir un historique de tous les prêts.
- c. Nous devons avoir une date de début de prêt.
- d. Nous devons avoir une date de fin prévue.
- e. Nous devons avoir une date de rendu réelle.
- f. Nous devons proposer, depuis Leonn, une liste des prêts du salarié.
- g. Nous devons savoir quel matériel a été prêté, à qui, pendant combien de temps, et si le matériel a été confronté à des incidents ou des problèmes durant cette période, tels que des dysfonctionnements ou des pannes signalées par l'utilisateur.

## **3. Gestion des tickets**

- a. Nous devons avoir un historique des tickets d'incident.
- b. Nous devons pouvoir suivre le statut d'un ticket d'incident pour le salarié/vacataire concerné (envoyé, en cours de traitement, résolu).

- c. Nous devons pouvoir suivre le statut du ticket d'incident pour le service informatique (envoyé, en cours de traitement, résolu).
- d. Nous devons pouvoir ajouter un commentaire sur un ticket afin de fournir des précisions ou des consignes à une autre personne chargée de finaliser le ticket.
- e. Nous devons pouvoir déclarer un ticket depuis l'intranet dans la liste des objets prêtés.
- h. Nous devons pouvoir filtrer nos tickets d'incident par date, par nom.

# **Rapport Technique**

Dans ce rapport technique, nous analyserons les choix informatiques de l'entreprise et leur influence sur les technologies utilisées ainsi que sur les décisions de conception. Nous aborderons les points suivants :

- Base de données : Nous expliquerons nos choix de structure et de champs pour répondre aux besoins et contraintes spécifiques.
- Sécurité de l'application : Il est crucial de préciser le niveau de sécurité mis en place, ainsi que les méthodes, les emplacements et les raisons de ces choix.
- Page d'accueil de l'application : Nous discuterons de la conceptualisation du home, des choix effectués pour garantir à l'utilisateur une accessibilité optimale aux informations essentielles.
- Processus de scan : Nous détaillerons la manière dont ce processus a été géré, depuis la conceptualisation jusqu'à la mise en œuvre.
- Amélioration de l'expérience utilisateur : Nous aborderons les actions entreprises pour améliorer l'expérience utilisateur, notamment les messages flash, l'autocomplétion des formulaires et la possibilité de créer des entités manquantes directement depuis un formulaire (par exemple, créer une marque depuis un produit). Cela inclut également l'utilisation des composants Twig et Twig live.
- API Platform : Nous parlerons de nos choix concernant l'API Platform, comment nous avons sécurisé cette API et pourquoi cela est important.
- Moteur de recherche Typesense : Nous expliquerons nos choix concernant l'utilisation de Typesense, comment nous l'utilisons et les avantages de cette intégration.

Pour chacun de ces points, nous ferons la distinction entre la conception et la réalisation. La conception couvrira les principes et les choix théoriques effectués en amont, tandis que la réalisation détaillera l'implémentation pratique de ces choix. À ces deux parties s'ajoutera une section dédiée aux problèmes rencontrés pour chacun des points abordés.

# ***Choix de l'entreprise***

## ***Environnement de développement***

### **Système d'exploitation**

Nous avons décidé de rester sur Windows pour profiter de ses avantages et répondre aux besoins des salariés, mais suite aux conseils de l'administrateur système sur les distributions Linux, nous avons opté pour une solution intermédiaire en utilisant VirtualBox. VirtualBox est un logiciel de virtualisation qui permet de créer et de gérer des machines virtuelles, permettant ainsi d'exécuter plusieurs systèmes d'exploitation sur un même ordinateur sans redémarrer.

### ***Outils de développement***

#### **IDE Vscode**

Visual Studio Code (VSCode) est très apprécié des développeurs pour ses nombreux avantages. Gratuit et open source, il offre une interface conviviale et personnalisable avec une vaste collection d'extensions. Parmi ses fonctionnalités puissantes, le développement à distance via SSH est particulièrement utile. Pour ces raisons, nous avons choisi VSCode.

Veuillez trouver en annexe ma configuration [vscode](#)

### ***La suite office o365***

#### **Microsoft Teams**

En outre, l'entreprise a mis en place Microsoft Teams couplé à OneDrive, ce qui facilite grandement le partage et la collaboration sur les fichiers. Microsoft Teams permet une communication fluide avec des fonctionnalités de chat, de visioconférence et de réunions en ligne, tandis que OneDrive offre un stockage cloud sécurisé et une gestion des fichiers en temps réel. Ensemble, ces outils permettent une collaboration transparente, avec des documents toujours à jour et accessibles de n'importe où. Cela améliore notre efficacité, réduit les risques de pertes de données et facilite le travail en équipe, même à distance.

## **Planner :**

Planner est un logiciel de planification de tâches. Il offre des fonctionnalités similaires à celles de Trello, étudiées à l'IUT, telles que l'affectation des tâches à un membre et l'ajout d'une date limite. Il permet ainsi de suivre globalement l'avancement des tâches de l'application, en distinguant les tâches prévues, en cours, terminées et validées.

## **VirtualBox**

Nous utilisons principalement VirtualBox pour coder.

Veuillez trouver en annexe ma configuration [virtualbox](#)

## ***Méthodologie et pratiques***

### **Hiérarchie et prise de décision**

Dans la structure de l'entreprise, l'administrateur système joue un rôle crucial dans la prise de décisions concernant l'environnement de développement. Il travaille en collaboration avec les développeurs pour trouver les solutions les plus adaptées. Par exemple, lorsqu'un développeur souhaite utiliser PHP 8.2, l'admin système identifie la meilleure façon de mettre en place côté serveur compte des aspects techniques comme la sécurité. C'est un processus d'échange entre les développeurs et l'administrateur qui assure la cohérence et la stabilité de l'environnement de développement. Bien que l'admin système puisse avoir un droit de veto pour des raisons de sécurité et inversement. Les décisions sont généralement prises de cette manière.

# **Technologies utilisées :**

Dans la réalisation de ce projet, nous intégrons un ensemble de technologies modernes, sécurisées et hautement fonctionnelles, sélectionnées pour leur ergonomie et leur facilité d'utilisation. Les technologies suivantes ont été choisies par le maître de stage pour répondre aux exigences du projet :

## **Symfony 6.4 avec PHP 8.2 :**

Symfony est un framework PHP qui nous permet de développer Leonn plus rapidement.

Il offre la possibilité de créer des entités, des contrôleurs, et de gérer la base de données simplement via Doctrine, parmi bien d'autres fonctionnalités.

Concernant le choix de la version, il est important de noter que les versions LTS (Long-Term Support) sont celles se terminant par .4, telles que 6.4, 7.4, etc. Afin d'éviter des migrations trop fréquentes, bénéficier d'une meilleure sécurité et de d'une meilleure stabilité, nous avons décidé d'opter pour la version la plus stable, qui est supportée jusqu'à fin 2026.

## **Tailwind CSS :**

Tailwind CSS est un framework CSS qui permet, après consultation de la documentation, de générer facilement du CSS à partir des noms donnés dans les classes, embellissant ainsi l'interface de Leonn. Gilles Gout a imposé ce choix car, contrairement à Bootstrap, un autre framework CSS, Tailwind ne charge que ce qui est nécessaire, alors que Bootstrap requiert l'importation de fichiers volumineux. Pour éviter de surcharger l'application, nous avons opté pour Tailwind.

## **Symfony UX :**

Symfony Ux est une technologie qui vient en complément du framework, elle va permettre de dynamiser sans avoir besoin d'écrire du javascript.

## ***Typesense :***

Moteur de recherche puissant et rapide, offrant des fonctionnalités avancées de recherche et de filtrage pour des applications web.

## ***Twig Component :***

Système de template flexible et extensible pour PHP, permettant de créer des composants réutilisables et modulaires pour la construction d'interfaces utilisateur.

## ***Twig Live Components :***

Extension de Twig Component qui permet le rendu dynamique des composants côté client, offrant une expérience utilisateur fluide et réactive sans rechargement de page.

## ***API Platform :***

API-Platform est une distribution Symfony qui permet de créer rapidement et simplement de puissantes API REST, au moyen de quelques annotations sur nos Entités.

# Conception et choix de la BDD

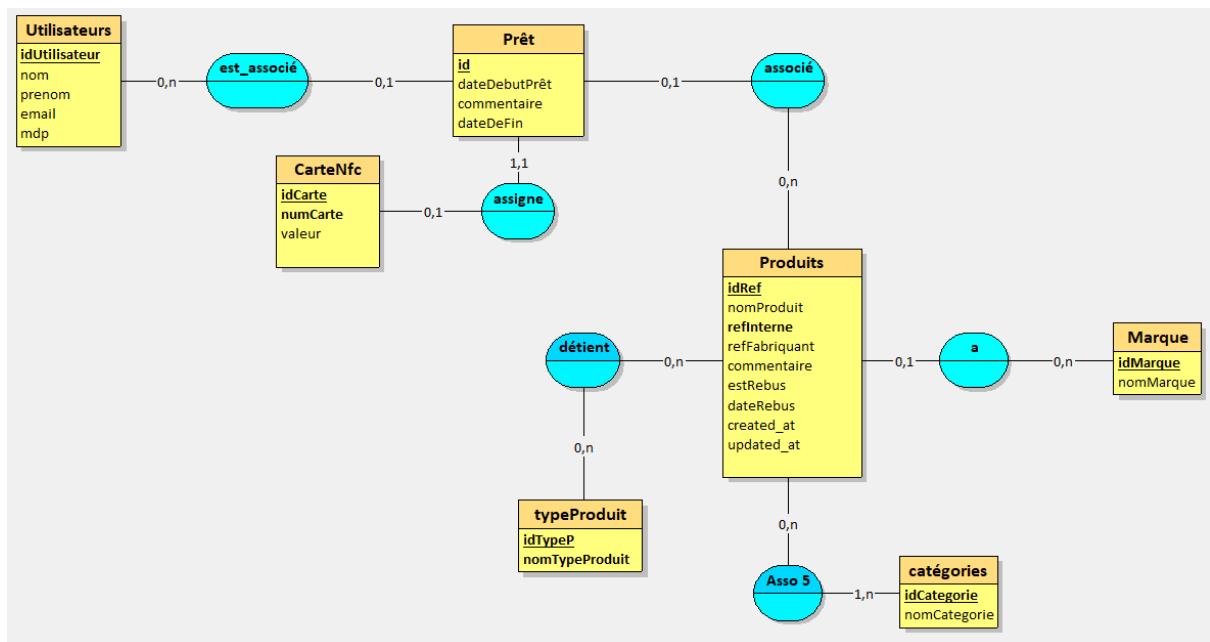
Pour la base de données, nous avons opté pour un modèle MySQL, également imposé. Nous avons rencontré des problèmes lors de la conception de ce modèle, car n'étant pas familier avec Symfony et son ORM Doctrine, j'ai d'abord tenté de concevoir la base de données en utilisant les méthodes traditionnelles enseignées à l'IUT. J'ai commencé par une réflexion sur papier, la mise en place des dépendances fonctionnelles, puis j'ai développé le schéma de la base de données en suivant le théorème de Casey-Delobel.

Les problèmes sont survenus lorsque j'ai essayé d'appliquer directement les notions de cours à un framework que je ne connaissais pas, ce qui m'a fait perdre du temps et a souvent perturbé ma démarche de réflexion. Durant la première semaine, nous avons donc perdu énormément de temps sur la modélisation de la base de données.

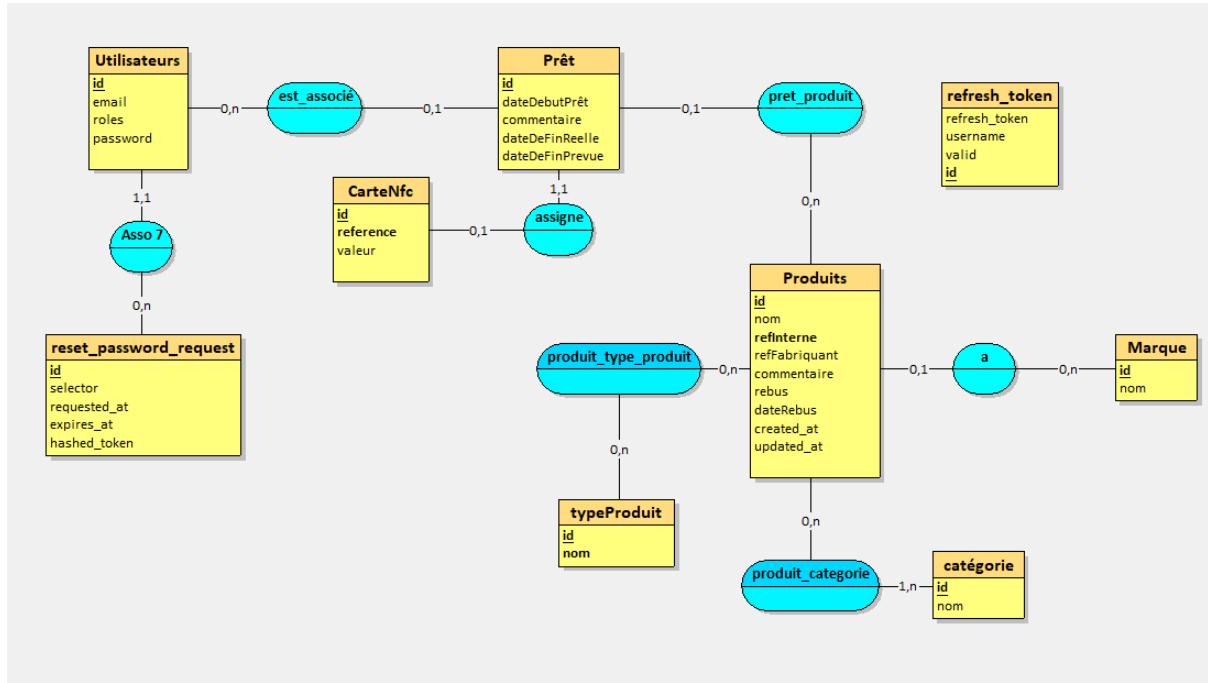
Par la suite, notre client et maître de projet nous a aidés à simplifier la modélisation. Nous avons commencé par modéliser l'entité "Produit" avec ses dépendances, que je vais expliquer plus en détail. Ensuite, nous avons créé le schéma de "Prêt" en intégrant les utilisateurs.

Cela nous a permis de mieux structurer notre base de données et de gagner en efficacité dans le développement du projet.

Je vous joins ici un schéma réalisé durant la phase de conception de la base de données de Leonn. Bien qu'il ne représente pas l'application actuelle, il offre un support visuel essentiel à mes propos.



Notre modèle MCD actuel :



Je vais ignorer les identifiants, car ils constituent la base du schéma ainsi que des attributs tels que le nom, l'e-mail, parce qu'ils sont des points évidents. Cela permettra de gagner en efficacité et en clarté.

Nous avons commencé par les produits, en incluant une référence interne (refInterne) et une référence fabricant (refFabricant). Cette distinction est importante, car il est possible que nous perdions la référence fabricant ou qu'un produit n'en ait pas (code-barres). Pour éviter de ne pas pouvoir répertorier le produit, nous générerons une référence interne unique pour chaque produit.

Un champ de commentaire permet d'expliquer l'utilisation du produit, par exemple, pourquoi il a été mis au rebut. Lorsqu'un produit est endommagé ou cassé, nous avons la possibilité de le mettre au rebut avec une date correspondante (dateRebus). Une date de dernière mise à jour (updated\_at) est également utile pour suivre l'évolution du produit.

Les produits peuvent être associés à une marque, mais ce n'est pas obligatoire. La distinction importante réside également entre le type de produit et sa catégorie.

Nous suivons principalement les objets prêts, mais uniquement ceux qui ont un type spécifique. Actuellement, il existe deux types de produits : "prêt" pour les objets que nous prêtons et suivons, et "consommable immo" pour des articles comme des cartouches d'imprimante ou des clés USB que nous donnons et qui ne sont pas forcément retournés. Suivre les prêts de ces derniers n'est pas nécessaire pour nous.

Voici un exemple concret de cette distinction :

Produit Immo : une cartouche d'imprimante ou une clé USB donnée aux utilisateurs sans attente de retour.

Produit Prêt : une PS5 avec suivi du retour prévu.

PS5	020620240039	Acer	Pret	Matériel	✓	02/06/2024 09:58		
Cartouche Imprimante	020620240040	Acer	Immo	Autre	✓	02/06/2024 09:59		

Ce schéma permet une gestion efficace des produits et des prêts, en tenant compte des différentes caractéristiques et besoins de suivi spécifiques.

## Conception du prêt

Pour cette partie-là, nous avons défini qu'un prêt devait se produire entre un utilisateur et un ou plusieurs produits.

L'origine de cette démarche peut provenir d'une carte avec la fonctionnalité de scan vue précédemment, mais elle n'est pas obligatoire.

De plus, une date de fin prévue a été établie pour estimer la durée du prêt. Par la suite, une date de retour réelle a été introduite pour confirmer que le prêt a été clôturé et que le matériel a bien été récupéré.

## Réalisation de la base de données

Pour la partie de la base de données, après avoir discuté et peaufiné le schéma qui répondrait aux besoins de Leonn. Nous avons configuré nos entités à l'aide de l'ORM de Symfony (Doctrine).

Mise en œuvre :

```
● ● ● leonn [SSH: 192.168.56.102] - setup.sh
1  symfony console make:entity
```

Suite à ça, le processus de création d'entité est lancé et nous allons devoir répondre à une suite de questions afin de déterminer :

- Nos attributs
  - nom
  - type
    - string, integer, float, boolean, etc...
    - ou relation avec d'autres entités
      - Choix de l'entité en relation
      - cardinalité
- Si nous souhaitons exposer à l'API notre entité, qu'elle soit accessible.
  - Prérequis (avoir installé api-platform)

Veuillez trouver tout le suivi de création d'une entité :

```
> ubuntu@ubuntu-VirtualBox:~/dev/leonn$ symfony console make:entity  
Class name of the entity to create or update (e.g. VictoriousKangaroo):  
> ExemplaireProduit  
Mark this class as an API Platform resource (expose a CRUD API for it) (yes/no) [no]:  
>  
Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:  
>  
created: src/Entity/ExemplaireProduit.php  
created: src/Repository/ExemplaireProduitRepository.php  
Entity generated! Now let's add some fields!  
You can always add more fields later manually or by re-running this command.  
New property name (press <return> to stop adding fields):  
> reference  
Field type (enter ? to see all types) [string]:  
>  
Field length [255]:  
>  
Can this field be null in the database (nullable) (yes/no) [no]:  
>  
updated: src/Entity/ExemplaireProduit.php
```

```
Add another property? Enter the property name (or press <return> to stop adding fields):
> products

Field type (enter ? to see all types) [string]:
> relation

What class should this entity be related to?:
> Produit

What type of relationship is this?
-----
Type      Description
-----
ManyToOne  Each ExemplaireProduit relates to (has) one Produit.
           Each Produit can relate to (can have) many ExemplaireProduit objects.

OneToMany  Each ExemplaireProduit can relate to (can have) many Produit objects.
           Each Produit relates to (has) one ExemplaireProduit.

ManyToMany  Each ExemplaireProduit can relate to (can have) many Produit objects.
           Each Produit can also relate to (can also have) many ExemplaireProduit objects.

OneToOne   Each ExemplaireProduit relates to (has) exactly one Produit.
           Each Produit also relates to (has) exactly one ExemplaireProduit.

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> ManyToOne

Is the ExemplaireProduit.products property allowed to be null (nullable)? (yes/no) [yes]:
>

Do you want to add a new property to Produit so that you can access/update ExemplaireProduit objects from it - e.g. $produit->getExemplaireProduits()? (yes/no) [yes]:
>

A new property will also be added to the Produit class so that you can access the related ExemplaireProduit objects from it.

New field name inside Produit [exemplaireProduits]:
>

updated: src/Entity/ExemplaireProduit.php
updated: src/Entity/Produit.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!
```

Next: When you're ready, create a migration with `symfony console make:migration`

## Configuration Docker

Pour avoir un environnement de travail sain et pour ne pas impacter les autres lors d'une modification, nous avons décidé de travailler localement avec un docker. Dans ce docker, nous avons mis en place différents services :

- Un service MySQL pour la base de données

● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml

```
1 db:
2   image: mysql
3   container_name: db_docker_symfony
4   restart: always
5   ports:
6     - "3306:3306"
7   volumes:
8     - db-data:/var/lib/mysql
9   environment:
10    MYSQL_ALLOW_EMPTY_PASSWORD: 'yes'
11   networks:
12     dev:
13       ipv4_address: 172.21.0.4
```

- Un service phpmyadmin pour accéder et interagir graphiquement avec notre base de données

● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml

```
1 phpmyadmin:
2   image: phpmyadmin
3   container_name: phpmyadmin_docker_symfony
4   restart: always
5   depends_on:
6     - db
7   ports:
8     - "8180:80"
9   environment:
10    PMA_HOST: db
11   networks:
12     dev:
13       ipv4_address: 172.21.0.6
```

- Un service maildev nous permettant de simuler un serveur de messagerie

```
● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml
1 maildev:
2   image: maildev/maildev
3   container_name: maildev_docker_symfony
4   command: bin/maildev --web 80 --smtp 25 --hide-extensions STARTTLS
5   ports:
6     - "8081:80"
7   restart: always
8   networks:
9     dev:
10    ipv4_address: 172.21.0.5
```

- un service typesense pour faire des recherches ultra-rapides sur notre application.

```
● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml
1 typesense:
2   image: typesense/typesense:26.0
3   container_name: typesense_docker_symfony
4   restart: on-failure
5   ports:
6     - "8108:8108"
7   volumes:
8     - typesense-data:/data
9   command: '--data-dir /data --api-key=xyz --enable-cors'
10  networks:
11    dev:
12      ipv4_address: 172.21.0.7
13  networks:
14    dev:
15      ipam:
16        config:
17          - subnet: 172.21.0.0/24
18  volumes:
19    db-data:
20    typesense-data:
```

## Sécurité

Nous avons voulu mettre en place un niveau de sécurité optimal pour notre application au niveau des points d'entrée de Leonn. Nous avons jugé les points suivants comme obligatoires pour garantir une solution sécurisée :

- Connexion/Déconnexion utilisateur
- Inscription utilisateur
- oubli de mot de passe, possibilité de récupération de compte
- API platform (JSON Web Token et JWT refresh token)
  - L'API servira d'Endpoint pour les accès hors application.
- Utilisation de token sur les formulaires de suppression

Nous souhaitions pouvoir accéder à nos prêts en cours hors de l'application (par exemple depuis l'intranet), afin de présenter à l'utilisateur connecté un widget contenant les objets qu'il a empruntés. Il est primordial que cette API ne soit pas accessible par n'importe qui, ce qui nécessite une sécurité à travers l'utilisation d'une clé API ou d'un système de token. Après avoir discuté avec les développeurs du projet, nous avons convenu que ce seraient des tokens.

## *Mise en application de la sécurité*

### **Système de token pour la connexion utilisateur**

Symfony utilise un système de tokens pour gérer la connexion des utilisateurs. Lorsqu'un utilisateur se connecte, un token unique est généré et associé à sa session. Ce token est utilisé pour authentifier l'utilisateur lors de chaque requête ultérieure, garantissant que seul l'utilisateur authentifié peut accéder à ses informations. Voici comment cela fonctionne techniquement :

- Génération du token : lors de la connexion, Symfony génère un token unique pour l'utilisateur.
- Stockage du token : le token est stocké côté serveur et dans la session de l'utilisateur.
- Validation du token : pour chaque requête, le token est vérifié pour s'assurer que l'utilisateur est bien authentifié.

Voici une illustration du flux de connexion utilisateur avec le système de token côté code :

```
leonn [SSH: 192.168.56.102] - SecurityController.php

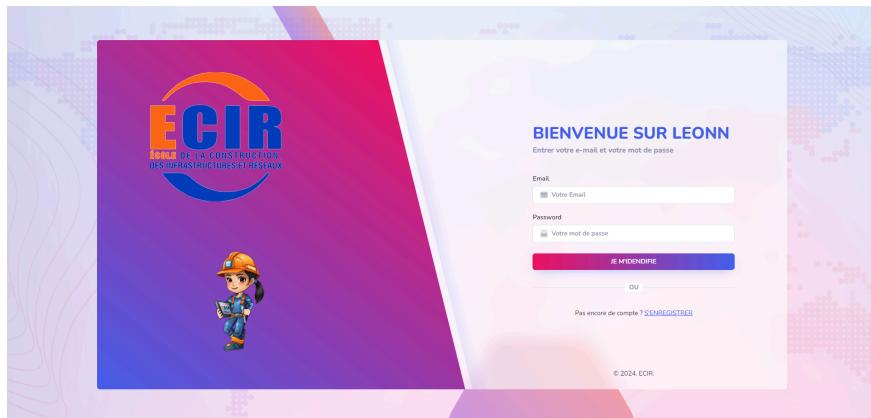
1 #[Route(path: '/login', name: 'app_login')]
2     public function login(AuthenticationUtils $authenticationUtils): Response
3     {
4         // if ($this->getUser()) {
5         //     return $this->redirectToRoute('target_path');
6         // }
7         // get the login error if there is one
8         $error = $authenticationUtils->getLastAuthenticationError();
9         // last username entered by the user
10        $lastUsername = $authenticationUtils->getLastUsername();
11        return $this->render('security/log.html.twig', ['last_username' => $lastUsername, 'error' => $error]);
12    }
```

```
leonn [SSH: 192.168.56.102] - log.html.twig

1 <input type="hidden" name="_csrf_token"
2     value="{{ csrf_token('authenticate') }}">
3 <button
4     type="submit"
5     class="btn btn-gradient !mt-6 w-full border-0 uppercase shadow-[0_10px_20px_-10px_rgba(67,97,238,0.44)]">
6     Je m'identifie
7 </button>
```

C'est la partie `name="_csrf_token"` `value="{{ csrf_token('authenticate') }}` qui va déclencher tout le fonctionnement d'authentification de token de symfony.

Voici une illustration du flux de connexion utilisateur :



## Contrôle d'accès avec le Bundle Security

Notre projet Symfony intègre le bundle Security, qui permet de réguler l'accès aux différentes parties de l'application en fonction des rôles des utilisateurs ou de certaines conditions spécifiques. Cela nous permet de définir des règles précises sur qui peut agir/interagir d'où et sur quoi.

- Rôle utilisateurs : les utilisateurs se voient attribuer des rôles spécifiques (par exemple, ROLE\_USER, ROLE\_ADMIN). Symfony attribut automatiquement le rôle « ROLE\_USER » par défaut.
- Règles d'accès : les règles d'accès sont définies dans les fichiers de configuration (security.yaml), spécifiant quelles routes ou actions sont accessibles par quels rôles.

Exemple de configuration des règles d'accès dans notre fichier security.yaml :

```
leonn [SSH: 192.168.56.102] - security.yaml

1 access_control:
2     - { path: ^/docs, roles: PUBLIC_ACCESS } # Allows accessing API documentations and Swagger UI docs
3     - { path: ^/auth, roles: PUBLIC_ACCESS }
4     - { path: ^/login, roles: PUBLIC_ACCESS } # Allows accessing the login page
5     # - { path: ^/api/prets, roles: IS_AUTHENTICATED_FULLY } # Allows accessing the prets API for everyone who is connected
6     - { path: ^/api/, roles: IS_AUTHENTICATED_FULLY } # Allows accessing the API for admins only
7     - { path: ^/api/(login|token/refresh), roles: PUBLIC_ACCESS }
8     - { path: ^/api/, roles: IS_AUTHENTICATED_FULLY } # Allows accessing the API for everyone who is connected
9     - { path: ^/, roles: ROLE_USER }
```

- Conditions d'accès : en plus des rôles, des conditions supplémentaires peuvent être appliquées. Exemple : afficher un bouton uniquement pour le «role\_admin»

```
leonn [SSH: 192.168.56.102] - index.html.twig

1  {% if 'ROLE_ADMIN' in app.user.roles %}
```

Système de token pour la suppression d'items

Pour renforcer la sécurité lors des opérations critiques comme la suppression d'items, Symfony utilise également un système de token.

- Génération du token de suppression : un token unique est généré pour chaque requête de suppression.
- Validation du token : avant de procéder à la suppression, le token est vérifié pour confirmer l'authenticité de la requête.

Exemple de code pour la génération et la vérification du token :



leonn [SSH: 192.168.56.102] - supprimer.html.twig

```
1 <form method="post" action="{{ path('app_produit_delete', {'id': p.document.id}) }}" onsubmit="return confirm('Are you sure you want to delete this item?');">
2   <input type="hidden" name="_token" value="{{ csrf_token('delete') ~ p.document.id }}">
3   <button class="text-slate-800 hover:text-white text-sm bg-white hover:bg-red-400 border border-slate-200 rounded-r-lg font-medium px-4 py-2 inline-flex space-x-1 items-center">
4     <twig:Delete />
5   </button>
6 </form>
```

Code côté contrôleur :



leonn [SSH: 192.168.56.102] - CategorieController.php

```
1 #[Route('/supprimer/{id}', name: 'app_categorie_delete', methods: ['POST'])]
2 public function supprimer(Request $request, Categorie $categorie): Response
3 {
4     if ($this->isCsrfTokenValid('delete' . $categorie->getId(), $request->request->get('_token'))) {
5         $this->entityManager->remove($categorie);
6         $this->entityManager->flush();
7
8         notyf()
9             ->position('x', 'center')
10            ->position('y', 'top')
11            ->duration(2000) // 2 seconds
12            ->addError('Catégorie ' . $categorie->getNom() . ' ' . ' a été supprimée');
13     }
14     return $this->redirectToRoute('app_categorie');
15 }
```

## Api Platform

Nous avons implémenté API Platform dans notre application pour répondre à un besoin spécifique.

À l'ECIR, il y a un intranet composé de nombreuses applications :

- **Lucca** : gestion des jours d'absences et du télétravail
- **Boomerang** : gestion des commandes en ligne
- **Marttin** : logiciel GMAO
- **Net Ypareo**
- **Moodle CFA**
- **Moodle Formation**
- **Wattson** : outil de suivi des habilitations et des CACES
- **Teams**
- **Resa ECIR** : Réservation d'hébergement
- **ecirapprentissage.fr**
- **ecirformation.fr**
- **OneDrive** : gestion de vos fichiers
- **ChatGPT** : modèle de langage (LLM) pour poser des questions ou demander de l'aide.
- **Texte vers voix (IA)** : transformer votre texte en voix

Étant donné que cet endroit est très fréquenté, nous avons eu l'idée d'y installer un tableau qui affichera les objets empruntés. Ce tableau indiquera à chaque personne les articles qu'elle a empruntés, la durée de l'emprunt, et rappellera la date de retour prévue.

## Sécurisation de l'API avec JWT

Nous avons également sécurisé notre API en utilisant des tokens JWT (JSON Web Token). Cette méthode offre plusieurs avantages, notamment la limitation de l'accès dans le temps grâce à l'expiration des tokens, garantissant que seules les personnes autorisées peuvent accéder aux ressources. En plus de cela, l'utilisation de JWT renforce la sécurité en exigeant un token valide pour chaque requête.

L'authentification JWT permet d'identifier un utilisateur sans devoir passer par la page login de Leonn.

Nous avons mis en place cette technologie en plusieurs étapes, que vous pouvez consulter dans la [documentation](#) que nous avons écrite.

## JWT Refresh Token

Pour éviter de surcharger les appels à la route d'authentification de notre API, qui génère un token et un token de rafraîchissement, nous avons mis en place un token de rafraîchissement valable pour une durée d'un mois. Ce jeton permet de rafraîchir le token actuel d'accès si celui-ci n'est plus valide.

## Problèmes rencontrés

Dans cette section, les principales difficultés ont été de comprendre comment intégrer les connaissances transmises par l'IUT concernant le développement web au sein de Symfony, notamment le fonctionnement et l'utilisation du framework. En effet, comprendre le fonctionnement d'une technologie nous permet de l'exploiter pleinement.

Par exemple, quelles sont les manières de sécuriser l'accès d'une application et où gérer ça.

Nous avons pour cela plusieurs options :

- Sécurisé depuis le fichier config/packages/security.yaml

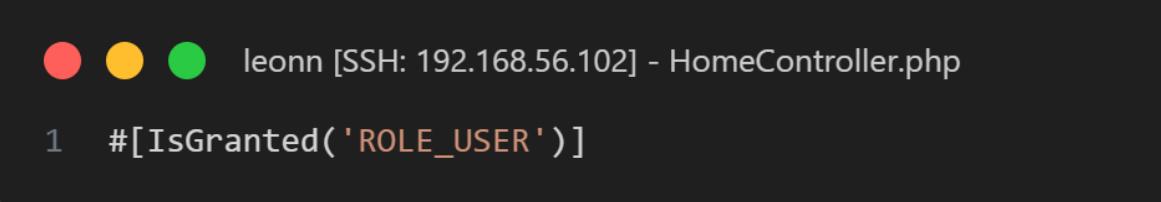
On peut simplement vouloir sécuriser tout ce qui suit une route particulière. Ici, on sécurise tout ce qui suit /api/.



leon [SSH: 192.168.56.102] - security.yaml

```
1 - { path: ^/api/, roles: IS_AUTHENTICATED_FULLY } # Allows accessing the API for admins only
```

- Sécuriser une route depuis le contrôleur



leon [SSH: 192.168.56.102] - HomeController.php

```
1 #[IsGranted('ROLE_USER')]
```

- Sécuriser une action depuis le contrôleur avec un voter

Nous n'avons pour le moment pas exploré cette solution, mais elle existe et elle permettrait en théorie de fixer des conditions plus précises sur les droits d'un utilisateur.

# Intégration de la page d'accueil de Leonn

The screenshot shows the Leonn application's dashboard. On the left, a sidebar menu includes Home, Prêts, Produits, Cartes, and Utilisateurs. The main area features three summary cards: 'Nombre d'utilisateurs totaux' (100019), 'Nombre de produits totaux' (17), and 'Nombre de prêts totaux' (13). Below these is a table titled 'Liste des prêts en cours : 13' with columns for Date de début, Liste des produits, Personne concernée, and Date de fin prévue. The table lists 13 entries from April 23 to May 29, 2024, detailing items like Smartphones and Imprimante, and users like a@b.15200 and clara.touche@ecirtp.fr.

## Écran d'accueil

Sur la page d'accueil, nous voulions un dashboard regroupant les informations cruciales, à savoir :

- Le nombre d'utilisateurs
- Le nombre de produits
- Le nombre de prêts totaux (pas seulement les prêts en cours)
- L'évolution des prêts par rapport au mois dernier

Nous voulions également avoir accès à une liste des prêts en cours, permettant de savoir :

- Ce qui a été prêté,
- Quand la personne est censée rendre le matériel emprunté,
- Depuis quand elle possède le matériel.

### Partie Header :

- Un lien vers la gestion des prêts par scan avec une carte NFC (détails à venir)
- Un champ de recherche
- Une option pour modifier le mode d'affichage, sombre, clair (recommandé), ou choix du système.
- Une vue de profil affichant le rôle, l'adresse e-mail connectée et un lien vers la déconnexion.

## **Partie Sidebar :**

Celle-ci est destinée à guider l'utilisateur vers les différentes pages grâce à une nomenclature claire et un système de sous-menus :

- Home : page d'accueil
- Prêt :
  - Historique des prêts
  - Création de prêts
- Produits :
  - Inventaire des produits
  - Listing des marques
  - Listing des types de produits
  - Listing des catégories
- Cartes :
  - Liste des cartes
- Utilisateurs :
  - Liste des utilisateurs
  - Liste des prêts de l'utilisateur connecté

## ***Réalisation de la page d'accueil***

Un template CSS basé sur le framework tailwind CSS a été imposé.

Pour la page d'accueil ainsi que pour toutes les autres pages de Leonn. Cette approche nous permet de nous concentrer davantage sur les fonctionnalités de l'application plutôt que sur son esthétique. Cependant, nous avons soigneusement sélectionné et adapté ce template à nos besoins afin de garantir une expérience utilisateur optimale.

Pour afficher nos données, nous avons utilisé des widgets dont les informations sont récupérées à l'aide de DQL (Doctrine Query Language). Les requêtes DQL, bien qu'elles soient similaires aux requêtes SQL, ont leur propre syntaxe. Cela permet à l'ORM Doctrine, le composant de Symfony chargé de la gestion des bases de données, d'interpréter et d'exécuter les requêtes de manière appropriée via le code.

### ***Les DQL de la page d'accueil***

Depuis notre contrôleur, nous faisons appel à des méthodes codées dans les repository respectifs.



leonn [SSH: 192.168.56.102] - HomeController.php

```
1 $users = $this->userRepository->findAll();
2     $produits = $this->produitRepository->findAll();
3     $prets = $this->pretRepository->findAll();
4     $pretsListe = $this->pretRepository->findByActive();
5     $pretsMois = $this->pretRepository->findByThisMonth();
6     $pretsPreviousMois = $this->pretRepository->findByPreviousMonth();
```

- findAll (fourni par symfony) qui récupère tous les prêts
- findByActive pour liste des prêts en cours



leonn [SSH: 192.168.56.102] - PretRepository.php

```
1 public function findByActive(): array
2 {
3     return $this->createQueryBuilder('p')
4         ->andWhere('p.dateFin is null')
5         ->getQuery()
6         ->getResult()
7     ;
8 }
```

- findByThisMonth pour récupérer la liste des prêts du mois actuel



leonn [SSH: 192.168.56.102] - PretRepository.php

```
1 public function findByThisMonth(): array
2 {
3     $premierJourDuMois = new \DateTime('first day of this month midnight');
4     $dernierJourDuMois = new \DateTime('last day of this month 23:59:59');
5
6     return $this->createQueryBuilder('p')
7         ->andWhere('p.datePret between :debutMois and :finMois')
8         ->setParameter('debutMois', $premierJourDuMois)
9         ->setParameter('finMois', $dernierJourDuMois)
10        ->getQuery()
11        ->getResult()
12    ;
13 }
```

- findByPreviousMonth pour récupérer les prêts du mois précédent

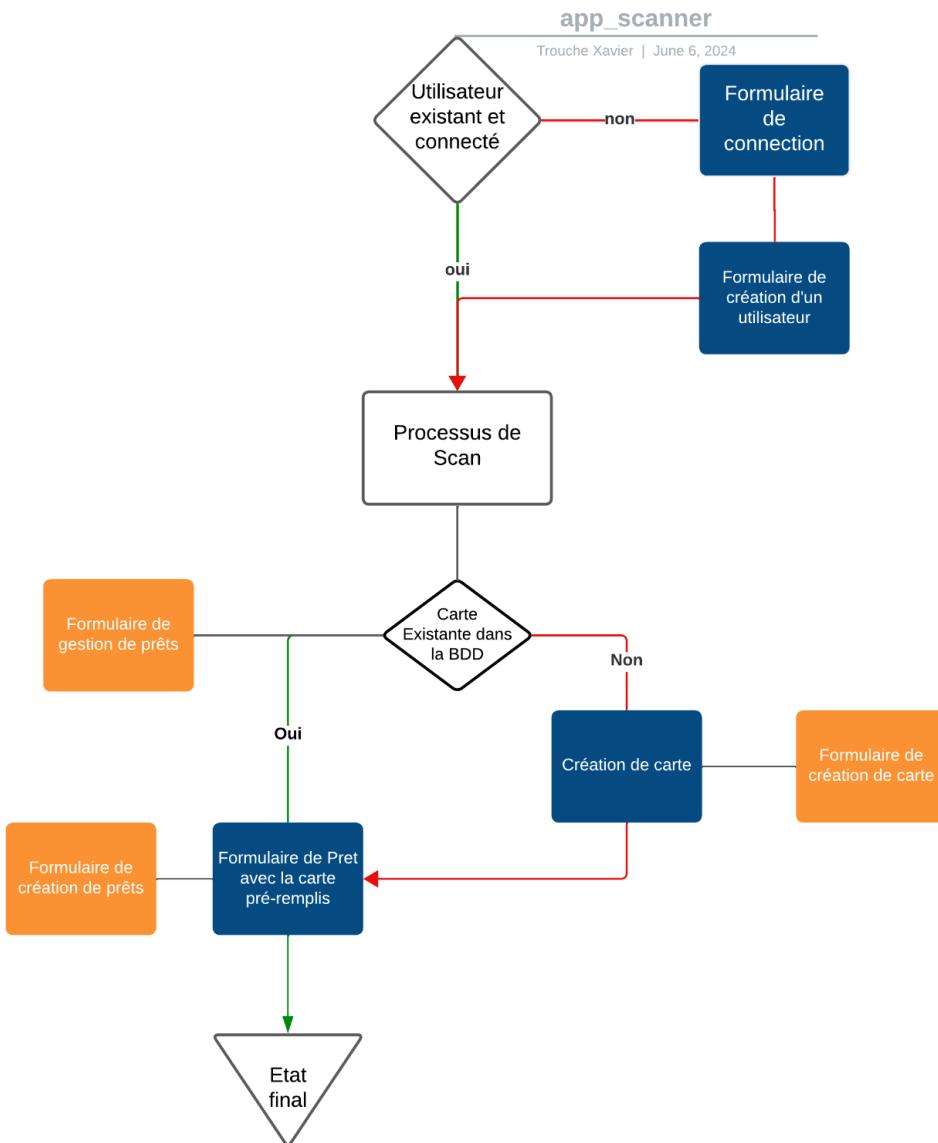


leonn [SSH: 192.168.56.102] - PretRepository.php

```
1 public function findByPreviousMonth(): array
2 {
3     $premierJourDuDernierMois = new \DateTime('first day of previous month midnight');
4     $dernierJourDuDernierMois = new \DateTime('last day of previous month 23:59:59');
5     return $this->createQueryBuilder('p')
6         ->andWhere('p.datePret between :debutMois and :finMois')
7         ->setParameter('debutMois', $premierJourDuDernierMois)
8         ->setParameter('finMois', $dernierJourDuDernierMois)
9         ->getQuery()
10        ->getResult()
11    ;
12 }
```

Pour ce qui est du tableau, nous avons mis en place un Twig component. Le principe est simple, le but est de faire ça et de généraliser l'utilisation d'un composant qui amenait à de la duplication de code. Désormais, on appelle simplement le composant en lui passant les bons paramètres et nous laissons le moteur de template Twig faire le reste pour nous. Nous verrons ce composant plus en détail dans la partie [Twig Components](#).

## Processus de Scan Gestion de prêt



Dans notre application, nous avons voulu gérer de manière rapide et efficace l'insertion de prêts effectués en urgence, notamment à partir d'une carte. Voici comment nous avons rendu cela possible :

Imaginons une situation d'urgence : une personne arrive dans le bureau et a besoin d'un produit immédiatement. Nous lui assignons une carte, scannons le produit, identifions la personne ou lui demandons son nom, et en quelques secondes, le prêt est créé. Tout cela peut se faire soit par le biais du formulaire de gestion de prêt, soit directement.

Nous allons maintenant entrer dans les détails du fonctionnement pour vous expliquer comment nous avons rendu ce processus aussi fluide et rapide.

# Conception Message Flash

## Avantages des messages flash

- Clarté et réactivité : les messages flash fournissent des retours instantanés sur les actions de l'utilisateur, ce qui permet de comprendre immédiatement si une action a été réussie ou si une correction est nécessaire.
- Amélioration de l'expérience utilisateur : en offrant des indications claires et visuelles, les messages flash aident l'utilisateur à naviguer dans l'application de manière plus fluide et intuitive.
- Réduction des erreurs : les messages d'erreur et d'avertissement aident à réduire le nombre d'erreurs en alertant les utilisateurs avant qu'ils ne commettent une action potentiellement problématique.
- Engagement utilisateur : les messages de succès, en particulier, créent un sentiment de satisfaction et de progression, incitant l'utilisateur à continuer d'utiliser l'application.

En intégrant ces types de messages flash, nous avons conçu une interface utilisateur qui non seulement guide les utilisateurs de manière efficace, mais aussi améliore leur interaction globale avec l'application.

Frameworks PHP, dont Symfony, sur lequel notre projet est basé. PHP Flasher est également disponible pour Laravel et bien d'autres frameworks. Avec un système de librairies, elle offre un choix plus large pour l'esthétique des messages flash. Nous avons opté pour la librairie Notyf.

## Réalisation Message flash

Nous avons mis en place plusieurs éléments pour rendre cette application-là plus intuitive possible, notamment grâce à la présence de logos, de messages d'erreur et de titres permettant de situer l'utilisateur dans ce qu'il va faire. Parmi ces éléments, les messages flash jouent un rôle crucial pour améliorer l'expérience utilisateur.

Nous avons mis en place ce système grâce à une bibliothèque appelée PHP Flasher. Vous pouvez consulter notre [documentation](#) pour voir comment nous l'avons implémentée et comment nous l'utilisons. Cette technologie a été très rapide à mettre en place et opérationnelle. Elle est pratique, car elle est adaptable à de nombreux.

### Types de messages flash

- Nous avons implémenté les quatre types de messages habituels.
  - Succès



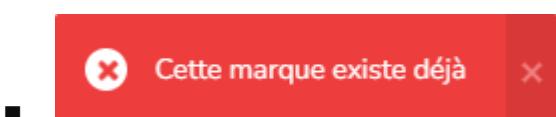
- Warning



- Info



- Erreur



## Légèreté de l'interface graphique

Nous avons conçu notre interface graphique pour qu'elle soit légère et intuitive, facilitant ainsi l'utilisation de l'application même en présence d'un grand nombre d'informations. Voici quelques points clés :

- Utilisation stratégique des couleurs : Les informations importantes sont mises en évidence par des couleurs distinctes, ce qui aide l'utilisateur à les repérer rapidement et facilement.

Sur la page home :



Sur la liste des produits :

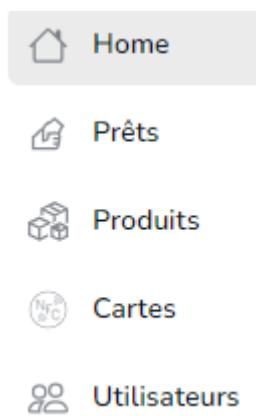
Liste des produits											Ajouter un produit
Nom	Référence Interne	Référence Fabricant	Marque	Type de produit	Catégorie	En Stock	Date de rebus	Commentaire	Dernière modification	Actions	
Ordinateur	REF_INTERNEO	REF_FABRICANTO		Prest Immo	Matériel, Bureautique, Autre	X	30/04/2024	fixture produit	30/05/2024 09:37		

- Logos explicites : les logos utilisés sont clairs et significatifs, permettant à l'utilisateur de comprendre leur fonction d'un simple coup d'œil.

### Logo de navbar :

- Scan
- 
- Adaptation du mode et des informations utilisateurs
-

## Logo SideBar :



- Clarté visuelle et titres explicites : le design visuel est soigné et les titres sont précis, aidant l'utilisateur à naviguer et à se situer dans l'application.

Liste des produits										Ajouter un produit
Nom	Référence Interne	Référence Fabricant	Marque	Type de produit	Categorie	En Stock	Date de rebus	Commentaire	Dernière modification	Actions
Ordinateur	REF_INTERNE0	REF_FABRICANT0		Pret Immo	Matériel, Bureautique, Autre	X	30/04/2024	fixture produit	30/05/2024 09:37	

Grâce à ces éléments, l'application Leonn permet à l'utilisateur de trouver plus facilement sa place et de gérer efficacement les informations, même lorsqu'elles sont nombreuses.

## Conception d'autocomplétion

Avec Symfony, nous avons intégré des composants d'autocomplétion pour améliorer encore davantage l'expérience utilisateur.

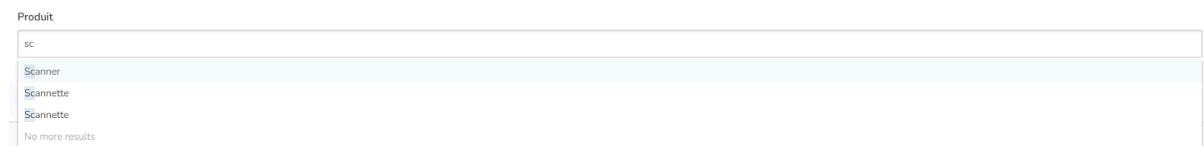
Les composants d'autocomplétion présentent plusieurs avantages :

- Gain de temps : ils permettent à l'utilisateur de trouver plus rapidement ce qu'il cherche en suggérant des options au fur et à mesure de la saisie.
- Réduction des erreurs : en proposant des suggestions pertinentes, ils diminuent les risques de fautes de frappe ou d'erreurs de saisie.
- Amélioration de l'accessibilité : les utilisateurs peuvent naviguer plus efficacement dans l'application, rendant l'expérience plus fluide et intuitive.

En intégrant ces composants, nous facilitons la recherche et la navigation au sein de l'application, ce qui permet aux utilisateurs de se concentrer sur leurs tâches principales sans se perdre dans les menus ou les options.

### Réalisation d'autocomplétion avec Symfony UX

Voici un exemple dans notre formulaire de création de prêt dans un champ :



Symfony intègre son propre fonctionnement de formulaire.

Vous pouvez donc créer un formType en ligne de commande en vous basant soit sur une entité existante, soit sur rien.

A screenshot of a terminal window. At the top, there are three colored circles (red, yellow, green) followed by the text "leonn [SSH: 192.168.56.102] - setup.sh". Below this, the command "1 symfony console make:form" is visible.

Veuillez trouver le code qui permet cette interaction :

Le champ dont je vais exposer le code permet d'automatiser la recherche de produits par nom depuis le formulaire de prêt.



leonn [SSH: 192.168.56.102] - PretType.php

```
1 ->add('produit', ProduitAutocompleteField::class, [
2     ])
```

## Les configurations de l'autocomplétion du produit



leonn [SSH: 192.168.56.102] - ProduitAutocompleteField.php

```
1 public function configureOptions(OptionsResolver $resolver)
2 {
3     $resolver->setDefaults([
4         'class' => Produit::class,
5         'placeholder' => 'Choisissez un produit',
6         'searchable_fields' => ['nom'],
7         'multiple' => true,
8         'choice_label' => 'nom',
9
10        // 'query_builder' => function (ProduitRepository $produitRepository) {
11        //     return $produitRepository->createQueryBuilder('produit');
12        // },
13        // 'security' => 'ROLE_SOMETHING',
14    ]);
15 }
16
17
18 public function getParent(): string
19 {
20     return BaseEntityAutocompleteType::class;
21 }
```

## Problèmes rencontrés

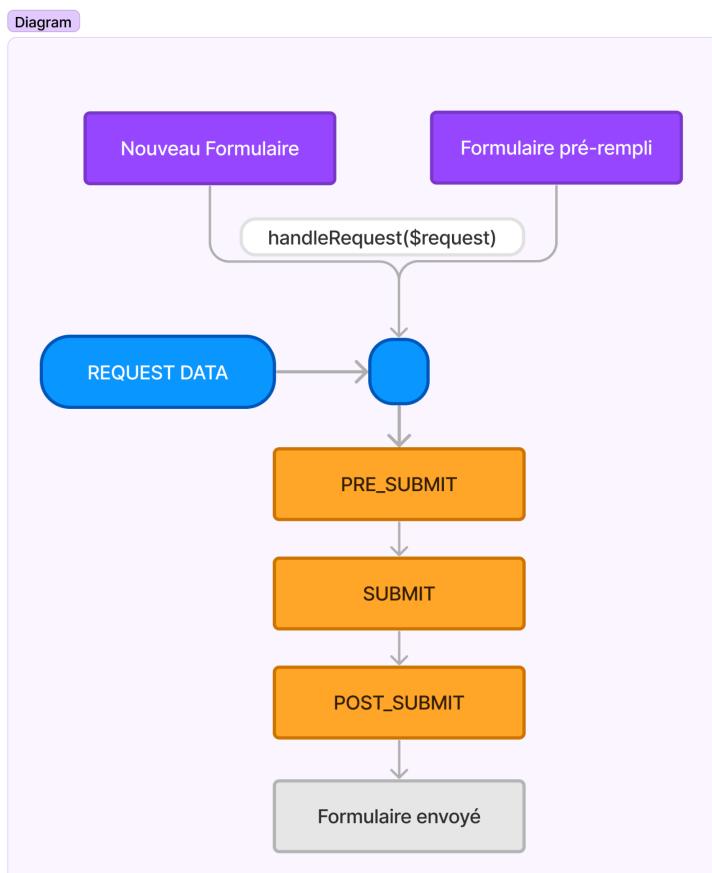
Il existe plusieurs façons de mettre en place un champ d'autocomplétion avec le FormType de Symfony. Une option consiste simplement à ajouter un attribut autocomplete → true. Une autre option est d'ajouter un champ d'autocomplétion en tant que classe distincte, ce qui permet de résoudre nos problèmes de manière plus efficace. En effet, nous avons de nombreux objets dans notre base de données, mais il n'est parfois pas nécessaire de tous les afficher. Par exemple, quand on crée un prêt et que l'on cherche un produit, on ne veut afficher que les produits qui ne sont pas déjà affectés à un prêt. Nous avons besoin de cibler certains attributs par des requêtes AJAX de l'entité concernée par l'autocomplétion pour obtenir une meilleure précision.

## Conception Subscribers

Nous avons rencontré un problème dans l'optimisation de l'expérience utilisateur de Leonn. Lors de la création d'un produit, qui inclut les attributs listés ci-dessous, il est nécessaire d'ajouter une marque ( facultative), une ou plusieurs catégories, ainsi qu'un ou plusieurs types de produit. Pour résoudre ce problème, nous avons dû mettre en place des subscriber.

id	nom	ref_interne	ref_fabricant	commentaire	rebus	date_rebus	create_at (DC2Type:datetime_immutable)	updated_at	marque_id
----	-----	-------------	---------------	-------------	-------	------------	---	------------	-----------

Un subscriber dans Symfony est un écouteur d'événements qui permet d'intercepter et de gérer les différentes étapes du cycle de vie d'un formulaire. Vous pouvez consulter le schéma ci-dessous pour une meilleure compréhension des étapes de validation d'un formulaire. Dans notre cas, nous avons utilisé un subscriber pour affiner l'expérience utilisateur de Leonn lors de la création d'un produit. Plus précisément, nous avons intercepté l'événement `pre_submit` afin de récupérer les données du formulaire avant leur soumission. Cela nous a permis de vérifier la marque fournie dans le champ, de la créer si elle n'existe pas, ou de laisser le formulaire suivre son cours normal si la marque est déjà présente.



## Réalisation du Subscriber

Nous avons réalisé des écouteurs pour :

- la création de marque
- la création de catégories

Pour éviter de la redondance, je vous montrerai seulement comment nous avons mis en place la classe MarqueSubscriber.

### Marque Subscriber

```
leonn [SSH: 192.168.56.102] - ProduitType.php

1  ->add('marque', EntityType::class, [
2      'required' => false,
3      'class' => Marque::class,
4      'autocomplete' => true,
5      'tوم_select_options' => [
6          // 'persist'=> false,
7          // 'createOnBlur'=> true,
8          'create'=> true
9      ],
10     ])
11     ->addEventSubscriber(new MarqueSubscriber($this->entityManager))
12     ->getForm()
```

```
leonn [SSH: 192.168.56.102] - MarqueSubscriber.php

1 <?php
2 namespace App\Form\EventSubscriber;
3
4 use App\Entity\Marque;
5 use Doctrine\ORM\EntityManagerInterface;
6 use Symfony\Component\EventDispatcher\EventSubscriberInterface;
7 use Symfony\Component\Form\Event\PreSubmitEvent;
8 use Symfony\Component\Form\FormEvent;
9 use Symfony\Component\Form\FormEvents;
10 class MarqueSubscriber implements EventSubscriberInterface
11 {
12     private $entityManager;
13     public function __construct(EntityManagerInterface $entityManager)
14     {
15         $this->entityManager = $entityManager;
16     }
17     public static function getSubscribedEvents()
18     {
19         return [
20             FormEvents::PRE_SUBMIT => 'onPreSubmit',
21         ];
22     }
23     public function onPreSubmit(PreSubmitEvent $event)
24     {
25         $form = $event->getForm();
26         $data = $event->getData();
27         if (!$data || !isset($data['marque'])) {
28             return;
29         }
30         $marqueName = $data['marque'];
31         $marque = $this->entityManager->getRepository(Marque::class)->findOneBy(['id' => $marqueName]);
32         if (!$marque) {
33             $marque = new Marque();
34             $marque->setNom($marqueName);
35             $this->entityManager->persist($marque);
36             $this->entityManager->flush();
37             $data['marque'] = $marque->getId();
38         }
39         $event->setData($data);
40     }
41 }
```

## Conception Twig components

L'utilisation des composants Twig est extrêmement bénéfique pour Leonn. Grâce à Twig, nous pouvons généraliser des composants réutilisables, ce qui simplifie grandement le développement et la maintenance.

Les raisons pour lesquelles nous utilisons Twig sont nombreuses :

- Réutilisabilité : nous pouvons créer des composants génériques qui peuvent être utilisés partout dans l'application. Par exemple, un composant Twig peut gérer n'importe quel type de tableau en passant simplement un titre, une liste d'objets et des options spécifiques.
- Simplification du code : en centralisant la logique dans des composants Twig, nous réduisons la redondance et facilitons la gestion du code.
- Flexibilité et personnalisation : les composants peuvent être facilement personnalisés pour répondre aux besoins spécifiques de différentes parties de l'application.
- Efficacité : le gain de temps est significatif, car il n'est plus nécessaire de recréer des éléments similaires pour chaque nouvelle fonctionnalité ou section de l'application.

En somme, Twig nous permet de développer plus rapidement, de maintenir un code propre et organisé, et d'offrir une expérience utilisateur cohérente et efficace.

### Réalisations avec Twig components

Pour éviter de surcharger le rapport, nous nous concentrerons sur le composant le plus complexe : le tableau. Les autres composants seront simplement évoqués en ce qui concerne leur utilisation.

Nous avons réalisé un composant de modif qui représente en fait toutes les actions de mise à jour de nos entités. Nous avons mis en place le même style de composant pour la suppression.

Un [snippet](#) pour le composant de modification, de suppression et pour les tableaux est accessible.

```
● ● ● leonn [SSH: 192.168.56.102] - index.html.twig

1 <twig:Modif url="app_pret_editer" id="{{p.id}}"/>
2 <twig:FormDelete url="app_pret_delete" id="{{p.id}}"/>
```

## Composant Tableau

Notre composant possède une liste d'attribut obligatoire et flexible en fonction du besoin de notre tableau.

```
● ● ● leonn [SSH: 192.168.56.102] - Tableau.php

1 use Symfony\Bridge\Doctrine\Form\Type\EntityType;
2 use Symfony\UX\TwigComponent\Attribute\AsTwigComponent;
3
4 #[AsTwigComponent]
5 final class Tableau
6 {
7     public string $entityType;
8     public string $titre;
9     public ?string $titreBouton;
10    public ?string $actionButton;
11    public array $listeTh;
12    public array $listeAttributs;
13    public $listeObjets;
14    public ?string $urlModif;
15    public ?string $urlSupp;
16 }
```

Ce composant va être lié à une vue Tableau.html.twig, comme cette vue est conséquente, nous allons simplement voir les points où interviennent les paramètres du composant.

```
● ● ● leonn [SSH: 192.168.56.102] - Tableau.html.twig

1 <span class="font-medium group-hover:text-white">{{ titreBouton }}</span>
```

pour la liste des th du tableau

```
● ● ● leonn [SSH: 192.168.56.102] - Tableau.html.twig

1 <thead>
2     <tr>
3         {% for th in listeTh %}
4             <th>{{ th }}</th>
5         {% endfor %}
6     </tr>
7 </thead>
```

Pour la liste des objets

```
● ● ● leonn [SSH: 192.168.56.102] - Tableau.html.twig

1 <tbody>
2     {% for item in listeObjets %}
3         <tr>
4             {% if entityType != 'user' %}
5                 <td>{{ attribute(item, attribut) }}</td>
6             {% endfor %}
7             <td>
8                 <div class="flex w-full h-full">
9                     {% if entityType != 'user' %}
10                        <twig:Modif url="{{urlModif}}" id="{{item.id}}"/>
11                    {% endif %}
12                    {% set supprimable = true %}
13                    {% if entityType == 'carte' or entityType == 'produit' or entityType == 'user' %}
14                        {% if item.prets|length > 0 %}
15                            {% set supprimable = false %}
16                        {% endif %}
17                    {% elseif entityType == 'marque' or entityType == 'categorie' or entityType == 'typeProduit' %}
18                        {% if item.produits|length > 0 %}
19                            {% set supprimable = false %}
20                        {% endif %}
21                    {% endif %}
22                    {% if supprimable %}
23                        <twig:FormDelete url="{{urlSupp}}" id="{{item.id}}"/>
24                    {% endif %}
25                </div>
26            </td>
27        </tr>
28    {% endfor %}
29 </tbody>
```

On ajoute un paginator côté contrôleur :

```
● ● ● leonn [SSH: 192.168.56.102] - CategorieController.php

1 $pagination = $paginator->paginate(
2     $categories,
3     $request->query->getInt('page', 1),
4     20
5 );
```

On ajoute le paginator coté Twig :

```
● ● ● leonn [SSH: 192.168.56.102] - Tableau.html.twig

1 {{ knp_pagination_render(listeObjets) }}
```

Nous appelons ce composant grâce à notre snippet Tableau. Voici un exemple avec la liste de nos catégories.

```
● ● ● leonn [SSH: 192.168.56.102] - index.html.twig

1 <twig:Tableau
2     entityType="{{ entityType }}"
3     titre="{{ titre }}"
4     titreBouton="{{ titreBouton }}"
5     actionBar="{{ actionBar }}"
6     listeTh="{{ liste_th }}"
7     listeAttributs="{{ listeAttributs }}"
8     listeObjets="{{ listeObjets }}"
9     urlModif="{{ urlModif }}"
10    urlSupp="{{ urlSupp }}"/>
```

# Conception Live Components

Nous souhaitons intégrer des composants dynamiques pour permettre la mise à jour en temps réel des données de nos tableaux en fonction des recherches effectuées dans un champ.

## Réalisation Twig Live Components

On fait appel au composant live depuis la vue twig.

```
● ● ● leonn [SSH: 192.168.56.102] - index.html.twig
```

```
1 {{ component('RechercheProduit') }}
```

Composant live :

```
● ● ● leonn [SSH: 192.168.56.102] - RechercheProduit.php
```

```
1 <?php
2
3 namespace App\Twig\Components;
4 use App\Repository\ProduitRepository;
5 use Symfony\UX\LiveComponent\Attribute\AsLiveComponent;
6 use Symfony\UX\LiveComponent\DefaultActionTrait;
7 use Symfony\UX\LiveComponent\Attribute\LiveProp;
8 #[AsLiveComponent('RechercheProduit')]
9 class RechercheProduit
10 {
11     use DefaultActionTrait;
12     #[LiveProp(writable : true)]
13     public string $produitRecherche = '';
14     private $produitFinder;
15     public function __construct($produitFinder,
16         private ProduitRepository $produitRepository)
17     {
18         $this->produitFinder = $produitFinder;
19     }
20     public function getData(): array {
21         return $this->produitRepository->findByQuery($this->produitRecherche);
22     }
23 }
24 }
```

Twig associé au composant :

```
leonn [SSH: 192.168.56.102] - RechercheProduit.html.twig

1 <div{{ attributes }}>
2
3     <div class="panel">
4         <div class="flex justify-between text-center align-middle mb-2">
5             <h1 class="text-2xl font-bold">Liste des produits</h1>
6             <input
7                 type="search"
8                 name="produitRecherche"
9                 data-model="produitRecherche">
10            <a href="{{ path('app_produit_ajouter') }}" class="inline-flex justify-center">
11                ...

```

# Manuel d'installation

## *Spécifications :*

Le projet étant réalisé en entreprise, le code produit appartient à l'ECIR. Le dépôt de ce projet est sur un GitLab auto hébergé. Il est donc nécessaire de rédiger un manuel d'installation pour expliquer comment nous procédons lors de la récupération du projet.

## **Étape d'installation du projet :**

### **1.0 Cloner le projet**

```
● ● ● leonn [SSH: 192.168.56.102] - setup.sh  
1 git clone git@git.Ecirtp.fr:dev/leonn.git
```

### **1.1 Installer les dépendances PHP avec Composer :**

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 composer install
```

### **2. Installer les dépendances JavaScript :**

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 yarn install
```

### **3. Créez un fichier .env.local avec les informations nécessaires au fonctionnement de la base de données et à typesense**

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 DATABASE_URL="mysql://root:@172.21.0.4:3306/leonn?serverVersion=5.7"  
2 TYPESENSE_URL=http://172.21.0.7:8108/  
3 TYPESENSE_KEY=xyz
```

### **4. Lancez la création du serveur typesense**

```
● ● ● leonn [SSH: 192.168.56.102] - setup.sh  
1 symfony console typesense:create
```

#### *Spécification*

Il faut savoir que pour certaines commandes, symfony permet de passer par des raccourcis, notamment pour celle-ci où t : c fait référence à typesense create.

### **5. Lancez l'import des données de votre base de données vers la base de données typesense**

```
● ● ● leonn [SSH: 192.168.56.102] - setup.sh  
1 symfony console typesense:import
```

#### *Spécification*

Il faut savoir que pour certaines commandes, symfony permet de passer par des raccourcis, notamment pour celle-ci où t: i fait référence à typesense import.

## 6. Lancez le server symfony, puis yarn

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local
```

```
1  symfony serve -d
```

Puis,

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local
```

```
1  yarn watch
```

La commande `yarn watch` est utilisée pour surveiller en temps réel les modifications apportées au code, en particulier les modifications esthétiques comme le CSS ou les fichiers JavaScript. Lorsqu'une modification est détectée, `yarn watch` recompile automatiquement les fichiers concernés.

# **Méthodologie et organisation du projet**

## ***Organisation du travail dans l'entreprise***

L'organisation du travail dans l'entreprise suit un processus bien structuré pour la prise de décision, la planification, l'attribution des tâches et le versioning de code. Guillaume et moi sommes les développeurs de ce projet. Notre maître de stage et l'administrateur de l'entreprise jouent le rôle de clients. Lorsque nous avons une idée sur une fonctionnalité , nous la concevons et l'expliquons aux clients. Ceux-ci valident l'idée et peuvent ajouter des spécifications supplémentaires. Ensuite, nous développons la fonctionnalité, la présentons au client qui peut proposer des modifications esthétiques ou techniques avant de la valider définitivement. Cette démarche est permise, car la politique de l'entreprise l'autorise, elle est ouverte à la proposition, ce qui n'est pas forcément le cas partout.

Pour la planification et l'attribution des tâches, les décisions se prennent initialement à l'oral. Une fois les choix fixés, nous les documentons sur Microsoft Planner. Bien que nous n'appliquions pas toutes les méthodologies Agile, nous tenons des réunions quotidiennes chaque matin. Pendant ces daily meetings, nous discutons de nos réalisations de la veille, des problèmes rencontrés et de nos objectifs pour la journée. Ces réunions peuvent également inclure des interactions avec d'autres employés en dehors du service informatique, qui nous informent de leurs problèmes techniques, comme des soucis d'affichage ou de licences logicielles, que nous résolvons ultérieurement.

En ce qui concerne le versioning de code, chaque nouvelle fonctionnalité est développée sur une branche distincte à partir de la branche de développement (dev) partagée avec Guillaume. Une fois la fonctionnalité terminée, Guillaume effectue une revue de code, suivie d'un briefing pour clarifier toute ambiguïté. Après validation, mon code est fusionné dans la branche DEV. Ensuite, notre maître de stage effectue une revue de code générale avant de décider de fusionner le code dans la branche de production (main). Ce processus garantit la qualité et la stabilité de notre code avant sa mise en production.

## **Méthodes de développement et de travail**

### **Autonomie et proactivité**

Mon degré d'autonomie a été notable, surtout considérant que je ne connaissais pas la plupart des technologies utilisées dans le projet initialement. J'ai dû me documenter pour comprendre et utiliser ces technologies. Par exemple, pour apprendre à utiliser Symfony, j'ai consulté des documentations officielles et sur GitHub, appris les commandes essentielles et exploré le fonctionnement des bundles.

Pour les autres technologies comme Typesense et API Platform, j'ai appliqué une méthode similaire : consulter la documentation officielle, suivre des tutoriels et explorer les bonnes pratiques. En particulier, pour sécuriser les connexions à l'API, nous avons intégré l'authentification JWT avec API Platform.

### **Prise d'initiative et proposition de solutions**

La plupart des solutions mises en place dans le projet résultent de l'initiative de Guillaume et moi. Par exemple, lorsque nous avions une idée pour améliorer l'application, nous en discutions avec l'équipe informatique. Ils pouvaient alors suggérer des technologies pertinentes, comme ce fut le cas avec Symfony UX.

### **Démarche pour trouver des informations**

Pour chaque nouvelle technologie ou fonctionnalité à implémenter, ma démarche consistait à me documenter.

En résumé, mon travail sur ce projet a nécessité de l'autonomie pour apprendre et proposer des solutions. Guillaume et moi avons souvent pris l'initiative d'améliorer et d'optimiser l'application, tout en restant en communication avec l'équipe informatique pour valider nos choix et recevoir des conseils sur les technologies à adopter.

## **Travail effectué.**

### **Tâches secondaires**

#### 1. Support technique lors d'événements

- Aménagement technique pour une conférence organisée par les Travaux publics méditerranéens nationaux : ajustement des télévisions, des micros pour les discours, et mise en place de vidéos publicitaires dans le hall.

#### 2. Résolution de problèmes techniques

- Réparation et sauvegarde de données d'un PC portable endommagé appartenant à une salariée.
- Résolution de problèmes d'accès réseau et de logiciels.

#### 3. Aménagement et configuration du réseau

- Réorganisation du réseau et installation du nouveau système de plateau du self avec le prestataire Turbo Self.
- Installation et configuration des machines, des Switch et branchements techniques pour garantir le bon fonctionnement du nouveau logiciel dans le réseau.

Ces tâches reflètent la diversité des missions que j'ai effectuées au cours de mon stage, allant du développement web et de la configuration technique à l'assistance aux événements et à la maintenance des équipements informatiques.

# Annexe

## Configuration VirtualBox

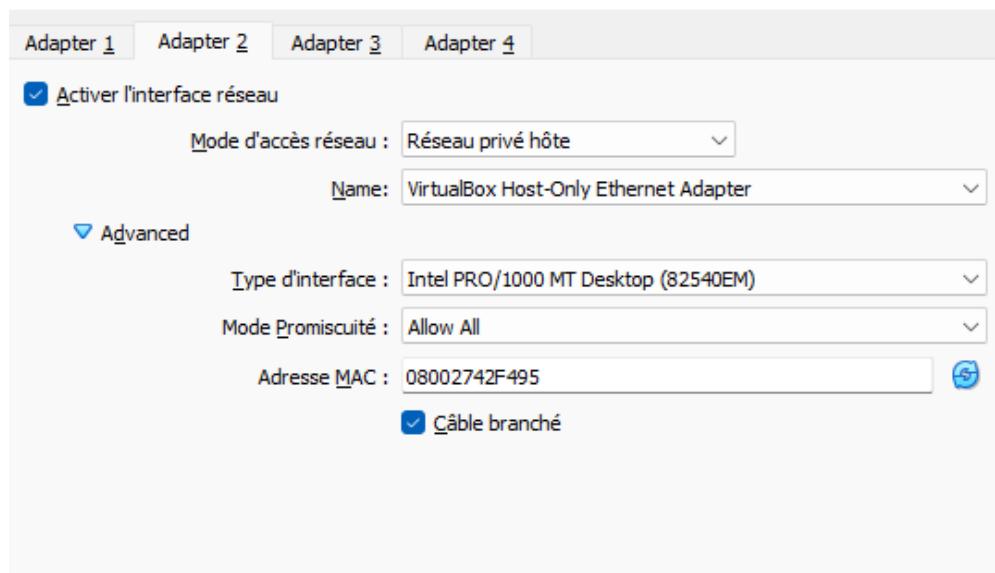
Version utilisée : [VirtualBox-7.0.14](#)

### Système d'exploitation :

- [ubuntu 22.04](#)

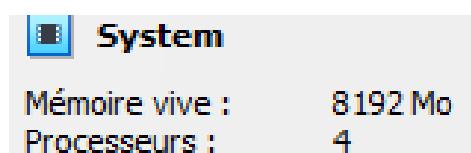
### Configuration réseau :

#### Réseau



Cette configuration me permet d'accéder à ma VM à distance, notamment pour utiliser les services Docker et projeter mon projet en cours via Symfony Serve.

### Configuration hardware :



J'ai décidé d'augmenter la RAM à 8Go, bien que 4 Go auraient été suffisants dans d'autres circonstances. De plus, l'IUT recommande fortement l'utilisation des outils Jetbrains pour le développement, et après avoir découvert que PhpStorm propose également une connexion SSH, il est apparu que pour des performances optimales, 4 CPU et 8 Go de RAM sont préconisés. Cela garantit une fluidité et une efficacité maximales dans l'exécution de nos tâches de codage.

# Configuration SSH

Pour se connecter en SSH à sa machine virtuelle, il est nécessaire de créer une clé SSH et de l'ajouter aux clés autorisées de la VM. Ensuite, il faut installer les packages OpenSSH Server et OpenSSH Client.

## Partie Clé SSH :

**WARNING : se positionner dans un powershell Windows en admin**

Deux cas possibles :

Vous possédait déjà une clé SSH 

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 type $env:USERPROFILE\.ssh\id_rsa.pub | ssh {IP-ADDRESS-OR-FQDN} "cat >> .ssh/authorized_keys"  
2 # Exemple pour ma part  
3 # type $env:USERPROFILE\.ssh\id_rsa.pub | ssh ubuntu@192.168.56.102 "cat >> .ssh/authorized_keys"
```

Créer une clé SSH

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 ssh-keygen
```

**Une fois votre clé SSH créée, je vous invite à suivre l'étape ci-dessus.**

**Source :** <https://chrisjhart.com/Windows-10-ssh-copy-id/>

## Partie Open SSH :

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local  
1 sudo apt-get install openssh-client openssh-server
```

## Script de configuration :

Étant donné les nombreuses commandes à effectuer, j'ai trouvé judicieux de faire un script avec des explications sous forme de commentaire.

```
● ● ● leonn [SSH: 192.168.56.102] - .env.local

1 # parti installation php 8.2
2 # si vous souhaitez installer une autre version de php, vous pouvez le faire en modifiant le numéro de version dans la commande qui sera précisée d'un #
3 sudo apt update && sudo apt upgrade
4 sudo apt install software-properties-common ca-certificates lsb-release apt-transport-https
5 LC_ALL=C.UTF-8 sudo add-apt-repository ppa:ondrej/php
6 sudo apt update
7 sudo apt install php8.2
8 # quelque option de base pour php
9 sudo apt install php8.2-mysql php8.2-mbstring php8.2-xml php8.2-curl php8.2-intl
10 # on met la version de php 8.2 par défaut
11 sudo update-alternatives --set php /usr/bin/php8.2
12 # Modification version
13 # sudo update-alternatives --set php /usr/bin/version_php
14 # Installation de nodejs
15 sudo apt install curl
16 # pour supprimer curl
17 # sudo apt remove curl
18 # acutellement nous sommes à la version 22.x qui est stable et fonctionnel peut être il faudra mettre à jour cette partie là
19 # le fait de mettre par défaut la dernière version peut causer des problèmes, a vous d'ajuster
20 curl -sL https://deb.nodesource.com/setup_22.x | sudo -E bash -
21 sudo apt-get install -y nodejs
22 # pour mettre à jour npm
23 npm install -g npm@latest
24 # Supprimer nodejs
25 # sudo apt-get remove nodejs
26 # ou
27 # sudo apt purge nodejs
28 # Installation de yarn
29 sudo apt update
30 curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
31 echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
32 sudo apt update
33 sudo apt install yarn --no-install-recommends
34 # désinstallation de yarn
35 # npm uninstall -g yarn
36 # Installation de composer
37 sudo apt update && sudo apt upgrade
38 sudo apt install php-cli php-zip unzip curl
39 sudo curl -sS https://getcomposer.org/installer -o composer-setup.php
40 sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
41 composer --version
42 # désinstallation de composer
43 # sudo rm -rf /usr/local/bin/composer
44 # Installation de symfonyCli
45 wget https://get.symfony.com/cli/installer -O - | bash
46 mv /root/.symfony5/bin/symfony /usr/local/bin/symfony
47 # Installer portainer
48 sudo apt update
49 sudo apt install docker.io -y
50 # regardez si le serveur tourne
51 sudo systemctl status docker
52 sudo systemctl start docker
53 sudo usermod -aG docker $USER
54 sudo docker pull portainer/portainer-ce:latest
55 # regardez si portainer est bien installé
56 docker images
57 docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer-ce:latest
58 # regardez si le service tourne
59 docker ps
60 sudo apt install docker-compose
61 docker-compose up -d
```

## Script de suppression de la configuration

Dans la même longueur d'onde, lors de la création du script précédent, j'ai rencontré des problèmes, qui parfois nécessitent la suppression de certaines choses. Par conséquent, j'ai également produit un script qui annule cette mise en place.

```
● ● ●
leonn [SSH: 192.168.56.102] - .env.local

1 #!/bin/bash
2 # Suppression de PHP 8.2
3 sudo apt purge -y php8.2 php8.2-mysql php8.2-mbstring php8.2-xml php8.2-curl php8.2-intl
4 sudo add-apt-repository --remove -y ppa:ondrej/php
5 # Suppression de Node.js et npm
6 sudo apt purge -y nodejs npm
7 sudo rm -rf /usr/local/lib/node /usr/local/include/node /usr/local/bin/npm /usr/local/share/man/man1/node* /usr/local/share/man/man1/npm*
8 # Suppression de Yarn
9 sudo npm uninstall -g yarn
10 # Suppression de Composer
11 sudo rm -rf /usr/local/bin/composer
12 rm -rf ~/.composer/composer.json
13 # Suppression de Symfony CLI
14 sudo rm -rf /usr/local/bin/symfony
15 # Suppression de Portainer
16 sudo docker stop $(docker ps -a -q)
17 sudo docker rm $(docker ps -a -q)
18 sudo docker rmi portainer/portainer-ce
19 # Suppression de Docker
20 sudo apt purge -y docker.io
21 # Nettoyage des dépendances inutiles
22 sudo apt autoremove -y
23 sudo apt clean
```

## Configuration Vscode

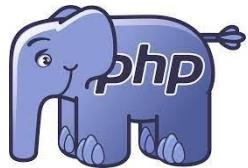
Développant exclusivement avec les outils de JetBrains, je n'étais donc pas du tout familier avec l'IDE vscode. En conséquence, j'ai dû m'adapter à ce nouvel environnement de travail. Cette adaptation a nécessité une compréhension approfondie du fonctionnement des extensions de VSCode, qui sont essentielles pour personnaliser l'éditeur et ajouter des fonctionnalités supplémentaires. J'ai également pris le temps d'apprendre les nombreux raccourcis clavier disponibles, ce qui m'a permis de gagner en efficacité et en productivité.

### Les extensions vscode :

Pour améliorer mon expérience de développement avec VSCode, j'utilise plusieurs extensions essentielles :



Cette extension enrichit les capacités Git de VSCode en affichant des informations détaillées sur l'historique des commits, les auteurs des lignes de code et les différences de fichiers. Elle aide à comprendre l'évolution du code et à collaborer plus efficacement.



Elle offre une auto complétion intelligente pour le code PHP, ainsi que des suggestions de méthodes et des aides contextuelles. Cela accélère le développement en réduisant les erreurs et en facilitant la navigation dans le code.



Cette extension fournit une autocomplétion pour les classes utilitaires de Tailwind CSS, rendant le style des éléments plus rapide et plus précis. Elle permet également de visualiser les classes disponibles directement dans l'éditeur.



Accessible via l'onglet Remote, cette extension permet de se connecter à une machine virtuelle (VM) distante via SSH. Elle offre une expérience de développement local tout en travaillant sur des serveurs distants, à condition que la VM soit allumée.



Développée par GitHub, Copilot utilise l'IA pour suggérer des lignes ou des blocs de code en fonction du contexte. Elle améliore la productivité en générant automatiquement du code, réduisant ainsi le temps de développement.



L'extension GitGraph affiche une représentation graphique de l'historique Git, facilitant la visualisation des branches, des merges et des commits. Elle est particulièrement utile pour comprendre les flux de travail complexes.



Elle ajoute une prise en charge syntaxique et une coloration pour le moteur de templates Twig, utilisé dans de nombreux projets PHP. Cela rend l'édition des fichiers Twig plus facile et plus lisible.

Ces extensions transforment VSCode en un environnement de développement puissant et personnalisé, optimisant ainsi ma productivité et la qualité de mon code.

# Configuration des réseaux et des volumes

## Partie networks :

L'utilisation de réseaux dans Docker améliore la communication entre conteneurs, offre une isolation et une sécurité accrue, facilite la scalabilité et la configuration et permet une gestion plus flexible et structurée des services.

● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml

```
1 networks:
2   dev:
3     ipam:
4       config:
5         - subnet: 172.21.0.0/24
```

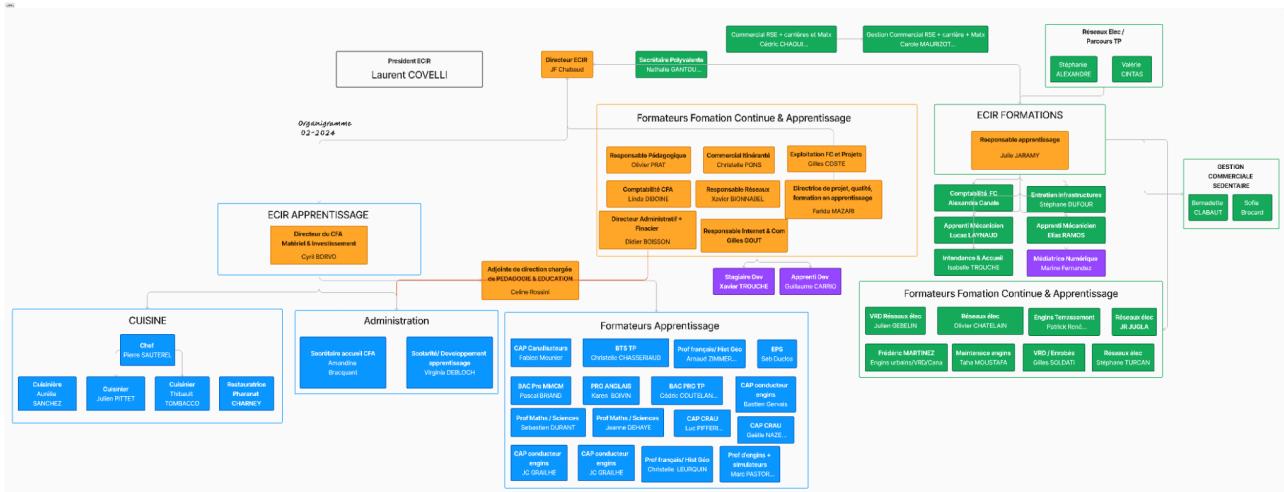
## Partie volumes :

Nous utilisons des volumes pour faire persister les données de notre application, ce qui nous permet de conserver l'état des données à des fins de test. En maintenant ces données, nous pouvons vérifier si les actions effectuées produisent les résultats attendus. Cela nous aide à valider le comportement de l'application et à assurer la fiabilité de ses fonctionnalités à travers différents scénarios de test.

● ● ● dev [SSH: 192.168.56.102] - docker-compose.yaml

```
1 volumes:
2   db-data:
3   typesense-data:
```

# Organigramme



Grâce aux compétences acquises à l'IUT, j'ai pu mettre en application mes connaissances en utilisant le logiciel Figma. En particulier, j'ai réalisé un organigramme de l'entreprise, ce qui m'a permis de visualiser et d'organiser efficacement la structure organisationnelle.

Pour y voir plus clair, je vous transmets le lien du [schéma](#).

# Documentation

Dans notre projet, nous avons décidé de créer une documentation complète pour informer l'équipe des nouveautés et des bonnes pratiques. Par exemple, nous avons inclus des snippets pour Visual Studio Code, qui permettent de gagner énormément de temps, ainsi que des guides d'installation et d'utilisation pour des technologies telles qu'API Platform, Typesense et Symfony UX (incluant Twig/Live Component, UX Autocomplete, Turbo, React, et Icon).

La mise en place de cette documentation offre plusieurs avantages :

- Gain de temps : les snippets et les guides pratiques permettent aux développeurs de travailler plus efficacement.
- Uniformité et qualité : une documentation centralisée aide à maintenir des standards élevés et cohérents dans le code et les pratiques de développement.
- Facilité d'intégration : les nouveaux membres de l'équipe peuvent rapidement se familiariser avec les technologies et les processus utilisés dans le projet.
- Réduction des erreurs : en fournissant des instructions claires et détaillées, la documentation aide à minimiser les erreurs de configuration et de développement.

En résumé, une documentation bien structurée et régulièrement mise à jour est essentielle pour améliorer la productivité, la qualité du travail et la cohésion au sein de l'équipe.

Je vais ensuite inclure toutes les documentations que nous avons élaborées jusqu'à présent.

En aparté, je tiens à préciser que j'ai pris l'initiative de convertir les fichiers Markdown (.md) en images (.png) afin de les joindre au rapport de manière plus visuelle et accessible.

# **Documentation Api-Platform**

## API Platform JWT Auth

Ce guide vous explique comment configurer l'authentification JWT dans un projet Symfony utilisant API Platform.

### Pré-requis

1. **Symfony** : Assurez-vous que Symfony est installé dans votre projet.
2. **API Platform** : Installez API Platform dans votre projet Symfony.

### Étapes de configuration

#### 1. Installer LexikJWTAuthenticationBundle

Installez le bundle LexikJWTAuthenticationBundle via Composer :

```
composer require "lexik/jwt-authentication-bundle"
```



#### 2. Générer la paire de clés JWT

Installez OpenSSL si ce n'est pas déjà fait, puis générez la paire de clés JWT :

```
sudo apt-get install openssl  
php bin/console lexik:jwt:generate-keypair
```



#### 3. Configurer les permissions des clés

Assurez-vous que les fichiers de clés sont accessibles par l'utilisateur approprié :

```
setfacl -R -m u:www-data:rX -m u:"$(whoami)":rwX config/jwt  
setfacl -dR -m u:www-data:rX -m u:"$(whoami)":rwX config/jwt
```



## 4. Configurer la sécurité

Ajoutez les configurations suivantes dans config/packages/security.yaml :

```
security:
    encoders:
        App\Entity\User:
            algorithm: bcrypt

    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email

    firewalls:
        api:
            pattern: ^/api/
            stateless: true
            provider: app_user_provider
            jwt: ~

        main:
            anonymous: true
            json_login:
                check_path: /auth
                username_path: email
                password_path: password
                success_handler: lexik_jwt_authentication.handler.authentication_success
                failure_handler: lexik_jwt_authentication.handler.authentication_failure

    access_control:
        - { path: ^/$, roles: PUBLIC_ACCESS }
        - { path: ^/docs, roles: PUBLIC_ACCESS }
        - { path: ^/auth, roles: PUBLIC_ACCESS }
        - { path: ^/login, roles: PUBLIC_ACCESS }
        - { path: ^/api/prets, roles: IS_AUTHENTICATED_FULLY }
        - { path: ^/api, roles: ROLE_ADMIN }
```

## 5. Configurer LexikJWTAuthenticationBundle

Créez un fichier de configuration config/packages/lexik\_jwt\_authentication.yaml :

```
lexik_jwt_authentication:  
    secret_key: '%env(resolve:JWT_SECRET_KEY)%'  
    public_key: '%env(resolve:JWT_PUBLIC_KEY)%'  
    pass_phrase: '%env(JWT_PASSPHRASE)%'  
    token_ttl: 3600
```

Ajoutez les variables d'environnement correspondantes dans le fichier .env :

```
##> lexik/jwt-authentication-bundle ##  
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem  
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem  
JWT_PASSPHRASE=your_passphrase  
##< lexik/jwt-authentication-bundle ##
```

## 6. Configurer API Platform

```
api_platform:  
    title: Hello API Platform  
    version: 1.0.0  
    formats:  
        jsonld: ['application/ld+json']  
    docs_formats:  
        jsonld: ['application/ld+json']  
        jsonopenapi: ['application/vnd.openapi+json']  
        html: ['text/html']  
    defaults:  
        stateless: true  
        cache_headers:  
            vary: ['Content-Type', 'Authorization', 'Origin']  
        extra_properties:  
            standard_put: true  
            rfc_7807_compliant_errors: true  
    event_listeners_backward_compatibility_layer: false  
    keep_legacy_inflector: false  
    swagger:  
        api_keys:  
            JWT:  
                name: Authorization  
                type: header
```

## Partie JWT Refresh Token

Pour une version de symfony 5.4+

### Etape 1 : Installer le bundle

```
composer require doctrine/orm doctrine/doctrine-bundle gesdinet/jwt-refresh-token-bundle
```

### Etape 2: Autoriser le bundle

```
<?php

return [
    ...
    Gesdinet\JWTRefreshTokenBundle\GesdinetJWTRefreshTokenBundle::class => ['all' : true];
];
```

### Etape 3: Configurer le bundle

Configurez la classe de refresh token

Créez le fichier config/packages/gesdinet\_jwt\_refresh\_token.yaml avec le contenu ci-dessous :

```
gesdinet_jwt_refresh_token:
    refresh_token_class: App\Entity\RefreshToken # This is the class name of the refresh token entity
```

### 2. CRÉÉ UNE CLASSE DANS SRC/ENTITY/REFRESHTOKEN.PHP

```
symfony console make:entity
# mettre REFRESH TOKEN quand il vous demande le nom de la classe
```

Une fois votre classe crée veuillez copié coller le contenu ci dessous

```
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;
use Gedinnet\JWTRefreshTokenBundle\Entity\RefreshToken as BaseRefreshToken;

/**
 * @ORM\Entity
 * @ORM\Table("refresh_tokens")
 */
#[ORM\Entity]
#[ORM\Table(name: 'refresh_tokens')]
class RefreshToken extends BaseRefreshToken
{}
```

#### Etape 4 : Définir la routes du refresh token

Dans le fichier config/routes.yaml

```
# config/routes.yaml
api_refresh_token:
    path: /api/token/refresh
# ...
```

Configurer le config/packages/security.yaml en ajoutant les lignes suivante :

```
refresh_jwt:
    check_path: /api/token/refresh # or, you may use the `api_refresh_`#
    # or if you have more than one user provider
    # provider: user_provider_name
- { path: ^/api/(login|token/refresh), roles: PUBLIC_ACCESS }
```

# ***Documentation Php Flasher***



## Installation pour Symfony

```
composer require php-flasher/flasher-symfony
```

Mettre en place les assets pour phpFlasher

```
php bin/console flasher:install
```

## Utilisation classique

```
#/ usage success
flash()->success('Your account has been re-verified.');
#/ usage error
flash()->error('There was a problem re-activating your account.');
#/ usage warning
flash()->warning('Your account may not have been de-registered.');
#/ usage info
flash()->info('Your account has been unlocked and a confirmation email has been se
```

## Usage complexe

### Info

Vous pouvez passer des options à php-flasher

```
flash()
->option('position', 'bottom-right')
->option('timeout', 3000)
->error('There was an issue submitting your feedback.');
```

Pour plus de détail, veuillez vous référer à la documentation officielle

## LIBRARIES PHPFlasher

### Info

Il faut savoir que intègre des librairies qui vous permettent d'ajuster le style de vos Message Flash. Actuellement nous utilisons une de ces librairie qui s'appelle notyf, il possède 3 autres, je vous invite à cliquer sur les lien pour en savoir d'avantage :

- [Noty](#)
- [Sweetalert](#)
- [Toastr](#)

## Installation de notyf pour Symfony

```
composer require php-flasher/flasher-notyf-symfony
```

Mettre en place les assets pour phpFalsher

```
php bin/console flasher:install
```

### Configuration fichier config/packages/flasher.yaml

Vous trouverez une ligne `default` qui par défaut prend `phpFlasher`. Pour modifier cette configuration, il faut mettre `notyf` à la place.

```
# default: votre ancienne config  
default: notyf
```

## Snippets vscode

Snippets twig component à rajouter et à expliquer

### Bouton edit

```
<twig:Modif url="ChangeMe" id="{{p.id}}"/>
```

#### Info

url = le path de suppression  
id = l'id de l'objet à supprimer  
color = la couleur du bouton (Facultatif) class de tailwindcss  
- exemple: "bg-red-500 text-white"

#### Warning

un snippet est disponible dans le projet

```
Modif
```

### Bouton supprimer

```
<twig:FormDelete url="app_pret_delete" id="{{p.id}}"/>
```

#### Paramètres

url = le path de suppression  
id = l'id de l'objet à supprimer

#### Attention

un snippet est disponible dans le projet

```
formdelete
```