

DOSSIER DE PROJET

JDRCONNEXION

par ROCHES Laurent

Pour le titre de

Développeur Web et Web mobile

année 2024

REMERCIEMENT

Je tiens à remercier Simplon Grenoble, particulièrement aux formateurs, pour les connaissances qu'ils m'ont apportées, leurs lumières sur les zones d'ombre de mon projet.

Je remercie aussi mes collègues de formation, pour l'ambiance et l'entraide dont ils ont fait part tout au long de l'année!

Un grand merci aussi à M. Bisillon Stéphane pour m'avoir donné une chance de réaliser mon stage dans son entreprise, pour les connaissances qu'il m'a transmises. Et merci à toute son équipe pour leur bienveillance et leur accompagnement!

Enfin, je remercie surtout ma famille et mes proches, et plus encore ma femme, pour son soutien et sa patience indéfectible sans qui rien n'aurait été possible!

SOMMAIRE

Remerciement	- - - - -	Page 1
Sommaire	- - - - -	Page 2
Résumé	- - - - -	Page 4
Compétences engagées	- - - - -	Page 5
Cahier des charges	- - - - -	Page 6
Le Front-End	- - - - -	Page 6
La liste des pages	- - - - -	Page 8
Le Back-End	- - - - -	Page 8
L'Administration	- - - - -	Page 11
Environnement technique	- - - - -	Page 11
Mise en oeuvre des compétences	- - - - -	Page 15
Compétence 1	- - - - -	Page 15
Compétence 2	- - - - -	Page 17
Compétence 3	- - - - -	Page 20
Compétence 4	- - - - -	Page 23

Compétence 5	-----	Page 27
Compétence 6	-----	Page 30
Compétence 7	-----	Page 35
Compétence 8	-----	Page 38
Annexes	-----	Page 41

RÉSUMÉ

Dans le cadre de ma formation suivie chez Simplon Grenoble du 08 Janvier 2024 au 23 Novembre 2024, et afin d'obtenir le diplôme de Développeur Web et Web mobile, titre RNCP 37674 de niveau 5, équivalent à bac+2, il m'est demandé de réaliser un site internet ainsi qu'un dossier projet.

C'est dans ce cadre que j'ai réalisé ce projet autour du site JDRConnexion.

Il s'agit d'un site dans l'univers du Jeu de Rôle et dont le but est de permettre aux rôlistes de trouver leurs prochains partenaires d'aventure grâce à un système de tri autour des disponibilités renseignées par ces joueurs.

D'autres options accompagneront cette fonctionnalité comme la recherche d'un Maître du Jeu, la recherche de joueurs pour un jeu précis, etc...

Enfin, une partie du site sera dédiée à la présentation, par l'intermédiaire d'articles, de certains aspect du Jeu de Rôle et de son univers.

Ce projet est réalisé seul, de la naissance de l'idée à sa mise en ligne, il m'aura permis de développer toutes les compétences nécessaires au métier de développeur web.

LISTE DES COMPÉTENCES

- I – Installer et configurer son environnement de travail**
- II – Maquetter des interfaces utilisateur**
- III – Réaliser des interfaces utilisateur**
- IV – Développer la partie dynamique des interfaces utilisateur**
- V – Mettre en place une base de données relationnelle**
- VI – Développer des composants d'accès aux données**
- VII – Développer des composants métier**
- VIII – Documenter le déploiement d'une application dynamique web ou web mobile**

CAHIER DES CHARGES

Pour toutes références à l'aspect visuel du projet, la maquette est disponible sur le lien Figma suivant :

<https://www.figma.com/design/M1peDzc1mxLlOLJRyIr9Se/Untitled?node-id=0-1&node-type=canvas&t=CX1j6h9SeoOgNOcj-o>

Ce cahier des charges présente le projet de site nommé
JDRConnexion

Il s'agit d'un site dédié à aider les joueurs de jeux de rôle à trouver des partenaires afin de former des groupes pour leurs prochaines parties.

Le point essentiel de ce projet sera tourné autour des disponibilités renseignées par les utilisateurs.

I - Le Front-End

L'architecture front-end sera divisée en deux grandes parties, à savoir une partie visible par tout le monde et une autre visible uniquement par les administrateurs et employés (modérateur). L'aspect commun à ces parties se fera par le respect de la charte graphique et des polices d'écriture.

La charte graphique sera centrée autour de ces cinq couleurs :

-> #f2efbd = la couleur 'parchemin' du site, ce sera la couleur de fond principale mais elle pourra être une couleur d'écriture lorsque le contraste se fera sur une couleur plus sombre.

-> #8C3842 = la couleur ‘encre’ du site, ce sera la couleur d’écriture principale. Elle servira de fond au Header et au Footer du site.

-> #86388C = la couleur ‘utilisateur’, elle sera utilisée pour toutes les actions en lien avec un utilisateur.

-> #38718C = la couleur ‘article’, cette couleur marquera les actions liées aux articles du site.

-> #5F558C = couleur à mi-chemin entre celle des utilisateurs et celle des articles, elle fera le lien entre ces actions principalement grâce au mécanisme du hover.

-> Enfin, en vrac, les couleurs = #286D22, #DB0000, #f2f2f2 et #292929 sont respectivement les couleurs du succès, de l’erreur, du blanc et du noir.

Pour un site ou un forum traitant des jeux de rôle (JDR), il est essentiel de choisir une police d’écriture qui reflète bien l’atmosphère immersive et souvent fantastique des JDR, tout en restant lisible pour les utilisateurs. C'est pourquoi j'ai fait le choix suivant :

-> Cinzel : Une police d’écriture inspirée des inscriptions romaines, idéale pour donner une touche épique et historique.
Elle sera utilisée principalement pour les titres.

-> Merriweather : J'ai choisi la police sans-serif, très lisible et élégante, qui s'adapte bien à des textes longs tout en gardant une certaine touche classique.

Ces combinaisons permettent d'apporter une touche immersive pour les passionnés de JDR, tout en assurant une bonne lisibilité et un confort de lecture pour les utilisateurs du site ou du forum.

II - Listes des principales pages public

Voici une liste non exhaustive des pages accessibles par le public :

- > Accueil
- > Inscription
- > Connexion
- > Profil
- > Liste des utilisateurs
- > Liste des articles
- > Article
- > Disponibilités
- > Jeux connus et voulus
- > Messages

III - Le Back-End

Dans cette partie, nous allons voir les principales fonctionnalités attendues sur chacune des pages précédemment mentionnées. Avant toute chose, toutes les pages du site se verront attribuer un Header et un Footer.

Le Header aura le logo et le nom du site ‘JDRConnexion’ servant de retour vers la page d’accueil, ainsi qu’une navbar guidant vers les principales pages du site. Ces liens seront soumis à condition! A savoir, pour toute personne arrivant sur le site, un lien vers les pages de ‘connexion’, ‘inscription’, ‘liste des utilisateurs’ et ‘liste des articles’.

Tandis qu'une personne déjà connectée verra en lieu et place des pages de connexion et d'inscription les liens 'mon profil' et 'déconnexion'.

Enfin, si la personne connectée est un administrateur ou un modérateur, une cinquième option se montrera : 'administration'.

Le Footer aura le logo du site servant de balise de retour vers l'accueil, une section vers les réseaux sociaux de JDRConnexion (éléments à venir), le lien vers les Conditions Générales d'Utilisation (CGU) et enfin une petite note sur le créateur du site.

La page d'accueil, après une partie statique présentant le site, mettra en avant les miniatures des trois articles les plus récents. Elles seront cliquables et mèneront vers la page dédiée à l'article correspondant. S'en suivra un bouton invitant à se rendre sur la page contenant la liste de tous les articles.

Puis un petit tableau montrera les trois utilisateurs ayant le meilleur ratio "aimé/nombre d'avis", l'idée étant d'encourager les bons comportements. De la même façon que les articles, les pseudonymes et images de profil des utilisateurs seront cliquables afin de mener vers leur profil. Puis, pour clôturer la page, un bouton conduira les utilisateurs vers le listing des utilisateurs enregistrés sur le site.

La page d'inscription sera un formulaire demandant les informations suivantes à l'utilisateur : son adresse e-mail, son nom, son prénom, son pseudonyme et sa date de naissance.

Le pseudonyme et l'adresse e-mail seront uniques. Une fois cette étape faite, un mail sera envoyé à la personne afin de finaliser son inscription. Dans cette deuxième étape, l'utilisateur enregistrera son mot de passe et validera ainsi son inscription.

La page de connexion permettra à l'utilisateur de se connecter grâce à son adresse e-mail et son mot de passe. Un lien lui permettra de recevoir un nouveau mail s'il venait à oublier son mot de passe.

La page de profil montrera l'image de profil, le pseudonyme de l'utilisateur visité, s'il est joueur ou bien Maître du Jeu, quand son compte a-t-il été créé, son niveau de joueur, une description qu'il se sera donnée.

Une liste de boutons mènera vers différentes parties à savoir : les messages, les disponibilités, les jeux connus et voulus. Deux boutons montreront les nombres de fois où le profil a été aimé ou non. Enfin, si le profil appartient à la personne connectée, deux boutons supplémentaires pourront amener vers la modification du profil ou vers sa suppression.

Globalement, si l'utilisateur est un simple visiteur non connecté, il ne peut qu'observer. S'il est connecté mais sur le profil différent du sien, il pourra envoyer un message, voter s'il aime ou non ce profil. Enfin, s'il est connecté et qu'il s'agit de son profil, il peut le modifier, le supprimer, voir la liste des messages qui lui ont été envoyés, modifier ses disponibilités, enfin il ne pourra pas modifier le nombre de fois où son profil a été aimé ou non.

La page présentant la liste des joueurs inscrits sur le site les affichera triés dans un premier temps par leur ratio "aimé/nombre d'avis" par paquet de dix avec un système de pagination.

L'utilisateur pourra affiner sa recherche grâce à un formulaire en haut de page. Il pourra trier cette liste par pseudonyme, par 'rôle' (joueur ou Maître Du jeu), par jeu voulu et enfin et surtout selon ses disponibilités.

La page listant les articles du site reprendra le fait de mettre un peu en avant les trois derniers articles, puis listera les suivants sous la forme de vignette.

La page détaillant un article permettra sa visualisation et, pour les utilisateurs connectés, il sera possible de laisser son avis sur cet article.

III - L'Administration

Cette partie s'adresse au cas où l'utilisateur, en plus d'être connecté, se révèle être un administrateur ou un modérateur. En effet, ils pourront ajouter, modifier ou supprimer notamment : Les articles et leurs catégories, les jeux et leurs catégories, la liste des mots tabous. Ils pourront aussi visualiser l'ensemble des messages et des avis laissés par les autres utilisateurs. Derniers éléments, ils pourront voir la liste des personnes inscrites sur le site et, s'il s'agit d'un administrateur, il pourra changer le rôle d'un usager en modérateur et inversement.

ENVIRONNEMENT TECHNIQUE

Je vais vous exposer mes choix de technologies pour réaliser ce projet.

Mon choix, influencé par les cours que j'ai reçus à Simplon ainsi que par les deux mois et demi de stage réalisés dans l'entreprise 'Web Media Com' à Voiron, s'est donc porté sur :
pour la partie Front-End

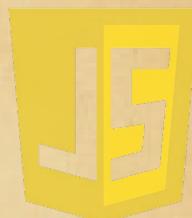
HTML



CSS

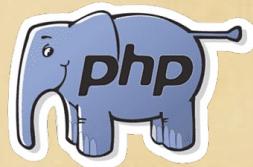


JS



HTML, CSS & JS.

Et pour la partie Back-End



PHP.

Comme vous le constatez, j'ai fait le choix de n'utiliser aucun framework. Plusieurs raisons à cela ! Premièrement, comme je l'ai enoncé, à Simplon, ce sont ces langages natifs que nous avons le plus vus. Ensuite, l'entreprise 'Web Media Com' où j'ai réalisé mon stage code elle-même en PHP natif, comme je compte réaliser, à leur côté, mon alternance et les premières années de ma carrière de développeur, c'est donc naturellement que mon choix s'est tourné vers ces technologies. Enfin, d'un point de vue réaliste, Le projet JDRConnexion ne nécessite pas tous les fichiers qui sont installés par défaut dans des projets utilisant des frameworks comme Symfony ou Next.js pour ne citer que ceux vus pendant ma formation.

Afin de pouvoir tester et développer mon application web sur un serveur local depuis mon environnement Windows, j'ai utilisé WAMP. Avec son serveur Apache intégré, sa gestion de base de données MySQL et son outil PhpMyAdmin, WAMP fournit un environnement de développement complet et prêt à l'emploi pour PHP, permettant de travailler efficacement en local.



J'ai aussi utilisé Git dans mon projet. Cet outil est un système de gestion de versions largement utilisé dans le développement. Git permet de suivre les modifications apportées au code au fil du temps, de revenir à une version antérieure en cas de problème, de garder un historique détaillé des modifications, ce qui est utile pour le suivi du projet et la résolution de bugs. Enfin dans mon cas, Git me permet d'alterner facilement entre mon ordinateur portable lorsque je suis à Simplon et mon ordinateur fixe lorsque je code depuis chez moi.



Enfin, dernier élément, pour la mise en ligne de mon site, j'ai fait appel à o2switch.

C'est une société française spécialisée dans l'hébergement web. Elle propose des services d'hébergement mutualisé, principalement destinés aux sites internet et applications web. Les serveurs d'o2switch sont situés en France et ils permettent de créer plusieurs sites, bases de données et d'utiliser une grande capacité de stockage sans restriction. Le service client est réputé et disponible 24h/24, 7j/7. Pour finir, grâce au cPanel, inclus dans l'offre, qui est un panneau de contrôle parmi les plus populaires et conviviaux du marché, et qui permet de gérer facilement les fichiers, bases de données, emails, et autres aspects de l'hébergement.



MISE EN OEUVRE DES COMPÉTENCES

I - Installer et configurer son environnement de travail

Dans cette partie, je vais vous exposer mon installation de WAMP sur mon ordinateur car il me semble être un élément essentiel et représentatif.

Tout commence avec une recherche internet pour trouver le site (Annexe 1), toutefois, comme nous l'avons vu en cours, je ne me rends pas sur le premier lien mais sur le deuxième : wampserver.aviatechno.net

Car cette page regroupe tous les éléments nécessaires à l'installation de WAMP et pas seulement le “.exe” de son installation.

Installers Wampserver full install version

Last Wampserver full install version

 [Wampserver 3.3.5 64 bit x64 - Apache 2.4.59 - PHP 7.4.33/8.0.30/8.1.28/8.2.18/8.3.6 - MySQL 8.3.0 - MariaDB 11.3.2 MDS](#)

Also includes PhpMyAdmin 5.2.1 - Adminer 4.8.1 - PhpSysInfo 3.4.3.
Install Wampserver as an administrator Launch Wampmanager via the Shortcut.

As of May 9, 2023 and in agreement with Maximus23, developer of the Aestan Tray Menu, discontinuation of 32bit support for Wampserver and installation under Windows 7.
From April 1, 2024, support from Windows 10 only

[See 32bit versions](#)

Qui récupère le “.exe”, mais aussi un outil permettant de vérifier si les paquetages VC++ nécessaires à WAMP sont correctement installés.

Tools

- [wampmanager.ini repair](#) MDS
Repair tool for wampmanager.ini file if it is corrupted or missing. You can download at any time, but install this tool only if you need it, because it takes into account the current configuration of Wampserver. A wampmamanger.ini corrupted file is reflected mostly by errors like "[EparsingError]" or "Could not execute run action" when launching Wampserver.
- [Checks VC++ packages installed](#) MDS
Tool to check if the VC++ packages needed to Wampserver 3 are installed correctly.
Do not use a previously loaded tool. Always make a new download to make sure you are using the correct version.

Visual C++ Redistributable Packages

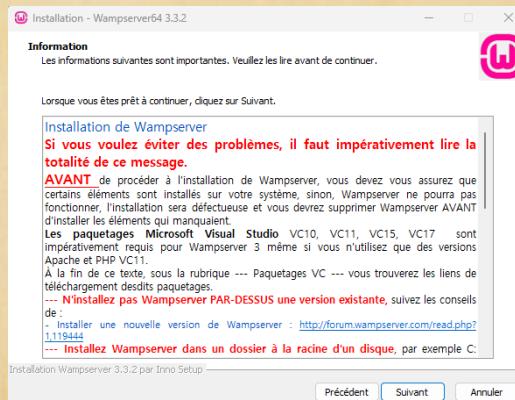
The latest Microsoft VC++ packages are available at : [Microsoft Visual C++ Redistributable Latest Supported Downloads](#)

[All available versions](#)

Microsoft VC++ Packages x86 (32 bits)	Microsoft VC Packages x64 (64bits)
VC 2008 (VC9) is required for PHP 5.3 and 5.4. VC 2010 SP1 Package (x86) <small>MDS</small> VC 2012 Update 4 (x86) <small>MDS</small> VC 2013 Package Up 5 (x86) <small>MDS</small> VC 2015-2022 (VC17 x86) 14.40.33810 <small>MDS</small>	VC 2008 (VC9) is required for PHP 5.3 and 5.4. VC 2010 SP1 Package (x64) <small>MDS</small> VC 2012 Up 4 (x64) <small>MDS</small> VC 2013 Up 5 (x64) <small>MDS</small> VC 2015-2022 (VC17 x64) 14.40.33810 <small>MDS</small>

VC2015-2022 (VC17) is backward compatible to VC2015 (VC14), VC2017 (VC15) and VC2019 (VC16). That means, a VC14, VC15 or VC16 module can be used inside a VC17 binary. Because this compatibility the version number of the Redistributable is 14.3x.xx and after you install the Redistributable VC2015-2022, the Redistributable packages VC2015 (VC14), VC2017 (VC15) and VC2019 (VC16) are eventually deleted but you can still use VC14, VC15 and VC16.

Il suffit dès lors de lancer l'installation du “.exe”, après avoir accepté les termes du contrat de licence, nous sommes prévenus des prérequis à l'installation :



A ce moment de l'installation, on peut soit installer tous les paquetages C++ présents sur le site, Windows nous informant lorsque la version installée est plus récente, soit utiliser l'outil proposé qui vérifiera les paquetages C++ pour nous et nous donnera la liste de ceux nécessaires à installer ou mettre à jour.
 Il faut ensuite placer cette installation à la racine de l'ordinateur : C:\ puis un dossier sera créé comme ceci : C:\wamp64

II – Maquetter des interfaces utilisateur

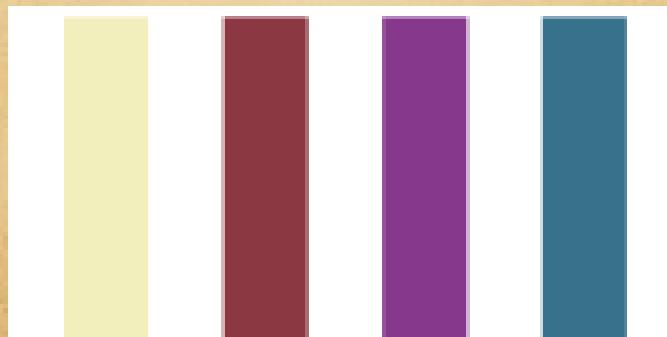
L'aspect visuel de mon site a été pensé après un “brainstorming”. J'ai cherché ce qui pourrait faire penser à l'image du Jeu de Rôle, de la convivialité et des rencontres. Et après avoir arrêté mon choix sur une thématique “héroïque fantaisie”, je me suis rendu sur l'IA copilot afin de créer ce qui sera le logo de mon site :

Une chope de bière, une dague et surtout des dés! Autant d'éléments évoquant une aventure d'Héroïque-Fantaisie commençant dans une taverne.



Ensuite, depuis le site Adobe Color et leurs outils “extraire le thème”, j'ai extrait les cinq couleurs principales liées à mon logo.

N'étant toutefois pas entièrement satisfait des couleurs ainsi récupérées. J'ai gardé la couleur la plus pertinente selon moi et j'ai ensuite utilisé l'outil “roue chromatique” toujours chez Adobe Color avec les options “Triade” ou “Carré” afin de former les couleurs de ma charte graphique. Je finis par garder les quatre couleurs suivantes :



Pour finaliser ce choix de couleurs, je me suis rendu sur deux sites afin de tester le contraste entre mes couleurs et ainsi m'assurer de l'accessibilité visuelle que j'offrirai :

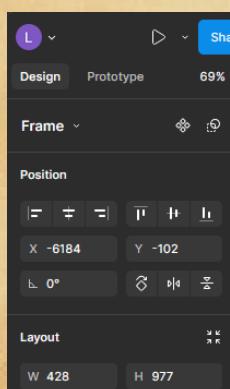
D'abord sur Adobe Color grâce à leur outil "Outils d'accessibilité", puis sur le site : webaim.org/resources/contrastchecker/
L'objectif étant d'avoir un score supérieur à 5,5.

Le dernier élément important pour l'aspect visuel du site et qu'il ne faut pas sous-estimer est la police d'écriture! Je voulais des polices d'écriture lisible mais restant dans le thème, j'ai donc arrêté mon choix sur les polices Cinzel et Merriweather Sans-Serif.

Une fois ces étapes préliminaires faites, je me suis rendu sur Figma afin de commencer le maquettage à proprement parler.

Vous trouverez un aperçu de la maquette finale dans l'annexe n°2. J'ai commencé par rassembler dans un grand ensemble "assets" les éléments généraux du projet, à savoir pour l'heure, le logo et la charte graphique.

Puis, pour commencer le maquettage de la version mobile, j'ai utilisé les "Frames" proposés par Figma.



Ici, on peut voir que les dimensions de cette Frame sont de 428 pixels de large pour 977 pixels de haut, soit la taille standard d'un écran de téléphone portable.

Le fait d'utiliser les Frames et le système de prototypage proposé par Figma permet ensuite de simuler la navigation dans la maquette comme s'il s'agissait du site internet.

Aussi, comme il est important de coder selon la méthodologie dite "mobile first", il m'a semblé logique de réaliser la maquette de mon site selon le même paradigme.

À savoir que, pour résumer, cette pratique repose sur le fait que les mobiles occupent une part importante de la navigation internet et aussi que les téléphones portables ont souvent moins de ressources que les ordinateurs. De plus, en codant d'abord pour le mobile, il est plus simple d'ajouter des éléments pour correspondre aux besoins des ordinateurs que d'en enlever pour leur version mobile. Et enfin dernier point important, le fait de coder en "mobile first" permet souvent d'avoir un code plus simple et plus propre, car on commence par les besoins essentiels, avant d'ajouter des éléments plus complexes pour les écrans plus grands.

En m'appuyant sur mon cahier des charges, j'ai réalisé les maquettes de toutes les pages principales, visibles par les utilisateurs, en commençant par l'accueil et en simulant le cheminement possible d'un usager.

Une fois la maquette mobile faite, je me suis penché sur celle pour ordinateur en reprenant le travail précédemment réalisé (Annexe 3), mais je n'ai mis que les pages principales qui m'ont permis de ressortir les différents éléments "redondants" du site.

Ainsi grâce à cette maquette, on peut voir le formulaire type, le tableau type, les vignettes type, etc...

J'ai aussi fait le choix, à ce moment du projet, de gagner un peu de temps en ne prototypant pas la maquette pour ordinateur puisque le cheminement était déjà réalisé sur la version mobile.

III - Réaliser des interfaces utilisateur

Dans cet aspect de la construction de JDRConnexion, plusieurs points essentiels ont été respectés.

Il faut que le site soit conforme à la maquette réalisée et, dans sa construction, il faut d'ores et déjà prendre en compte le référencement naturel.

Pour ce faire, il est important de respecter la hiérarchisation des titres dans la page (h1, h2, h3, etc.)

```
<div class="accueil_presentation">
  <h2 class="accueil_titre">Bienvenue sur JDRConnexion - Le Portail des Aventuriers</h2>
  <p>Vous êtes un passionné de jeux de rôle, toujours à la recherche de nouvelles aventures et de compagnons pour partager des quêtes épiques? JDRConnexion est là pour vous! Notre plateforme est spécialement conçue pour vous aider à trouver d'autres joueurs qui partagent vos envies, vos horaires, et votre passion pour l'univers du jeu de rôle. Que vous soyez un maître du jeu chevronné ou un novice en quête de votre première campagne, vous trouverez ici une communauté accueillante prête à explorer ensemble les contrées les plus fantastiques.</p>
  <div class="accueil_presentation_2">
    <img src=<?= HOME_URL ?>img/accueil_social.jpg" alt="Image symbolisant la sociabilité du site JDRConnexion" class="accueil_image_social">
    <div class="accueil_texte_social">
      <h3 class="accueil_titre_section">Grâce à JDRConnexion, vous pouvez facilement entrer en contact avec des joueurs partageant vos goûts et votre disponibilité.</h3>
      <p>Que vous préfériez les aventures médiévales fantastiques, les explorations spatiales, ou les intrigues mystérieuses, notre système de recherche vous permet de trouver des partenaires de jeu en quelques clics. Plus besoin de chercher pendant des heures, le groupe parfait est à portée de main. Plongez dans l'univers du jeu de rôle!</p>
```

Dans cette illustration, on visualise un titre h2 englobant une section h3.

Pour le référencement, j'ai aussi utilisé d'autres méthodes comme le choix d'un titre adapté et court dans la balise `<title> </title>` (Annexe 4), l'utilisation de 'Méta description' qui reflètent avec précision le contenu de chaque page, ou encore le fait de désindexer certaines pages afin que les bots les ignorent telles que les pages Conditions Générales d'Utilisation ou 404 not found.

```
<head>
  <meta name="robots" content="noindex">
</head>
```

```
<head>
  <meta name="description" content="JDRConnexion, le site idéal pour trouver des partenaires et former un groupe pour vos parties de Jeu de Rôle. Rejoignez-nous pour vivre des aventures épiques!">
</head>
```

Il est aussi important de penser à l'expérience utilisateur et notamment pour les personnes en situation de handicap. Pour ce faire, j'ai utilisé, entre autres, des balises ** ou **** qui permettent visuellement de mettre son contenu en italique ou en gras mais aussi aux lecteurs d'écrans d'utiliser une intonation différente pour signaler l'emphase. Il est aussi important de compléter les attributs "alt" des balises . Cela permet aux lecteurs d'écrans de décrire l'image grâce à son contenu, il permet aussi un meilleur référencement naturel, et enfin, en cas de mauvais chargement de l'image, il permet d'afficher ce texte aux visiteurs.

```
img/dnd_logo.png" alt="Logo du jeu Donjon & Dragon pour le site JDRConnexion">

```

Afin de produire un site conforme à la maquette précédemment réalisée, j'ai codé trois fichiers de code CSS (Annexe 4). Il aurait été possible de le faire en seulement deux fichiers, mais c'est ici un choix purement personnel afin de faciliter ma gymnastique mentale entre les fichiers.

L'ordre d'utilisation de ces fichiers assure le respect du paradigme 'mobile first' et assure la responsivité de mon site.

De façon générale, j'ai instancié ma charte graphique, et défini les polices d'écriture selon le respect du cahier des charges :

```
:root {
  --blanc: #f2f2f2;
  --parchemin: #f2efbd;
  --ecriture: #80C3842;
  --utilisateur: #86388C;
  --hover: #F5558C;
  --article: #38718C;
  --noir: #292929;
  --succes: #286D22;
  --erreur: #DB0000;
}

body {
  font-family: 'Merriweather', sans-serif;
  background-color: var(--parchemin) !important;
  color: var(--ecriture) !important;
}

h1, h2, h3 {
  font-family: 'Cinzel';
}
```

Enfin, il est possible, dans une certaine mesure, de sécuriser notre site internet grâce au code HTML, notamment lors de la soumission de formulaires. Pour ce faire, nous pouvons nous assurer de l'envoi des informations attendues par le biais de l'attribut ‘requière’ et aussi de leur forme lors de l'envoi en choisissant le bon type de `<input>`.

```
<div class="connexion_champs">
    <label for="str_email" class="">Adresse email :</label>
    <input id="str_email" name="str_email" type="email" autocomplete="email" required class="">
</div>

<div class="disponibilte_champs">
    <label for="time_debut_0">Heure de début :</label>
    <input id="time_debut_0" name="time_debut[]" type="time" required>
</div>
<div class="disponibilte_champs">
    <label for="time_fin_0">Heure de fin :</label>
    <input id="time_fin_0" name="time_fin[]" type="time" required>
</div>
```

Même si ces sécurités sont facilement contournables, elles permettent une meilleure expérience utilisateur pour les visiteurs de bonne foi qui commettraient seulement une erreur, en les guidant, sans nécessiter le rechargeement de la page, avant même l'envoi du formulaire.

IV - Développer la partie dynamique des interfaces utilisateur

Cette étape est cruciale pour offrir une expérience interactive et accessible aux utilisateurs, tout en respectant les bonnes pratiques en matière de sécurité et de réglementation, notamment celles relatives à l'accessibilité.

L'objectif principal étant de créer une interface dynamique, en harmonie avec le dossier de conception initial, tout en garantissant une expérience utilisateur fluide, inclusive et sécurisée. Je vais détailler dans cette section comment j'ai mis en œuvre ces principes à travers différentes actions concrètes.

Dans un premier temps, de façon très visuelle, j'ai codé mon élément central pour la navigation sur mon site : la 'navbar' qui, en plus de remplir son rôle d'aiguilleur, est aussi responsive en passant d'un 'menu burger' depuis sa version mobile à un header dans sa version ordinateur comme demandé depuis la maquette.

Pour parvenir à ce résultat, j'ai utilisé un peu de code JavaScript et du code CSS comme ceci :

```
public > js > burger.js > ...
1 const burger = document.querySelector('.burger');
2 const navLinks = document.querySelector('.nav-links');
3
4 burger.addEventListener('click', () => {
5   navLinks.classList.toggle('active');
6   burger.classList.toggle('toggle');
7 });


```

```
.navbar .nav-links.active {
  transform: translateX(0%);
  background-color: var(--ecriture);
  z-index: 3;
  display: flex;
}
.burger.toggle .line1 {
  transform: rotate(-45deg) translate(-5px, 6px);
}
.burger.toggle .line2 {
  opacity: 0;
}
.burger.toggle .line3 {
  transform: rotate(45deg) translate(-5px, -6px);
}
```

En haut le code JS et à droite un extrait du code CSS pour le menu burger / navbar

Toujours grâce à JavaScript, j'ai aussi pu permettre aux utilisateurs rempliesant leur formulaire de disponibilités, d'ajouter des choix avant l'envoi de leurs données.

Ceci permet d'ajouter autant d'entrées dans le formulaire que le souhaite l'utilisateur sans qu'il n'ait besoin d'envoyer ses disponibilités une à une.

On voit ainsi que dans ce code, j'incrémente de façon dynamique mon HTML grâce à JS et j'anticipe mon traitement de l'information en PHP avec \${index}.

```
public > js > form_dispojs > ...
1  document.getElementById('ajouter_dispo').addEventListener('click', function() {
2
3      let container = document.getElementById('disponibilite_container');
4
5      let index = container.getElementsByClassName('disponibilite_item').length;
6
7      let newDispo = document.createElement('div');
8      newDispo.classList.add('disponibilite_item');
9      newDispo.innerHTML = `
10         <div class="disponibilite_champs">
11             <label for="str_jour_${index}">Jour disponible :</label>
12             <select name="str_jour[$index]" id="str_jour_${index}">
13                 <option value="lundi">lundi</option>
14                 <option value="mardi">Mardi</option>
15                 <option value="mercredi">Mercredi</option>
16                 <option value="jeudi">Jeudi</option>
17                 <option value="vendredi">Vendredi</option>
18                 <option value="samedi">Samedi</option>
19                 <option value="dimanche">Dimanche</option>
20             </select>
21         </div>
22         <div class="disponibilite_champs">
23             <label for="time_debut_${index}">Heure de début :</label>
24             <input id="time_debut_${index}" name="time_debut[$index]" type="time" required>
25         </div>
26         <div class="disponibilite_champs">
27             <label for="time_fin_${index}">Heure de fin :</label>
28             <input id="time_fin_${index}" name="time_fin[$index]" type="time" required>
29         </div>
30     `;
31
32     container.appendChild(newDispo);
33 });

```

Ainsi, avec ce code JS, lorsque l'utilisateur m'envoie des données contenant plusieurs disponibilités en même temps, je peux traiter l'information sous forme d'un tableau proprement indexé grâce à la récursivité du code.

Un autre aspect du code dynamique présent dans mon site JDRConnexion est la complétion de mon code HTML directement avec les données récupérées directement depuis la Base De Données (BDD).

Un exemple simple de cet aspect est le tableau contenant les trois utilisateurs avec le meilleur ratio “aimé / nombre d'avis” présent sur la page d'accueil. Après avoir récupéré cette liste par l'intermédiaire d'une requête SQL, après l'avoir sécurisée, je récupère un tableau \$liste_user (en PHP) et j'applique le code comme dans l'annexe 5

à savoir, l'utilisation de l'outil 'foreach' qui me permet de parcourir ce tableau et d'en extraire chaque utilisateur, et ainsi de construire ma page HTML avec les éléments dont j'ai besoin.

J'ai évoqué la récupération des données depuis la BDD, je reviendrai sur ce point plus loin dans ce dossier. Cependant, je vais expliciter maintenant comment j'ai sécurisé ces données.

La sécurité est un élément important du développement. Il est nécessaire à toutes les étapes du cheminement de la donnée. Nous avons déjà aperçu avec le développement statique d'une page la première étape de cette sécurisation avec les attributs 'required' et 'type' des inputs.

L'étape suivante survient lors de l'envoi de ces données. Je vais ici prendre l'exemple d'une inscription d'un utilisateur car cela me semble suffisamment représentatif.

```
use Reponse;
use Securite;
use Censure;

public function connexion():void { ... }

public function inscription() {
    $data = $_POST;
    $data = $this->sanitize($data);
```

Dans cette étape je récupère les données envoyées avec la ligne \$data = \$_POST, et on peut observer que j'utilise ici trois services : 'use Reponse', 'use Securite' et 'use Censure'. Les deux derniers sont directement liés avec

la sécurisation des données. Vous pouvez les observer dans leur intégralité dans l'annexe n°6.

La fonction sanitize passe les données en fonction de leur typage, soit dans la fonction native 'intval()' pour les nombres entiers, soit dans la fonction native 'htmlspecialchars()' pour les chaînes de caractères, avant de les retourner.

La fonction filtrer quant à elle me sert à vérifier que le pseudonyme, les messages ou les avis laissés par les utilisateurs ne contiennent pas d'injures, et le cas échéant, les remplacer par des astérisques '***'. Toujours à cette étape de l'inscription, pour sécuriser les données soumises, je soumets le tableau récolté à une série de tests comme

vous pouvez apercevoir une grande partie dans l'annexe n°7. Je vérifie ici que j'ai bien reçu le nombre de données attendues, que leur typage correspond bien à ce que j'avais prévu, je vérifie aussi certains éléments comme le fait que l'adresse e-mail ou le pseudonyme ne soient pas déjà utilisés par un autre utilisateur, ou encore que la date de naissance ne soit ni dans le futur, ni avant le 01 Janvier 1900.

Cette sécurisation de la donnée se poursuivra encore, mais nous verrons la suite plus tard lorsque je traiterai de l'interaction avec la base de données.

Un autre élément ne participant pas à la sécurité directement mais permettant une meilleure maintenabilité du code est la pratique de commenter son code afin de savoir ou se rappeler plus rapidement son utilité.

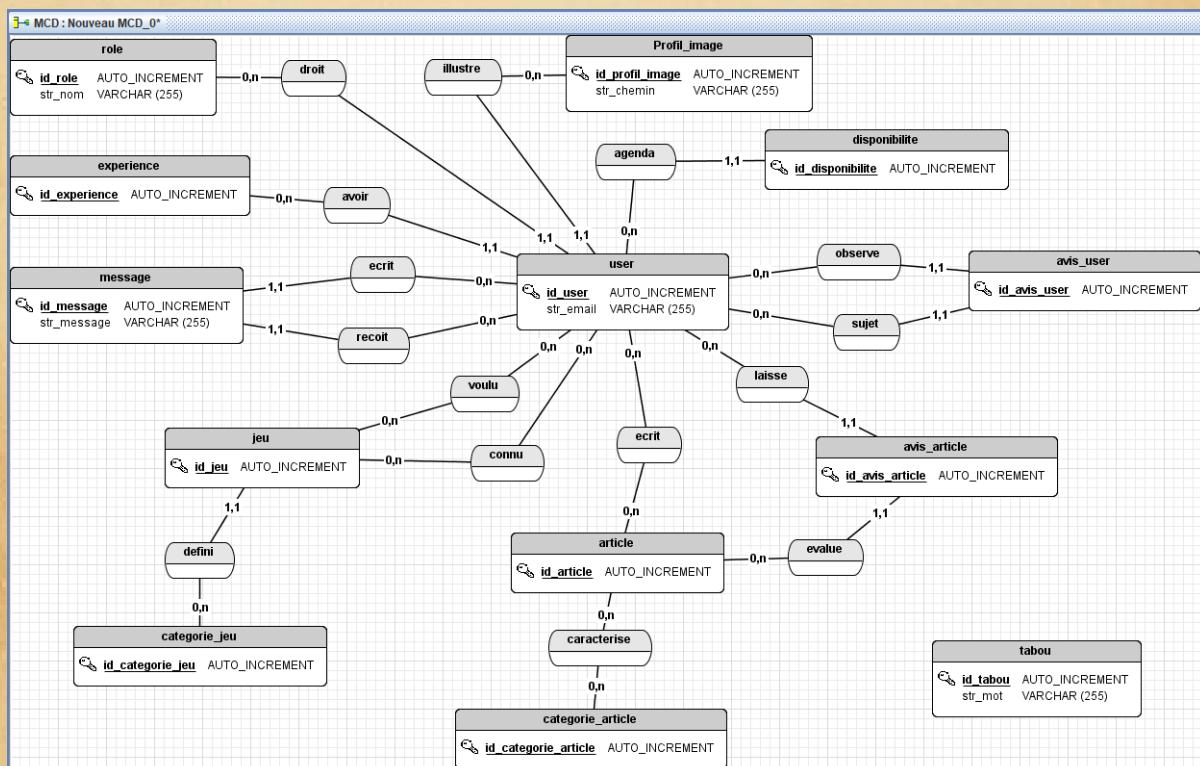
```
/**  
 * Get the value of id_user  
 */  
public function getIdUser(): int  
{  
    return $this->id_user;  
}  
  
/**  
 * Set the value of id_user  
 */  
public function setIdUser(int $id_user): self  
{  
    $this->id_user = $id_user;  
  
    return $this;  
}
```

```
/**  
 * Crée un utilisateur  
 *  
 * @param User $user les données sous forme de l'objet User  
 *  
 * @return User      Retourne vrai si l'enregistrement s'est bien opéré  
 */  
public function createUser(User $user): bool {...}  
  
/**  
 * Récupère tous les utilisateurs en fonction du tri, paginer et classer selon leur ratio de Like  
 *  
 * @param int     $id_game   identifiant du jeu souhaité  
 * @param string  $str_pseudo pseudonyme du joueur rechercher  
 * @param int     $bln_mj    booléen de Maître du Jeu  
 * @param string  $str_jour  jour disponible  
 * @param string  $time_debut heure de début  
 * @param string  $time_fin  heure de fin  
 * @param int     $page     numéro de la page souhaité  
 * @param int     $perPage   nombre de résultat par page  
 *  
 * @return array        Un tableau contenant la liste des utilisateurs trouvés  
 */  
public function getAllUser(int $id_game = null, string $str_pseudo = '', int $bln_mj = null, string  
$str_jour = null, string $time_debut = null, string $time_fin = null, int $page = 1, int $perPage = 10)  
{  
    $array = $this->getDB()->query("SELECT * FROM users WHERE 1=1");  
    if ($str_pseudo != '') {  
        $array = $array->where("pseudo LIKE '%{$str_pseudo}%'");  
    }  
    if ($id_game != null) {  
        $array = $array->where("id_game = {$id_game}");  
    }  
    if ($bln_mj != null) {  
        $array = $array->where("bln_mj = {$bln_mj}");  
    }  
    if ($str_jour != null) {  
        $array = $array->where("str_jour = {$str_jour}");  
    }  
    if ($time_debut != null) {  
        $array = $array->where("time_debut = {$time_debut}");  
    }  
    if ($time_fin != null) {  
        $array = $array->where("time_fin = {$time_fin}");  
    }  
    $array = $array->order("ratio DESC")->limit($page, $perPage);  
    $array = $array->get();  
    $array = $array->toArray();  
    return $array;  
}  
  
/**  
 * Décompte de tous les utilisateurs trouvé selon les critères de recherche  
 *  
 * @param int     $id_game   identifiant de jeu souhaité  
 * @param string  $str_pseudo pseudonyme du joueur rechercher  
 * @param int     $bln_mj    booléen de Maître du Jeu  
 * @param string  $str_jour  jour disponible  
 * @param string  $time_debut heure de début  
 * @param string  $time_fin  heure de fin  
 *  
 * @return int      La valeur numérique récupérée  
 */  
public function countAllUser(int $id_game = null, string $str_pseudo = '', int $bln_mj = null, string  
$str_jour = null, string $time_debut = null, string $time_fin = null)  
{  
    $array = $this->getDB()->query("SELECT COUNT(*) AS count FROM users WHERE 1=1");  
    if ($str_pseudo != '') {  
        $array = $array->where("pseudo LIKE '%{$str_pseudo}%'");  
    }  
    if ($id_game != null) {  
        $array = $array->where("id_game = {$id_game}");  
    }  
    if ($bln_mj != null) {  
        $array = $array->where("bln_mj = {$bln_mj}");  
    }  
    if ($str_jour != null) {  
        $array = $array->where("str_jour = {$str_jour}");  
    }  
    if ($time_debut != null) {  
        $array = $array->where("time_debut = {$time_debut}");  
    }  
    if ($time_fin != null) {  
        $array = $array->where("time_fin = {$time_fin}");  
    }  
    $array = $array->get();  
    $array = $array->toArray();  
    return $array[0]['count'];  
}
```

En haut des commentaires détaillés en français et à gauche des commentaires plus succincts en anglais.

V - Mettre en place une base de données relationnelle

Pour réaliser ce projet, il me fallait une base de données centrée sur l'utilisateur. Je me suis aidé d'un outil JMerise afin de visualiser les tables nécessaires et leurs connexions :



Cette image est, certes grande, mais essentielle pour illustrer les connexions entre les tables de ma base de données. Il s'agit d'un MCD (Modèle Conceptuel des Données). J'ai volontairement, à cette étape, omis le contenu des tables car je voulais coder le script SQL personnellement sans avoir recourt à un outil extérieur et il n'apportait donc rien de plus à ce moment de la construction. Plusieurs schémas se dégagent du MCD. Premièrement, les tables comme "expérience", "rôle" ou encore

“profil_image” ont des relations ‘many-to-one’ avec la table “user” ce qui veut dire que la table “user” va recevoir les clés étrangères (foreign keys) des tables “rôle”, “expérience” et “profil_image” en son sein.

Deuxièmement, les tables “article” et “categorie_article” ont une relation dite ‘many-to-many’ ce qui signifie qu’une table intermédiaire sera créée afin de recevoir les clés étrangères de ces deux tables.

Un autre schéma se manifeste notamment entre les tables “user” et “avis_user” et entre “user” et “message”. Dans ce cas-ci, c'est la table “message” ou la table “avis_user” qui reçoivent pas une, mais deux clés étrangères de la part de la table “user” reflétant ainsi la dualité ‘expéditeur / destinataire’ et ‘évaluateur / évalué’.

Enfin, dernier élément, quelque peu anecdotique, la table “tabou” qui n'a aucune relation!

La suite de la création passe par le MLD (Modèle Logique des Données), vous pourrez le voir à l'annexe n°8, que j'ai déjà commencé à expliciter précédemment. Je ne vais pas m'attarder sur ce point pour passer directement au script SQL que j'ai réalisé.

```
CREATE TABLE role (
    id_role          Int Auto_increment PRIMARY KEY,
    str_role         Varchar (50) NOT NULL
)ENGINE=InnoDB;

INSERT INTO role (str_role) VALUES ("USER");
INSERT INTO role (str_role) VALUES ("ADMIN");
INSERT INTO role (str_role) VALUES ("MODO");
```

Comme nous l'avons vu, avant de créer la table “user” centrale au projet JDRConnexion, il faut que les tables “rôle”, “expérience” et “profil_image” soient déjà codées

et incrémentées de données. Voici donc un exemple de code, ici la table “role”, répondant à cette demande.

Une fois cette étape faite, j'ai pu enfin coder la table “user”. Elle est intéressante car regroupe un certain nombre de spécificités. Déjà, les contraintes imposées assurent l'intégrité des données et la cohérence des relations entre les tables. Les NOT NULL garantissent que

certaines informations essentielles sont toujours présentes. Les contraintes UNIQUE et les clés étrangères renforcent la validité des données. Les valeurs par défaut, comme DEFAULT 0 ou CURRENT_TIMESTAMP, facilitent la gestion des utilisateurs en définissant un comportement attendu sans qu'il soit nécessaire de préciser certaines valeurs à la création d'un utilisateur.

```
CREATE TABLE user (
    id_user          Int Auto_increment PRIMARY KEY,
    str_email        Varchar (255) NOT NULL UNIQUE,
    str_nom          Varchar (255) NOT NULL,
    str_prenom       Varchar (255) NOT NULL,
    dtm_naissance   Datetime NOT NULL,
    bln_active      TINYINT(1) NOT NULL DEFAULT 0,
    str_mdp          Varchar (255) DEFAULT '',
    bln_notif       TINYINT(1) NOT NULL DEFAULT 0,
    str_pseudo       Varchar (255) NOT NULL UNIQUE,
    str_description Varchar (255) DEFAULT '',
    bln.mj           TINYINT(1) NOT NULL DEFAULT 0,
    id_experience   Int NOT NULL DEFAULT 1,
    id_role          Int NOT NULL DEFAULT 1,
    id_profil_image Int NOT NULL DEFAULT 1,
    dtm_creation    Datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
    dtm_maj          Datetime,
    CONSTRAINT id_experience_FK FOREIGN KEY (id_experience) REFERENCES experience(id_experience),
    CONSTRAINT id_role_FK FOREIGN KEY (id_role) REFERENCES role(id_role),
    CONSTRAINT id_profil_image_FK FOREIGN KEY (id_profil_image) REFERENCES profil_image(id_profil_image)
)ENGINE=InnoDB;
```

Pour revenir à la particularité principale des tables “message” et “avis_user” qu'est la double clés étrangères, voici comment j'ai procédé:

```
CREATE TABLE avis_user (
    id_avis_user     Int AUTO_INCREMENT PRIMARY KEY,
    id_observateur  Int NOT NULL,
    id_evalue        Int NOT NULL,
    bln_ame          TINYINT(1) NOT NULL DEFAULT 0,
    dtm_creation    DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    dtm_maj          Datetime,
    CONSTRAINT id_observateur_FK FOREIGN KEY (id_observateur) REFERENCES user(id_user),
    CONSTRAINT id_evalue_FK FOREIGN KEY (id_evalue) REFERENCES user(id_user)
)ENGINE=InnoDB;
```

Il y a bien les deux contraintes, mais elles se voient attribuées à chacune un nom différent et unique.

Toujours lors de la création du script SQL, j'ai essayé d'anticiper les cas de suppression ou de mise à jour des éléments d'une table et leurs répercussions sur les autres tables auxquelles elle serait liée. Ainsi, comme dans l'exemple suivant, L'option ON DELETE CASCADE signifie que si un enregistrement dans la table “user” est supprimé,

toutes les disponibilités associées à cet utilisateur seront automatiquement supprimées. L'option ON UPDATE CASCADE permet de propager toute modification de l'ID utilisateur dans la table "disponibilité".

```
CREATE TABLE disponibilite (
    id_disponibilite  INT AUTO_INCREMENT PRIMARY KEY,
    id_user           INT NOT NULL,
    str_jour          ENUM('lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi', 'dimanche') NOT NULL,
    time_debut        TIME NOT NULL,
    time_fin          TIME NOT NULL,
    CONSTRAINT id_user_disponibilite_FK FOREIGN KEY (id_user) REFERENCES user(id_user) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB;
```

Dans cet exemple, on remarque aussi le cas particulier du champ "str_jour". Ce champ utilise le type ENUM pour restreindre les valeurs possibles aux jours de la semaine : "lundi", "mardi", etc. Cela garantit que seuls ces jours peuvent être insérés dans la table, éliminant ainsi les erreurs de saisie.

VI - Développer des composants d'accès aux données

Pour illustrer cette partie, je vais poursuivre l'exemple pris précédemment en lien avec l'annexe n°7, à savoir l'inscription d'un utilisateur.

Ainsi, nous pourrons suivre l'intégralité des éléments d'un CRUD (Create, Read, Update, Delete ou créer, lire, mettre à jour, supprimer en français) autour d'un utilisateur.

Nous nous étions stopper après avoir récupéré les informations données par l'utilisateur, les avoir vérifiées et sécurisées.

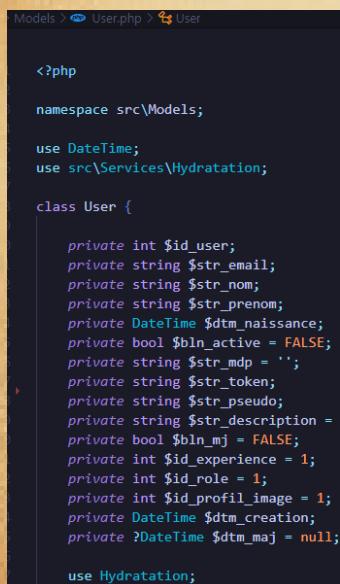
Maintenant, grâce à la fonction "createUser()" de mon UserRepository qui attend en entrée un objet User et qui donne en sortie un booléen :

L'utilisation d'un bloc try-catch, pour gérer les exceptions, permet d'intercepter les erreurs liées à la base de données et d'éviter que des détails internes ne soient exposés à l'utilisateur. Cela améliore la sécurité

```
public function createUser(User $user): bool {
    try {
        $sql = "INSERT INTO user (str_email, str_nom, str_prenom, dtm_naissance, str_pseudo)
                VALUES (:str_email, :str_nom, :str_prenom, :dtm_naissance, :str_pseudo)";
        $statement = $this->DB->prepare($sql);
        $retour = $statement->execute([
            ':str_email' => $user->getStrEmail(),
            ':str_nom' => $user->getStrNom(),
            ':str_prenom' => $user->getStrPrenom(),
            ':dtm_naissance' => $user->getDtmNaissance(),
            ':str_pseudo' => $user->getStrPseudo()
        ]);
        if ($retour) {
            return TRUE;
        }
        else {
            return FALSE;
        }
    }
    catch (PDOException $error) {
        throw new \Exception("Database error: " . $error->getMessage());
    }
}
```

en évitant la fuite d'informations sensibles. Le code utilise des requêtes préparées avec PDO, ce qui protège contre les attaques par injection SQL. En effet, les variables utilisateurs comme ‘:str_email’, ‘:str_nom’, etc. sont passées en tant que paramètres, ce qui permet à PDO de les échapper correctement avant de les exécuter.

J'ai évoqué l'objet User, c'est une classe PHP qui contient des propriétés et des méthodes pour gérer les informations utilisateur.



```
<?php

namespace src\Models;

use DateTime;
use src\Services\Hydratation;

class User {

    private int $id_user;
    private string $str_email;
    private string $str_nom;
    private string $str_prenom;
    private DateTime $dtm_naissance;
    private bool $bln_active = FALSE;
    private string $str_mdp = '';
    private string $str_token;
    private string $str_pseudo;
    private string $str_description =
    private bool $bln.mj = FALSE;
    private int $id_experience = 1;
    private int $id_role = 1;
    private int $id_profil_image = 1;
    private DateTime $dtm_creation;
    private ?DateTime $dtm_maj = null;

    use Hydratation;
}
```

Chaque propriété correspond à un attribut stocké dans la base de données ou calculé par l'application.

Une fois l'utilisateur ainsi créé, il est possible de le récupérer afin de l'afficher ou de travailler avec ses caractéristiques. Pour ce faire, je peux recourir à plusieurs de ses caractéristiques car elles sont définies de façon unique et peuvent donc permettre de retrouver l'utilisateur auquel elles sont liées. Dans le cas de l'usager, je peux notamment me servir de son adresse email, de son pseudonyme ou encore de son identifiant unique,

mais aussi de n'importe quelle combinaison d'au moins un de ces éléments. Grâce à ces fonctions de récupération, il est tout à fait possible de récupérer une ou plusieurs entrées de la base de données depuis une ou plusieurs tables. Je ne vais ici présenter que la récupération d'une entrée en base de données car je réserve le traitement des entrées multiples à l'illustration de mes composants métier. Vous pouvez voir sur cet exemple trois fonctions remplissant le même rôle, récupérer un utilisateur spécifique. Seule leur condition 'WHERE' varie. Même si les trois facteurs de tri dans ces fonctions sont des éléments uniques à chaque utilisateur et qu'on s'attend donc à ne retrouver qu'une seule entrée à ces recherches, l'utilisation de 'LIMIT 1' reste une bonne pratique pour optimiser la performance et la clarté de la requête, en évitant à La base de données recherche plus d'enregistrements que nécessaire. Dans l'exemple, on constate que \$user résulte du fetch(PDO::FETCH_ASSOC), c'est donc un tableau associatif si une entrée est trouvée ou 'FALSE' sinon. L'utilisation d'un objet User au lieu d'un tableau associatif permet de mieux structurer, sécuriser et maintenir mon code. Cela apporte des avantages en termes de contrôle des accès aux données, d'intégration avec le modèle objet, de flexibilité dans les évolutions futures, et de lisibilité du code.

```
/** * Récupérer un utilisateur spécifique * * @param int $id_user identifiant unique de l'utilisateur * * @return User Retourne l'utilisateur trouvé */ public function getThisUserById(int $id_user): ?User { ... }

/** * Récupérer un utilisateur spécifique * * @param string $str_email Email de l'utilisateur * * @return User Retourne l'utilisateur trouvé */ public function getThisUserByEmail(string $str_email): ?User { ... }

/** * Récupérer un utilisateur spécifique * * @param string $str_pseudo Pseudo unique de l'utilisateur * * @return User Retourne l'utilisateur trouvé */ public function getThisUserByPseudo (string $str_pseudo): ?User { try { $sql = "SELECT * FROM user WHERE str_pseudo = :str_pseudo LIMIT 1"; $statement = $this->DB->prepare($sql); $statement->execute([":str_pseudo" => $str_pseudo]); $user = $statement->fetch(PDO::FETCH_ASSOC); if($user) { return new User($user); } return null; } catch (PDOException $error) { throw new \Exception("Database error: " . $error->getMessage()); } }
```

Il est ensuite possible de se servir des données récupérées afin d'afficher une page de profil de l'utilisateur et ainsi lui donner la possibilité de modifier certains de ces éléments. C'est la partie 'mettre à jour' ou 'Update' du 'CRUD'.

Dans le projet JDRConnexion, l'utilisateur peut compléter son profil soit par des entrées dans des tables liées, soit par des champs supplémentaires dans la table user. Ceci n'est pas porté à sa connaissance, mais dépend du formulaire qu'il complète.

Illustrons ici l'exemple où il viendrait à compléter les champs de la table user. Le formulaire de données serait recueilli et sécurisé comme vu précédemment. Puis, grâce à la fonction updateThisUser() Qui prend en entrée un objet User et retourne un objet User, j'actualise les données de la BDD avec celles fournies par l'utilisateur.

```
public function updateThisUser(User $user): User {
    try {
        $sql = "UPDATE user SET
            str_email = :str_email,
            str_nom = :str_nom,
            str_prenom = :str_prenom,
            str_pseudo = :str_pseudo,
            bln_mj = :bln_mj,
            str_description = :str_description,
            id_experience = :id_experience,
            id_profil_image = :id_profil_image,
            dtm_maj = :dtm_maj";

        $params = [
            ":str_email" => $user->getEmail(),
            ":str_nom" => $user->getNom(),
            ":str_prenom" => $user->getPrenom(),
            ":str_pseudo" => $user->getPseudo(),
            ":bln_mj" => $user->isBlnMj(),
            ":str_description" => $user->getDescription(),
            ":id_experience" => $user->getIdExperience(),
            ":id_profil_image" => $user->getIdProfilImage(),
            ":dtm_maj" => $user->getDtmMaj(),
            ":id_user" => $user->getIdUser()
        ];

        if (!empty($user->getMdp())) {
            $sql .= ", str_mdp = :str_mdp";
            $params[":str_mdp"] = $user->getMdp();
        }

        $sql .= " WHERE id_user = :id_user";

        $statement = $this->DB->prepare($sql);
        $statement->execute($params);

        return $this->getThisUserById($user->getIdUser());
    }
    catch (PDOException $error) {
        throw new \Exception("Database error: " . $error->getMessage());
    }
}
```

Dans cet exemple, on utilise UPDATE pour mettre à jour la table 'user' avec les valeurs (SET) recueillies précédemment.

Le mot de passe est conditionnel dans cette situation car je ne pré-remplis pas le formulaire avec le mot de passe enregistré dans la BDD. Et donc si l'utilisateur ne veut pas le modifier, je ne dois pas l'inclure dans la mise à jour au risque de le corrompre. A noter que les mots de passe dans JDRConnexion sont gérés grâce aux fonctions natives de PHP : password_hash(\$mdp, PASSWORD_DEFAULT) et password_verify(\$mdp, \$resultat['str_mdp']).

J'utilise aussi un champ ‘dtm_maj’ qui enregistre la date et l'heure de la dernière modification pour cette entrée. C'est une bonne pratique pour garantir une bonne gestion des informations actualisées en base de données.

Enfin, un utilisateur sera aussi en mesure de supprimer son compte, c'est le ‘DELETE’ du ‘CRUD’.

Si cela venait à arriver, j'ai volontairement instancié à la création de la base de données un utilisateur fictif (avec l'ID = 2) qui me servira pour la gestion des contraintes liées aux clés étrangères. En effet, je ne souhaite pas que les messages et les avis que l'utilisateur aurait laissés aux autres usagers ou articles soient supprimés. Pour ces cas particuliers, lors de la suppression, je substitue le compte qui va être supprimé par ce compte fictif dans les différentes tables avant de procéder à la suppression de l'entrée dans la table user. Etant donné

que la fonction enchaîne plusieurs opérations, il est préférable d'utiliser une transaction. Ainsi, si une des étapes échoue, toutes les modifications seront annulées, évitant des incohérences dans la base de données.

C'est pourquoi j'utilise beginTransaction() pour la commencer, puis commit() pour la valider ou rollBack() pour revenir en arrière s'il y a eu une erreur.

```
public function deleteThisUser(int $id_user): bool {
    try {
        $this->DB->beginTransaction();

        $sql_message = "UPDATE message SET id_destinataire = 2 WHERE id_destinataire = :id_user;";
        $statement_message = $this->DB->prepare($sql_message);
        $statement_message->execute([":id_user" => $id_user]);

        $sql_message_2 = "UPDATE message SET id_expediteur = 2 WHERE id_expediteur = :id_user;";
        $statement_message_2 = $this->DB->prepare($sql_message_2);
        $statement_message_2->execute([":id_user" => $id_user]);

        $sql_avis = "UPDATE avis_user SET id_observateur = 2 WHERE id_observateur = :id_user;";
        $statement_avis = $this->DB->prepare($sql_avis);
        $statement_avis->execute([":id_user" => $id_user]);

        $sql_avis_2 = "UPDATE avis_user SET id_evalue = 2 WHERE id_evalue = :id_user;";
        $statement_avis_2 = $this->DB->prepare($sql_avis_2);
        $statement_avis_2->execute([":id_user" => $id_user]);

        $sql_avis_article = "UPDATE avis_article SET id_user = 2 WHERE id_user = :id_user;";
        $statement_avis_article = $this->DB->prepare($sql_avis_article);
        $statement_avis_article->execute([":id_user" => $id_user]);

        $sql = "DELETE FROM user WHERE id_user = :id_user;";
        $statement = $this->DB->prepare($sql);
        $statement->execute([":id_user" => $id_user]);

        $this->DB->commit();

        return true;
    } catch (PDOException $error) {
        $this->DB->rollBack();
        throw new \Exception("Database error: " . $error->getMessage());
    }
}
```

VII - Développer des composants metier

Je vais ici décrire la fonctionnalité principale du site JDRConnexion, à savoir sa fonctionnalité de tri lors de la recherche d'utilisateurs. Elle permet à un usager de trouver le partenaire idéal pour ses prochaines parties. Il peut ainsi affiner sa recherche parmi tous les profils enregistrés selon les jeux sur leur liste de souhaits, directement par un pseudonyme, et surtout par les disponibilités renseignées par les utilisateurs de JDRConnexion. Dans l'annexe n°9, on peut voir sur la page listant les profils que le formulaire permettant le tri renvoie sur sa propre page avec une méthode 'GET' ce qui me permet de récupérer les filtres demandés par l'utilisateur grâce aux différents :

```
isset($_GET[pseudo]) ? htmlspecialchars($_GET[pseudo]) : '';
```

Cette ligne signifie que, s'il existe dans l'URL une valeur : 'pseudo=' alors on la récupère et on l'attribut à une variable, sinon cette variable est vide.

Une fois toutes les variables reflétant le tri de l'utilisateur récupérées, je les passe en paramètre de la fonction getAllUser() qui me sert à récupérer les entrées de la table user ainsi que celles des entrées d'autres tables utiles à l'affichage et au tri des profils. Cette fonction est visible à l'annexe n°10.

Je vais maintenant analyser cette fonction. Premièrement, les données passées en paramètre de fonction sont typées, nommées correctement et renseignées dans un ordre défini. Elles sont aussi prédéfinies au cas où elles ne seraient pas un élément de filtrage donné par l'utilisateur afin d'éviter les erreurs.

Ensuite, un point ergonomique, cette fonction permet un système de pagination. C'est pourquoi elle commence par calculer son 'offset' avec la formule : (\$page - 1) * \$perPage

qui permet de sauter un certain nombre d'enregistrements en fonction de la page demandée.

La requête principale sélectionne tous les utilisateurs (user.*) et joint plusieurs tables pour obtenir des informations supplémentaires :

- profil_image : pour récupérer le chemin de l'image de profil de l'utilisateur.
- avis_user : pour compter le nombre d'avis laissés sur l'utilisateur et le nombre d'avis positifs.
- game_voulu : potentiellement pour récupérer les jeux auxquels l'utilisateur souhaite jouer.
- disponibilite : pour filtrer les utilisateurs par disponibilité.

Elle effectue des calculs importants pour fournir des statistiques sur les avis laissés pour chaque utilisateur afin par la suite d'ordonner les résultats du meilleur ratio "aimé / nombre d'avis" au plus mauvais.

La ligne : COALESCE(SUM(CASE WHEN avis_user.bln_aime = 1 THEN 1 ELSE 0 END),0) AS aime, calcule le nombre total d'avis positifs reçus par un utilisateur. SUM() permet de faire l'addition, CASE WHEN() est une expression conditionnelle qui vérifie si 'bln_aime' est égale à 1. Si le cas est vérifié, on ajoute 1 sinon on ajoute 0. COALESCE : Cette fonction SQL retourne la première valeur non nulle. Si la somme est nulle (par exemple, si l'utilisateur n'a pas reçu d'avis), le résultat sera remplacé par 0. 'AS aime' permet de récupérer ce résultat sous la colonne 'aime'.

La ligne : COUNT(avis_user.id_avis_user) AS total_avis, a pour objectif de calculer le nombre total d'avis reçus par un utilisateur, qu'ils soient positifs ou négatifs sous la colonne 'total_avis'.

Enfin la ligne : COALESCE(SUM(CASE WHEN avis_user.bln_aime = 1 THEN 1 ELSE 0 END) / NULLIF(COUNT(avis_user.id_avis_user), 0), 0) AS ratio, calcule le ratio d'avis positifs par rapport au nombre total d'avis pour chaque utilisateur. Pour ce faire, le numérateur est la somme des avis positifs (déjà expliquée dans le premier point),

le dénominateur est le nombre total d'avis (déjà expliqué dans le deuxième point), la fonction :

NULLIF(COUNT(avis_user.id_avis_user), 0) permet de renvoyer NULL si le nombre total d'avis est 0, ce qui permet d'éviter une division par zéro. Le fait de réaliser tout ceci dans la fonction COALESCE() permet si le résultat de la division est NULL (c'est-à-dire s'il n'y a aucun avis pour cet utilisateur), de renvoyer 0 à la place de NULL. Le résultat est enregistré sous la colonne 'ratio'.

La requête SQL comprend ensuite un système de conditions dynamiques : La clause WHERE 1=1 permet d'ajouter des filtres optionnels de manière dynamique :

- Filtre par jeu : Si \$id_game est défini, la fonction filtre les utilisateurs ayant joué à un jeu donné à travers une sous-requête.
- Filtre par pseudo : Si \$str_pseudo est renseigné, elle cherche les utilisateurs dont le pseudo correspond partiellement à la chaîne passée.
- Filtre par maître de jeu (MJ) : Si \$bln_mj est défini, la fonction filtre selon ce critère.
- Filtre par jour : Si \$str_jour est défini, elle filtre les utilisateurs disponibles un certain jour.
- Filtre par plage horaire : Les utilisateurs sont filtrés selon leur disponibilité en fonction des plages horaires \$time_debut et \$time_fin.

Ensuite, Les résultats sont groupés par utilisateur (GROUP BY user.id_user) pour éviter les doublons causés par les jointures.

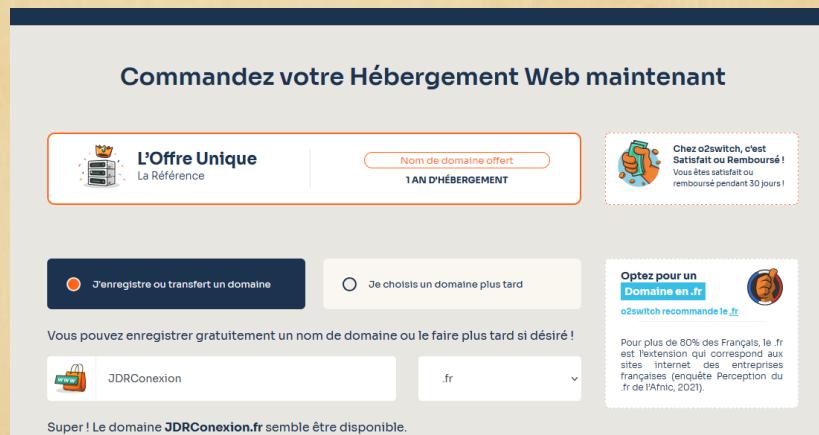
Les utilisateurs sont ensuite triés par leur ratio d'avis positifs en ordre décroissant (ORDER BY ratio DESC).

La requête retourne seulement un certain nombre d'enregistrements par page, défini par \$perPage, ici 10 entrées, avec l'offset calculé précédemment.

VIII – Documenter le déploiement d'une application dynamique web ou web mobile

Une fois toutes ces étapes réalisées, il me fallait mettre mon projet en ligne. Pour ce faire, je me suis rendu sur le site de o2switch.fr car il propose un excellent rapport qualité/prix et aussi car c'est un hébergeur français! Ceci limitera un peu l'impact écologique du stockage de mon site.

Parmi les offres qui me sont proposées, j'ai fait le choix de prendre l'offre unique. J'ai ensuite dû choisir le nom de domaine pour mon site, heureusement jdrconnexion.fr était disponible !



L'étape d'après étant de remplir ses informations personnelles et de paiement, je ne la documente pas.

On reçoit par email ses identifiants permettant de se connecter à son cPanel, c'est un panneau de contrôle utilisé par de nombreux hébergeurs web pour permettre aux utilisateurs de gérer facilement leur site internet et les ressources associées, sans avoir à interagir directement avec les fichiers ou à écrire du code complexe.

Depuis ce panneau, j'ai commencé par créer la base de données pour mon site. Je me suis donc rendu sur l'option 'Assistant de base de

données MySQL®’ de la rubrique ‘Base de données’.



Cet assistant m'a accompagné étape par étape pour la création de la base de données, annexe n°11:

Premièrement nommer celle-ci, deuxièmement créer un utilisateur ainsi que son mot de passe pour cette BDD, enfin définir les droits que cet utilisateur a sur la base de données.

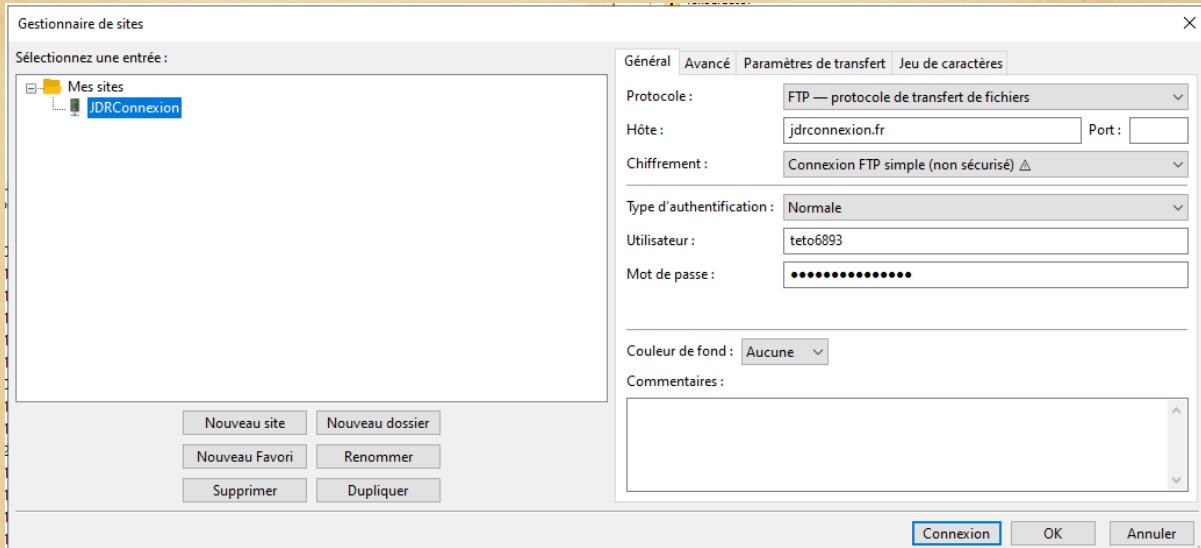
Toujours depuis le cPanel, dans la rubrique Domaine sous l'option ‘Domaines Configurés’ j'ai modifié la racine du document pour mon domaine vers, dans un premier temps, ‘jdrconnexion.fr’ ce qui a créer un dossier au même nom à sa racine.

Domaines supplémentaires	Racine du document	Redirige vers	Actions
jdrconnexion.fr	http://jdrconnexion.fr/app	not redirected	Supprimer Gérer la redirection

Dans un deuxième temps, je fais pointer la racine du document vers le dossier voulu.

Bien qu'il existe un système de gestion des fichiers depuis le cPanel de o2switch, j'ai fait le choix de recourir au logiciel FileZilla pour la gestion de transfert de mes fichiers. Une fois le logiciel installé dans son interface, je me rends sur Fichier/gestionnaire de sites et je

complète les infos demandées avec celles de mon compte o2switch comme ceci :



Cela me permet de me connecter au gestionnaire de fichiers de mon site JDRConnexion et ainsi transférer depuis la partie gauche de l'interface, mes fichiers locaux, vers la partie droite, les fichiers en ligne.

Nom de fichier	Taille de fic...	Type de fichier	Dernière modifcat...	Nom de fichier	Taille de fi...	Type de fic...	Dernière modif...	Droits d'ac...	Propriétaire...
public	Dossier de fichiers		21/09/2024 15:04:13	...	Dossier de ...		18/09/2024 15...	0755	1022 1025
src	Dossier de fichiers		21/09/2024 15:04:13	cgi-bin	Dossier de ...		21/09/2024 15...	0755	1022 1025
.htaccess	178 Fichier HTACCESS		24/08/2024 16:12:49	public	Dossier de ...		21/09/2024 15...	0755	1022 1025
config.php	409 Fichier source PHP		21/09/2024 15:05:00	src	Dossier de ...		21/09/2024 15...	0755	1022 1025
index.php	46 Fichier source PHP		24/08/2024 16:12:49	.htaccess	173 Fichier HT...		21/09/2024 15...	0644	1022 1025
README.md	6,287 Fichier source Mar...		24/08/2024 16:12:49	config.php	397 Fichier sou...		21/09/2024 15...	0644	1022 1025
				index.php	45 Fichier sou...		21/09/2024 15...	0644	1022 1025
				README.md	6,287 Fichier sou...		21/09/2024 15...	0644	1022 1025

Il ne faut pas oublier, même si c'est faisable après, de penser à modifier son fichier config.php contenant les informations nécessaires à la connexion à la BDD et aux cheminements à l'intérieur des fichiers du site.

Il faut aussi penser à modifier le fichier .htaccess qui permet de définir la redirection des URLs comme nous le voulons.

ANNEXES

Annexe n° 1:

The screenshot shows a Google search results page with a dark theme. The search query 'wamp' is entered in the search bar. The results are categorized under 'Tous' (All) and include the following entries:

- WampServer**
https://www.wampserver.com :
[WampServer, la plate-forme de développement Web sous Windows](#)
...
Installation · Création automatique d'un répertoire « www » lors de l'installation (typiquement c:\wamp\www). · Créez un sous répertoire pour votre projet et ...
- Download Wampserver 64bits**
WampServer is a Windows web development environment. It ...
- WAMP**
WampServer is a Windows web development environment. It ...
- Installation**
Installation · Création automatique d'un répertoire « www » lors de l ...
- Forum Wampserver**
Sujet, Envois, Commencé par, Dernier envoi. Note ...
- Télécharger les addons Php**
Installation · Création automatique d'un répertoire « www » lors de l ...
Autres résultats sur wampserver.com »
- Wampserver - Files and addons**
https://wampserver.aviaTechno.net · Traduire cette page :
[Wampserver - Files and addons](#)
WampServer is a Windows-based Web development platform, without Internet access, for dynamic Web applications using the Apache 2.4 server, PHP scripting ...
- Wikipédia**
https://fr.wikipedia.org › wiki › WAMP :
[Wikipédia](#)

Annexe n° 2:



Annexe n° 3:

Bienvenue sur [Nom du site] - Le Portail des Aventuriers

Vous êtes un passionné de jeu de rôle, qui a pour la recherche de nouvelles aventures et de compagnons pour partager des parties épique? [Nom du site] est là pour vous! Nous avons développé un système de recherche et de connexion pour trouver des partenaires de jeu qui partagent vos envies, vos horaires, et votre passion pour l'aventure du jeu de rôle. Que vous soyez un maître de jeu expérimenté ou un débutant dans votre première campagne, nous trouvons le jeu communautaire accueillant et prêt à explorer ensemble les contenus les plus fantastiques.

Grâce à [Nom du site], vous pouvez facilement entrer en contact avec des joueurs partageant vos goûts et votre disponibilité.

Que vous préfériez les aventures médiévales, fantastiques, les explorations spatiales, ou les intrigues mystérieuses, notre système de recherche vous aidera à trouver des partenaires de jeu en quelques clics. Plongez dans l'univers du jeu de rôle, le groupe parfait est à portée de main.

D&D **Sorcery**

PARK HERESY

En plus de vous connecter avec d'autres joueurs, [Nom du site] nous proposons également une sélection d'articles et d'outils pour l'aventure de jeu de rôle. Nos conseils pour les maîtres de jeu, des astuces pour les joueurs, des analyses des meilleures systèmes de jeu, et des tutoriels pour apprendre de nouvelles compétences et échanger sur entre-lignes. Que vous cherchiez à perfectionner vos techniques ou à découvrir de nouvelles idées, notre contenu est là pour nourrir votre imagination.

L'Art du Maître de Jeu : Conseils et Astuces pour Devenir un MJ Inoubliable

Les Règles d'Or pour Créer et Gérer un Groupe de Jeu de Rôle

Les Règles d'Or pour Order et Gérer un Groupe de Jeu de Rôle

En savoir plus

PSEUDO MJ POSITIVE NÉGATIVE

PSEUDO	M.J.	POSITIVE	NÉGATIVE	JEU(S) SOUHAITÉ(S)
Killer	Non	125	15	Plusieurs
Killer	Non	125	15	DnD 5,5e
Killer	Non	124	19	Non renseigné

En voir plus

Avenir:

Conditions Générale d'utilisation

Fait sous licence, © Nom du site par ROCHET L'Agence

Bienvenue sur [Nom du site] - Le Portail des Aventuriers

Le site de référence pour trouver tes prochaines partenaires de jeu!

BIENVENUE SUR [Nom du site] - LE PORTAIL DES AVENTURIERS

Vous êtes un passionné de jeu de rôle, qui a pour la recherche de nouvelles aventures et de compagnons pour partager des parties épique? [Nom du site] est là pour vous! Nous avons développé un système de recherche et de connexion pour trouver des partenaires de jeu qui partagent vos envies, vos horaires, et votre passion pour l'aventure du jeu de rôle. Que vous soyez un maître de jeu expérimenté ou un débutant dans votre première campagne, nous trouvons le jeu communautaire accueillant et prêt à explorer ensemble les contenus les plus fantastiques.

Grâce à [Nom du site], vous pouvez facilement entrer en contact avec des joueurs partageant vos goûts et votre disponibilité.

Que vous préfériez les aventures médiévales, fantastiques, les explorations spatiales, ou les intrigues mystérieuses, notre système de recherche vous aidera à trouver des partenaires de jeu en quelques clics. Plongez dans l'univers du jeu de rôle, le groupe parfait est à portée de main.

D&D **PARK HERESY** **Sorcery**

En plus de vous connecter avec d'autres joueurs, [Nom du site] nous proposons également une sélection d'articles dédiés à l'aventure de jeu de rôle. Nos conseils pour les maîtres de jeu, des astuces pour les joueurs, des analyses des meilleures systèmes de jeu, et des tutoriels pour apprendre de nouvelles compétences et échanger sur entre-lignes. Que vous cherchiez à perfectionner vos techniques ou à découvrir de nouvelles idées, notre contenu est là pour nourrir votre imagination.

DERNIERS ARTICLES SORTIS :

L'Art du Maître de Jeu : Conseils et Astuces pour Devenir un MJ Inoubliable

Les Règles d'Or pour Créer et Gérer un Groupe de Jeu de Rôle

Les Règles d'Or pour Order et Gérer un Groupe de Jeu de Rôle

En savoir plus

UTILISATEURS MIS EN AVANT :

PSEUDO	M.J.	POSITIVE	NÉGATIVE	JEU(S) SOUHAITÉ(S)
Killer	Non	125	15	Plusieurs
Killer	Non	125	19	DnD 5,5e
Killer	Non	124	19	Non renseigné

En voir plus

Avenir:

Conditions Générale d'utilisation

Tout droit réservé, © Nom du site par ROCHET L'Agence

Annexe n° 4:

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JDRConnexion - trouve tes partenaires de JDR</title>
    <link href="https://fonts.googleapis.com/css2?family=Merriweather+Sans&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Cinzel&display=swap" rel="stylesheet">
    <link rel="icon" type="image/png" href=<?= HOME_URL ?>img/logo_site.png" />
    <link rel="stylesheet" href=<?= HOME_URL ?>css/commun.css">...
    <link rel="stylesheet" href=<?= HOME_URL ?>css/mobile.css" media="only screen and (max-width: 768px)">
    <link rel="stylesheet" href=<?= HOME_URL ?>css/pc.css" media="only screen and (min-width: 769px)">
</head>
```

Annexe n° 5:

```
<table class="accueil_tableau">
    <thead>
        <tr>
            <th>M.J.</th>
            <th>j'aime</th>
            <th>non aimé</th>
            <th>jeux souhaités</th>
        </tr>
    </thead>
    <tbody>
        <?php
        if(!empty($liste_user)){
            foreach($liste_user as $utilisateur) { ?>
                <tr>
                    <td><a href=<?= HOME_URL ?>profil/<?= $utilisateur['str_pseudo'] ?>"><img src=<?= HOME_URL ?><?= $utilisateur['str_chemin'] ?>" alt="miniature de l'image de profile" class="accueil_minieture_profil"></a></td>
                    <td><a href=<?= HOME_URL ?>profil/<?= $utilisateur['str_pseudo'] ?>"><?= $utilisateur['str_pseudo'] ?></a></td>
                    <td>
                        <?php if($utilisateur['bln_mj'] == 1) { ?>
                            <img src=<?= HOME_URL ?>img/icons/valider_icon.png" alt="Validé" class="accueil_icon_mj">
                        <?php } else { ?>
                            <img src=<?= HOME_URL ?>img/icons/croix_icon.png" alt="Non validé" class="accueil_icon_mj">
                        <?php } ?>
                    </td>
                    <td><?= $utilisateur['aime'] ?></td>
                    <td><?= ($utilisateur['total_avis'] - $utilisateur['aime'])?></td>
                    <td>
                        <?php
                            $tab_game = $GameRepository->getAllGameVoulu($utilisateur['id_user']);
                            $int_game_voulu = count($tab_game);
                            if($int_game_voulu === 0) {
                                echo "Non renseigné.";
                            }
                            elseif($int_game_voulu >= 2) {
                                echo "Plusieurs jeux souhaités.";
                            }
                            else {
                                echo $tab_game[0]['str_nom'];
                            }
                        <?php
                    </td>
                </tr>
            <?php
        }.
    </tbody>
</table>
```

Annexe n° 6:

```
<?php

namespace src\Services;

use stdClass;

trait Securite {
    public static function sanitize(array|stdClass $data): array {
        if ($data instanceof stdClass) {
            $data = (array) $data;
        }

        $dataSanitized = [];

        foreach ($data as $key => $value) {
            if (is_array($value)) {
                $dataSanitized[$key] = self::sanitize($value);
            }
            elseif (is_int($value)) {
                $dataSanitized[$key] = intval($value);
            }
            else {
                $dataSanitized[$key] = htmlspecialchars($value);
            }
        }
        return $dataSanitized;
    }
}
```

```
<?php

namespace src\Services;

use src\Models\Database;
use src\Repositories\TabouRepository;

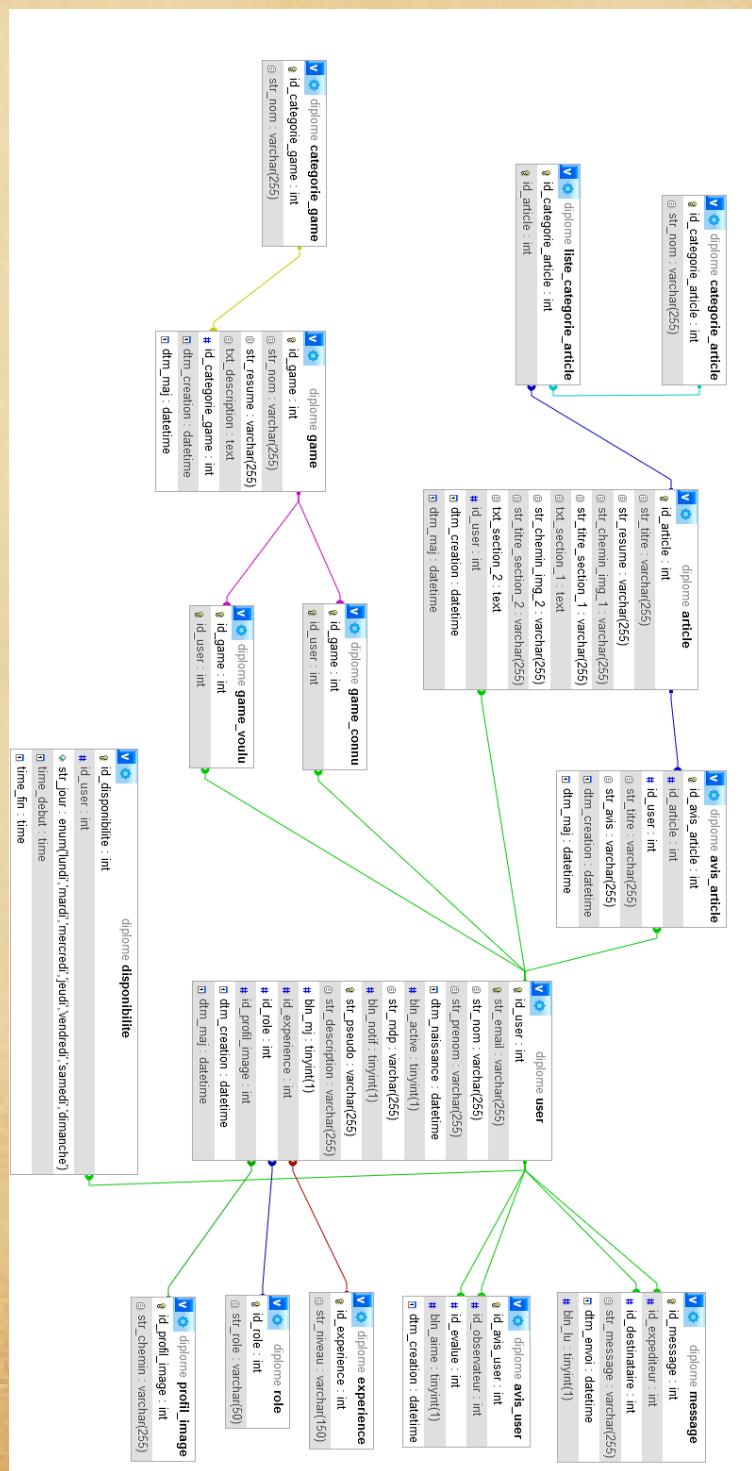
trait Censure {
    public function filtrer(string $texte):string {
        $database = new Database();
        $TabouRepository = TabouRepository::getInstance($database);
        $tab_tabou = $TabouRepository->getAllTabou();
        foreach($tab_tabou as $tabou) {
            $texte = str_ireplace($tabou['str_mot'], "****", $texte);
        }
        return $texte;
    }
}
```

Annexe n° 7:

```
public function inscription() {
    if (!isset($data['str_email']) || $data['str_email'] == '') {
        $_SESSION["erreur"] = "Le champs 'adresse email' est obligatoire et n'a pas été remplis.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if(filter_var($data['str_email'], FILTER_VALIDATE_EMAIL) == FALSE) {
        $_SESSION["erreur"] = "Le champs 'adresse email' doit être au format mail.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if($userRepository->getThisUserByEmail($data['str_email'])) {
        $_SESSION["erreur"] = "Un utilisateur avec cette adresse email existe déjà.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if(!isset($data['str_nom']) || $data['str_nom'] == '') {
        $_SESSION["erreur"] = "Le champs 'nom' est obligatoire et n'a pas été remplis.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if(!isset($data['str_prenom']) || $data['str_prenom'] == '') {
        $_SESSION["erreur"] = "Le champs 'prénom' est obligatoire et n'a pas été remplis.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if(!isset($data['str_pseudo']) || $data['str_pseudo'] == '') {
        $_SESSION["erreur"] = "Le champs 'pseudo' est obligatoire et n'a pas été remplis.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    if($userRepository->getThisUserByPseudo($data['str_pseudo'])) {
        $_SESSION["erreur"] = "Ce pseudonyme est déjà utilisé.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    $data['str_pseudo'] = $this->filtrer($data['str_pseudo']);

    if (!isset($data['dtm_naissance']) || $data['dtm_naissance'] == '') {
        $_SESSION["erreur"] = "Le champ 'date de naissance' est obligatoire et n'a pas été rempli.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    $naissances = explode('-', $data['dtm_naissance']);
    if (count($naissances) != 3 || !ctype_digit($naissances[0]) || !ctype_digit($naissances[1]) || !ctype_digit($naissances[2])) {
        $_SESSION["erreur"] = "Le format de la date de naissance est incorrect. Veuillez utiliser JJ-MM-AAAA.";
        $this->render("inscription", ["erreur" => $_SESSION["erreur"]]);
        return;
    }
    try {
        $naissance = new DateTime($data['dtm_naissance']);
        $maintenant = new DateTime();
        if ($naissance >= $maintenant) {
            $_SESSION["erreur"] = "La date de naissance ne peut pas être dans le futur.";
        }
    }
}
```

Annexe n° 8:



Annexe n° 9:

```
$id_game = isset($_GET['id_game']) ? intval($_GET['id_game']) : null;
$str_pseudo = isset($_GET['str_pseudo']) ? htmlspecialchars($_GET['str_pseudo']) : '';
$bln_mj = isset($_GET['bln_mj']) ? intval($_GET['bln_mj']) : null;
$str_jour = isset($_GET['str_jour']) ? htmlspecialchars($_GET['str_jour']) : '';
$time_debut = isset($_GET['time_debut']) ? htmlspecialchars($_GET['time_debut']) : '';
$time_fin = isset($_GET['time_fin']) ? htmlspecialchars($_GET['time_fin']) : '';

$total_utilisateur = $UserRepository->countAllUser($id_game, $str_pseudo, $bln_mj, $str_jour, $time_debut, $time_fin);
$parPage = 10;
$total_pages = ceil($total_utilisateur/$parPage);

$page = isset($_GET['page']) ? intval($_GET['page']) : 1;

if($page < 1 || $page > $total_pages) {
    $page = 1;
}

$liste_user = $UserRepository->getAllUser($id_game, $str_pseudo, $bln_mj, $str_jour, $time_debut, $time_fin, $page, $parPage);
$liste_game = $GameRepository->getAllGame();

?>
<head>
    <meta name="description" content="Trouvez des joueurs et maîtres de jeu sur JDRConnexion. Explorez la liste des utilisateurs."/>
</head>
<h2 class="accueil_titre_section">Les utilisateurs :</h2>

<div class="form_bg user_liste_filter">
    <div class="form_post_texte">
        <h3 class="connexion_titre">Filtre de tri</h3>
        <?php
        if($erreur !== '') { ?>
            <p class="erreur_texte"> <?= $erreur ?> </p>
        <?php } .
        if($succes !== '') { ?>
            <p class="succes_texte"> <?= $succes ?> </p>
        <?php } ?>
        <form class="connexion_form" action="<?=HOME_URL?>userliste" method="GET">
```

Annexe n° 10:

```
Récupère tous les utilisateurs en fonction du tri, paginer et classer selon leur ratio de like

@param int $id_game identifiant du jeu souhaité
@param string $str_pseudo pseudonyme du joueur rechercher
@param int $bln_mj booléen de Maître du Jeu
@param string $str_jour jour disponible
@param string $time_debut heure de début
@param string $time_fin heure de fin
@param int $page numéro de la page souhaitée
@param int $perPage nombre de résultat par page

@return array Un tableau contenant la liste des utilisateurs trouvés

lic function getAllUser(int $id_game = null, string $str_pseudo = '', int $bln_mj = null, string $str_jour = '', string $time_debut = '', string $time_fin = '') {
    try {
        $offset = ($page - 1) * $perPage;
        $sql = "SELECT
            user.*,
            profil_image.str_chemin,
            COALESCE(SUM(CASE WHEN avis_user.bln_aime = 1 THEN 1 ELSE 0 END), 0) AS aime,
            COUNT(avis_user.id_avis_user) AS total_avis,
            COALESCE(SUM(CASE WHEN avis_user.bln_aime = 1 THEN 1 ELSE 0 END) / NULLIF(COUNT(avis_user.id_avis_user), 0), 0) AS ratio
        FROM user
        LEFT JOIN avis_user ON user.id_user = avis_user.id_evalue
        LEFT JOIN profil_image ON user.id_profil_image = profil_image.id_profil_image
        LEFT JOIN game_voulu ON game_voulu.id_user = user.id_user
        LEFT JOIN disponibilite ON user.id_user = disponibilite.id_user
        WHERE 1=1";
        $params = [];
        if($id_game !== null) {
            $sql .= " AND user.id_user IN (SELECT id_user FROM game_connexion WHERE id_game = :id_game)";
            $params[':id_game'] = $id_game;
        }
        if(!empty($str_pseudo)) {
            $sql .= " AND str_pseudo LIKE :str_pseudo";
            $params[':str_pseudo'] = '%' . $str_pseudo . '%';
        }
        if($bln_mj !== null) {
            $sql .= " AND bln_mj = :bln_mj";
            $params[':bln_mj'] = $bln_mj;
        }
        if(!empty($str_jour)) {
            $sql .= " AND disponibilite.str_jour LIKE :str_jour";
            $params[':str_jour'] = '%' . $str_jour . '%';
        }
        if(!empty($time_debut)) {
            $sql .= " AND time_fin > :time_debut";
            $params[':time_debut'] = $time_debut;
        }
        if(!empty($time_fin)) {
            $sql .= " AND time_debut < :time_fin";
            $params[':time_fin'] = $time_fin;
        }
        $sql .= " GROUP BY user.id_user ORDER BY ratio DESC LIMIT $perPage OFFSET $offset";
        $statement = $this->db->prepare($sql);
        $statement->execute($params);
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $error) {
        throw new \Exception("Database error: " . $error->getMessage());
    }
}
```

Annexe n° 11:

Assistant de base de données MySQL®

Les bases de données MySQL vous permettent de stocker de grandes quantités d'information les utilisateurs. De nombreuses applications Web, notamment des forums et des systèmes de base de données, vous devez la créer. Seuls les utilisateurs MySQL (différents des utilisateurs cette base ou y écrire des données).

Étape 1 : Créer une base de données

Nouvelle base de données :

teto6893_ JDRC connexion

Remarque : 54 caractères max.

[Étape suivante](#)

Assistant de base de données MySQL®

✓ Vous avez créé une base de données MariaDB/MySQL nommée « teto6893_JDRC connexion ».

Étape 2 : Créer des utilisateurs de base de données :

Nom d'utilisateur :

teto6893_ JDRC connexion

Remarque : 58 caractères max.

Mot de passe :

Confirmation du mot de passe :

Niveau de sécurité

Très élevé (100/100)

Générer

[Créer un utilisateur](#)

Étape 3 : Ajouter un utilisateur à la base de données

Utilisateur: teto6893_JDRC connexion
Base de données: teto6893_JDRC connexion

TOUS LES PRIVILÉGES

ALTER

ALTER ROUTINE

CREATE

CREATE ROUTINE

CREATE TEMPORARY TABLES

CREATE VIEW

DELETE

DROP

EVENT

EXECUTE

INDEX

INSERT

LOCK TABLES

REFERENCES

SELECT

SHOW VIEW

TRIGGER

UPDATE

[Apporter des modifications](#) [Réinitialiser](#)

[Étape suivante](#)