Greta-Cfa
Formation | Accompagnement | Apprentissage

wavy
WAVE OF SUCCESS

DOSSIER PROJET SITE INTERNET

« Organisation de tournois de jeu vidéo Fortnite » pour l'agence Wavy Group

Réalisé par
Christelle Pakulic

Résumé

Dossier projet site internet « Organisation de tournois de jeu vidéo Fortnite »
pour l'agence Wavy Group
Titre Professionnel Développeur Web & Web Mobile - 2021

Le projet a été initié dans le cadre du stage professionnel (juillet-août 2021) au sein de l'agence d'e-sport Wavy Group, avec deux autres stagiaires. J'ai choisi de le poursuivre et de le perfectionner durant la seconde partie de la formation. Il a pour objectif d'organiser des tournois du jeu vidéo Fortnite. Les joueurs doivent pouvoir se créer un compte, gérer leurs données, organiser des tournois, y participer et accéder aux classements.

Dès les prémices du projet, nous avons défini des méthodes de travail, le cahier des charges, les maquettes et la conception de la base de données (méthode MERISE). Sur la base de ces travaux, nous avons réalisé les développements HTML/CSS en mode responsive, en respectant les standards W3C. Pour la partie dynamique, j'ai utilisé JavaScript et le Framework VueJS pour quelques évènements au clic, mais surtout pour sécuriser les champs de formulaires qui sont ensuite revérifiés en PHP.

Pour la partie back-end, nous avons réalisé les développements en PHP. J'ai ensuite retravaillé ces derniers en intégrant la programmation orientée objet, en appliquant les recommandations de sécurité et de référencement.

En termes de perspectives, il est prévu de faire évoluer le projet vers une architecture MVC afin d'améliorer la maintenance et renforcer la sécurité. Des travaux seront à poursuivre, notamment pour la partie détail d'un tournoi (modifications, inscriptions et classements), la traduction en anglais, le système d'e-mailing et le partage avec les réseaux sociaux, ainsi que la connexion avec la base de données d'Epic Games.

Summary

This project started during the internship (July-August 2021) in the e-sport agency Wavy Group with two other interns. I chose to continue and perfect this in the second part of the training. The objective of this project is to organize tournaments for all fornite players. Players must be able to create an account, manage their data, organize tournaments, participate and access to rankings.

First, we established a project management, wrote the specifications with the agency's director, created the mock up and models of the database (MERISE method). Based on these works, we developed the frond-end of the application in native html/css in responsive mode while respecting the W3C standards. Then, I realized the dynamic part with VueJs framework to finalize the navigation and secure the form fields, that are then rechecked in PHP.

For the back-end part, we realized the developpements in php. I improved this development with the technique we learned in the second part of the training : the object-oriented programming, applying security recommendations (checking input and preparing sql queries) and referencing.

This is something I am going to finish after the conclusion of the training. I'm especially going to focus on applying the MVC architecture in order to improve maintenance, creating the detailed part of a tournament, adding the English translation, sharing on social networks, emailing and connecting to the game's database (Epic Games).

Sommaire

I. Introduction 4
II. Cahier des charges 4
A. Le projet 4
1. Les objectifs 4
2. Le périmètre et les contraintes du projet 4
B. Conceptualisation de l'application 5
C. Graphisme 6
D. Description fonctionnelle 7
1. Page d'accueil 7
2. Espace utilisateur : création de comptes et connexion 8
3. Espace utilisateur : accueil et paramètres de compte 9
4. Espace utilisateurs : organiser un tournoi 10
5. Interface administrateur : 11
E. Conception de la base de données 12
III. La gestion du projet : 14
1. Ressources : équipe et budget 14
2. Organisation du travail 14
3. Planning : 14
IV. Spécifications techniques 15
1. Technologies front-end : 15
2. Technologies back-end : 15
3. Sécurité de l'application 16
4. Versionning 17
V. Compétences du référentiel 18
A. Maquetter une application (CP1) : 18
B. Interface statique et adaptable (CP2) 20
C. Interface web dynamique (CP3) 22
D. Création d'une base de données (CP5) : 24
E. Composants d'accès aux données (CP6) 26
F. Partie back-end de l'application web (CP7) 28
VI. Réalisation personnelle 30
VII. Jeu d'essai 31
VIII. Situation de travail : veille et recherche de solutions 32
A. Description de la veille en termes de sécurité 32
B. Situation de travail ayant nécessité une recherche (à partir d'un site anglophone) 33
IX. Conclusion et perspectives 34
X. Annexes 35

I. Introduction

Le projet de site internet a été initié, avec deux autres stagiaires, dans le cadre d'un stage professionnel au sein de **l'agence de communication et de e-sport Wavy Group**. J'ai choisi ensuite de poursuivre le projet et de le perfectionner durant la seconde partie de la formation.

Il a pour objectif d'organiser des tournois du jeu vidéo Fortnite. Il s'adresse à des joueurs de tout âge, de tout niveau et quelle que soit la technologie utilisée (pc, mobile, Nintendo, Playstation, Xbox). Les joueurs doivent, entre autres, pouvoir se créer un compte, gérer leurs données, organiser des tournois, y participer et accéder aux classements.

Wavy est un groupe Esport Européen fondé en 2018. Il possède trois branches d'activités :

- Une Agence de communication, elle accompagne les marques dans leurs objectifs de visibilité et notoriété auprès des millenials (15-35 ans)
- Une Agence événementielle, elle crée et fait la promotion de compétitions ou d'événements online et physique.
- Une Agence Esport, Wavy est un club Esport professionnel.

L'Esport désigne la pratique sur internet d'un jeu vidéo seul ou en équipe par le biais d'un ordinateur, d'une technologie mobile (tablette, téléphone) ou d'une console de jeux (Nintendo, Xbox, Playstation). Wavy Group possède des équipes de joueurs pour les jeux Fortnite, Rocket league, League of legends, Fifa, GTA (source : https://wavygroup.fr/team/)

II. Cahier des charges

A. Le projet

1. Les objectifs

Les objectifs stratégiques de ce site sont :

- **Donner un univers aux joueurs** : tournois, classements, avec la possibilité d'organiser leurs propres tournois
- **Générer du trafic sur le site de Wavy Group**
- **Faire connaître Wavy auprès d'un maximum de personne** (notoriété)

L'objectif quantitatif serait dans l'idéal de 2000 joueurs par semaine

☛ *Plus de détails sur le profil des utilisateurs et sur les objectifs en annexe 1.*

2. Le périmètre et les contraintes du projet

Le site doit être :

- Multilingue : anglais/français
- Adapté aux différents écrans (mobile / tablette) et aux principaux navigateurs actuels
- Bien référencé

Des connexions avec les outils existants :

- La base de données du jeu (constructeur : Epic Games) pour l'authentification et la remontée de données (classement)
- Lien avec le site principal : https://wavygroup.fr/ (accès au site et leaderboard)

À noter : ces travaux de connexion ne font pas partie du périmètre du projet, mais ils doivent être pris en considération.

Considérations relatives à la sécurité :

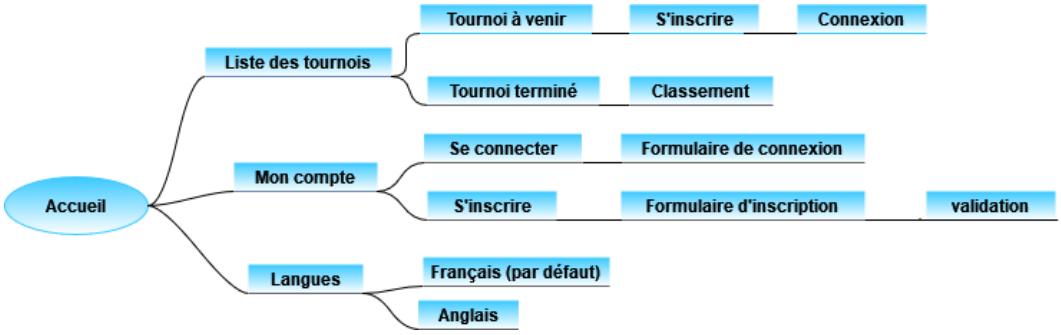
- Le site contient une base de données et de nombreux formulaires à sécuriser (création de compte et paramètre de compte, organisation d'un tournoi, création d'une équipe).

B. Conceptualisation de l'application

Un produit minimum viable (MVP ou Minimum Viable Product en anglais) indique le périmètre (minimum requis) pour la première version du site avant sa mise en production. Pour notre projet, il est défini ci-dessous (avant et après authentification).

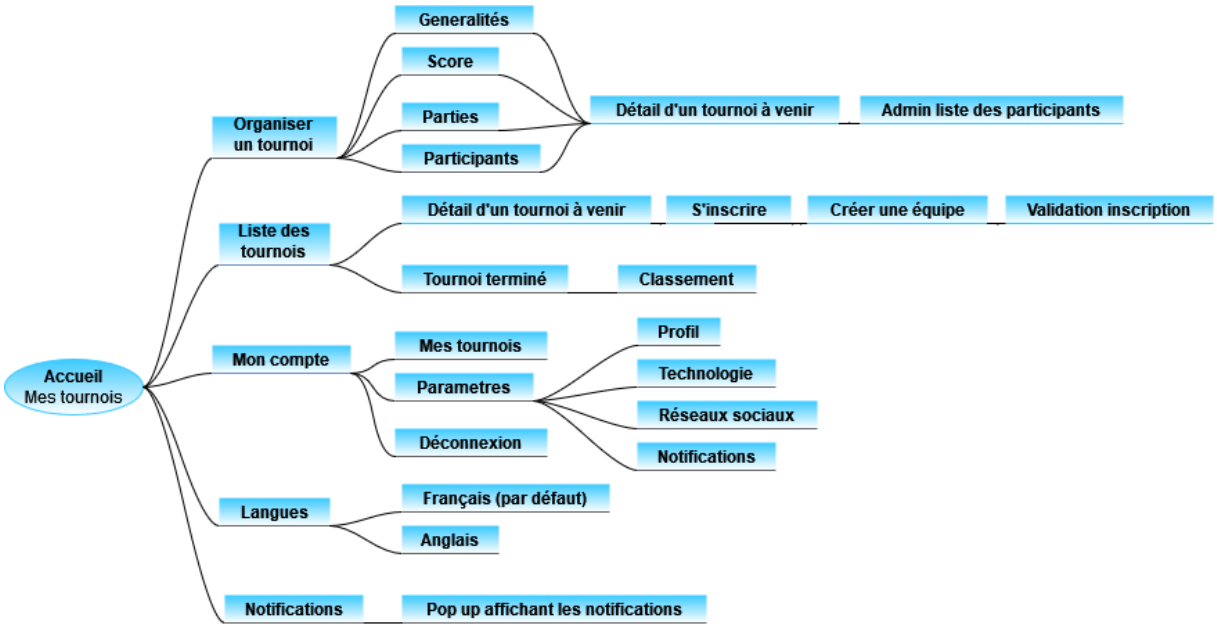
Disponible sans authentification :

- Menu principal (logo, accueil et lien vers la page tournois), avec la **traduction** anglais/français et l'accès à la connexion au compte.
- **Page d'accueil**
- **Page « liste des tournois » et « détail d'un tournoi »** (« à venir » ou « terminé », avec classement quand terminé)
- **Formulaires d'inscription / connexion** à l'espace utilisateur



Disponible seulement après authentification

- Menu principal (logo, accueil, liens vers la page tournois et la page organisation de tournois), la traduction anglais/français, les **notifications (avec une fenêtre « notifications »).**
- Menu utilisateur avec :
  - **Mes tournois** (tournois auxquels l'utilisateur participe ou qu'il organise)
  - **Mes paramètres de compte** (profil, technologies, réseaux sociaux, notifications)
- Page **« organiser un tournoi »,** ainsi qu'une page d'administration des participants
- Page « liste des tournois » (comme hors connexion)
- **Possibilité d'inscription aux tournois,** depuis la page « détail d'un tournoi » (avec réutilisation ou création d'une équipe)



À noter : le footer reste identique : liens vers des pages statiques (mentions légales, CGU), liens vers le site principal (qui sommes-nous, boutique) *et les réseaux sociaux de Wavy.*

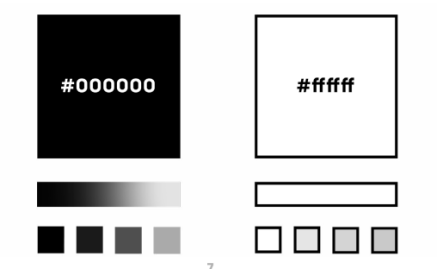
C. Graphisme

1. Aperçu de la charte graphique :

Logo

Le logo représente deux vagues l'une sur l'autre.
2 symboles sont cachés. Il s'agit de deux poissons qui sourient.
Il est noir, car les vecteurs de Wavy sont : simplicité, classe et sérieux.

Palettes de couleurs :



Polices :

2. Univers graphique et planche de style choisie

Le site internet requiert une palette de couleurs assez sombres notamment pour le fond de l'écran, car il s'adresse à un public (joueurs de jeu vidéo) qui reste très longtemps devant l'écran.
Les couleurs choisies sont une déclinaison de la charte (noir, blanc) auxquelles nous avons choisi d'ajouter des nuances de gris et une couleur d'appel : le bleu. Cette dernière nous permettra d'attirer le regard à des endroits précis (appel au clic…).

Palette de couleur :

Noir #000000	Gris Foncé #272727	Gris Moyen #4A4A4A	Gris clair #9E9E9E	Blanc #FFFFFF	Bleu #3AC9FF

Typographie de titre

Mazzard H Bold

Typographie de contenus

Mazzard H ExtraLight

Design graphique : Flat
En lien avec l'univers de jeu vidéo, le site pourra aussi contenir ponctuellement quelques animations pour l'affichage des contenus et des images.

☛ *Vous trouverez les **références utilisées** pour le graphisme en annexe 2 et l'explication des **travaux de maquettage dans le chapitre « maquetter une application »** .*

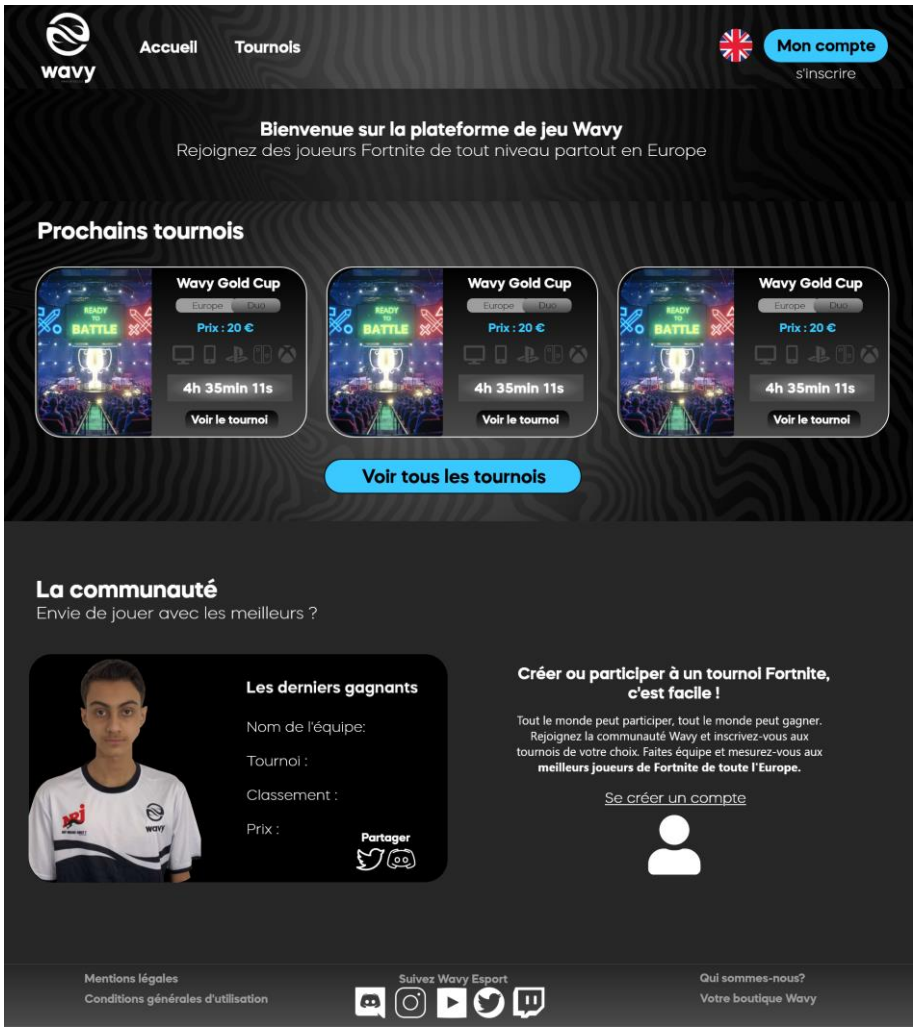
D. Description fonctionnelle

1. Page d'accueil

Cette page doit permettre de :

- Présenter l'objectif du site (accroche)
- Promouvoir les derniers tournois (cf. détail d'un résumé ci-dessous).
- Donner envie de faire partie de la communauté Wavy : jouer en équipe (lien vers Discord) et mettre en avant les gagnants et ce qu'ils ont gagné et proposer la création de comptes.

Maquette Page d'accueil version desktop



Extrait version mobile

Le résumé d'un tournoi à venir contient :

- Le nom du tournoi (titre)
- La photo de couverture du tournoi
- La région et le mode de jeu
- Le prix
- Les technologies sur lesquelles on peut jouer
- Un timer (dd hh :mm ou hh :mm :ss le jour j)
- Un bouton « voir le tournoi » qui renvoie vers la page de détail du tournoi

Wireframe résumé d'un tournoi à venir

☛ *Les pages « liste tournois », « détail d'un tournoi » et la gestion des inscriptions sont décrites en annexe 3.*

2. Espace utilisateur : création de comptes et connexion

Les connexions possibles :

Il est possible de se connecter :

- **Directement via l'interface Wavy** avec le pseudo Epic Games. Dans ce cas, c'est le pseudo qui permettra la connexion avec la base de données Epic Games.
- **Avec un compte Epic Games.** En cas de connexion avec Epic Games, on fait remonter les informations disponibles dans ces bases de données pour compléter les champs du profil.

À noter : l'option avec Epic Games ne sera pas décrite dans ce cahier des charges, car elle ne fait pas partie du périmètre des travaux définis avec l'entreprise, néanmoins je me suis intéressée à cette authentification (cf. travaux de recherche p.33).

Création d'un compte avec le site Wavy

Si l'utilisateur choisi de se créer un compte Wavy, un formulaire d'inscription lui demandera de renseigner les champs suivants :

- Pseudo Epic Games
- Mot de passe
- Confirmation du mot de passe
- Mail
- Compte Discord

Les conditions pour pouvoir se créer un compte Wavy :

- Acceptation des conditions générales d'utilisation
- Acceptation de l'utilisation de ses données personnelles (pour recevoir de l'information de la part de Wavy)
- Un seul compte : pas de possibilité d'utiliser deux fois le même mail ou pseudo (lien avec le compte Epic Games)
- Confirmation du mail à partir du lien

Connexion de compte :

Pour se connecter via l'interface Wavy, l'utilisateur doit :

- Saisir son login (adresse mail de l'utilisateur)
- Saisir son mot de passe
- Cliquer sur "valider".

En cas d'oubli du mot de passe, il clique sur "mot de passe oublié". Ce lien renvoie vers un formulaire de réinitialisation du mot de passe.
On note qu'on laissera à l'utilisateur la possibilité de se connecter via Epic Games.

Maquette formulaire de création de comptes Wavy –version mobile

3. Espace utilisateur : accueil et paramètres de compte

Page d'accueil de l'espace : « mes tournois »

Quand il est connecté, l'utilisateur est redirigé vers la page « mes tournois ».
Cette page ressemble au niveau design à la page « liste des tournois », car elle donne accès à la liste des tournois auxquels l'utilisateur participe ou qu'il organise, à la différence qu'elle affiche également les tournois privés.
Cette page contient la possibilité de filtrer les tournois :

- Par visibilité (public/privé)
- Niveau de participation (participant /organisateur)

Les notifications

L'utilisateur peut recevoir des informations via l'interface (et/ou par mail) concernant :

- « Mon classement » : photo, nom de l'équipe, tournois, classement, prix (s'il existe), lien vers le tournoi et la possibilité de partage sur les réseaux sociaux : Discord et Twitter
- « Les derniers gagnants » : même information que pour « mon classement »

L'activation/ désactivation de ces notifications se fait via les paramètres de compte.

Les paramètres de compte

Depuis la page « paramètre de compte », l'utilisateur a la possibilité de modifier différents paramètres.

La modification du profil - Photo, sinon il y aura une image par défaut : logo Wavy. - Pseudo joueur* - Adresse mail* - Mot de passe* - Confirmation du mot de passe* - Téléphone - Code postal (option qui pourrait permettre de filtrer les joueurs d'une même zone géographique)	**La (ou les) technologies utilisée(s)** - PC - Mobile - Nintendo - Playstation - Xbox. Ces informations permettront de filtrer automatiquement la liste des tournois dès que l'utilisateur est connecté
Les réseaux sociaux de l'utilisateur - Discord* - Twitter - Twitch - Instagram. Ces informations servent au partage d'information (gagnants, tournois, classements) sur les réseaux sociaux notamment Twitter et Discord.	**La configuration des notifications** - Choisir les informations : classement et/ou derniers gagnants - Choisir le mode de notifications : par mail et/ou dans l'interface Ces informations conditionnent l'envoi des informations dans la fenêtre notifications et/ou par mail.

*(Les champs obligatoires sont notés avec un *)*

4. Espace utilisateurs : organiser un tournoi

Pour créer un nouveau tournoi, l'utilisateur doit être connecté avec son compte. Il accède à la page en cliquant sur "organiser un tournoi" dans le menu principal. Il doit ensuite renseigner tous les champs de chaque étape (généralité, score, parties, participants). Le tournoi ne peut être validé qu'après le passage par les 4 étapes et avoir cliqué sur le bouton "valider".

Ces 4 étapes sont :

Étape 1 : "Généralité" ❖ Le nom du tournoi* ❖ La photo (affiche du tournoi) * ❖ Le prix ❖ Le lien discord ❖ La description ❖ La région * ❖ Les technologies utilisées *	**Étape 2 : "Score"** ❖ Le nombre de joueurs ou d'équipes minimum par partie (sinon la partie sera annulée) * ❖ Les champs 'top' et 'points' ❖ Le nombre de points par élimination ❖ Le nombre d'éliminations maximales ❖ Les règles du jeu *Si les informations « top » et « points » ne sont pas renseignés, il n'y aura pas de classement*
Étape 3 : "Parties d'un tournoi" ❖ Les dates et heures de début et fin de tournois. * ❖ Les nom(s), date(s) et heure(s) des parties	**Étape 4 : "Participants"** ❖ Le nombre maximum de participants* ❖ Le mode de jeu (solo/ duo/trio / squad) * ❖ La visibilité (public/privé) * ❖ Le début et la fin des inscriptions. Par défaut, ils seront identiques à ceux du tournoi

*(Les champs obligatoires sont notés avec un *)*

Quand l'utilisateur a créé un tournoi, un mail de confirmation lui est envoyé avec :

- ❖ L'URL du tournoi : l'utilisateur peut inviter d'autres joueurs (via Discord) à s'inscrire grâce au lien du tournoi
- ❖ L'URL pour modifier les participants (Il doit être connecté pour y avoir accès) : l'organisateur doit pouvoir créer ou supprimer une équipe, modifier l'attribution des équipes (changer un joueur d'équipe) et/ou supprimer un joueur.

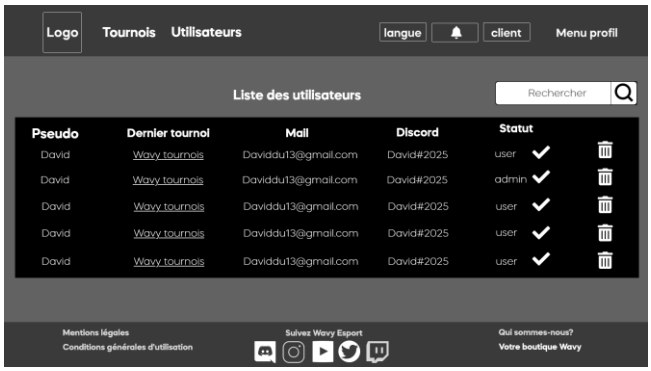
5. Interface administrateur :

Règles de fonctionnement :

- Un administrateur est un utilisateur ayant plus de droits sur le site qu'un utilisateur (cf. droits supplémentaires ci-dessous).
- **Il peut y avoir plusieurs administrateurs, mais un seul administrateur principal.**
  Un administrateur doit pouvoir attribuer un rôle d'administration à un utilisateur existant ou le lui retirer. Cependant aucun administrateur ne peut retirer le rôle d'administration à l'administrateur principal.

Les droits d'administration des utilisateurs :

Wireframe de la page liste des utilisateurs



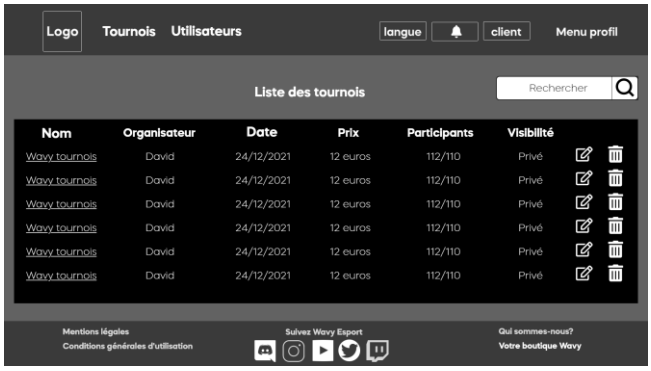
Cette page affiche la liste des utilisateurs de la base de données de Wavy.

Elle permet de :

- Rechercher un utilisateur par son nom(pseudo)
- Visualiser les informations : pseudo/dernier tournoi /mail / Discord
- Modifier le statut (user/admin)
- Supprimer un utilisateur

Les droits d'administration des tournois :

Wireframe de la page liste des tournois



Cette page liste tous les tournois dans un tableau (comme utilisateur) avec :

- Nom du tournoi
- Organisateur
- Date du tournoi
- Prix
- Nombre de participants
- Visibilité (public/privé)

Cette page permet de :

- Rechercher un tournoi par son nom
- Modifier ou supprimer un tournoi, quel que soit l'organisateur
- Pouvoir intervenir sur la liste des participants, quel que soit l'organisateur du tournoi

E. Conception de la base de données

Afin de concevoir la base de données, j'ai utilisé la méthode MERISE. Elle se compose de 3 étapes : le modèle conceptuel de données, le modèle logique de données et le modèle physique de données.

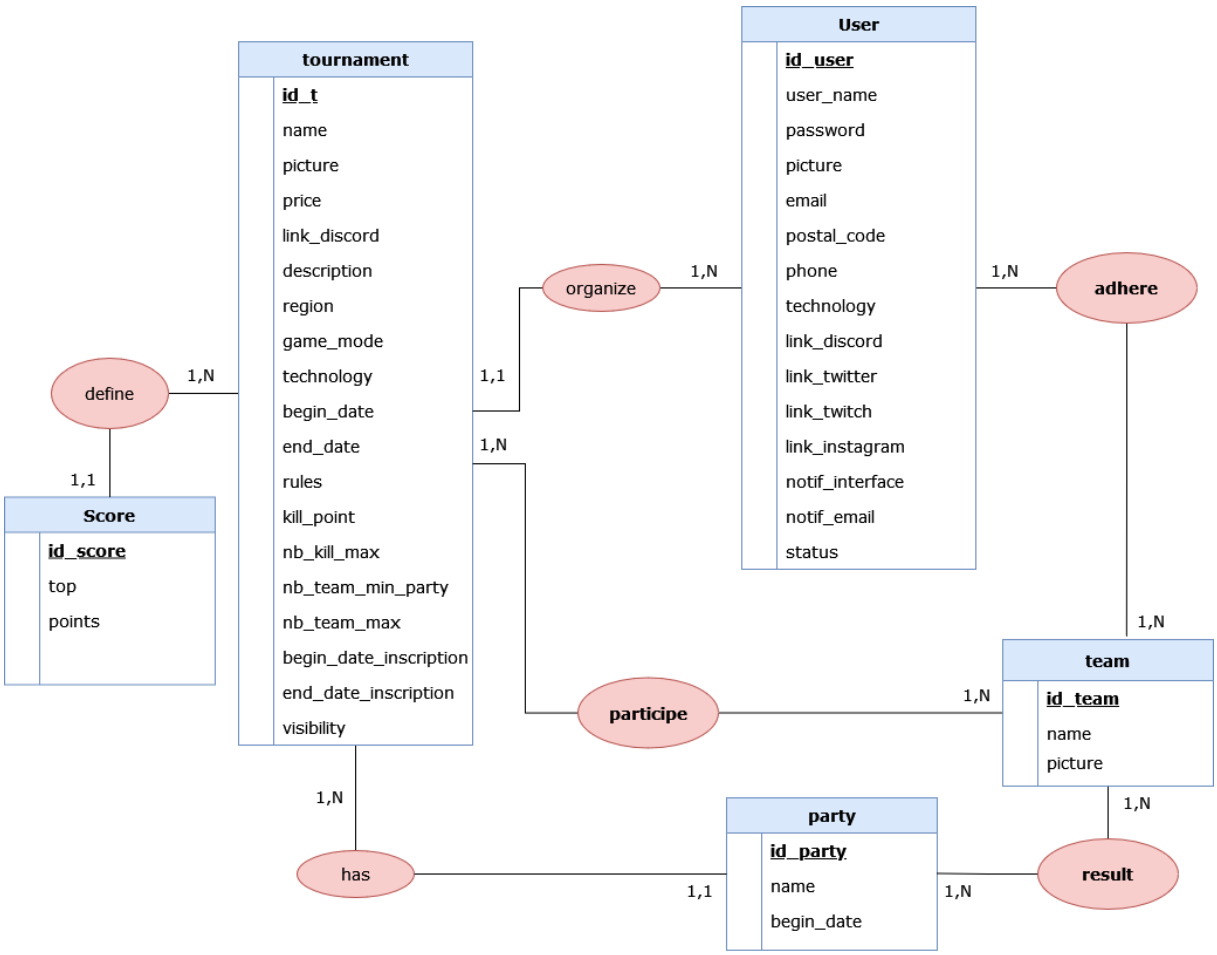
1. Modèle conceptuel de données (MCD)

Ce premier modèle permet de définir les entités et leurs relations.
Nous avons donc **5 entités** (tournament, user, team, party, score). Chaque entité possède une liste de **propriétés (ou attributs).**

On définit ensuite :

- **Les relations entre les entités**, qui sont nommées avec des verbes (par exemple 'organize' est la relation entre « user » et « tournament »)
- **Les cardinalités** (les quantités minimum et maximum qui peuvent exister entre deux entités). Exemple pour la relation user-tournament :
  Un utilisateur peut créer un ou plusieurs tournois => minimum :1, maximum : plusieurs (N)
  Un tournoi peut être créé par un seul utilisateur => minimum : 1, maximum 1.

Représentation graphique du modèle conceptuel de données (réalisé avec https://app.diagrams.net/)



2. Modèle logique de données (MLD)

Ce deuxième modèle permet, à partir du modèle conceptuel et notamment des cardinalités, de définir les clés primaires, les clés étrangères et dans le cas où les deux cardinalités sont « N », il y a **création d'une table de jonction.** Ici, nous en créons trois : participant (participe), result et adhesion (adhere).

Ci-dessous les tables avec les **clés primaires** et ***étrangères*** :

- PARTY (**id_party**, name, begin_date, ***id_t***)
- TOURNAMENT **(id_t**, name, picture, price, link_discord, description, region, game_mode, technology, begin_date, end_date, rules, kill_point, nb_kill_max, nb_team_min_party, nb_team_max, begin_date_inscription, end_date_inscription, visibility, ***id_user***)
- TEAM (**id_team**, name, picture)
- SCORE **(id_score**, top, points, ***id_t***)
- USER (**id_user**, user_name, password, picture, email, postal_code, phone, technology, link_discord, link_twitter, link_twitch, link_instagram, notif_interface, notif_email, status)
- PARTICIPANT (**id_participant**, rank, price, nb_party, nb_victory, points, ***id_t, id_team***)
- RESULT (**id_result,** status, placement, kills, ***id_party, id_team***)
- ADHESION **(id_adhesion,** ***id_user, id_team***)

3. Modèle physique de données (MPD)

Le MPD est l'implémentation directe du MLD. On ajoute **les types de données de chaque champ**.

- PARTY (**id_party [INT UNSIGNED NOT NULL AUTO_INCREMENT],** name[VARCHAR(30)], begin_date[datetime], ***id_t***)
- TOURNAMENT **(id_t [INT UNSIGNED NOT NULL AUTO_INCREMENT],** name[VARCHAR(50)], picture[blob], price[VARCHAR(50)], link_discord[VARCHAR(250)], description[VARCHAR(250)], region[SET('Monde','Europe','NAE','NAW','Asie','Oceanie','Moyen_Orient','Bresil')], game_mode[SET('solo', 'duo ','trio','squad')], technology[SET('pc', 'mobile', 'nintendo', 'xbox', 'playstation')], begin_date[datetime], end_date[datetime], rules[VARCHAR(250)], kill_point[INT], nb_kill_max[INT], nb_team_min_party[INT], nb_team_max[INT], begin_date_inscription[datetime], end_date_inscription[datetime], visibility[SET('public','private')], ***id_user***)
- TEAM (**id_team[INT UNSIGNED NOT NULL AUTO_INCREMENT],** name[VARCHAR(50)], picture [blob])
- SCORE **(id_score[INT UNSIGNED NOT NULL AUTO_INCREMENT],** top[INT], points[INT], ***id_t***)
- USER (**id_user[INT UNSIGNED NOT NULL AUTO_INCREMENT**], user_name[VARCHAR(50)], password[VARCHAR(250)], picture[blob], email[VARCHAR(250)], postal_code[VARCHAR(30)], phone[VARCHAR(20)], technology[SET('pc', 'mobile', 'nintendo', 'xbox', 'playstation')], link_discord[VARCHAR(250)], link_twitter[VARCHAR(250)], link_twitch[VARCHAR(250)], link_instagram[VARCHAR(250)], notif_interface[SET('classement', 'gagnant')], notif_email[SET('classement', 'gagnant')], status[SET('user', 'admin')])
- PARTICIPANT (**id_participant[INT UNSIGNED NOT NULL AUTO_INCREMENT],** rank[INT], price[VARCHAR(50)], nb_party[INT], nb_victory[INT UNSIGNED NOT NULL AUTO_INCREMENT], points[INT], ***id_t, id_team***)
- RESULT (**id_result[INT UNSIGNED NOT NULL AUTO_INCREMENT],** status[VARCHAR(50)], placement[INT], kills[INT], ***id_party, id_team***)
- ADHESION **(id_adhesion[INT UNSIGNED NOT NULL AUTO_INCREMENT**], ***id_user, id_team***)

☛ *Pour la ba*se de données, cf. chapitre « création d'une base de données ».

III. La gestion du projet :

1. Ressources : équipe et budget

L'équipe est constituée d'un directeur, trois stagiaires bénévoles et un développeur back-end senior.

	Ressources
Conception (cahier des charges, graphisme, développement)	Unités d'œuvres (bénévoles)
Nom de domaine : non validé (proposition : wavy-Fortnite.com)	Environ 10 euros/ an chez ovh
Hébergement	Hébergement chez le développeur senior
Emailing	Sendinblue (gratuit jusqu'à 300 mails/jour puis 19 euros/ mois pour 10 000 mails/jours).
Maintenance	Unités d'œuvres (développeur back-end senior bénévole)

2. Organisation du travail

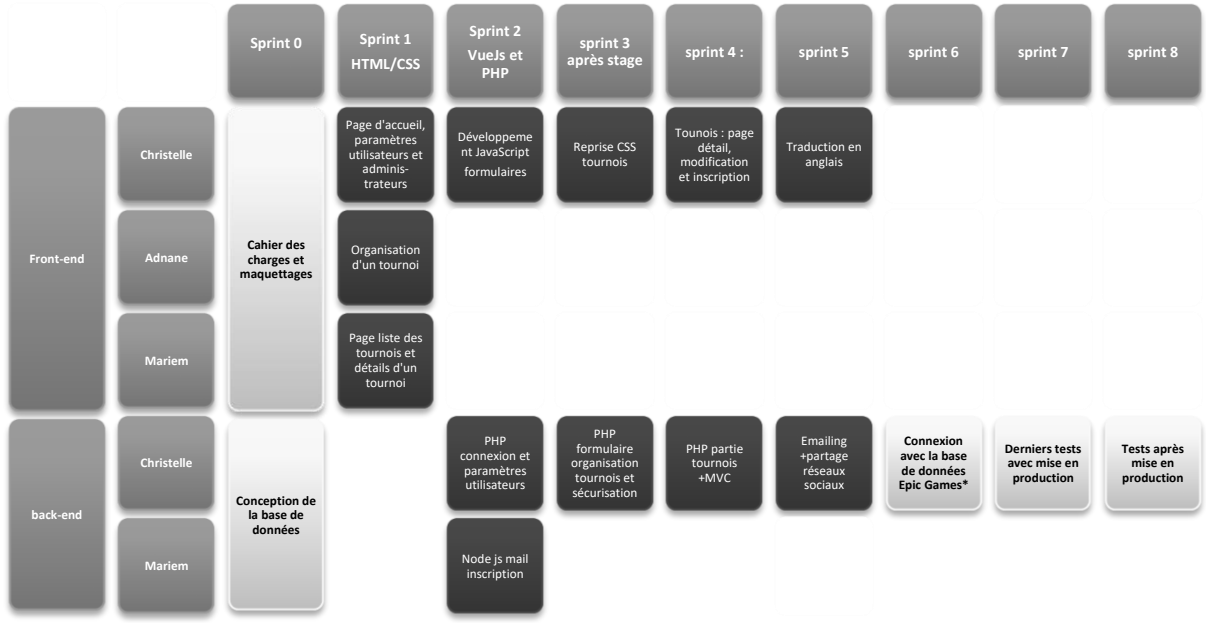
Dès les prémices du projet, nous avons défini des méthodes d'organisation du travail avec :

- Des réunions hebdomadaires avec le directeur de Wavy Group afin échanger régulièrement sur les besoins, valider chaque étape de travail.
- Des points quotidiens avec l'équipe et rapports d'activité de quelques lignes (action réalisée/ à réaliser /points de blocage éventuels) via Discord.

Nous avons également utilisé différents outils : Discord pour les échanges au quotidien, Trello pour la planification, Google drive pour le partage des fichiers importants (cahier des charges, charte graphique…) et GitHub pour le partage des développements et le versionning.

3. Planning :

Les travaux ont été planifiés selon 8 sprints. Le stage s'est terminé après le 2e sprint. Le projet n'a pas pu être finalisé durant les 34 jours de stages. J'ai choisi ensuite de poursuivre le projet et de le perfectionner durant la seconde partie de la formation.



IV. Spécifications techniques

1. Technologies front-end :

La partie front-end a été réalisée en langage **HTML, CSS et JavaScript (VueJs)**. Même si cela aurait pu réduire le temps des travaux au niveau responsive, j'ai choisi de ne pas utiliser de Framework pour le CSS afin de ne pas être dépendante de cette technologie et pouvoir bien dissocier les trois langages.

À propos du responsive, il est important de noter que, comme les utilisateurs (joueurs) peuvent utiliser aussi bien la technologie mobile que desktop, une version mobile demeure indispensable. Les maquettes et les développements CSS (**media queries**) ont donc été réalisés pour répondre à ce besoin selon la méthode **« mobile first** », c'est-à-dire réfléchir et développer pour répondre en premier à ce besoin.

Les développements VueJs concernent notamment la sécurisation des formulaires (complexité et correspondance des mots de passe, regex…) qui est ensuite revérifiée en PHP.

Le code HTML respecte les **normes W3C** et est adapté dans la mesure du possible pour répondre au référencement naturel.

En termes de perspectives, il est prévu d'utiliser **JavaScript** et sa **méthode fetch ()** afin d'améliorer la performance du site, notamment en ce qui concerne les champs de recherche et filtres de contenus (par exemple dans les pages « listes des tournois », « mes tournois », « création d'équipe » (pour la recherche de joueurs, mais aussi dans l'espace administrateurs).

2. Technologies back-end :

La partie back-end a été développée **en PHP,** avec une base de données sous **PhpMyAdmin** (langage mysql). L'objet **PHP Data Objects (PDO)** est utilisé comme interface pour accéder à cette base de données.

Le langage PHP permet de générer du contenu dynamique et de collecter les données de formulaires en respectant les recommandations de sécurité (nettoyage des champs input, préparation des requêtes en base de données…). J'ai également utilisé la **programmation orientée objet** (POO) afin d'optimiser le code (méthode DRY : don't repeat yourself).

Le langage PHP fournit d'autres fonctionnalités comme la fonction **Mail () de PHP** utilisée pour la gestion des mails, tels que la confirmation pour la création de comptes. En termes de perspectives, il est prévu d'implémenter un système d'emailing (Sendinblue) pour l'envoi des notifications (classements, derniers gagnants).

Par ailleurs, **afin de faciliter la maintenance**, les fichiers de traitements sont séparés des fichiers Models (Class POO qui gère les requêtes en base de données) et le site est commenté. Il est également prévu de mettre en place **l'architecture MVC (Model-View-Controller).**

3. Sécurité de l'application

La sécurité de l'application a été réalisée à différents niveaux.

Nettoyage / vérification des données saisies par l'utilisateur

Afin d'éviter les **failles XSS (Cross-Site Scripting) au niveau des inputs,** j'ai vérifié et nettoyé les données saisies par l'utilisateur. Selon les cas, on utilise :

- **htmlentities**() qui convertit tous les caractères éligibles en entités HTML
  stripslashes (), qui supprime les antislashs d'une chaîne
  et **trim**(), qui supprime les espaces (ou d'autres caractères) en début et fin de chaîne.

```php
public static function validInput($input)
{
  htmlentities($input);
  stripslashes($input);
  trim($input);
  return $input;
}
```

  Cette méthode est notamment utilisée pour toutes les chaines de caractères, par exemple le « nom de l'utilisateur ».

- **Avec preg_match() et une REGEX,** qui vérifient que la saisie de l'utilisateur correspond bien au pattern défini par le REGEX. La fonction preg_match() renvoie le chiffre 1 lorsque c'est vrai et 0 lorsque c'est faux. Cette méthode a été utilisée par exemple pour le téléphone :

```php
public static function validTel($input)
{
  if (preg_match("/^([+|00]?33|0) ?[1-9] ?(\d{2} ?){4}$/
  ", $input) == 1) {
    return trim($input);
  } else {
   $_SESSION['error'] .= "le format de téléphone n'est
   pas bon <br>";
  }
}
```

- **Avec filter_var()**, une fonction définie par PHP et qui applique un filtre spécifique. Cette méthode est par exemple utilisée pour les e-mails :

```php
public static function validEmail($input)
{
  if (
    filter_var($input, FILTER_VALIDATE_EMAIL) &&
    preg_match("/^(\w+)\.?(\w+)@(\w+)\.\w{2,}$/m", $input) == 1
  ) {
    return trim($input);
  } else {
    $_SESSION['error'] .= "le format de mail n'est pas bon<br>";
  }
}
```

En plus de la vérification des champs inputs, l'utilisation d'exit() après la redirection avec un header('Location :… ') permet également de sécuriser l'application en arrêtant le script .

Protection des requêtes d'accès à la base de données

Afin de protéger la base de données contre **les injections SQL,** j'ai préparé les requêtes de données avec **la méthode prepare()** de l'objet PDO(). L'exécution de la requête se fait dans un second temps avec la méthode execute(). La vérification de cette protection est d'ailleurs facilitée par l'utilisation de la POO où l'on peut distinguer les requêtes de lecture et d'écriture.

Exemple : préparation de la requête de lecture de toutes les données de la table user

```php
//On prépare la requête
$resultat = $this->getPDO()->prepare('SELECT * FROM user');
//On execute la requête
$resultat->execute(array());
```

Protection de l'authentification utilisateur

Afin de protéger l'authentification utilisateur, je génère **une clé de confirmation** unique au moment de l'inscription qui est envoyée dans le lien de confirmation et permet de vérifier que l'utilisateur existe avant de confirmer son compte. L'utilisateur a ensuite le droit de se connecter uniquement si son compte est confirmé. Le même système est utilisé pour la réinitialisation du mot de passe.

La force et le chiffrement du mot de passe :

Afin de protéger le mot de passe, l'utilisateur doit saisir un mot de passe fort, il doit également confirmer la saisie de son mot de passe. Cette vérification est réalisée en VueJs puis en PHP. Ce mot de passe est ensuite **crypté avec la fonction password_hash()** et vérifié au moment de la connexion avec password_verify().

```php
if (preg_match("/^(?=.{8,}$)(?=.*?[a-z])(?=.*?[A-Z])(?=.*?[0-9])(?=.*?[^a-zA-Z0-9]).*$/", $input) == 1) {
  return password_hash($input, PASSWORD_DEFAULT);
```

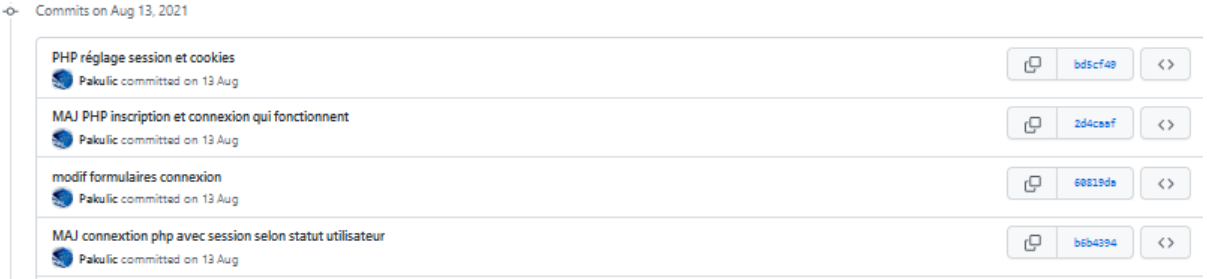
En termes de perspective, la mise en place de l'architecture MVC permettra ensuite de rajouter un contrôle d'accès aux pages grâce au rooter.

4. Versionning

Depuis le début du projet, nous avons partagé nos travaux via **GitHub**. Il existe de nombreux autres logiciels de versionning. Mais GitHub a l'avantage d'être libre.

L'outil GitHub permet de **synchroniser** les développements des différents membres du projet, mais également de **sauvegarder régulièrement** le projet en gardant une trace de chaque version. Il offre donc la possibilité de revenir à tout moment sur une **version antérieure du projet**.

Le **versionning** consiste à figer le code source d'un projet à un instant donné. On dit qu'on réalise des versions de notre code. Cette notion est associée au terme de **commit** dans le vocabulaire de Git.



V. Compétences du référentiel

A. Maquetter une application (CP1) :

Cette étape est importante **en termes d'expérience utilisateur (UX),** car elle doit permettre de répondre aux besoins et habitudes de navigation, de gérer l'organisation visuelle du site (limiter la surcharge visuelle), de garder une cohérence tout au long de la navigation (via le header et le footer par exemple) et de faciliter la compréhension (utiliser des mots et des symboles simples).

Tout d'abord, nous avons pris en compte la charte graphique de Wavy et quelques sites de référence (cf. annexe 2).

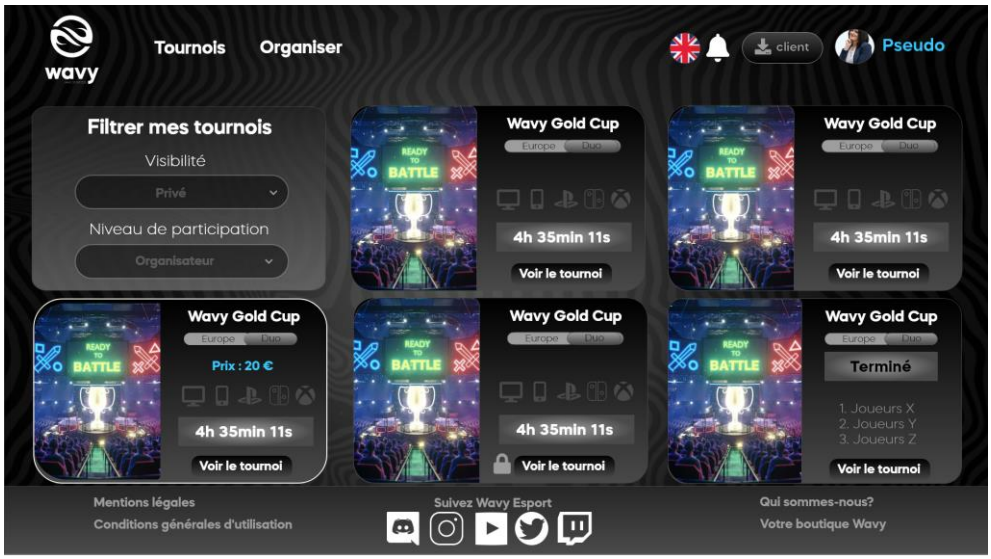
Nous avons ensuite utilisé **le logiciel adobe Xd** et réalisés ces travaux en plusieurs étapes :

- **Le zoning,** qui permet de définir l'organisation générale des pages web. Il consiste à définir les zones de contenus.
- **Le wireframe,** qui permet de mettre au propre les croquis, de détailler chaque bloc de contenus et de commencer à positionner les éléments sur la page, sans tenir compte du graphisme.
- À partir de cette étape, nous avons commencé à faire **le prototypage :** simulation de la navigation (enchainement des pages) et des interactions avec les contenus : au survol, au clic, incrustations.
- Enfin, nous avons défini et mis en œuvre **l'habillage graphique.**

<u>Utilisation des couleurs pour mettre en avant des contenus</u>
Nous avons utilisé **une palette de couleurs assez sombres (noir et nuances de gris),** car l'utilisateur passe souvent des heures devant l'écran et nous avons rajouté des **contours blancs** pour mettre en valeur certains tournois (avec prix) ou **encore la couleur bleue pour le « call to action ».**

<u>Différence d'ergonomie entre mobile et desktop</u>
Sur la version mobile, les boutons par exemple dans le header sont plus grands et arrondis pour améliorer l'accès au clic. Les liens du menu principal passent dans un menu hamburger à gauche.

Maquette graphique « mes tournois » version desktop



Extrait maquette version mobile

Exemples d'évolution de maquettages

J'ai réalisé quelques évolutions de maquettage pour améliorer l'ergonomie du côté utilisateurs, par exemple la possibilité de sélectionner directement une zone contenant le label plutôt qu'une case à cocher. Cela permet également d'améliorer encore plus l'ergonomie de la version mobile.

Paramètre de l'utilisateur
Technologie

WIREFRAME

MAQUETTE

Paramètre de l'utilisateur
Notifications

WIREFRAME

MAQUETTE

B. Interface statique et adaptable (CP2)

Un développement en mode responsive (media queries)

Le développement en mode responsive permet un ajustement de l'affichage de la page web en fonction de la taille de l'écran. Il est important dans notre projet, car les utilisateurs peuvent jouer à Fortnite sur un appareil mobile (téléphone ou tablette) sans compter que les internautes sont de plus en plus nombreux à consulter internet sur ces technologies. Ce développement permet donc de faciliter la navigation et d'améliorer l'expérience utilisateur.

Pour le réaliser, j'ai utilisé les **medias queries** selon 4 tailles d'écran :

- <767px : mobile
- Entre 767px et 940px : tablette
- Entre 940px et 1200px : tablette ou petit écran
- >1200 px : grand écran

```
@media (max-width: 767px) {…
}

@media (min-width: 767px) and
(max-width: 940px) {…
}

@media (min-width: 940px) and
(max-width: 1200px) {…
}

@media (min-width: 1200px) {…
}
```

Exemple pour la page « Mes tournois » :

Pour la page « Mes tournois » (cf. maquette p. 18), la section main (class « container) va contenir le filtre et le résumé des tournois.

J'ai utilisé **les flexbox et notamment le paramètre wrap** pour répartir les contenus sur la page. Cette fonctionnalité CSS permet d'adapter facilement les contenus d'une taille d'écran à l'autre.

Pour une meilleure mise en page et afin d'ajuster les différents éléments en fonction de celle des écrans, **quelques modifications sont réalisées grâce aux médias queries.**

```
/* main qui contient le filtre et
les cartes tournois*/
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  height: 100%;
  width: 95%;
  margin-top: 2%;
}
```

Par exemple, on fait varier :

- La taille de la section filtre
- La taille et la mise en page des résumés des tournois (class « tournoisavecprix » avec prix et « tournoisFortniteSimple » sans prix)

```
/* CARTE TOURNOIS- configuration des
cartes contenant les tournois (page
accueil, recherche et mes tournois)*/
.tournoisFortniteSimple,
.tournoisavecprix {
  display: flex;
  flex-direction: row;
  height: 18em;
  min-width: 32%;
  width: 30%;
}
```

Dans le cas du résumé des tournois, il passe de « flex-direction : row » (cf. illustration ci-contre), c'est-à-dire un alignement horizontal, à « flex-direction : column » (alignement vertical) lorsque la taille d'écran est inférieure à 940px.

Standard W3C et référencement

J'ai réalisé les **développements HTML/CSS, en respectant les standards W3C** (normes définies par les World Wide Web Consortium). Le code HTML a été vérifié avec le site https://validator.w3.org/.

Ils respectent également :

- **La structure avec des balises sémantiques** (notamment les balises «html » avec « head » et « body ». La balise « body » contient « header », « main », et « footer ».)
- L'utilisation **d'id et de class** en attribut pour les balises, qui sont ensuite utilisées pour les **feuilles de style.**
- Des **textes alternatifs** pour les contenus multimédias (alt) et pour les champs de formulaires (label).

Exemple pour un champ de formulaire (champ email dans la page paramètre de compte utilisateur) :

```
<label for="email">Email*</label>
<input type="email" name="email" class="input" id="email"
placeholder="email" value="<?php echo $profilUser['email']; ?>">
```

Exemple pour un contenu image : extrait de code de la photo du gagnant sur la page d'accueil

```
<img class="player" src="img/photo-gagnants-tournoi.png"
alt="photo du gagnant du tournoi Fortnite" title="photo
des derniers gagnants de tournoi du jeu Fortnite">
```

En ce qui concerne le référencement, le projet intègre, dans la mesure du possible, des mots clés dans les balises h1, h2, h3, dans les balises « images » en utilisant les attributs « alt » et « title » (cf. exemple ci-dessus, pour un contenu image) ainsi que dans les liens et boutons.

Ce travail pourra être amélioré par la suite, grâce à l'architecture MVC, en ajoutant un titre et une balise métadescription adaptée pour chaque page, mais aussi en ajoutant des liens de qualité internes/externes et en augmentant la réputation (partage sur les réseaux sociaux, création d'une FAQ pour répondre aux questions des utilisateurs…)

- 👉 cf. travaux de référencement annexe 6.

Exemple de balise métadescription pour la page d'accueil :

```
<meta name="description"
  content="Rejoignez la commaunauté Wavy, organisez votre
  tournoi Fortnite ou participez à un tournoi de jeu vidéo
  gratuit">
```

Exemple de titre pour la page d'accueil :

```
<!-- TITRE DE LA PAGE -->
<title>Page d'accueil : Organisez ou participez à un tournoi
Fortnite partout en Europe avec Wavy</title>
```

C. Interface web dynamique (CP3)

Dans le projet, j'ai réalisé tous les développements JavaScript (avec le Framework VueJs) afin de fluidifier la navigation entre les différents contenus (création des évènements au clic) et notamment **de vérifier la saisie dans les champs de formulaires,** qui seront ensuite revérifiés en PHP.

Le site internet contient de nombreux formulaires (inscription, connexion, réinitialisation de mot de passe, création d'un tournoi…) qu'il convient de sécuriser.

Parmi les formulaires proposés, **le site internet offre la possibilité à l'utilisateur de modifier les informations de son profil** : pseudo, mail, mot de passe et d'autres paramètres. En cohérence avec le RGDP (règlement général sur la protection des données), il a également la possibilité de demander la suppression de son compte auprès de l'administrateur.

Exemple pour la confirmation du mot de passe (paramètre de compte utilisateur):

L'application VueJs est montée sur le formulaire, qui récupère, entre autres, la valeur du champ «password » et du champ «confirmpassword».

J'ai créé une méthode «confimPassword()» qui va comparer les valeurs des deux champs et va renvoyer une erreur si les deux mots de passe ne correspondent pas.

```
/* vérification de la correspondance des mots de
passe */
confirmPassword() {
  if (this.password != this.confirmpassword) {
    this.erreur2 = 'Les mots de passe ne
    correspondent pas';
  } else {
    this.active = false;
    this.inactive = true;
    this.erreur2 = '';
  }
},
```

Côté HTML, la balise input comprend un attribut « v-model » qui indique le nom de la variable (data) pour VueJs et un attribut « @input = 'confirmPassword()' » qui exécute la méthode en fonction de la saisie de l'utilisateur.

```
<label
for="confirm_password">Confirmation*</
label>
<input v-model="confirmpassword"
type="password" class="input"
id="confirm_password"
name="confirmpassword"
placeholder="Confirmez votre mot de passe"
pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.
*[0-9]).{8,20}" @input="confirmPassword()"
required>
```

Exemple de la suppression du compte utilisateur depuis l'espace administrateur

Ci-dessous l'exemple du développement qui a été réalisé en php afin de répondre au besoin de suppression du compte utilisateur par l'administrateur.

```
} else if (isset($_POST['supprUser'])) {
  // je récupère l'id de l'utilisateur
  $idsupprUser = $_POST['supprUser'];
  // j'execute la requête pour supprimer les données
  $updateUser->removeUser($idsupprUser);
  // je retourne l'information et je redige vers la
  page en cours
  $_SESSION['success'] = "Le compte à été supprimé";
  header('Location: ../admin/liste_utilisateurs.php');
  exit();
```

Explication :
Lorsque le fichier de traitement reçoit l'information du formulaire (clic sur le bouton), il récupère l'id de l'utilisateur et envoie les données à la méthode qui prépare et exécute la requête en base de données.

Exemple de double vérification VueJs/PHP pour le mail dans le formulaire de connexion :

Je monte une application VueJs sur le formulaire de connexion.

Je déclare, dans la partie data, les variables : champ mail « emailConnexion », champ erreur «erreurMailConnexion » et la regex.

Je crée ensuite une méthode « verifMail() » (cf. illustration ci-contre) qui vérifie que le mail saisi correspond à la regex sinon il renvoie une erreur au clic sur le bouton « valider ».

```
methods: {
  // vérification du mail
  verifMail() {
    //si le mail est vide, on indique de saisir le
    mail
    if (this.emailConnexion == '') {
      this.erreurMailConnexion = 'Merci de saisir
      votre adresse mail';
      // on vérifie le format du mail
    } else if (this.mailregExp.test(this.
    emailConnexion)) {
      this.erreurMailConnexion = '';
    } else {
      this.erreurMailConnexion = 'Veuillez saisir
      une adresse email valide';
    }
  },
```

En PHP, ce même champ est vérifié et nettoyé avec la méthode « validEmail() ». Cette méthode vérifie la saisie en utilisant la fonction PHP filter_var() ainsi que la fonction preg_match() avec une regex.

```
// fonction pour sécurer les champs email
public static function  validEmail($input)
{
  if (
    filter_var($input, FILTER_VALIDATE_EMAIL) &&
    preg_match("/^(\w+)\.?(\w+)@(\w+)\.\w{2,}$/m",
    $input) == 1
  ) {
    return trim($input);
  } else {
    $_SESSION['error'] .= "le format de mail n'est pas
    bon<br>";
  }
}
```

À noter : la structure (HTML), le style (CSS) et la logique (JavaScript) **ont été dissociés** afin de mieux identifier le code propre à chaque page (exemple : DOCTYPE HTML) et d'améliorer la **sécurité** et la maintenance.

D. Création d'une base de données (CP5) :

La conception de la base de données a été élaborée grâce à la méthode MERISE en 3 étapes (cf chapitre « conception de la base des données ») :

- **Modèle conceptuel de données** pour définir les entités, leurs relations et leurs cardinalités
- **Modèle logique de données** pour déterminer les clés primaires de chaque table, les clés étrangères et les tables de jonction selon les cardinalités précédemment définies.
- **Modèle physique de données** pour transformer cette modélisation en base de données physiques, en tenant compte des résultats du MLD et en ajoutant les types de données pour chaque champ.

Création de la base de données avec mysql

Une fois l'organisation physique des données construite, la création de la base fait appel à un langage sql (Structured Query Language), plus précisément **mysql**. Pour créer la base, on intègre le code directement dans PhpMyAdmin. Pour plus de sécurité et donner un accès restreint à la base, on utilisera un compte utilisateur 'non root'.

Le code sql « CREATE DATABASE `wavytournament` » permet de créer la base de données. Il sera suivi de la création des tables (cf. exemple ci-dessous)

Exemple du code sql pour la création de la table adhésion :

```
CREATE TABLE `adhesion` (
  `id_adhesion` int(10) UNSIGNED NOT
  NULL AUTO_INCREMENT,
  `id_user`,
  `id_team`,
  CONSTRAINT `PK_adhesion_id`
  PRIMARY KEY(`id_adhesion`))
  ENGINE=InnoDB
  DEFAULT CHARSET=utf8;
```

Explication du code sql ci-contre :

- « CREATE TABLE 'adhesion' » permet de créer la table nommée 'adhesion'
- 'id_adhesion' est le champ dont le type de données est : entier sans signe (-) et non nul qui s'auto-incrémente. Pour déclarer qu'il s'agit de la clé primaire de la table, on ajoute : « CONSTRAINT PK_adhesion_id PRIMARY KEY (id_adhesion) ».
- On crée les deux champs `id_user` , `id_team`, qui deviendront ensuite les clés étrangères. Ils récupèreront le type de données après la contrainte.
- « ENGINE=InnoDB » correspond au moteur utilisé.
- « DEFAULT CHARSET=utf8» correspond au paramétrage des caractères

```
ALTER TABLE `adhesion`
  ADD CONSTRAINT `adhesion_ibfk_1`
  FOREIGN KEY (`id_team`)
  REFERENCES `team` (`id_team`),
  ADD CONSTRAINT `adhesion_ibfk_2`
  FOREIGN KEY (`id_user`)
  REFERENCES `user` (`id_user`);
```

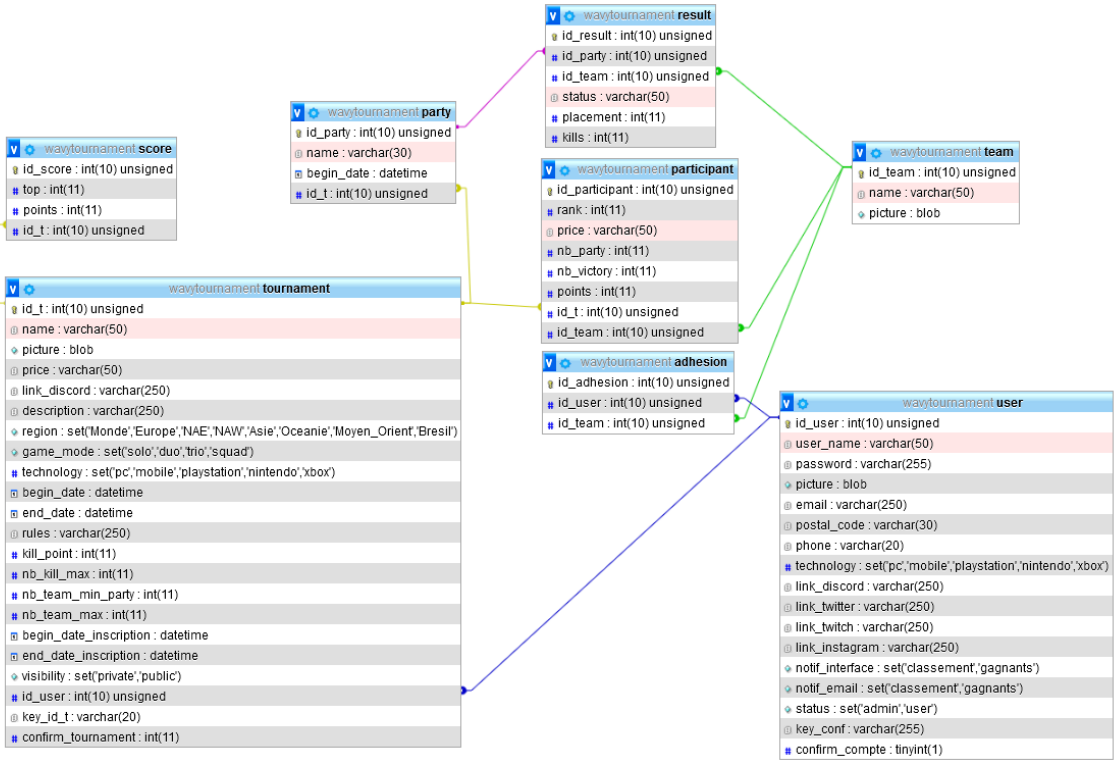
Pour implémenter les clés étrangères :

- « ALTER TABLE » permet de modifier la table
- « ADD CONSTRAINT » indique que l'on ajoute une contrainte. Il est suivi du nom de la contrainte ('adhesion_ibfk_1')
- « FOREIGN KEY (`id_team`) REFERENCES `team` (`id_team`)» signifie que l'id team est une clé étrangère qui se référence à la table team et le champ « id_team»

La base de données de notre projet est constituée de 8 tables :

- Chacune d'entre elles possède plusieurs champs, dont une **clé primaire (primary key)**
- **Les 3 « tables de jonction » :** participant (participe), result et adhesion (adhere), possèdent en même temps **une clé primaire (id de la table) et deux clés étrangères.**
- D'autres possèdent une clé étrangère (foreign key) en plus de la clé primaire, comme la table score, party et tournament.

Schéma relationnel de la base de données (réalisé avec PhpMyAdmin)



<u>Requête en base de données depuis PHP</u>

La lecture et l'écriture des données se font ensuite via PHP et l'objet **PHP Data Objects (PDO).**

<u>Exemple pour l'insertion de données de score (création d'un tournoi)</u>

```php
//méthode pour insérer les données de scores
public function updateScore($top, $points, $id)
{
  // on prépare la requête
  $result = $this->getPdo()->prepare("INSERT
  INTO score (top,points,id_t) VALUES (:top,
  :points,:id_t)");
  // on execute la requête
  $result->execute(array('top' => $top, 'points
  => $points, 'id_t' => $id));
  // on ferme la requête
  $result->closeCursor();
}
```

Cette méthode « updateScore() » permet l'insertion des données grâce à la méthode getPdo(), qui contient l'objet PDO .

*La requête d'insertion des données est composée de « **INSERT INTO** » suivie du nom de la table et des champs ainsi que des valeurs (**VALUES**).*

En termes de perspectives, il semblerait pertinent **d'instaurer des règles de gestion** relatives au volume de données de cette base (limiter l'historique des tournois à deux ans et supprimer les utilisateurs non connectés depuis plus d'un an).

E. Composants d'accès aux données (CP6)

Nous allons voir plus précisément dans ce chapitre comment j'ai appliqué la programmation orientée objet pour la lecture et l'écriture des données en base.

Les composants d'accès aux données

Les class permettent de définir les variables et les méthodes propres à un objet.

Par exemple dans le projet, j'ai réalisé, pour le moment :

- **1 class Database** qui contient une méthode protégée d'accès à la base getPdo(). Cette méthode fait appel à l'objet PDO() fournit par PHP. L'utilisation de PDO() permet de gérer les exceptions (les erreurs de connexions sont récupérées avec un "try/catch").
- **4 autres class,** qui sont des extensions de la class Database, car elles font appel à sa méthode protégée getPdo().

Ces 4 class contiennent les requêtes CRUD (create - read - update - delete). Ce sont les class :

- User : pour la lecture des données de la table user
- Updateuser : pour l'écriture des données de la table user
- Tournament : pour la lecture des données des tables tournament, score et party
- UpdateTournament : pour l'écriture des données des tables tournament, score et party.

D'autres class en mode CRUD seront créées par la suite pour les tables 'team' (avec l'adhésion des joueurs=> table 'adhesion'), 'participant' et 'result'.

Chaque class est contenue dans un fichier avec le même nom. Ces noms sont écrits en PascalCase. Ces fichiers sont documentés (commentés) afin de faciliter la maintenance.

Exemple de la class User :

La class User est une extension de la class Database (extends Database), qui contient plusieurs méthodes dont getAllUser() ;

La méthode getAllUser() récupère la méthode protégée d'accès à la base getPdo() afin de :

- Préparer - avec prepare() - la requête de lecture « SELECT * FROM user » qui signifie sélectionner toutes les données de la table user,
- Exécuter la requête avec execute(),
- Récupérer les données dans un tableau grâce à fetchAll() et retourner les données dans la méthode getAllUser().
- Fermer la connexion avec closeCusor() ;

```php
<?php
//on crée un nom d'espace(cela servira pour dissocier
du nom du controller)
namespace Models;
// on inclut le fichier Database
include_once 'Database.php';
//on utilise la class Database de l'espace Models
use Models\Database;

// LECTURE DES DONNEES UTILISATEURS

// Class qui contient toutes les requêtes de données
qui concernent les utilisateurs
class User extends Database
{
  // LECTURE DES DONNEES
  // récupérer tous les utilisateurs
  public function getAllUser()
  {
    //On prépare la requête
    $resultat = $this->getPDO()->prepare('SELECT *
    FROM user');
    //On execute la requête
    $resultat->execute(array());
    //On récupère les résultats
    return $resultat->fetchAll();
    // on ferme la connexion à la base
    $resultat->closeCursor();
  }
```

Utilisation de ces composants d'accès aux données

L'instanciation est le fait de créer un objet via une classe, ce qui génère une instance de la classe dont elle dépend.

Exemple d'utilisation de la class User pour l'affichage de la liste des utilisateurs (espace administrateur)

J'instancie la class User avec une variable $users. $users devient dons une instance de la class User.

J'exécute ensuite la méthode getAllUser() qui va récupérer toutes les données de la table user et les stocker dans un tableau de la variable $result.

Pour afficher chaque donnée de chaque utilisateur, je parcours les données du tableau avec la fonction 'foreach()'.

```php
<?php
// j'instancie la class User
$users = new User;
// j'execute la requête pour récupérer les
données
$result = $users->getAllUser();
// On affiche chaque entrée une à une
foreach ($result as $user) {
?>
```

Sécurité de ces composants d'accès aux données

Il est important de sécuriser l'accès aux données contre les potentielles injections sql. Au niveau de la base de données, j'ai mis en place les sécurités suivantes :

- **La méthode getPdo()** qui contient la connexion à la base, est **protégée**

```php
protected function getPDO()
```

- **Chaque requête d'accès aux données est préparée,** que ce soit pour la consultation, la création, la mise à jour et la suppression
- **La connexion à la base est fermée après chaque requête**
- **Sécurisation de l'authentification avec une clé de confirmation** (cf. chapitre suivant)

F. Partie back-end de l'application web (CP7)

Le projet comprend différents fichiers de traitement de l'information. Ils évolueront prochainement vers une l'architecture Model-View-Controller (MVC). Cette architecture qui sépare bien chaque rôle de fichiers (vues, modèles et contrôleurs) permet notamment de faciliter la maintenance, de restreindre l'accès à certaines pages avec un routeur et de travailler en équipe sur le projet.

Ces fichiers de traitement servent, par exemple, à sécuriser l'authentification et ne sont appelés que lorsque le traitement est nécessaire. On passe des informations dans l'URL qui permettent de vérifier si le traitement doit s'exécuter.

Extrait du fichier de traitement pour l'inscription

On vérifie si le formulaire d'inscription a bien été soumis via la valeur passée dans le lien ('inscription').

Dans ce cas, on vérifie si les champs obligatoires ne sont pas vides.

On vérifie et on sécurise la saisie des données avec la class ChampsForm et les méthodes « statics » appropriées (cf. chapitre sur la sécurité)

```php
//on vérifie que l'utilisateur a bien saisi puis envoyé le formulaire
if (isset($_POST['inscription']) && $_POST['inscription'] == 'inscription') {
  // on verifie si les champs obligatoires ont bien été renseignés
  if ($_POST['pseudo'] != '' && $_POST['email'] != '' && $_POST['password'] != '' && $_POST['compte_discord'] != '') {
    //on déclare les variables et on les sécurise (sanitize)
    $pseudoUser = ChampsForm::validInput($_POST['pseudo']);
    $emailUser = ChampsForm::validEmail($_POST['email']);
    $passwordUser = ChampsForm::validPwd($_POST['password']);
    $discordUser = ChampsForm::validUrl($_POST['compte_discord']);
```

Création d'une clé de confirmation (au moment de l'inscription)

Afin de sécuriser l'authentification de l'utilisateur, l'utilisateur doit confirmer son compte. On génère pour cela une clé de confirmation unique.

Avant de créer un nouveau compte, on vérifie que l'email n'existe pas déjà dans la base.

Si ce n'est pas le cas, on génère une clé de confirmation avec la fonction PHP mt_rand(). Cette fonction génère une valeur aléatoire.

On enregistre ensuite toutes les données (avec la clé de confirmation) dans la base de données. Ces données seront uniquement confirmées si l'utilisateur a cliqué sur le lien de confirmation.

```php
// j'instancie ma class User
$user = new User;
// j'execute la méthode qui va chercher tous les utilisateurs de la base avec l'email saisi
$result = $user->getUserbyEmail($emailUser);
// je compte le nombre d'email correspondant à celui saisi et si le nombre d'email est <1, j'execute la suite
if ($result->rowCount() < 1) {
  // je crée une clé à chaque fois que des données sont postées par un utilisateur
  foreach ($_POST as $key) {
    //génération d'une clé
    $key_length = 15;
    $key_conf = "";

    for ($i = 0; $i < $key_length; $i++) {
      $key_conf .= mt_rand(0, 9);
    }
    // j'instancie la class pour écrire une donnée utilisateur
    $newUser = new UpdateUser;
    // j'ajoute l'utilisateur dans la base
    $newUser->addUser($pseudoUser, $passwordUser, $emailUser, $discordUser, $key_conf);
```

Envoi de la clé de confirmation dans le lien de confirmation par mail

La fonction **Mail () de PHP** est utilisée pour la gestion des mails tels que la confirmation pour la création de comptes.

Elle comprend différents paramètres ($to, $subjects, $message ,$headers) qui sont définis ci-contre.

Le pseudo et la clé de confirmation, passés dans l'URL, permettront de vérifier lors du clic sur le lien que l'utilisateur existe et confirmer son compte (modification du champ « confirm_compte » de la table user).

```php
// ENVOI DE L'EMAIL -avec la clé de confirmation
$to      = $emailUser;
$subject = "veuillez confirmer votre compte Wavy";
$message = '
    Bonjour ' . $pseudoUser . ',
    Nous vous remercions pour votre demande
    d\'inscription.
    Afin de procéder à la confirmation de votre
    compte, nous vous remercions de bien vouloir
    cliquer sur le lien ci-dessous ou de le copier/
    coller dans votre navigateur :"http://localhost/
    Site_Wavy_Tournois/traitement/confirm.php?
    user_name=' . $pseudoUser . '&key_conf=' .
    $key_conf . '>
    Cordialement
    L\'équipe de Wavy
';
$headers = "From:'test.site13920@gmail.com";
```

Connexion et variable de session

Ensuite, lorsque l'utilisateur demande à se connecter, je vérifie si le mail existe. Dans le cas où le compte est confirmé, je vérifie avec password_verify() que le mot de passe est équivalent au hachage présent dans la base.

Si c'est le cas, la connexion est acceptée et les informations de l'utilisateur sont enregistrées dans la variable globale $_SESSION. L'utilisateur sera ensuite redirigé vers son espace en fonction de son statut (user ou admin)

```php
if ($confirmCompte == 1) {
  if (password_verify($pwd, $passwordbdd)) {
    // on la démarre la session
    session_start();
    // l'utilisateur existe dans la table
    // on ajoute ses infos en tant que variables de
    session
    $_SESSION['pseudo'] = $pseudo;
    $_SESSION['id'] = $id;
    $_SESSION['status'] = $statut;
    $_SESSION['picture'] = $photo;
    // on redirige l'utilisateur vers une page
    d'accueil où il sera redirigé vers sa section
    membre en fonction de son statut
    header('Location: ../index.php');
    exit();
  }
```

Afin de sécuriser les permissions d'accès aux espaces utilisateurs et administrateurs, le même mécanisme de clé de confirmation est utilisé lors du renouvellement de mot de passe.

VI. Réalisation personnelle

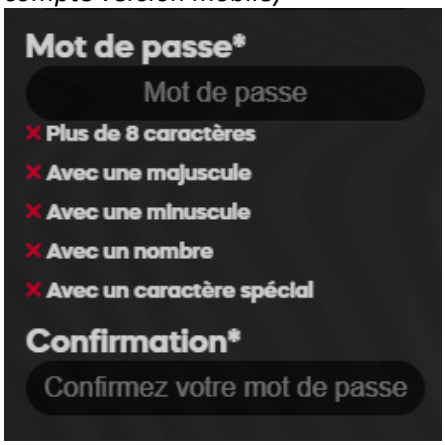
Plusieurs réalisations phares ont été abordées dans le chapitre précédent (comme l'inscription, la connexion des utilisateurs et les paramètres de compte). J'aimerais faire ici un zoom sur l'indicateur de « force » du mot de passe qui est un plus que j'ai souhaité apporter au projet.

Zoom sur la réalisation : l'indicateur de mot de passe

Afin d'améliorer la sécurité du site, il me semblait pertinent d'aider l'utilisateur à générer un mot de passe complexe. J'ai donc créé un indicateur de « force » du mot de passe en VueJs qui lui indique au fur et à mesure de sa saisie s'il a rempli tous les critères requis.

Cet indicateur est utilisé pour l'inscription, la modification dans les paramètres de compte et lors de la réinitialisation du mot de passe.

Zoom sur le champ 'mot de passe' et son indicateur (paramètre de compte version mobile)



Extrait du code HTML de l'indicateur

```
<ul id="indicateurMdp" :class="{'none':inactive, 'wrap':active }">
  <li class="frmValidation" :class="{'frmValidation--passed' :
  password.length ⩾ 8}"><i class="frmIcon fas" :class="password.
  length ⩾ 8 ? 'fa-check' : 'fa-times'"></i> Plus de 8
  caractères</li>
  <li class="frmValidation" :class="{'frmValidation--passed'
  :has_uppercase }"><i class="frmIcon fas" :class="has_uppercase ?
  'fa-check' : 'fa-times'"></i> Avec une majuscule</li>
  <li class="frmValidation" :class="{'frmValidation--passed'
  :has_lowercase }"><i class="frmIcon fas" :class="has_lowercase ?
  'fa-check' : 'fa-times'"></i> Avec une minuscule</li>
  <li class="frmValidation" :class="{'frmValidation--passed' :
  has_number }"><i class="frmIcon fas" :class="has_number ?
  'fa-check' : 'fa-times'"></i> Avec un nombre</li>
  <li class="frmValidation" :class="{'frmValidation--passed' :
  has_special }"><i class="frmIcon fas" :class="has_special ?
  'fa-check' : 'fa-times'"></i> Avec un caractère spécial</li>
</ul>
```

L'indicateur s'affiche au clic sur l'input « password », car la class CSS passe d'inactive ('none') à active ('wrap').

L'indicateur teste ensuite la saisie et indique à l'utilisateur si la condition est remplie (par exemple l'icône 'fa-check' si elle est remplie ou 'fa-times' si elle ne l'est pas).

Cet indicateur se ferme ensuite lorsque l'utilisateur a bien confirmé son mot de passe.

Extrait du code VueJs – méthode d'activation de l'indicateur activeIndicateur() et méthode qui teste la saisie du mot de passe nommée password_check().

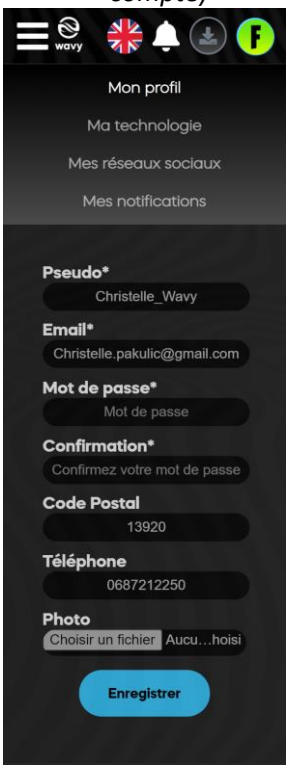
```
methods: {
  /* méthode qui affiche l'indicateur */
  activeIndicateur() {
    this.active = true;
    this.inactive = false;
  },
  /* méthode qui créer l'indicateur */
  password_check() {
    this.has_number = /\d/.test(this.password);
    this.has_lowercase = /[a-z]/.test(this.password);
    this.has_uppercase = /[A-Z]/.test(this.password);
    this.has_special = /[^a-zA-Z\d]/.test(this.
    password);
  },
```

VII. Jeu d'essai

J'ai réalisé de nombreux tests manuels pour vérifier le bon fonctionnement des formulaires, notamment pour la lecture et l'écriture des données en base. L'objectif étant de vérifier que les champs sont correctement saisis avant d'être envoyés en base et que l'utilisateur voit si la modification a été ou non prise en compte et ce qu'il doit éventuellement corriger.

Exemple du formulaire paramètre de compte

Screenshot de l'onglet profil (paramètre de compte)



Plusieurs tests ont été effectués afin de vérifier que :

- Les champs pseudo, email et mot de passe ne sont pas vides dans l'onglet profil, car ils sont requis. (Ou Discord si on est sur l'onglet réseaux sociaux).
- Les deux mots de passe correspondent.
- Le champ email, mot de passe et téléphone sont dans le bon format (si le téléphone est saisi).

- L'utilisateur peut modifier (ou non) sa photo de profil et enregistrer ses données

- Les erreurs s'affichent sur la page
- Les checkbox des onglets technologies et notifications peuvent s'activer et leur activation est prise en compte à chaque validation
- Si les données sont correctes, les informations doivent être modifiées en base de données et remontées dans le formulaire (sauf le mot de passe et la photo)

Pour améliorer ce formulaire, il est prévu de vérifier si l'email existe déjà dans la base et de demander à l'utilisateur de confirmer son mail.

En termes de perspectives, il serait par ailleurs intéressant de mettre en place des tests unitaires en natif ou via un Framework (par exemple Symfony).

VIII. Situation de travail : veille et recherche de solutions

A. Description de la veille en termes de sécurité

Les menaces évoluant quotidiennement, il est important de disposer des informations les plus récentes possible pour adapter les dispositifs de protection.

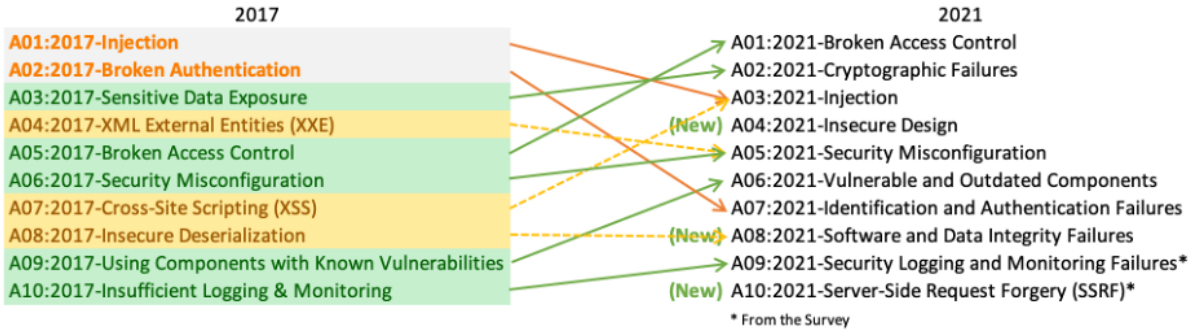
Pour réaliser la veille, j'ai fait une recherche sur internet avec les mots clés ci-dessous. **Mots clés utilisés :** cybersecurity, news, hacker, hacking, cybersecurité, web, actualité

Le résultat de la recherche est disponible en annexe 4. J'ai classé les sites selon 3 catégories : les sites de références qui sont des sites institutionnels, les sites d'actualités qui me semblent pertinents (incontournables ou cohérents avec l'information recherchée) et les autres sites d'informations.

Top 10 de l'OWASP

Une des informations qui m'a semblé la plus pertinente à traiter ici est celle de l'OWASP. Son dernier classement, en 2021, place la rupture de contrôle d'accès en 1re position, les échecs cryptographiques en 2e position et l'injection (sql et XSS) en 3e position. Des sécurités ont déjà été mises en place sur le projet afin de répondre à ces risques (cf. chapitre sur la sécurité).

Illustration du top 10 de l'OWASP (source : https://owasp.org/www-project-top-ten/).



Sécurisation de l'authentification :

Moyens d'authentification en fonction du contexte

Exemple de contexte	Sensibilité des données ou du service	Importance de la menace	Moyen d'authentification
Plate-forme de réservation de terrain de tennis	Peu sensible	Faible (modification de la réservation)	Mot de passe
Site Web publicitaire	Moyennement sensible	Moyenne (interruption ou défiguration du site)	Mot de passe robuste
Messagerie professionnelle	Sensible	Moyenne (interruption du service, compromission d'informations métier sensibles, qui peuvent être d'ordre industriel, financier, concurrentiel, etc.)	Mot de passe robuste + second facteur possible
Système d'information d'administration	Très sensible	Importante (compromission complète du système d'information)	Multifacteur fort (ex. : carte à puce et code PIN)

Source ; https://www.ssi.gouv.fr/guide/recommandations-relatives-a-lauthentification-multifacteur-et-aux-mots-de-passe/

En recherchant des informations sur l'authentification utilisateurs, j'ai trouvé des recommandations de l'agence nationale de la sécurité des systèmes d'information.

Selon cette agence, l'authentification multifacteur est requise dans certaines conditions (cf. tableau ci-joint).

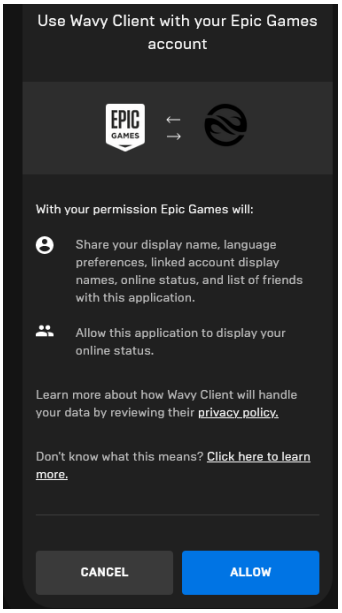
Après analyse, une authentification multifacteur ne semble pas essentielle pour mon projet (situé dans des niveaux de sensibilité de données plutôt peu à moyennement sensibles).

Cependant j'ai retenu **d'autres recommandations qui peuvent être intéressantes** pour mon projet :

Limitation de la durée de validité d'une session (avec un cookie)	Réalisé
Proposer une méthode de recouvrement d'accès	Réalisé
Ne pas donner d'information sur l'échec de l'authentification	À prévoir
Définir un délai d'expiration des facteurs d'authentification	A prévoir
Limiter dans le temps le nombre de tentatives d'authentification	À prévoir
Imposer une longueur minimale pour les mots de passe *Source :https://www.ssi.gouv.fr/administration/precautions-elementaires/calculer-la-force-dun-mot-de-passe/*	À prévoir
Ne pas imposer de longueur maximale pour les mots de passe	

B. Situation de travail ayant nécessité une recherche (à partir d'un site anglophone)

Le projet doit intégrer, dans sa version finale, une authentification de l'utilisateur via Epic Games. Même si ce développement ne fait pas partie du périmètre des travaux que nous devons réaliser pour le projet, je me suis intéressée à la façon dont cela pouvait se faire.



En accord avec la direction de Wavy, je me suis donc connectée au portail d'Epic Games dédié au développeur d'applications et j'ai recherché les différentes étapes à réaliser pour avoir accès à ce service.

> ☛ *Vous trouvez un extrait des instructions du site et sa traduction en annexe 5.*

Suite à cette traduction, j'ai effectué les différentes étapes demandées (authentification multifacteur, création d'organisation avec le compte du directeur de Wavy et création d'un produit). J'ai ainsi pu activer le Compte Epic Services (EAS). Le développement semble s'opérer à partir d'un code en C++ ou C#.

IX. Conclusion et perspectives

Dès les prémices du projet, nous avons défini des **méthodes de travail**, le cahier des charges, les maquettes et le schéma conceptuel de données qui nous ont permis de **structurer le travail à réaliser**. II est important d'anticiper les développements et de **tester régulièrement son code**, c'est pourquoi **la méthode agile** (scrum) est bien adaptée à ce type de projet, car elle permet de réaliser une phase de tests après chaque développement, de corriger les éventuelles régressions et d'appliquer éventuellement de nouvelles sécurités. Pour ce qui est des tests, j'ai réalisé de nombreux tests manuels durant le projet, mais il serait pertinent de pouvoir mettre en place des tests unitaires pour la suite. **Un outil de versionning** est également très important pour partager et sauvegarder les travaux. Dans le cadre de ce projet, nous avons utilisé GitHub.

Sur la base des travaux réalisés (cahier des charges et maquettes), nous avons réalisé le **développement front-end en HTML/CSS natif en mode responsive,** en respectant les standards W3C. Pour la partie dynamique, j'ai utilisé JavaScript et le Framework VueJS pour quelques évènements au clic mais **surtout afin de sécuriser les champs de formulaires** qui sont ensuite vérifiés en PHP. Le projet intègre actuellement, dans la mesure du possible, quelques notions de référencement naturel, comme des mots clés dans les balises h1, h2, h3, dans les balises images en utilisant les attributs « alt » et « title » ainsi que dans les intitulés des liens et boutons. Ce travail pourra être amélioré par la suite grâce à l'architecture MVC qui permet de définir un titre et une balise métadescription par page, mais aussi par le partage sur les réseaux sociaux, la création d'une FAQ et d'un sitemap XML.

Pour la partie back-end, nous avons réalisé les développements avec PHP et sa base de données PhpMyAdmin. J'ai retravaillé ces développements en fonction des méthodes apprises durant la seconde partie de la formation, c'est-à-dire **la programmation orientée objet**, et en appliquant les **recommandations de sécurité**. **Afin de faciliter la maintenance**, les fichiers de traitements sont séparés des fichiers Models et **le site est commenté**. Il est également prévu de mettre en place **l'architecture MVC (Model-View-Controller),** qui permet de faciliter la maintenance et de travailler à plusieurs sur le site, mais aussi d'utiliser par la suite **JavaScript** et sa **méthode fetch()** notamment en ce qui concerne les champs de recherche et filtres de contenus (par exemple dans les pages « listes des tournois », « mes tournois », « création d'équipe » (pour la recherche de joueurs, mais aussi dans l'espace administrateurs).

Les sécurités actuellement appliquées concernent la **vérification des données saisies, le chiffrement du mot de passe, la protection de l'accès à la base de données** (notamment la préparation des requêtes) et quelques sécurisations au niveau **de l'authentification** (clé de confirmation, méthode de recouvrement d'accès, cookies de session). Comme nous l'avons vu, pour ce site, l'authentification multifacteur n'est pas nécessaire pour ce projet, mais la sécurité peut être renforcée en limitant les informations sur l'échec de l'authentification, en définissant un délai d'expiration des facteurs d'authentification, en limitant le temps et le nombre de tentatives et en améliorant la complexité du mot de passe (augmenter la longueur minimale et ne pas imposer de maximum).

En ce qui concerne la base de données, la **méthode MERISE** nous a permis de créer une base qui répond aux besoins, en structurant les bonnes relations entre les tables. En termes de perspectives, il semblerait pertinent **d'instaurer des règles de gestion** relatives au volume de données (limiter l'historique des tournois à deux ans et supprimer les utilisateurs non connectés depuis plus d'un an).

Enfin, même si de **nombreux développements ont été réalisés,** il reste encore **d'autres travaux à poursuivre**, notamment pour la partie détail d'un tournoi (modification, gestion des inscriptions et classement), la traduction en anglais, le système d'emailing et le partage avec les réseaux sociaux, ainsi que l'accompagnement du développeur de Wavy à la connexion avec la base de données d'Epic Games.

X. Annexes

Annexe 1 : Profil de l'utilisateur et objectifs en 3 axes

Profil de l'utilisateur

Fortnite age demographics

Age	Percentage of users
18-24	62.7
25-34	22.5
35-44	12.7
45-54	2
55+	0.1

Source: Verto Analytics

Selon certaines statistiques menées aux États-Unis en 2018 :

- **72%** de ces joueurs seraient de **genre masculin.**
- Le plus grand nombre d'utilisateurs auraient **entre 18 et 24 ans** (62.7%).
- Plus de 70% d'entre eux passeraient **moins de 10 heures** par semaines à jouer

Source :
https://www.statista.com/statistics/865616/fortnite-players-age/
https://www.statista.com/statistics/865625/fortnite-players-gender/

Objectifs en 3 axes :

Faciliter l'accès au site internet

- Le site doit être accessible depuis l'onglet « tournois » du site principal de Wavy Group.
- Le site peut être accessible également directement via internet (référencement Google). Dans ce cas, l'utilisateur peut y accéder par une page d'accueil ou une autre page du site accessible sans connexion.
- Faire le maximum de liens possibles entre ce site et le site principal de Wavy Group pour générer du trafic sur le site principal (par exemple avec le leaderboard)

Capter l'utilisateur dès la page d'accueil

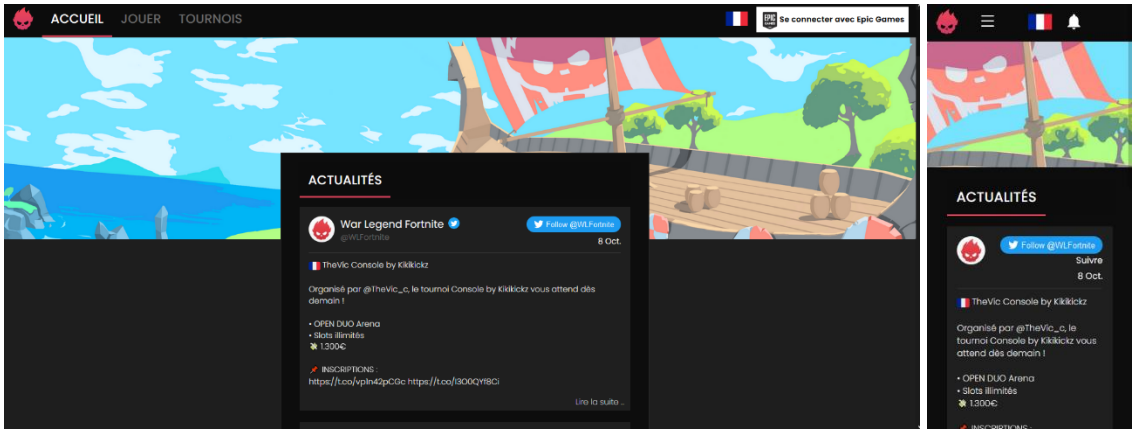
- Promouvoir des évènements (tournois) dès la page d'accueil et inciter l'utilisateur à se créer un compte sur le site.
- Proposer à l'utilisateur de faire partie de la communauté Wavy, jouer en équipe (Discord)
  - Mettre en avant les gagnants et ce qu'ils ont gagné
  - Témoignages de joueurs de wavy (cf. Twitch ou YouTube Wavy)
  - Lien vers la boutique Wavy
- Utiliser éventuellement des photos du jeu
- Option : Proposer éventuellement à l'utilisateur de jouer avec des personnes de sa ville (possibilité de rechercher un tournoi en fonction du code postal)

Fidéliser l'utilisateur

- Lui donner envie de se créer un compte : créer un univers propre à l'utilisateur
- Recevoir et/ou partager des informations : calendrier des tournois, classements ...
- Option : éventuellement, lui proposer d'acheter ou de gagner des supports (vêtements…) pour soutenir son équipe ou d'autres équipes

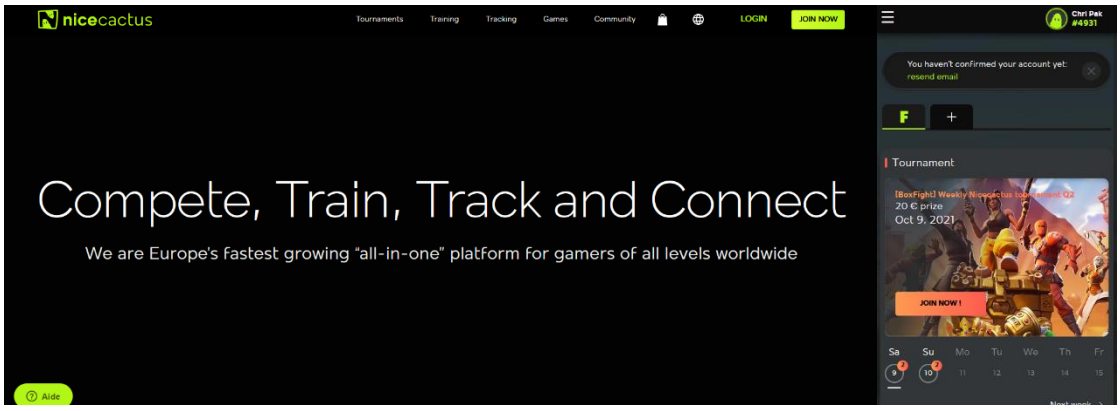
Annexe 2 : Références pour l'ergonomie et le graphisme

- **War Legends Fortnite :** https://system-beta.warlegend.net



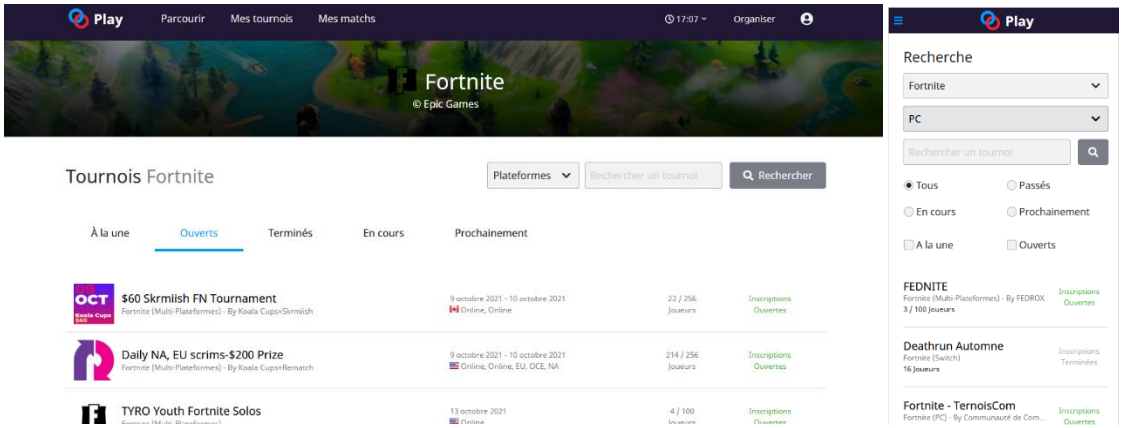
Points forts	Points faibles
• Ergonomie • Simplicité	• Graphisme, notamment pour les tournois et les formulaires

- **Nice Cactus :** https://app.nicecactus.gg/



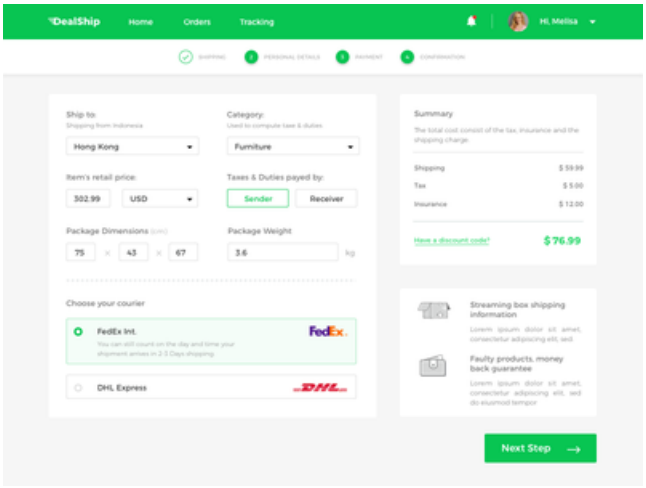
Points forts	Points faibles
• Graphisme • Mise en avant des zones de clics • Animation • Gestion des langues	• Rapidité du site internet • Parfois manque de lisibilité des informations

- **Toornament :** https://www.toornament.com/fr/games/fortnite#



Points forts	Points faibles
• Gestion des formulaires • Éventail d'informations délivrées • Gestion de l'information	• Graphisme (photos et charte graphique) • Navigation • Gestion des langues

- **Inspiration graphique pour la mise en page de formulaire -** https://dribbble.com/shots/3321645-Shipping-App-Concept



Points forts	Points faibles
• Formulaires étape par étape • Ergonomie moderne des champs de saisie	• Adaptable uniquement à une partie du projet => formulaires

Annexe 3 : Détail d'un tournoi et inscription

1. Page liste des tournois

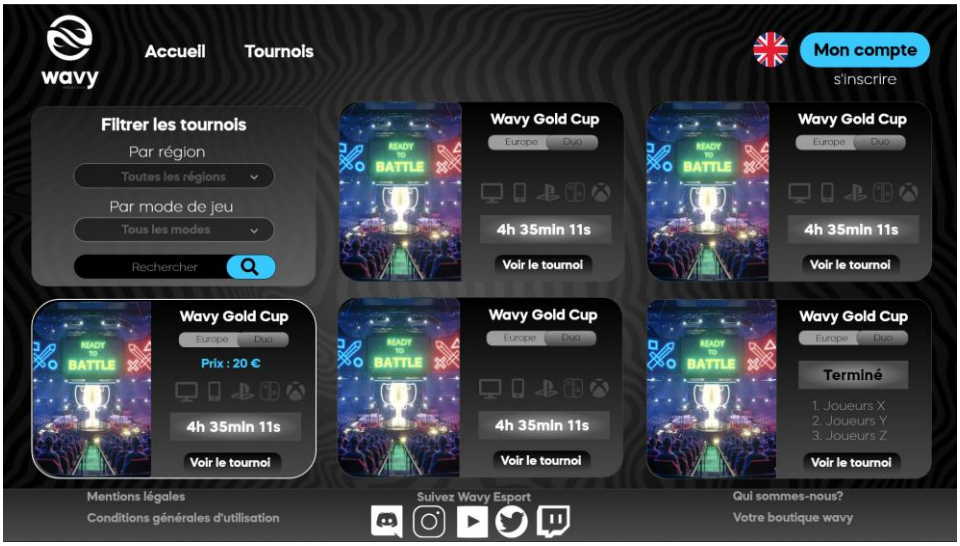
Objectif :

- Lister les tournois en cours et à venir
- Mettre en avant les prochains tournois avec prix (par rapport à ceux qui n'en ont pas)
  Prévoir la possibilité de modération par l'administrateur de supprimer le prix (s'il est jugé comme non adapté)
- Filtres : région du jeu / type de jeu (solo/duo/trio/quad)
- Recherche de tournois (par nom de tournoi)
- Résumé de chaque tournoi avec plus de détails au clic sur le bouton «voir le tournoi ».

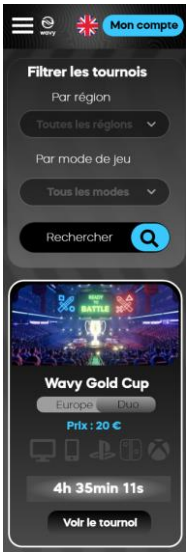
Fonctionnement du filtre par défaut :

- Les tournois sont rangés par chronologie : tournois à venir le plus proche jusqu'à celui dont la date est la plus éloignée, suivi des tournois les plus anciens.
- En statut utilisateur : il y a un filtre automatique sur les technologies utilisées par l'utilisateur et sur la visibilité « public ». L'utilisateur peut accéder à ses tournois privés dans la partie "mes tournois".
- L'administrateur peut voir tous les tournois : toutes technologies et toutes les visibilités

Maquette graphique de la page « liste des tournois »



Extrait maquette version mobile



2. Page détail d'un tournoi :

Cette page donne toutes les informations sur le tournoi et la possibilité de s'inscrire aux tournois.

Wireframe page détail d'un tournoi - version desktop

Extrait version mobile

Description :

- Nom du tournoi
- Photo (affiche du tournoi)
- Région et mode de jeu : solo/duo/trio/squad
- Prix
- Lien Discord
- Description
- Technologies (icônes)
- Date et heure de début et de fin
- Timer
- Visibilité : un cadenas s'affiche si privée
- Bouton « s'inscrire »

Information relative aux inscriptions :

- Nombre de joueurs inscrits / nombre maximum de participants

Informations complémentaires

À droite de la partie « description » :

Liste des parties avec :

- Nom de la partie
- Date et heure de début

Deux onglets en bas de page (Sections déroulantes, qui s'affichent au clic)

- Scores et règles
- Joueurs inscrits

Détail d'un tournoi terminé

Une fois le match terminé :

- Le bouton « inscription » et le timer disparaissent au profit d'une mention « terminé »
- Un onglet « classement » apparaît en dessous de la description, avec le placement de chaque équipe, le nom de l'équipe et ses membres, le prix, le nombre de parties jouées, les victoires et les nombres de points.

Le joueur s'inscrit à un tournoi donc à toutes les parties relatives à ce dernier. Toutes les parties participent au classement (qualifications, finales…). C'est l'organisateur qui définit le nombre de parties et à quoi elles correspondent.

3. Gestion des inscriptions :

Étape 1 : conditions d'accès à l'inscription

Au clic sur le bouton « s'inscrire », il faut vérifier si l'utilisateur est connecté et si le client est téléchargé par l'utilisateur avant de donner accès à l'inscription.

Étape 2 : création d'une équipe selon le mode de jeu :

Si le mode de jeu n'est pas solo :

L'utilisateur a la possibilité de récupérer une équipe qu'il a déjà créée ou de former une nouvelle équipe. Il n'a pas accès aux équipes des autres joueurs.

Si le mode de jeu est solo :

Un joueur = une équipe

Création d'une équipe :

- Nom
- Photo (avec une photo par défaut)
- Les membres (joueurs) avec un champ de recherche

Le joueur doit pouvoir rechercher d'autres joueurs pour les ajouter à l'équipe et les inscrire dans les champs prévus à cet effet.

Il est obligé de rajouter autant de joueurs que le prévoit le mode de jeu sinon il ne peut pas valider son inscription. Le nombre de champs pour ajouter un nouveau joueur s'affiche en fonction du nombre de joueurs maximum. Ce dernier est défini par l'organisateur (mode de jeu et nombre maximum de participants).

Étape 3 : Confirmation des inscriptions et ajout à la liste des participants

Les joueurs doivent recevoir une notification pour intégrer l'équipe et confirmer l'inscription au tournoi.

Si un des joueurs ne confirme pas, l'inscription au tournoi n'est pas validée.

Si tous les joueurs confirment l'inscription, il est inscrit au tournoi :

- L'équipe et les joueurs sont ajoutés à la liste des participants
- On envoie un mail de confirmation aux joueurs inscrits avec les principales informations du tournoi (nom, date de début, région et mode de jeu, lien Discord, prix) et un lien vers la page détaillée du tournoi.

À savoir : l'organisateur a le droit de modifier et de supprimer des participants.

Annexe 4 : Veille sur la sécurité : Liste des sites issus de la recherche

Français	English
Référence : https://www.cert.ssi.gouv.fr/ https://www.ssi.gouv.fr/	**Reference** : https://owasp.org/ https://cert.europa.eu/cert/filteredition/en/CERT-LatestNews.html
Actualités : https://www.silicon.fr/tag/cybersecurite https://www.usine-digitale.fr/cybersecurite/ https://www.zataz.com/	**News** : https://www.itsecurityguru.org/
Autres: https://www.lemagit.fr/ https://www.lemondeinformatique.fr/securite-informatique-3.html https://siecledigital.fr/cybersecurite/ https://www.futura-sciences.com/tech/cybersecurite/actualites/	**Others** : https://threatpost.com/ https://www.wired.com/category/security/ https://www.cyberscoop.com/ https://www.zdnet.com/ https://securityaffairs.co/wordpress/ https://www.infosecurity-magazine.com/ https://thehackernews.com/ https://cyware.com/cyber-security-news-articles https://cybersecuritynews.com/ https://www.reuters.com/subjects/cybersecurity https://www.cnbc.com/cybersecurity/ https://veille-cyber.com/

Annexe 5 : Extrait de la documentation d'Epic Games Services

Auth Interface :

The Auth interface allows logging in the local user with their Epic. This enables access to the features provided by Epic Account Services (EAS), such as Friends, Presence, UserInfo and Ecom interfaces. The Auth Interface handles Epic account-related interactions with EOS, providing the ability to authenticate users and obtain access tokens.

To use the Auth Interface, your must have Epic Account Services (EAS) active, and must obtain [user consent] to access Basic Profile data. You can activate EAS on the [Developer Portal], or learn more in [Epic's documentation]. Without EAS and user consent, you will still be able to initialize the EOS SDK and the Auth Interface, but all Auth Interface function calls to the back-end service will fail.

L'interface d'authentification :

L'interface d'authentification permet à l'utilisateur de se connecter localement avec son Epic Games. Cela permet d'accéder à des services Epic Games, tels que Amis, Présence, InfoUtilisateur et les interfaces Ecom. L'interface d'authentification gère les interactions liées au compte Epic avec EOS (Epic Online Services = Services Epic Games en ligne), offrant la possibilité de l'authentification des utilisateurs et d'obtenir des jetons d'accès.

Pour utiliser l'interface d'authentification, vous devez avoir un compte Epic Services (EAS) actif et devez obtenir le consentement de l'utilisateur pour l'accès à la base de données. Vous pouvez activer EAS sur le portail de développement ou lire la documentation d'Epic Games. Sans EAS et le consentement de l'utilisateur, vous pourrez toujours initialiser le EOS SDK (developer framework for online services = services en ligne pour les développeurs d'infrastructures) et l'interface d'Authentification, mais tous les appels d'interfaces d'authentifications vers les services back-end échoueront.

Source : https://dev.epicgames.com/docs/services/en-US/EpicAccountServices/AuthInterface/index.html

Epic Account Services (EAS) :

Before you can begin setting up an Epic Account Services and application, be sure that you fulfill these prerequisites:

- Signup for the Developer Portal and enable multi-factor authentication on the account.
- Create your own organization or join an existing one. You need an Epic Account or an Admin role to proceed.
- Create or pick a product to use Epic Account Services.
- Download the latest version of EOS SDK. Check the EOS Quick Start guide for more details.

Compte Epic Services (EAS) :

Avant de pouvoir commencer à configurer un compte et une application Epic Account Services, assurez-vous de remplir ces conditions préalables :

- Inscrivez-vous au portail des développeurs et activez l'authentification multifacteur sur le compte
- Créez votre propre organisation ou rejoignez une organisation existante. Vous avez besoin d'un compte Epic ou d'un rôle d'administrateur pour continuer.
- Créez ou choisissez un produit pour utiliser les services de compte Epic.
- Téléchargez la dernière version du SDK EOS . Pour plus de détails, lisez la documentation "démarrage rapide avec EOS".

Source : https://dev.epicgames.com/docs/services/en-US/EpicAccountServices/GettingStarted/index.html

Annexe 6 : Pistes de réflexions en termes de référencement

Choix des mots clés :

Pour la requête principale (test réalisé avec Google Trends)

- ⇨ Jeux vidéos
- ⇨ Wavy
- ⇨ Fortnite
- ⇨ Tournoi

Pour la requête secondaire (réalisé avec Ubersuggest : https://app.neilpatel.com/)

- ⇨ Fortnite jeu
- ⇨ jeux vidéo gratuits
- ⇨ tournoi Fortnite

Réalisation d'une FAQ à partir des questions que se posent les gens (réalisé avec answerthepublic.com)

Pour améliorer le référencement, il est intéressant de réaliser la FAQ à partir des questions que se posent les gens, car cela utilise les mots clés et cela augmente la fréquentation de la page et donc du site. Une FAQ permet également d'insérer des liens externes de qualité. Cette FAQ, en lien notamment avec les tournois et le jeu Fortnite, pourrait être accessible depuis le footer.

Exemple de questions pour la FAQ :

- **À partir de quel âge peut-on joueur à Fortnite ?**

  Selon la classification PEGI, Fortnite est déconseillé aux joueurs de moins de 12 ans, notamment du fait de « fréquentes scènes de violence modérée »

- **Est-ce qu'on jouer à Fortnite sur mobile?**

  Oui, il est possible de jouer à Fortnite sur Mobile. Vous pouvez d'ailleurs organiser des tournois sur Mobile avec Wavy.

- **Quel skin Fortnite es-tu?**

  Vous pouvez choisir votre personnage en fonction de sa personnalité

  Voir la description des personnages

- **Quelle configuration pour Fortnite ?**

  Fortnite est conçu pour fonctionner correctement sur divers systèmes PC

  Configuration recommandée :

  Carte graphique : *Nvidia GTX 960, AMD R9 280 ou processeur graphique DX11 équivalent*
  Mémoire vidéo : *2 Go de VRAM*

  Processeur : *Core i5-7300U 3,5 GHz, AMD Ryzen 3 3300U ou équivalent*

  Mémoire : *8 Go de RAM ou plus*

  SE : *Windows 10 64 bits*

  En savoir plus sur la configuration requise pour Fortnite sur le site d'Epic Games

- **Quels sont les tournois Fortnite aujourd'hui?**

  Des tournois sont organisés tous les jours

  Pour consulter tous les tournois prévus aujourd'hui (lien vers liste des tournois)