DOCUMENTATION TECHNIQUE - ESPORTIFY

1. Réflexions technologiques

Avant de vous expliquer mes choix, je vais vous lister les technologies utilisées tout au long du projet.

TECHNOLOGIE	**Rôle dans mon projet**
PHP	C'est le langage principal côté serveur
MySQL	Base de données relationnelle pour les users, events, newsletter
MongoDB	Stockage des scores en NoSQL (utilisé localement)
HTML/CSS	Pour la structure et le design de l'interface utilisateur (et responsive)
SCSS	C'est un préprocesseur CSS (compilé) pour le style
JavaScript	Interaction, filtres dynamiques et AJAX
Always Data	Déploiement du site

Pour chaque technologie utilisée, je vais vous expliquer mon choix.

PHP : utiliser PHP sans framework s'est imposé naturellement dans le cadre de l'ECF pour maîtriser pleinement les fondamentaux du back-end. J'ai pu construire manuellement un système de routage, de contrôleurs et de sécurité avec des sessions utilisateur. J'ai vu en profondeur la mécanique HTTP, la gestion des formulaires, une architecture MVC simplifiée et des appels AJAX.

MySQL : la base relationnelle MySQL a été utilisée pour modéliser toutes les entités centrales de l'application à savoir user, role, event, eventParticipant, favorite_event, validation (et newsletter pour une éventuelle continuité dans le développement). L'usage de MySQL m'a permis d'avoir un langage SQL structuré, idéal pour l'intégrité des données, rapide et simple à mettre en place et à importer/exporter.

MongoDB : pour inclure une base de données NoSQL, j'ai choisi MongoDB pour stocker l'historique des scores de chaque joueur. Le format JSON est parfaitement

adapté à ce type de donnée souple et imbriquée (score, timestamp, event_id, user_id). Ce choix m'a permis d'explorer une approche non relationnelle, complémentaire à MySQL. La consultation des scores se fait uniquement en local, car l'hébergement gratuit AlwaysData ne permet pas l'utilisation de MongoDB en ligne. Le code est cependant prévu pour s'adapter automatiquement : en production, la connexion est désactivée pour éviter toute erreur.

HTML : j'ai créé une structure HTML5 conforme aux bonnes pratiques de développement web. Chaque page est organisée avec des balises explicites afin de garantir une bonne lisibilité du code, une accessibilité de base et une compatibilité avec le référencement naturel (SEO). L'utilisation de balises de titre hiérarchiques (<h1> à <h4>) et de structures en grille contribue à améliorer l'expérience utilisateur. Tous les formulaires sont conçus avec des champs required et des labels associés, ce qui améliore la compatibilité avec les lecteurs d'écran et les outils d'assistance.

SCSS (compilé en CSS) : pour le style, j'ai choisi SCSS, qui est un préprocesseur CSS qui facilite grandement l'organisation et la lisibilité du code. Il permet de réduire le nombre de lignes de code et gérer au mieux chaque partie du site. J'ai pu centraliser mes variables de couleurs et polices, découper le style par composants (modal, dashboard, popup etc), éviter les répétitions (grâce aux @mixin qui sont réutilisés). J'ai utilisé l'extension Live Sass Compiler qui m'a permis de compiler en CSS pour être compatible avec le navigateur.

JavaScript (AJAX) : les actions les plus interactives de l'application utilisent des appels AJAX natifs avec fetch(), sans bibliothèque externe. Cela m'a permis de proposer une expérience utilisateur fluide et moderne, notamment pour les filtres dynamiques dans la liste des événements, les panneaux ouverts dans les dashboards (que j'avais initialement codé avec des redirections vers d'autres pages), la mise à jour des favoris et l'affichage de popups d'informations. Ces requêtes sont gérées proprement côté back avec des scripts dédiés dans api/.

Je reviendrais sur Always Data dans la partie déploiement (page 7)

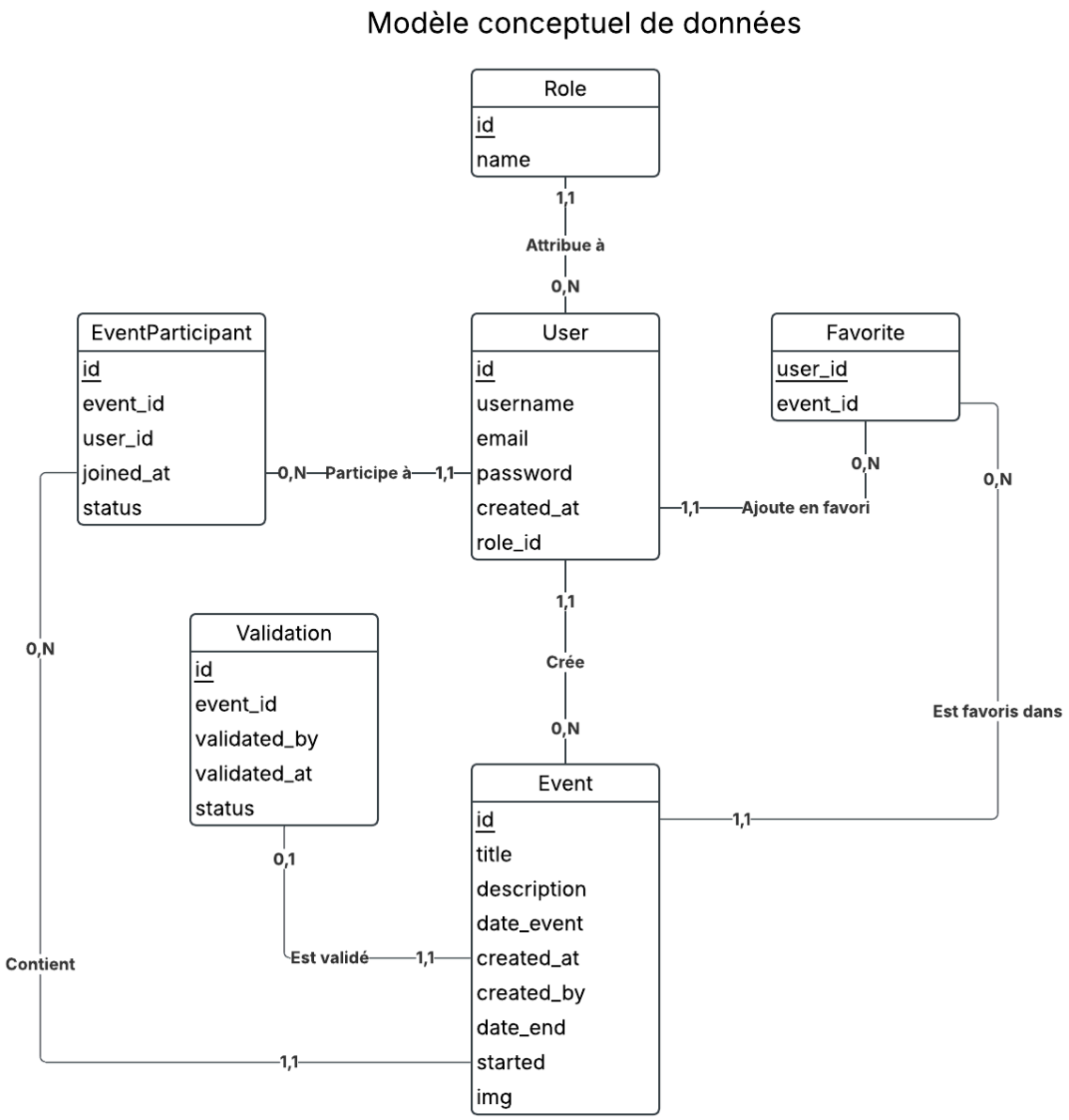
2. Environnement de travail

En ce qui concerne la configuration de mon environnement de travail :

- mon système d'exploitation est Windows 10
- j'ai utilisé un serveur local avec WAMP (que j'avais mis en place avant pour d'autres projets)
- mon éditeur a été VSCode avec les extensions que j'ai ajoutées où que j'avais déjà installées (Intelephense, Live Sass Compiler, Prettier seulement pour la lecture plus simple du code, Intellisense)
- en version PHP j'ai la 8.1
- Git et Github en passant par le bash Git et directement depuis VSCode pour effectuer des commit (Github Copilot extension)
- pour ma BDD relationnelle j'ai utilisé phpMyAdmin en me connectant localement
- pour ma BDD noSQL j'ai installé MongoDB Compass qui est une interface simple pour gérer les collections
- FileZilla pour le transfert en FTP (où je me suis connecté avec mes identifiants Always Data)
- Always Data pour le serveur en production (hébergement gratuit)

3. Modèle conceptuel de données (MCD)

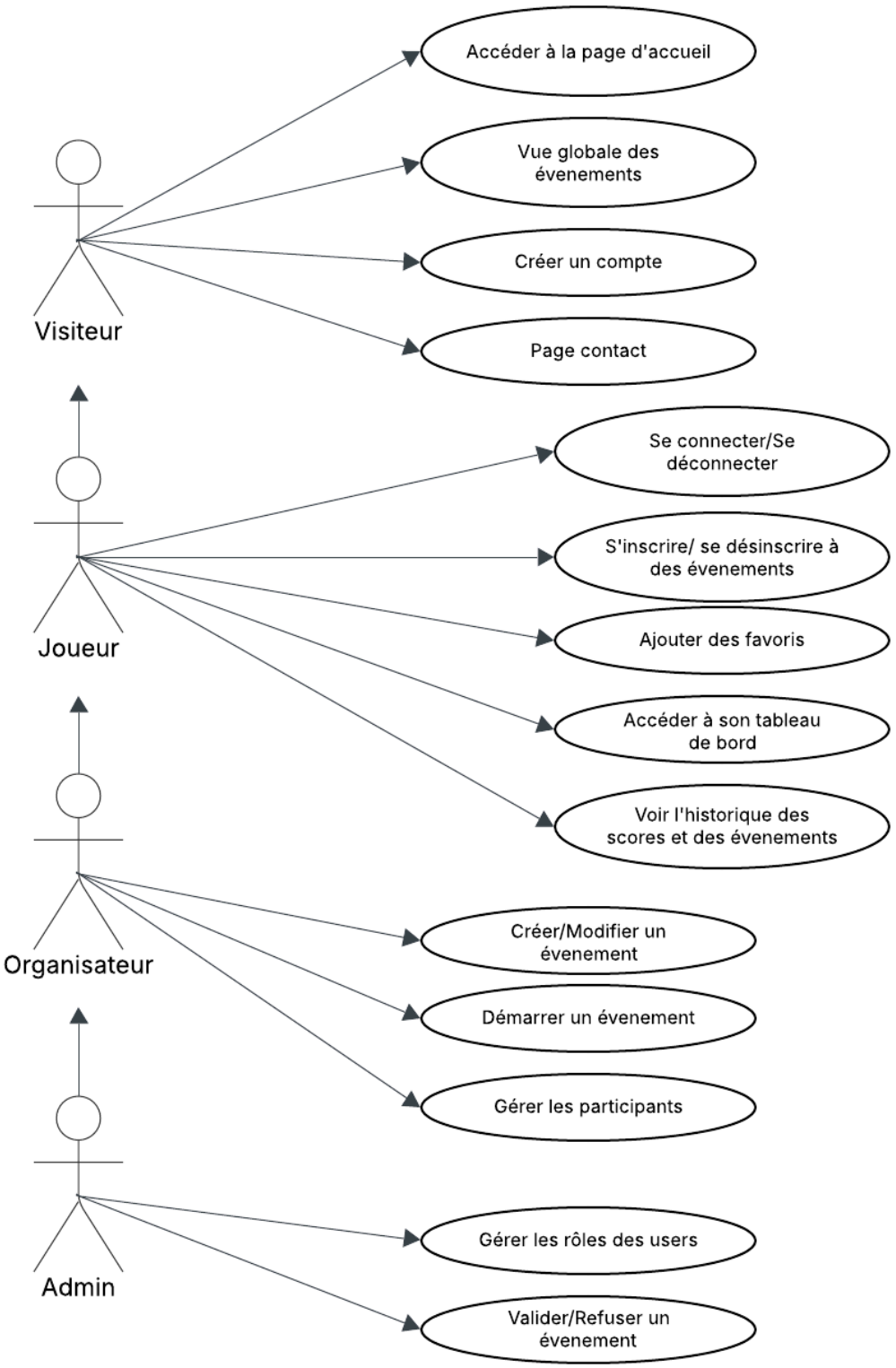
Dans mon projet, il y a plusieurs entités principales qu'il a fallu représenter pour pouvoir appréhender les différents attributs et relations entre entités. Voici mon MCD que j'ai créé sur LucidChart :



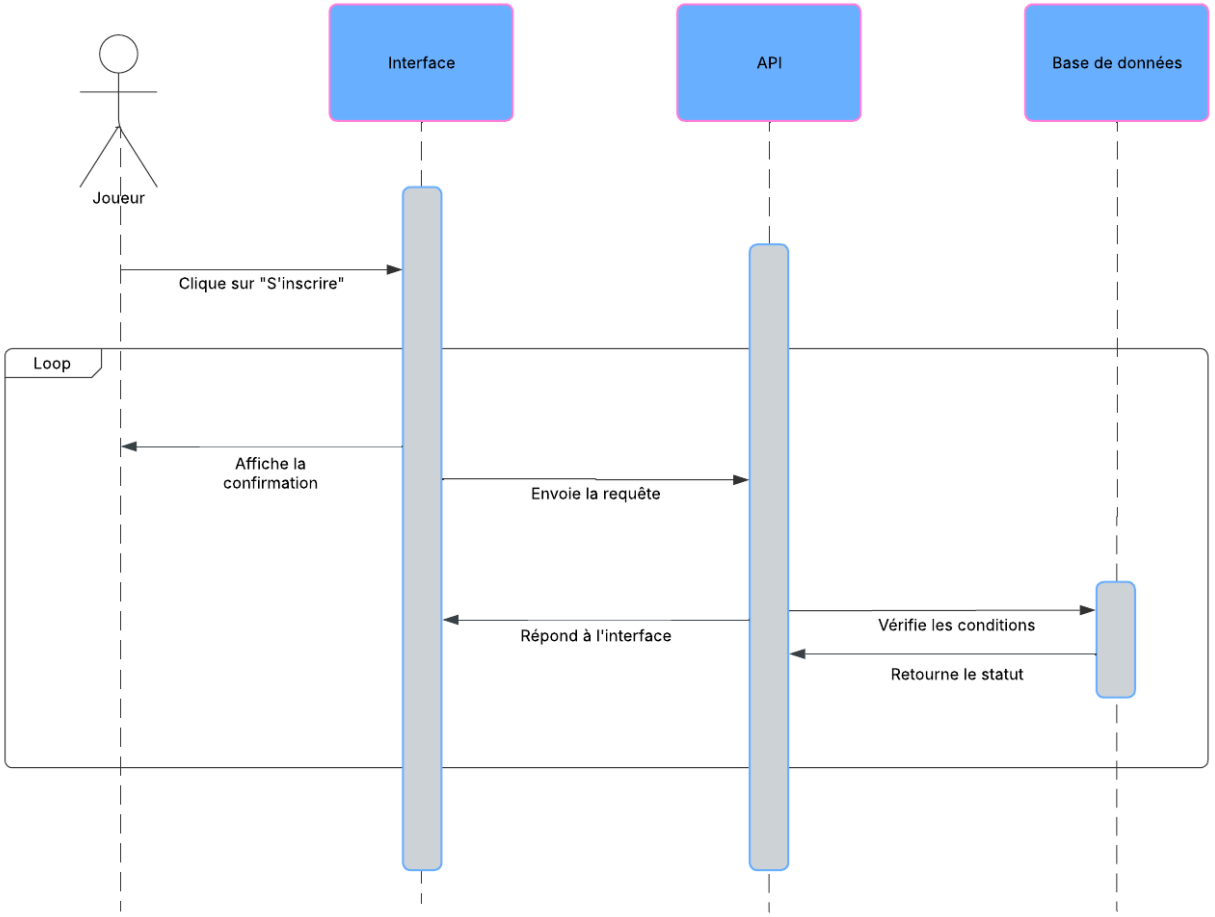
Il faut savoir que j'ai ajouté la table "newsletter" dans ma base de données relationnelle plus tard.

4. Diagrammes d’utilisation & de séquence

Pour comprendre au maximum les interactions des différents utilisateurs ainsi que les fonctionnalités à développer, il a fallu créer des diagrammes les plus représentatifs possibles de la réalité. Voici le diagramme d’utilisation globale :



Et voici le diagramme de séquence qui représente l'inscription d'un joueur à un event (fait sur LucidChart aussi) :



5. Déploiement

Mon projet a été mis en ligne avec la plateforme Always Data. J'ai choisi cet hébergeur car il est gratuit, français et compatible avec PHP et MySQL. Je n'ai pas anticipé le côté payant avec le noSQL. J'ai réalisé le déploiement de manière manuelle pour conserver la maîtrise complète de chaque étape. Je vais vous détailler ma démarche et les étapes que j'ai réalisées.

Création d'un site web sur Always Data :
J'ai créé un site type PHP avec la version PHP 8.1 configuré sur le dossier racine /www/public (car mon index.php ainsi que les fichiers accessibles par les utilisateurs sont dans le dossier public/). L'adresse a été attribuée automatiquement.

Import de la base MySQL :
J'ai utilisé l'outil phpMyAdmin intégré à Always Data en créant une base nommée esportify et en important dans cette base mon fichier esportify.sql (structure + exemples de données).

Configuration de la connexion :
J'ai modifié le fichier config/db.php uniquement pour le serveur distant pour qu'il soit lié à la BDD. J'ai changé le host, dbname, username et password pour qu'ils correspondent à ceux de mon Always Data.

Transfert des fichiers du projet :
J'ai utilisé FileZilla pour pouvoir transférer mes fichiers en FTP sur le serveur distant. J'ai déposé mon dossier public/ ainsi que views/, config/, controllers/, scripts/, helpers/ et data/. Je n'ai pas transféré vendor/ car MongoDB ne fonctionnant pas, les dépendances PHP n'étaient pas requises.

Blocage automatique de MongoDB :
J'ai eu un bug au premier test du déploiement et j'ai fini par comprendre qu'il venait du fait que l'hébergeur ne supporte pas MongoDB. J'ai dû changer une condition dans le fichier scores.php pour qu'il désactive automatiquement la tentative de connexion au client Mongo.

Je remets ci-après le lien de mon projet en ligne :
https://ocewan.alwaysdata.net