

NOM	DE CARVALHO TEIXEIRA
Prénom	Manuel
Date de naissance	21/08/1990

TP – Développeur Web et Web Mobile

Lien du git :

<https://github.com/ManuDC7/ECF-ZOO-ARCADIA>

Lien du déploiement :

<https://preprod-arcadia.lamphie.moe/>

Lien de l'outil de gestion de projet :

<https://trello.com/invite/b/RO02reLs/ATTI01eeb726de02850dc083d3c1ae69460b2A62457E/ecf-arcadia>

Requêtes SQL à la main :

Présent dans [src/requete.sql](#)

Login et mot de passe administrateur :

eMail: josearcadia1@hotmail.fr

Mot de passe : [arcadia](#)

Partie 1 : Analyse des besoins

Résumé du projet

José est directeur du zoo Arcadia, situé près de la foret de Brocéliande, en Bretagne depuis les années 1960. Le zoo se porte très bien, le personnel et les animaux sont heureux et José souhaite moderniser son établissement en développant une application web. C'est par le biais de son assistante que je suis contacté.

Il souhaite que l'application respire l'écologie et que chaque visiteur ressente les valeurs du zoo dont il est fier. Il nous explique ensuite comment il entrevoit son application: une page d'accueil de présentation qui mentionne également les avis laissés par les visiteurs, un menu pour accéder aux différentes pages, la possibilité pour le personnel de se connecter. Le visiteurs doit également être en mesure de voir les services proposés par le zoo, les habitats reconstitués ainsi que les animaux présent dans chaque habitat. Un formulaire de contact doit également lui être mis à disposition.

La solution technologique idéale serait une application web responsive, développée avec des technologies telles que HTML5, CSS3, JavaScript et PHP pour assurer une expérience utilisateur fluide et intuitive. Pour la gestion des données, une base de données SQLite et une base de donnée MongoDB sont adaptées.

En travaillant 8 heures par jour, ce projet devrait prendre environ 30 jours. Je vais commencer par définir clairement les besoins, puis concevoir l'architecture de l'application avant de passer au développement. Enfin, des phases de tests et de corrections seront nécessaires pour garantir la qualité du produit final.

Cahier des charges

Une application web sera développée pour le zoo Arcadia, permettant aux visiteurs de découvrir les valeurs écologiques du directeur via la page d'accueil. Ils auront également accès aux avis post-visite des clients, aux services, et à la liste des animaux présents dans chaque habitat. Les visiteurs auront la possibilité de contacter le zoo via un formulaire de contact, qui transmettra directement leurs demandes aux employés.

Le directeur aura un accès complet au site une fois connecté. Il pourra créer, supprimer et modifier des utilisateurs (vétérinaires ou employés), des animaux, des services, des habitats et les horaires d'ouverture du zoo. Il pourra consulter les comptes rendus des vétérinaires, avec des filtres pour trier et filtrer les comptes rendus par animal ou date. Un dashboard lui montrera le nombre de consultations par animal.

Les employés pourront approuver les avis des visiteurs pour qu'ils apparaissent sur la page d'accueil de l'application web et modifier les services proposés par le zoo. Ils auront également la possibilité d'ajouter des consommations de nourriture pour les animaux à partir de leur espace, en spécifiant la date, l'heure, la nourriture et la quantité.

Les vétérinaires auront un accès administratif limité. Ils pourront rédiger des comptes rendus sur les habitats et saisir des informations sur les animaux, telles que leur état, la nourriture proposée, le grammage et la date de leur intervention.

Partie 2 : Spécifications technique

Technologies

Tenant compte de ma formation en cours et souhaitant exploiter mes compétences actuelles, j'ai délibérément choisi de ne pas avoir recours à des frameworks ni à des préprocesseurs. À la place, j'ai opté pour l'utilisation de technologies que je maîtrise déjà, notamment HTML pour la structure, CSS pour le style visuel, JavaScript pour l'interactivité, et PHP pour la gestion de la base de données. jQuery assurera la liaison entre la base de données et l'interface utilisateur principalement grâce au protocole AJAX.

Pour ce qui est des bases de données, j'ai opté pour SQLite, une base de données SQL légère, ainsi que MongoDB, une base de données NoSQL. Cette décision découle de mon désir d'explorer et de me familiariser avec différents types de bases de données. De plus, je suis déjà familier avec MySQL, ce qui m'aide à appréhender rapidement les concepts fondamentaux des bases de données relationnelles.

J'ai choisi ces solutions en fonction des besoins spécifiques du projet, cherchant ainsi à acquérir une expérience variée dans ce domaine.

Environnement de travail

Pour commencer le développement, sur mon environnement MacOS, j'ai créé un nouveau dossier à l'intérieur de mon répertoire GitHub en utilisant la commande mkdir, que j'ai nommé "ECF-ZOO-ARCADIA". J'ai ensuite initialisé le projet en lançant la commande git init et créé immédiatement un fichier .gitignore pour exclure les fichiers temporaires indésirables générés par MacOS, tels que .DS_Store. Cette approche me permettra de développer le projet de manière efficace en utilisant des technologies familières et en organisant proprement le code source.

Par la suite, j'ai utilisé Homebrew pour installer tous les outils nécessaires au développement, notamment SQLite, MongoDB, PHP, PDO (pour interagir avec la base de données de manière sécurisée) et Composer (pour manipuler les données de manière efficace et flexible avec MongoDB). Cette démarche garantit la disponibilité de tous les composants indispensables au développement de l'application web, tout en simplifiant le processus d'installation et de gestion des dépendances.

En parallèle, j'ai configuré un serveur Apache pour tester l'application localement, privilégiant XAMPP pour sa facilité d'utilisation et sa compatibilité avec les environnements de développement. Après l'installation, j'ai pris soin de configurer le fichier php.ini de XAMPP pour activer les extensions requises, assurant ainsi le bon fonctionnement de l'environnement de développement.

Sécurité

En plus de l'aspect fonctionnel, la sécurité SQL a été au cœur de ma démarche lors de la création des tables et des collections. Chaque commande SQL a été minutieusement examinée pour garantir le respect des principes de sécurité établis. Par exemple, lors de la définition des champs, j'ai veillé à utiliser des types de données appropriés pour éviter les risques de débordement ou de troncature. De plus, j'ai défini des contraintes d'intégrité pour assurer la cohérence des données et prévenir les vulnérabilités potentielles.

Après avoir établi la structure de ma base de données, j'ai entrepris l'implémentation de fonctionnalités PHP sécurisées visant à relier cette base de données à mon application web. Cette étape revêt une importance capitale dans le processus de développement, car elle garantit la sécurité et l'intégrité des données manipulées par l'application.

Pour assurer la sécurité des données, j'ai mis en place plusieurs mesures de protection. Tout d'abord, j'ai intégré des mécanismes de validation des données d'entrée. Cela m'a permis de filtrer et de valider les données fournies par les utilisateurs, afin d'éviter les attaques d'injection SQL. En validant les données dès leur saisie, j'ai pu empêcher l'exécution de requêtes malveillantes qui pourraient compromettre la sécurité de ma base de données.

Parallèlement, j'ai opté pour l'utilisation systématique de requêtes préparées. Cette technique de programmation sécurisée consiste à séparer les données des instructions SQL, ce qui permet d'éliminer le risque d'injection SQL. En préparant les requêtes SQL à l'avance et en liant les paramètres de manière sécurisée, j'ai pu garantir que seules les données légitimes seraient utilisées dans mes requêtes, renforçant ainsi la sécurité de mon application contre les attaques potentielles.

En outre, j'ai mis en place des mécanismes de chiffrement pour les données sensibles stockées dans ma base de données. En chiffrant ces données, j'ai assuré leur confidentialité et leur protection contre tout accès non autorisé. Cette couche supplémentaire de sécurité garantit que même en cas de violation de la base de données, les données sensibles restent inaccessibles et illisibles pour les tiers.



Veille technologique

TEXTE_A_ENTRER_ICI

Partie 3 : Recherche

Recherche à partir de site anglophone

Le processus de création de collection implique l'intégration d'une sécurité dans la base de données MongoDB en ajoutant un utilisateur avec un mot de passe. Cependant, après avoir effectué cette opération, je rencontre une erreur lors du démarrage du service MongoDB, identifiée comme l'erreur 3584. Cette erreur bloque ma progression dans le travail.

En parcourant différents forums de discussion, j'ai découvert une solution sur un forum coréen (Source : <https://kimbarbie.tistory.com/49>) qui redirige vers un fil de discussion sur Stack Overflow (Source : <https://stackoverflow.com/questions/63562177/mongod-aborts-on-mac>). Ce dernier explique que macOS génère des fichiers temporaires qui perturbent le fonctionnement de MongoDB. Ces fichiers temporaires peuvent causer des conflits ou des problèmes de permissions avec les fichiers nécessaires au fonctionnement de MongoDB, tels que le fichier de socket.

Pour résoudre ce problème, je dois exécuter la commande sudo rm -rf /tmp/mongodb-27017.sock pour supprimer le fichier de socket MongoDB temporaire.

Ensuite, je redémarre le service MongoDB en utilisant la commande sudo brew services restart mongod. Cette séquence d'opérations me permet de réinitialiser correctement les fichiers temporaires et de redémarrer le service MongoDB avec succès, me permettant ainsi de reprendre le travail sans rencontrer l'erreur 3584.

Extrait du site anglophone

2 Answers

Sorted by: Highest score (default) 

Please use one of the following solutions.

18**for Linux**

```
$ sudo rm -rf /tmp/mongodb-27017.sock  
$ sudo service mongod start
```

**for Mac**

```
$ sudo rm -rf /tmp/mongodb-27017.sock  
$ sudo brew services restart mongod
```

Partie 4 : Informations complémentaire

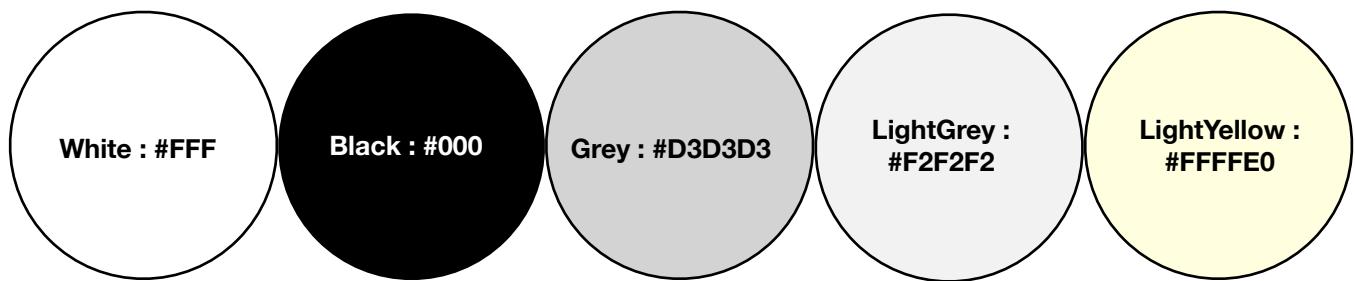
Font :

Source Sans Pro Font Family

 By Adobe



Colors :



Wireframes de l'application web

Desktop :

The screenshot shows a desktop application interface for creating wireframes. At the top, there's a toolbar with various icons for file operations. Below it, the title bar reads "Drafts / Wireframe desktop ECF". The main area features a grid of seven wireframe prototypes. To the left, a sidebar lists components under categories such as "Wireframe - animals" and "Wireframe - housing_Next". On the right, a panel provides options for "Design" and "Prototype", including color selection (F5F5F5), zoom (100%), and export functions.

Mobile :

The screenshot shows a mobile application interface for creating wireframes. The layout is similar to the desktop version, with a toolbar at the top, a title bar reading "Drafts / Wireframe mobile ECF", and a central grid of wireframe prototypes. The sidebar on the left contains a more detailed list of components and sections. The right-hand panel offers design and prototype tools, including a color palette set to F5F5F5 and a zoom level of 15%.

Mockup de l'application web

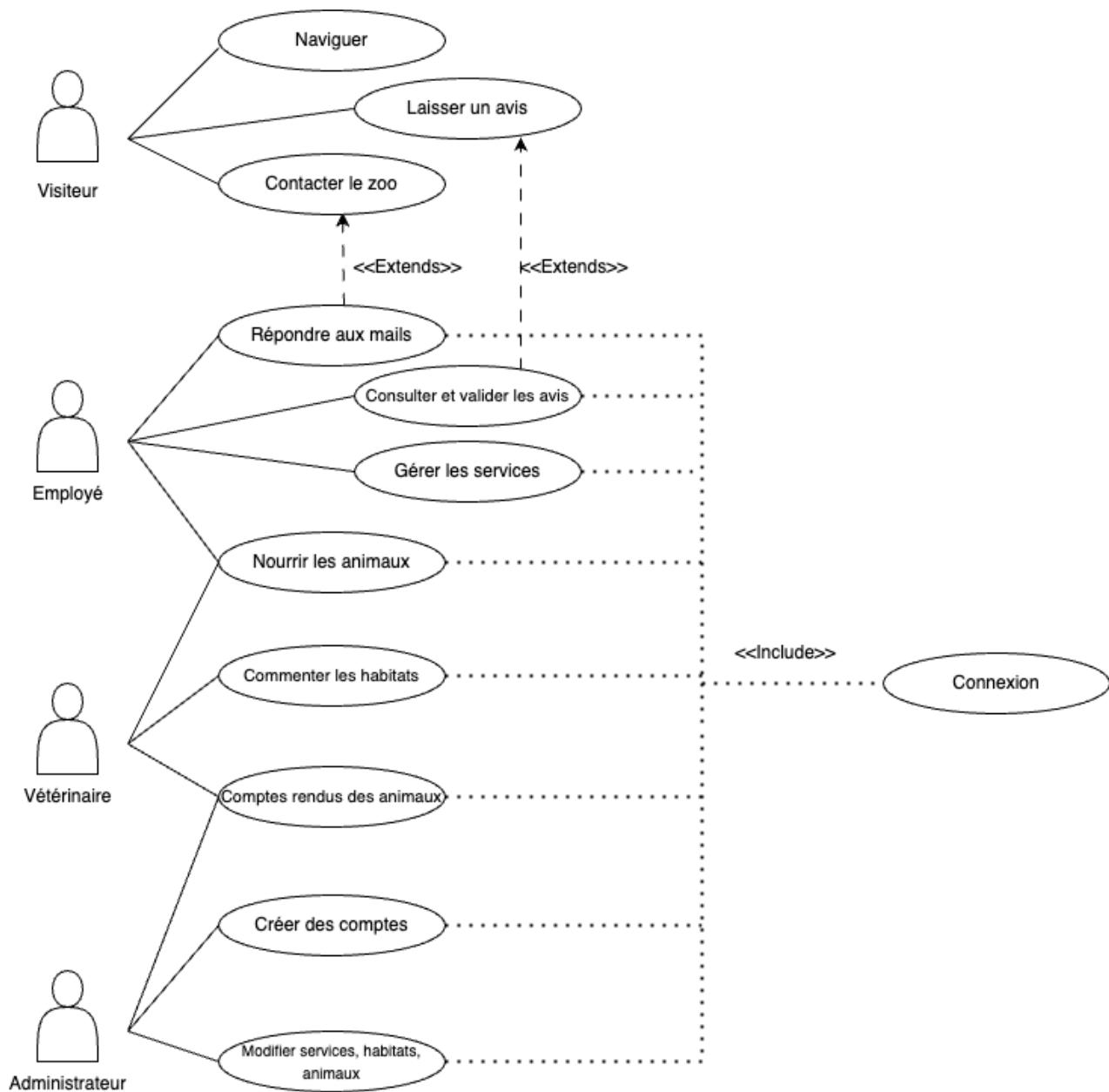
Desktop :

The desktop application interface is shown in a wireframe tool. The top navigation bar includes icons for file operations (New, Open, Save, Print, Find, Copy, Paste, Undo, Redo), a search bar, and tabs for 'Drafts / Mockup desktop ECF'. On the right, there's a toolbar with 'Share', 'Copy', 'Print', and a zoom level of 5%. The left sidebar lists components like 'adminPanel_delete', 'adminPanel_add', 'adminPanel', 'Contact', 'Login_role', 'Login', 'Animals', 'Housing_Next', 'Housing', 'Services_02', 'Services_01', and 'Index'. The main workspace displays a grid of cards representing different pages: 'Login_role', 'Login', 'Services 01', 'Services 02', 'adminPanel', 'adminPan...', and 'adminPan...'. Each card shows a preview of the page content, which includes various animal images and service-related text.

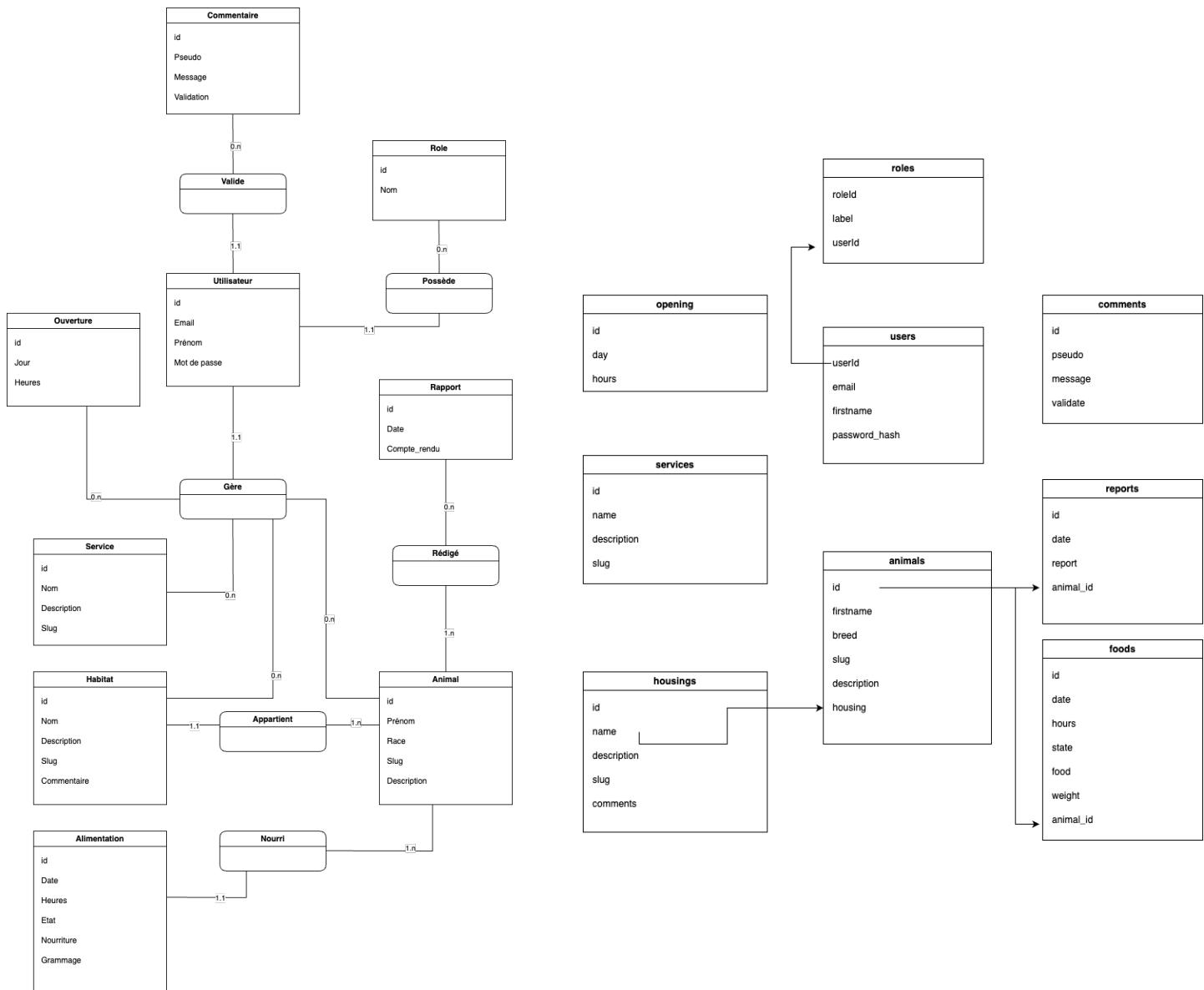
Mobile :

The mobile application interface is shown in a wireframe tool. The top navigation bar includes icons for file operations (New, Open, Save, Print, Find, Copy, Paste, Undo, Redo), a search bar, and tabs for 'Drafts / Mockup mobile ECF'. On the right, there's a toolbar with 'Share', 'Copy', 'Print', and a zoom level of 14%. The left sidebar lists components for an iPhone 13, including 'housing_Next', 'housing_Next', 'housing', 'services_02', 'login_role', 'login', 'contact', 'services', and 'index'. The main workspace displays a grid of cards representing mobile pages for various services: 'ARACADA', 'SERVICES', 'HABITATS', 'HOUSE', 'NAME', 'MARAI', 'JUNGLE', 'SAVANE', 'CONTACT', 'CONNEXION', and 'CONNEXION'. Each card shows a preview of the mobile-optimized page content, featuring animal images and service details.

Diagramme de cas d'utilisation

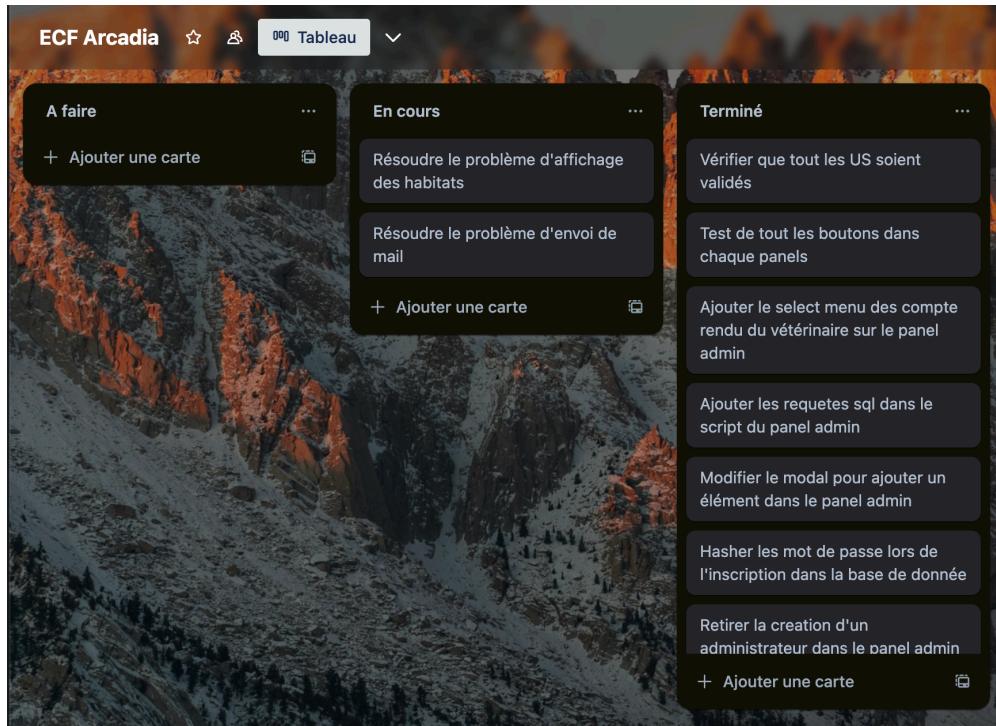


Modèle conceptuel et logique de données



Gestion de projet

Lien : <https://trello.com/b/RO02reLs/ecf-arcadia>



Spécifications techniques

Serveur

PHP 8.3.3

Extension PHP : PDO

Extension MongoDB : Composer

Front-End

HTML 5

CSS 3

Javascript

Back-End

PHP 8.3.3 avec PDO

JQuery et le protocole AJAX

SQLite

MongoDB

Déploiement

VPS personnel Nexus-Game