Dossier Projet

Application de collection de modèles 3D (version alpha)

Septembre 2022 - Septembre 2023

Réalisé par :

Timothée BAUDEQUIN

CDA - Bachelor Développeur IA

EPSI (2022/2023)

Remerciements

J’aimerais remercier deux personnes qui ont pu donner vie à ce projet d'après la propriété de Chasles.

Je voudrais également remercier une personne pour sa présence silencieuse, mais rassurante tout au long de ce projet. Parfois, il suffit d'un simple auditeur pour nous aider à surmonter les obstacles et à avancer. Merci encore à lui pour son soutien et pour être toujours alpaguable en cas de besoin.

Enfin, je souhaite remercier une personne que je n'apprécie pas plus que ça (oula) mais qui m’a malgré tout beaucoup aidé sur la relecture de ce rapport et qui risque de m’aider pour la création et l’inspiration des modèles 3D.

Table des matières

Table des figures

Glossaire

API : Ensemble de règles qui permet à différents logiciels de se parler et de partager des informations de manière structurée. (par exemple : http://app.meteo/paris n'est pas un lien vers un site web, mais nous retourne des informations sur la météo actuelle de Paris que l'on peut ensuite exploiter)

Application en couches : Application informatique développée en séparant les différentes fonctions en plusieurs couches logiques, pour plus de modularité et de réutilisabilité des composants logiciels.

Assertion : Vérification ou affirmation faite à un point spécifique du code pour s'assurer qu'une condition donnée est vraie.

Back-end : Partie invisible et non interactive d'une application web ou mobile qui assure le traitement des données et des fonctionnalités côté serveur.

Booléen : Valeur qui peut être soit vraie, soit fausse, souvent utilisée en informatique pour prendre des décisions.

Duaphin : Créé par erreur lors d'une frappe sur le clavier. Bien que dépourvu de signification, ce terme a été conservé à titre humoristique.

Fixtures : ensemble de données préconfigurées et stables utilisées lors des tests pour fournir un état de départ connu et reproductible au système ou à l'application en cours de test.

Front-end : Partie visible et interactive d'une application web ou mobile destinée aux utilisateurs finaux.

Mocking : Dans le cadre de test, c'est une technique qui permet de créer des objets ou des comportements simulés qui se comportent de la même manière que les objets réels

Scalabilité : Capacité à s'adapter

Token : chaîne de caractères utilisée pour authentifier et gérer l'état de la session utilisateur.

Versionner : Gérer et contrôler les différentes versions d'un logiciel ou d'un projet de développement.

XSS : Cross-Site Scripting est une technique d'attaque qui permet à un attaquant d'injecter du code malveillant dans une page web pour exploiter les vulnérabilités des navigateurs web des utilisateurs.

Compétences professionnelles

Les compétences professionnelles requises sont regroupées en 3 activités :

Activité 1- Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité :

- Maquetter une application
- Développer une interface utilisateur de type desktop
- Développer des composants d'accès aux données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web

Activité 2 - Concevoir et développer la persistance des données en intégrant les recommandations de sécurité :

- Concevoir une base de données
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données

Activité 3 - Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité :

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
- Concevoir une application
- Développer des composants métier
- Construire une application organisée en couches
- Développer une application mobile
- Préparer et exécuter les plans de tests d'une application
- Préparer et exécuter le déploiement d'une application

Introduction

Dans le cadre de ma troisième année de Bachelor Développeur IA à l'EPSI de Montpellier, j'ai réalisé des projets afin de répondre aux compétences professionnelles requises pour l'obtention du diplôme de CDA (Conception et Développement d'Applications).

Parmi les projets que je vais présenter, il y a un projet de collection de modèle 3D et une application mobile de météo.

Les deux sont des projets personnels que j'ai réalisés seul.

Le projet de collection de modèle 3D valide les compétences suivantes :

Activité 1- Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité :

- Maquetter une application
- Développer des composants d'accès aux données
- Développer la partie front-end* d'une interface utilisateur web
- Développer la partie back-end* d'une interface utilisateur web

Activité 2 - Concevoir et développer la persistance des données en intégrant les recommandations de sécurité :

- Concevoir une base de données
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données

Activité 3 - Concevoir et développer une application multicouche* répartie en intégrant les recommandations de sécurité :

- Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
- Concevoir une application
- Développer des composants métier
- Construire une application organisée en couches

- Préparer et exécuter les plans de tests d’une application
- Préparer et exécuter le déploiement d’une application

Quant au projet sur l’application mobile sur la météo, il valide les compétences suivantes :

Activité 3 - Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité :

- Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement
- Concevoir une application
- Construire une application organisée en couches
- Développer une application mobile

Tout d'abord, je vais présenter le cahier des charges des besoins de ces projets, en insistant sur les spécificités de chacun d'entre eux.
Ensuite, je vais me concentrer sur la conception et la réalisation de ces projets, en exposant les choix techniques et les étapes de développement qui ont été retenus.
Enfin, je conclurai en évoquant la gestion des projets, en abordant notamment la planification, le suivi de l'avancement et la communication avec les différentes parties prenantes.

1. Résumé des projets en anglais

The “Duaphin*” project is an online platform that enables 3D modeling enthusiasts and novices alike to discover this fascinating world while having fun collecting 3D models. Users can create an account, collect gold coins daily and buy decks of cards containing 3D models. What's more, they can trade or sell duplicate cards, making the collecting experience more interactive and social.

The site is developed using modern technologies such as React JS for the frontend, THREE JS for 3D model management, Node JS for backend development and MySQL for the database. These technologies ensure a smooth and pleasant user experience.
It's important to note that “Duaphin” will first be developed in an alpha version to check that all the elements and structure work well. Other features will then be added in the future to offer users an even richer, more immersive collecting experience. These include gift cards, models with special environments, trophies and much more. These features will add a touch of personalization and gamification to the collecting experience, giving users a unique experience on the platform.

The project also aims to encourage people to take up modeling or development by offering them an interesting and interactive experience. In short, “Duaphin” is a 3D model collection platform that combines technology, innovation and social interaction to offer a unique and fascinating experience for 3D modeling enthusiasts and novices alike.

My second project concerns a weather application. I developed a Java mobile application that uses a public API* to provide real-time weather information for a specific city. Users can retrieve accurate, up-to-date data on temperature, sky conditions and wind speed. This application enables users to access up-to-date weather information for any location they wish, helping them to stay informed of current weather conditions in real time.

2. Cahier des charges

2.1 Introduction

Afin de répondre aux compétences requises pour l'obtention du diplôme de CDA, j'ai choisi de réaliser un projet qui me tenait à cœur : l'application de collection de modèles 3D appelé "Duaphin"*.

Le principe de ce site est de permettre aux utilisateurs de découvrir et de collecter les différents modèles 3D disponibles. Cette idée est née en début d'année 2022, suite aux connaissances acquises lors de mes cours de DUT Informatique, ainsi qu'à une formation en ligne que j'ai suivie appelée Three JS Journey. Ce cours en ligne m'a appris comment créer des applications 3D interactives à l'aide de la bibliothèque JavaScript Three.js.

J'ai commencé à travailler sur ce projet en 2023, en mettant en pratique les compétences que j'ai acquises au fil de ma formation. Ce projet m'a permis de mobiliser plusieurs compétences clés pour l'obtention du diplôme de CDA, telles que la conception et le développement d'applications, la gestion d'une base de données, la gestion de projet et la veille technologique.

2.2 Contexte

Le projet "Duaphin" vise à offrir à tous les utilisateurs une expérience immersive et interactive de collection de modèles 3D. Il est conçu pour permettre à chacun de découvrir l'univers de la 3D fascinant tout en s'amusant à collectionner des modèles 3D. Les utilisateurs peuvent également proposer leurs propres modèles 3D ou idées pour les inclure sur le site, ce qui leur permet de participer à la communauté et de partager leur passion pour la modélisation 3D, le dessin et l'art.

2.3 Objectifs

Le but du projet “Duaphin” est de créer un site web de collection de modèles 3D avec les fonctionnalités suivantes :

- Permettre aux utilisateurs de créer un compte unique et devenir des membres du site.
- Permettre aux utilisateurs de posséder une collection de modèles 3D.
- Permettre aux utilisateurs de récupérer chaque jour un certain nombre de pièces d'or.
- Permettre aux utilisateurs d’acheter avec des pièces d’or des decks qui contiennent des cartes de modèle 3D afin d’alimenter leur collection.
- Permettre la collection de cartes avec des raretés différentes (pourcentages d'obtention différents) et appartenant à des séries de cartes.
- Permettre aux utilisateurs d'échanger ou vendre des cartes en double.
- Offrir des cartes cadeaux (événements temporaires) pour les utilisateurs.
- Offrir des modèles avec des environnements spéciaux pour les cartes très rares.
- Offrir des trophées pour récompenser les statistiques des utilisateurs.

2.4 Exigences fonctionnelles

- Une page d'accueil avec une présentation du site.
- Une page de connexion ou de création de compte pour les nouveaux utilisateurs.
- Une page de collection de tous les modèles 3D disponibles (possibilité de trier).
- Une page pour récupérer des pièces d'or et pour acheter des decks de cartes contenant des modèles 3D.
- Une page pour afficher les cartes possédées par l'utilisateur (possibilité de trier).
- Une page pour échanger ou vendre des cartes en double.

- Une page pour afficher les trophées des utilisateurs.
- Une page d'information pour afficher la version actuelle de l'application et les modifications des anciennes versions et de celles à venir.

2.5 Exigences techniques après analyse.

Après une analyse approfondie des différentes options disponibles, j'ai choisi les langages et logiciels suivants pour mon projet de développement de site web :

React JS : une bibliothèque JavaScript populaire pour la création d'interfaces utilisateur dynamiques et réactives. J'ai choisi React JS pour sa modularité, sa facilité de développement et sa grande communauté de développeurs. Cela m'a permis de créer une interface utilisateur fluide et dynamique, ainsi que d'intégrer facilement des fonctionnalités complexes telles que la gestion des états et des événements.

THREE JS : une bibliothèque JavaScript pour la création d'animations et de graphismes en 3D. J'ai choisi THREE JS pour sa flexibilité et sa capacité à créer des graphismes en 3D de haute qualité qui peuvent être intégrés à mon site web. Cela me permettra de créer une expérience utilisateur immersive et engageante.

Node JS : une plateforme de développement JavaScript pour la création de serveurs. J'ai choisi Node JS pour sa facilité de développement, sa scalabilité* et sa grande communauté de développeurs. Cela m'a permis de créer une API robuste et sécurisée pour mon site web.

MySQL : une base de données relationnelle open-source. J'ai choisi MySQL pour sa fiabilité, sa sécurité et sa capacité à gérer des données volumineuses. Cela m'a permis de stocker et de gérer efficacement les données de mon site web. J'utilise également MAMP pour avoir une interface PhpMyAdmin qui facilite l'utilisation de MySQL.

Blender : un logiciel open-source pour la création de modèles 3D. J'ai choisi Blender pour sa facilité d'utilisation, sa flexibilité et sa grande communauté de développeurs. Cela me permettra de créer des modèles 3D de haute qualité pour mon site web.

Github : une plateforme de développement collaborative populaire qui permet de stocker, collaborer et partager du code source avec d'autres développeurs. En choisissant Github pour

versionner* mon code, je bénéficie d'une gestion de version efficace, d'une collaboration facilitée entre les membres de l'équipe (s'il devait en avoir une dans le futur), d'un suivi des bugs et d'un hébergement sécurisé pour mon code. Github peut être intégré à de nombreux autres outils de développement pour permettre une automatisation et une intégration continue.

En choisissant ces langages et logiciels, j'ai pu créer un site web performant, sécurisé et évolutif, tout en offrant une expérience utilisateur immersive et engageante grâce à l'intégration de graphismes en 3D.

2.6 Contraintes

Une version alpha fonctionnelle doit être terminée afin de pouvoir présenter une première version du site.

La version alpha doit comporter les fonctionnalités suivantes :

- Permettre aux utilisateurs de créer un compte unique et devenir des membres du site.
- Permettre aux utilisateurs de posséder une collection de modèles 3D.
- Permettre aux utilisateurs de récupérer chaque jour un certain nombre de pièces d'or pour acheter des decks de cartes contenant des modèles 3D.
- Permettre aux utilisateurs d’acheter avec des pièces d’or des decks qui contiennent des cartes de modèle 3D.
- Permettre la collection de cartes avec des raretés différentes et appartenant à des séries de cartes.

Le projet n’a pas de budget financier, ni de contrainte graphique ou ergonomique, car c’est un projet personnel.

Le site web doit être sécurisé pour protéger les informations des utilisateurs.

L’application doit proposer une expérience dynamique et innovante aux utilisateurs.

Je dois apprendre à maîtriser Node JS afin de concevoir un back-end robuste et structuré. Je dois également maîtriser React JS dans le but de proposer une interface utilisateur dynamique et moderne. Pour parvenir à maîtriser Node JS et React JS, je me suis appuyé sur la documentation et les tutoriels en ligne pour me familiariser avec ces outils de développement modernes.

2.7 Planification

Phase 1 : Analyse de la demande et modélisation du projet. (2 mois)

Phase 2 : Conception et création de la version Alpha (développement du backend (API*), de la base de données et du frontend. Tests et débogages.) (1 mois)

Phase 3 : Création des modèles 3D (1 mois)

Phase 4 : Ajout des fonctionnalités (trophées, échange/vendre, etc.) (1 mois)

Phase 5 : Maintenir le site et continuer d'ajouter des modèles.

2.8 Validation et tests

Des tests doivent être effectués pour s'assurer que les fonctionnalités du site web telles que la récupération de pièces d'or, l'achat de decks de cartes et l'échange de cartes fonctionnent correctement.

J'ai choisi d'utiliser le framework JavaScript Jest pour réaliser les tests de mon API, car Jest est un outil puissant et facile à utiliser pour les tests d'applications JavaScript. Il offre une variété de fonctionnalités utiles pour la mise en place de tests, notamment la configuration, le mocking*, l'assertion* et la gestion des erreurs. De plus, Jest est simple à intégrer dans le flux de développement grâce à sa compatibilité avec de nombreux outils de développement et de CI/CD.

Des tests de sécurité doivent être effectués pour s'assurer que le site web est sécurisé contre les attaques informatiques.

2.9 Support et maintenance

Un système de support pour les utilisateurs en cas de problème doit être mis en place. La mise en place d'une adresse mail de support qui permettra aux utilisateurs de contacter facilement l'équipe de développement en cas de besoin.

Une maintenance régulière doit également être mise en place, pour assurer le bon fonctionnement du site web. Cette maintenance doit être effectuée de manière régulière pour corriger les bugs éventuels, mettre à jour les dépendances et assurer la sécurité du site web.

Une page dédiée pour les mises à jour et les changements apportés au site web sera mise en place. Cette page permettra aux utilisateurs de suivre l'évolution de l'application, ainsi que les ajouts, corrections et autres changements effectués. Elle peut contenir des informations sur la version actuelle de l'application, les changements apportés à cette version et les ajouts prévus pour les versions futures. Cette transparence permettra aux utilisateurs de mieux comprendre les évolutions du site web et de se sentir impliqués dans son évolution.

2.10 Normes

Le développement du site web doit respecter les normes de développement pour React et Node.js, ainsi que les normes du web telles que HTML5, CSS3 et JavaScript pour garantir un code bien structuré, facile à lire et à maintenir. Il est également important de commenter le code pour faciliter la compréhension et la maintenance du projet.

Pour assurer la qualité et la fiabilité de la base de données de l'application, il est crucial de respecter les normes de MySQL, étant donné que c'est la technologie utilisée pour son développement.

Pour respecter les réglementations de la RGPD, il est primordial de stocker de manière sécurisée les informations des utilisateurs en suivant les normes de sécurité appropriées.

3. Rapport technique

3.1 Conception

Dans cette section du rapport technique, je vais aborder les aspects clés de la conception de mon système, en examinant en détail la conception de la base de données ainsi que du front-end et du back-end, afin de mieux comprendre les choix techniques et les solutions mises en place pour répondre aux besoins du projet.

3.1.1 Conception de la base de données

Pour la conception de la base de données, j'ai choisi d'utiliser MAMP, qui propose MySQL avec une interface PhpMyAdmin. Ayant déjà utilisé cette interface pour plusieurs projets, je suis habitué à son utilisation. Dans un premier temps, j'ai créé un diagramme entité-association pour la version alpha de la base de données. Ce diagramme permet de visualiser la structure de la base de données.

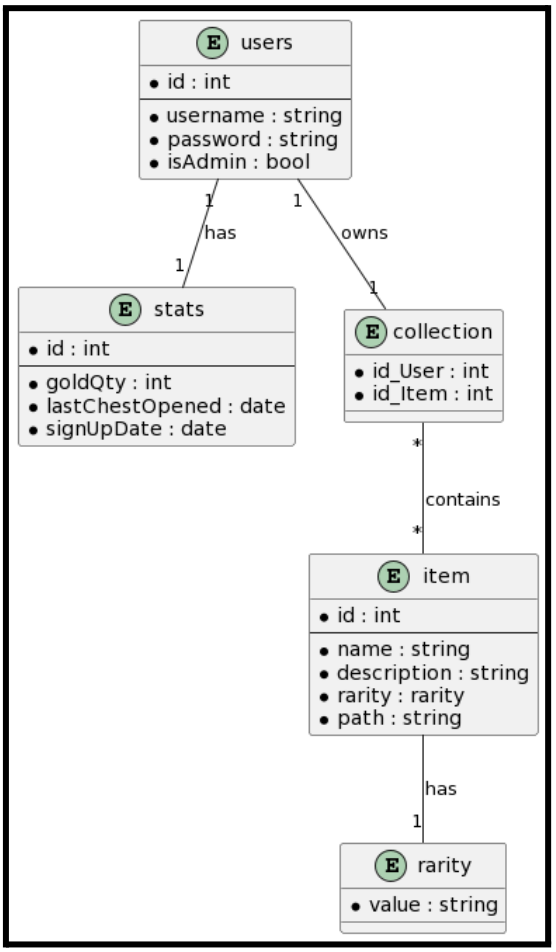


Figure 1 : Diagramme entité association pour la base de données

Le schéma de base de données suivant met en évidence les relations entre les tables en présentant les clés primaires et les clés étrangères.

USERS (id, username, password, isAdmin)
STATS (id #, goldQty, lastChestOpened, signUpDate)
COLLECTION (id_User #, id_Item #)
ITEM (id, name, description, rarity#, path)
RARITY (value)

Le diagramme montre qu'un utilisateur est identifié par un identifiant unique qui s'incrémente automatiquement grâce à une fonctionnalité de MySQL et possède un nom d'utilisateur et un mot de passe. L'attribut "isadmin" permet de déterminer si un utilisateur est administrateur grâce à un booléen*.
Chaque utilisateur possède des statistiques permettant la gestion de leur compte et qui détermine les actions possibles sur le site telles que la quantité d'or, le dernier coffre ouvert et la date d'inscription. De plus, chaque utilisateur possède une collection, qui est une liste d'objets. "Collection" est une table 'many to many' permettant qu'un utilisateur puisse avoir plusieurs items, et plusieurs items peuvent appartenir à différents utilisateurs. L'identifiant unique sur "id_User" et "id_Item" permet de garder chaque combinaison unique. Pour que l'utilisateur puisse posséder plusieurs fois la même carte, il suffirait d'ajouter un attribut "quantity" à cette table.
Chaque objet est identifié par un identifiant unique (qui s'incrémente automatiquement) et possède un nom, une description, un chemin d'accès pour l'image de l'objet, ainsi qu'une rareté qui est déterminée parmi les raretés de la table "rarity".

3.1.2 Conception du back-end

Pour concevoir le back-end de mon application, j'ai choisi d'utiliser Node JS pour créer une API qui servira de passerelle entre le front-end et la base de données. J'ai opté pour le modèle MVC (Modèle-Vue-Contrôleur) pour structurer mon code et séparer les différents composants du back-end.

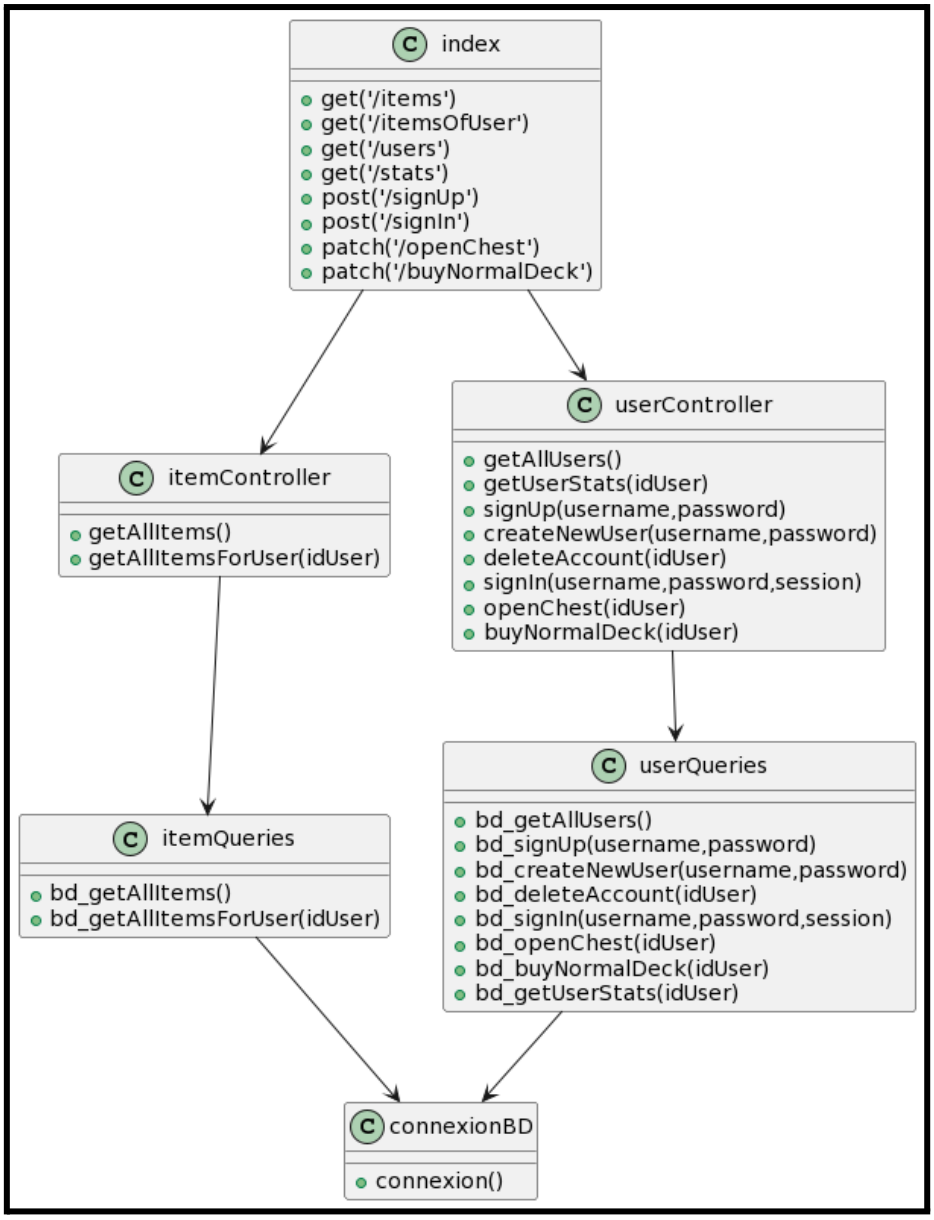


Figure 2 : Diagramme de classe pour le back-end

Le diagramme de classes que j'ai créé montre clairement la division en couches de mon back-end.

La première couche est l'index qui répertorie toutes les routes de mon API, c'est-à-dire tous les appels possibles. Ensuite, j'ai créé les contrôleurs qui contiennent des fonctions pour chaque route (itemController et userController).

La deuxième couche, celle des contrôleurs, est chargée de recevoir les appels à l'API, de traiter les données et de renvoyer une réponse appropriée. Pour cela, les contrôleurs utilisent des requêtes spécifiques pour chaque fonction. Les requêtes sont gérées par les "queries", qui

sont des fonctions qui encapsulent les requêtes SQL vers la base de données (itemQueries et userQueries).

Les "queries" utilisent la classe "connexionBD" pour se connecter à la base de données et exécuter les requêtes SQL. La classe "connexionBD" est responsable de la création et de la gestion de la connexion à la base de données. Elle fournit également les méthodes pour exécuter des requêtes et récupérer les résultats.

En utilisant ce modèle, le back-end serait bien structuré et facile à maintenir. Les différentes couches sont clairement définies et les responsabilités sont bien séparées, ce qui facilite la compréhension et la modification du code. Cette séparation des responsabilités permet de plus de rendre le code plus modulaire et réutilisable, ce qui facilitera l'ajout de nouvelles fonctionnalités à l'avenir.

3.1.3 Conception du front-end

Afin de faciliter le processus de conception de l’application, j'ai réalisé une maquette de l'application à l'aide de l'outil Figma (outil de conception d'interface utilisateur) avant de commencer le développement de la partie frontale. J'ai puisé mon inspiration auprès de plusieurs sites web, tels que celui de la marque Bugatti pour leur présentation de voitures en grille et leur ambiance muséale. De plus, pour la présentation de mes modèles, j'ai utilisé le design des photos des appareils photo Polaroid, avec une photo centrée et encadrée d'un contour blanc.

Voici les différentes fenêtres que j'ai conçues :

- La page d'accueil de l'application (Home) :

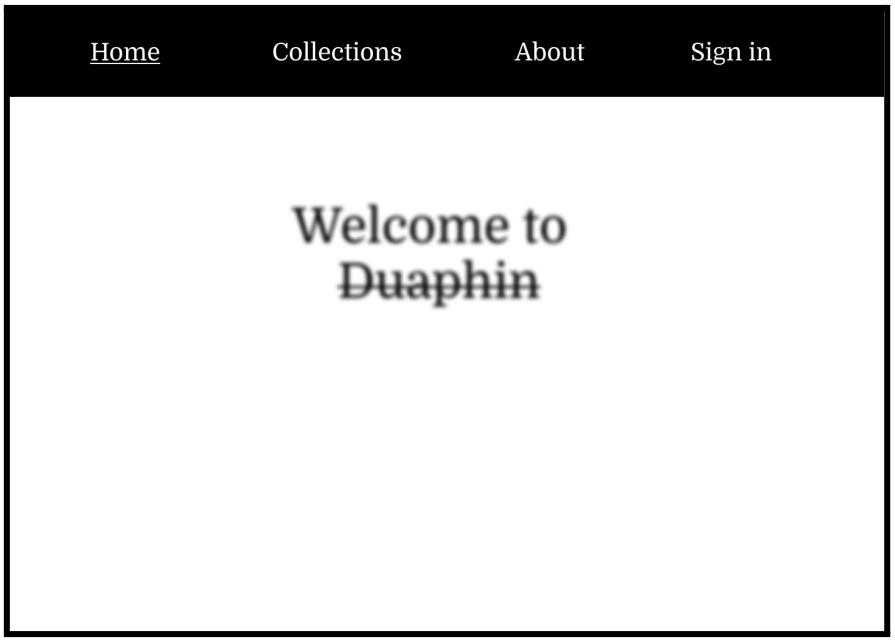


Figure 3 : Page d'accueil de la maquette

Cette page possède une barre de navigation qui permet de naviguer entre les différentes pages. Pour l'instant, il n'y a que le titre de l'application. Afin de faciliter la compréhension de l'application pour les nouveaux utilisateurs, j'ai prévu d'ajouter des informations supplémentaires sur la moitié inférieure de la page. Ces informations comprendront des explications brèves et une présentation des fonctionnalités principales de l'application, pour expliquer rapidement le but de l'application et d'inciter les visiteurs à en savoir plus et à explorer les fonctionnalités de l'application. En fournissant ces informations claires et concises, les nouveaux utilisateurs auront une meilleure compréhension de l'application et seront plus enclins à rester sur l'application pour découvrir ses fonctionnalités.

- La page de collection avec tous les objets (Collections) :

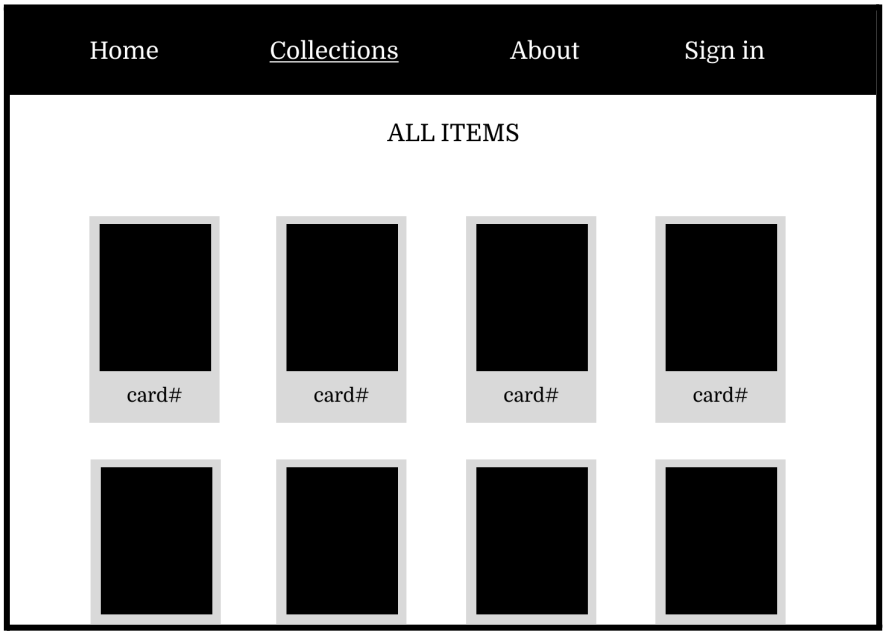


Figure 4 : Page de collection de la maquette

Cette page possède également la barre de navigation. Elle présente l'ensemble des cartes des modèles 3D disponibles dans la collection. Les modèles pourront être triés selon le nom, la rareté ou l'id (ordre d'apparition dans le jeu). Cette page permettra aux utilisateurs de visualiser facilement l'ensemble des modèles disponibles et de trouver plus facilement ceux qui les intéressent.

- La page d'à propos (About) :

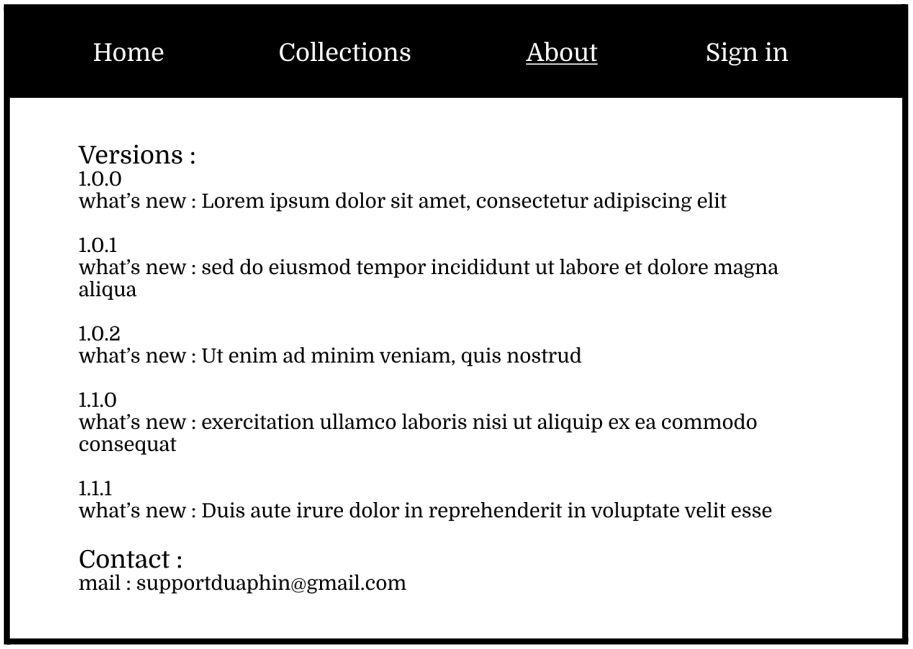


Figure 5 : Page à propos de la maquette

Cette page possède également la barre de navigation. Elle présente l'ensemble des informations relatives aux versions de l'application, comme les bugs corrigés, les fonctionnalités ajoutées et les cartes/modèles ajoutés. Les utilisateurs pourront par ailleurs y trouver les informations nécessaires pour contacter le support de l'application en cas de problème. Cette page permettra aux utilisateurs de rester informés sur les mises à jour de l'application et d'entrer en contact facilement avec le support en cas de besoin.

- La page d'inscription ou de connexion (Sign in) :

Figure 6 : Page de connexion/inscription de la maquette

Cette page possède également la barre de navigation. Elle présente un formulaire avec deux boutons permettant aux utilisateurs de s'inscrire ou de se connecter à leur compte. En fonction du bouton sur lequel l'utilisateur clique. Cette page permettra aux utilisateurs de facilement accéder à leur compte ou de s'inscrire pour accéder à toutes les fonctionnalités de l'application.

- La page de statistique de l'utilisateur (My Stuff) :

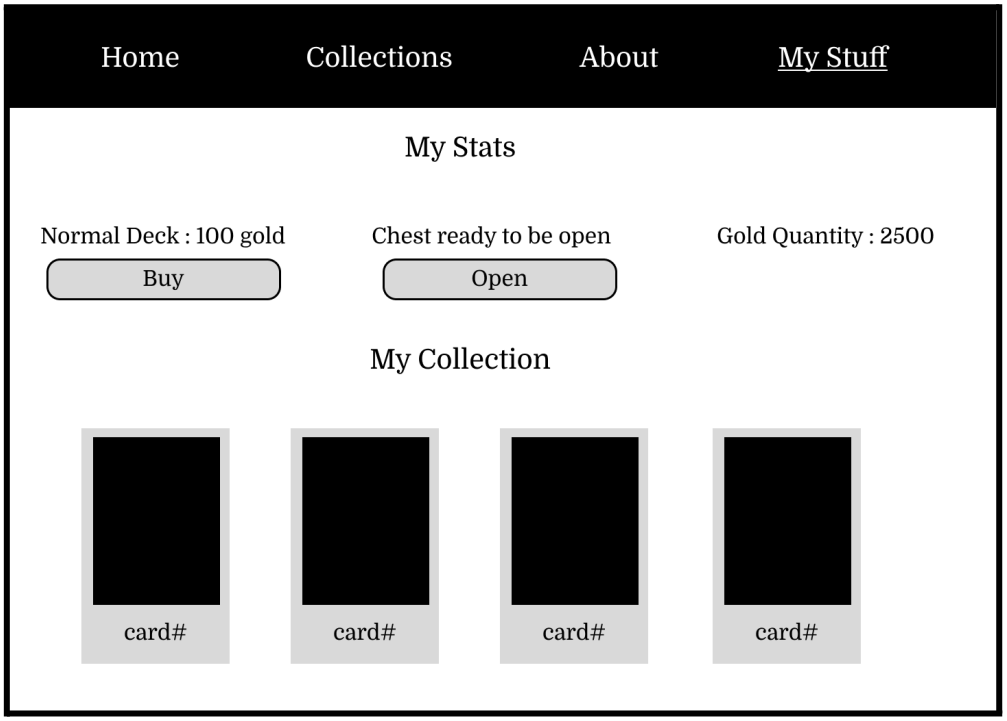


Figure 7 : Page d'information du compte utilisateur de l'application

Cette page possède également la barre de navigation. Elle présente l'ensemble des cartes des modèles 3D que l'utilisateur a réussi à collecter, avec la possibilité de les trier selon le nom, la rareté ou l'id (ordre d'apparition dans le jeu). La page permet aussi à l'utilisateur d'acheter un deck contenant une carte de la collection. En outre, un coffre quotidien permet de récupérer une quantité d'or, et la quantité d'or actuelle de l'utilisateur est visible à droite de la page. Cette page permettra aux utilisateurs de suivre leur progression dans la collection et de gérer leur compte en achetant des decks et en collectant de l'or.

3.2 Réalisation

Dans cette partie du rapport technique, je vais explorer les éléments fondamentaux de la conception de mon système en examinant attentivement la réalisation de la base de données, ainsi que celle du front-end et du back-end. Mon objectif est de vous fournir une compréhension approfondie des choix techniques et des solutions mises en place pour répondre aux besoins du projet.

3.2.1 Réalisation de la base de données

Dans cette section, nous allons examiner comment j'ai réalisé la base de données de mon application. La base de données est l'une des parties les plus importantes de toute application, car elle stocke les données essentielles de l'application et assure leur disponibilité et leur intégrité. Dans cette partie, nous allons examiner comment j'ai conçu et mis en place la base de données pour mon application, en utilisant des pratiques de sécurité et de fiabilité pour garantir que les données de l'utilisateur sont stockées de manière sûre et accessible.

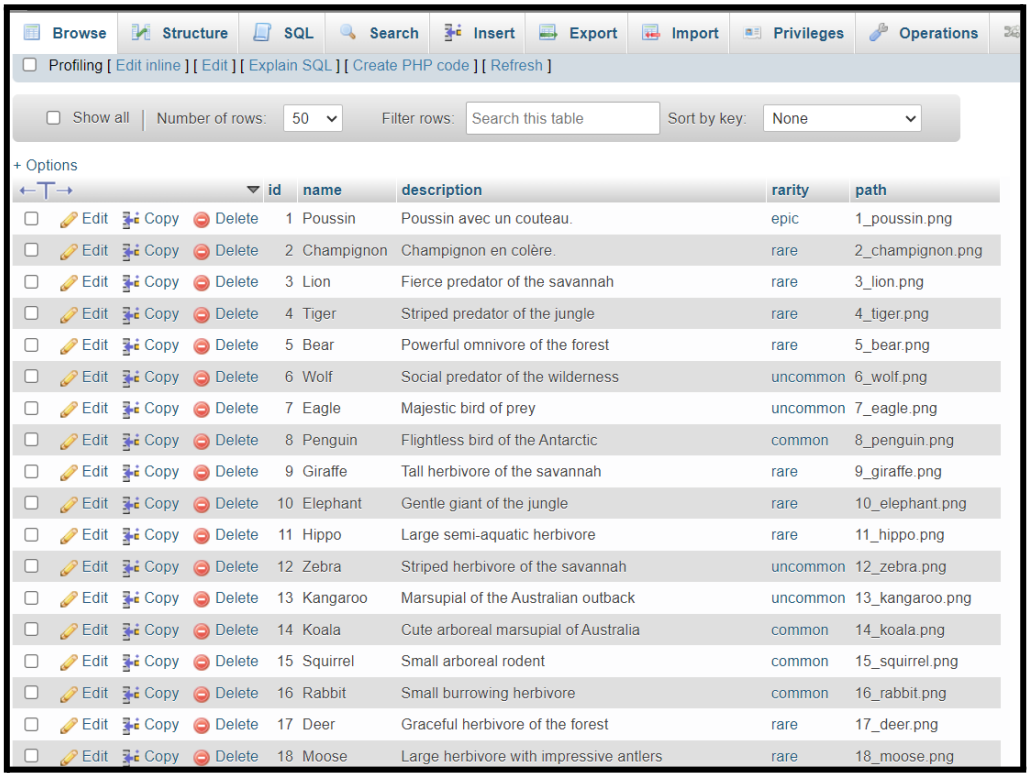
Pour réaliser la base de données, j'ai utilisé un diagramme entité-association et l'interface PHPMyAdmin. J'ai veillé à respecter les règles de typage. J'ai également mis en place des contraintes de clé primaire, de clé étrangère et d'auto-incrémentation pour garantir l'intégrité des données et faciliter la gestion de la base de données.
Ma base de données respecte :

- La première forme normale, car chaque table ne contient que des données atomiques, c'est-à-dire que dans chacune des colonnes, il n’y a qu'une unique information.
- La deuxième forme normale, parce que chaque colonne dépend uniquement de la clé primaire de la table correspondante, sauf la table Item, puisque les objets auront des noms et des images (path) uniques.
- La troisième forme normale, car il n'y a pas de dépendance transitive entre les colonnes de chaque table, autrement dit, que toutes les informations relatives à une table sont seulement dans cette table.

En respectant les formes normales, l'application est mieux structurée, plus facile à maintenir et moins sujette aux erreurs de données.
En utilisant ces outils et en appliquant ces bonnes pratiques, j'ai pu créer une base de données robuste et cohérente qui répond aux besoins du projet.

Après avoir créé les tables de la base de données pour mon application de collection, j'ai ajouté un utilisateur avec ses informations de connexion et les cinq niveaux de rareté d'objets que les utilisateurs peuvent collecter. J'ai également ajouté 50 objets de collection (qui me servent de fixtures*) pour faciliter le développement et les tests du front-end et du back-end. Ces objets ont été générés par une intelligence artificielle pour offrir une certaine diversité dans un échantillon suffisamment grand pour les tests futurs. Il est important de noter que pour la version définitive de l'application, tous les objets seront créés par mes soins sur Blender.

Sur la capture d'écran de l'interface de phpMyAdmin, on peut apercevoir certains des objets de test ajoutés à la base de données.



	id	name	description	rarity	path
Edit Copy Delete	1	Poussin	Poussin avec un couteau.	epic	1_poussin.png
Edit Copy Delete	2	Champignon	Champignon en colère.	rare	2_champignon.png
Edit Copy Delete	3	Lion	Fierce predator of the savannah	rare	3_lion.png
Edit Copy Delete	4	Tiger	Striped predator of the jungle	rare	4_tiger.png
Edit Copy Delete	5	Bear	Powerful omnivore of the forest	rare	5_bear.png
Edit Copy Delete	6	Wolf	Social predator of the wilderness	uncommon	6_wolf.png
Edit Copy Delete	7	Eagle	Majestic bird of prey	uncommon	7_eagle.png
Edit Copy Delete	8	Penguin	Flightless bird of the Antarctic	common	8_penguin.png
Edit Copy Delete	9	Giraffe	Tall herbivore of the savannah	rare	9_giraffe.png
Edit Copy Delete	10	Elephant	Gentle giant of the jungle	rare	10_elephant.png
Edit Copy Delete	11	Hippo	Large semi-aquatic herbivore	rare	11_hippo.png
Edit Copy Delete	12	Zebra	Striped herbivore of the savannah	uncommon	12_zebra.png
Edit Copy Delete	13	Kangaroo	Marsupial of the Australian outback	uncommon	13_kangaroo.png
Edit Copy Delete	14	Koala	Cute arboreal marsupial of Australia	common	14_koala.png
Edit Copy Delete	15	Squirrel	Small arboreal rodent	common	15_squirrel.png
Edit Copy Delete	16	Rabbit	Small burrowing herbivore	common	16_rabbit.png
Edit Copy Delete	17	Deer	Graceful herbivore of the forest	rare	17_deer.png
Edit Copy Delete	18	Moose	Large herbivore with impressive antlers	rare	18_moose.png

Figure 8 : Interface PhpMyAdmin de la table items

3.2.2 Réalisation du back-end

Dans cette section, nous allons voir comment j'ai réalisé le back-end pour créer une API avec Node.js. Node.js est une plate-forme logicielle libre et multiplateforme qui permet d'exécuter du code JavaScript côté serveur. Cette technologie est idéale pour créer des applications web évolutives et performantes. Dans cette partie, nous allons examiner comment j'ai utilisé

Node.js pour créer une API robuste pour mon application, en utilisant des pratiques de sécurité solides pour protéger les données de l'utilisateur.

Afin de structurer le back-end de l'application, j'ai séparé les différentes couches de l'architecture. Comme on peut le voir sur la capture d'écran ci-dessous, l'index, les contrôleurs, les requêtes et la connexion à la base de données sont clairement séparés.

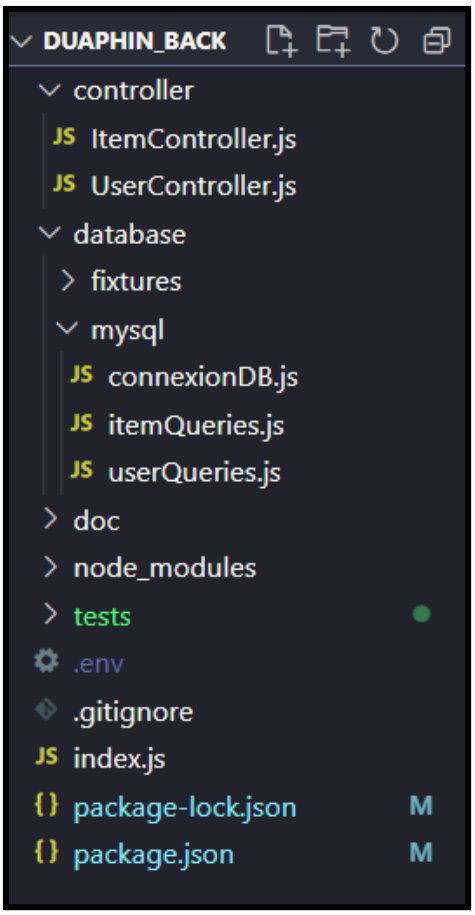


Figure 9 : Structure de fichier de mon back-end

Ensuite, j'ai mis en place l'API en utilisant Node.js et Express, qui offrent des fonctionnalités puissantes pour le développement web. J'ai créé la connexion à la base de données, ce qui me permet de récupérer ou modifier des informations de la base de données via des requêtes HTTP en utilisant l'API. J'ai utilisé Postman pour tester mes routes, telles que la route '/items', qui permet de récupérer tous les objets disponibles dans la base de données, comme on peut le voir sur la capture d'écran ci-dessous.

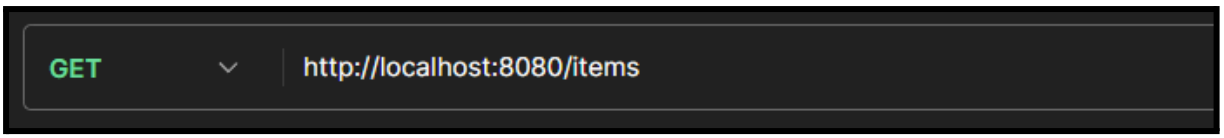


Figure 10 : Requête à l'API sur Postman

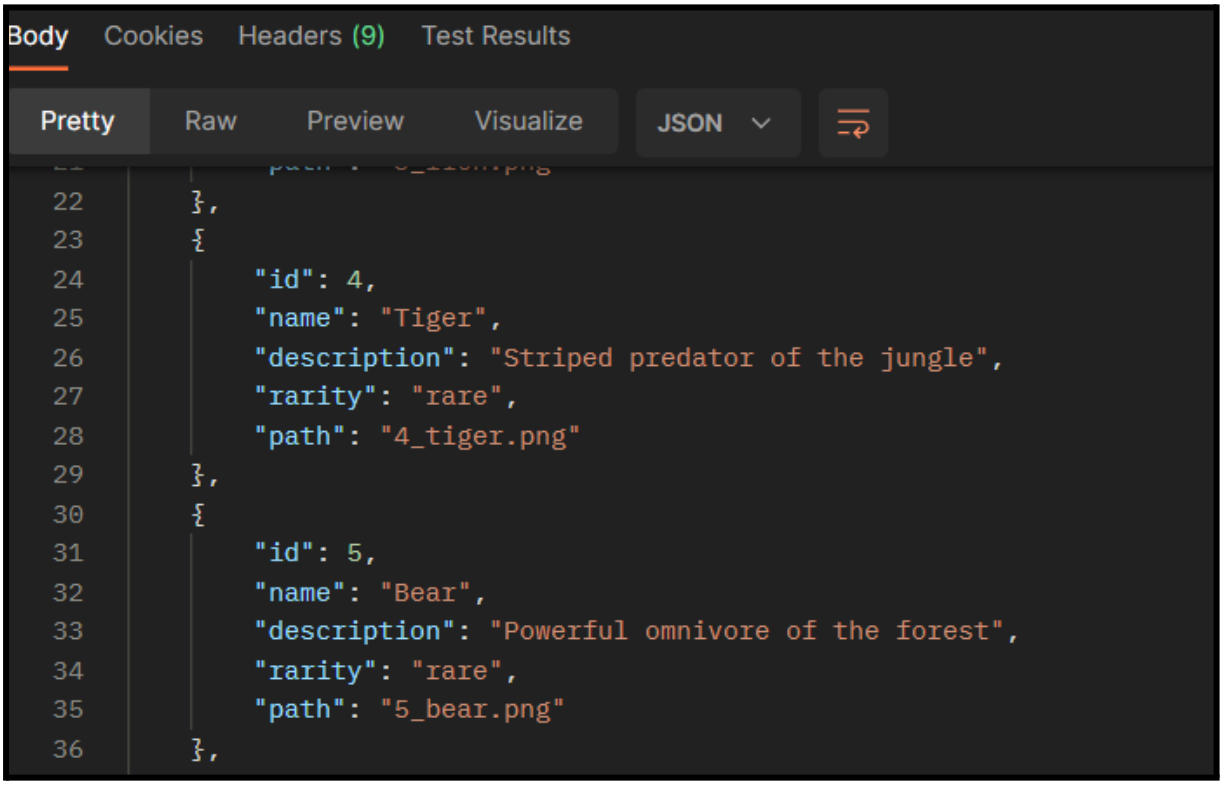


Figure 11 : Réponse de la requête à l'API sur Postman (cf. figure 10)

Une des fonctionnalités intéressantes de l'application est l'inscription d'un nouvel utilisateur, où l'utilisateur saisit un nom d'utilisateur et un mot de passe. Le mot de passe est ensuite haché avant d'être stocké dans la base de données, ce qui permet de protéger les données de l'utilisateur. Par exemple, le mot de passe "test123" peut être stocké dans la base de données sous la forme "$2a$10$MlDvdYkC5Uj1eCp9gNVnkeUpW3ENxnoP8i1jx4Laygo7Ad.K6pkwu". Cette technique de hachage de mot de passe contribue à renforcer la sécurité de l'application en protégeant les informations sensibles de l'utilisateur.

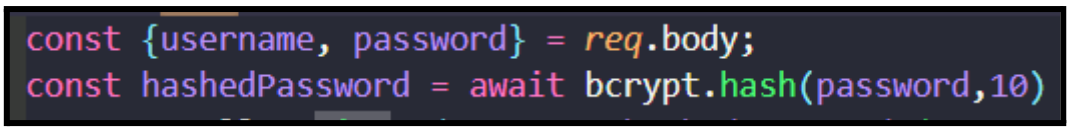


Figure 12 : Hachage du mot de passe utilisateur

Afin de renforcer la sécurité de l'application, les requêtes sont préparées en utilisant des paramètres de requête avec des marqueurs de position (" ? ") pour éviter les injections SQL. Cette technique consiste à transmettre les données utilisateur à la base de données via des paramètres de requête, plutôt que d'insérer directement les données dans la requête SQL. En utilisant cette méthode, on évite les risques d'injections SQL qui peuvent permettre à des pirates informatiques de modifier, supprimer ou récupérer des données de la base de données de manière malveillante. La préparation des requêtes est donc une pratique essentielle pour garantir la sécurité des données de l'application.

```
function bd_createNewUser(username, hashedPassword, callback) {
    // NEW USERS WITH STATS
    const query = 'INSERT INTO users (username, password,isAdmin) VALUES (?,?,?);';
    connection.query(query, [username, hashedPassword, 0], (err, results) => {
        if (err) {
            console.error('Error executing query:', err);
            callback(err, null);
            return;
        }
        const idUser = results.insertId;
        // NEW STATS
        const querybis = "INSERT INTO stats (id,goldQty,lastChestOpened,signUpDate) VALUES (?,?,DATE_SUB(NOW(), INTERVAL 24 HOUR),NOW()) "
        connection.query(querybis, [idUser, 1000], (err, results) => {
            if (err) {
                console.error('Error executing query:', err);
                callback(err, null);
                return;
            }
            return idUser;
        })
    });
}
```

Figure 13 : Préparation d'une requête à la base de données

J'ai mis en place toutes les fonctions nécessaires pour récupérer ou modifier les informations de la base de données, telles que définies dans le diagramme de classe, afin de répondre aux besoins de l'application et des utilisateurs. Pour cela, j'ai créé un controller utilisateur (userController) qui contient les fonctions nécessaires pour gérer les requêtes liées aux utilisateurs. Vous pouvez voir ci-dessous une capture d'écran montrant les fonctions disponibles dans le userController.

```
function updateUser(updateUser,idUser,req,res) {
    userQueries.bd_updateUser(updateUser,idUser,req,res)
}

function signUp(username,hashedPassword,req,res){
    userQueries.bd_signUp(username, hashedPassword,req,res)
}

function signIn(username,Password,session,req,res){
    userQueries.bd_signIn(username,Password,session,req,res)
}

function openChest(id,req,res){
    userQueries.bd_openChest(id,req,res)
}

function buyNormalDeck(id,req,res){
    userQueries.bd_buyNormalDeck(id,req,res)
}

function getUserStats(id,req,res){
    userQueries.bd_getUserStats(id,req,res)
}

function deleteAccount(id,req,res){
    userQueries.bd_deleteAccount(id,req,res)
}
```

Figure 14 : Appel aux fonctions dans le controller

3.2.3 Réalisation du front-end

J'ai structuré le front-end de mon application web en suivant les directives fournies dans la documentation de React JS. Cette structure est conçue pour séparer les différents composants de l'application, ce qui facilite la navigation et la modification du site. En suivant cette structure, je peux apporter des modifications au site plus efficacement et avec moins de risques d'erreurs.

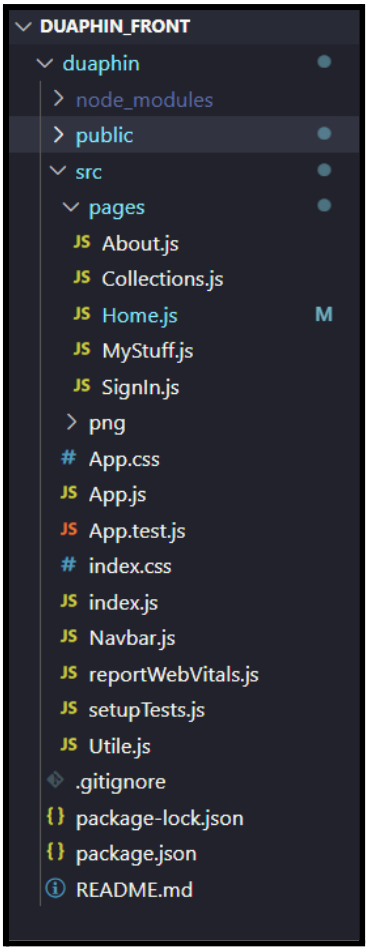


Figure 15 : Structure de fichier de mon front-end

En suivant la maquette que j'avais réalisée, j'ai essayé de m'en rapprocher le plus possible lors de la création de mon application web. Grâce à React JS, j'ai pu créer des éléments réutilisables, comme la barre de navigation qui est composée de 5 CustomLink (un élément que j'ai créé) et qui affiche différents éléments en fonction des paramètres qui lui sont donnés (“to” qui redirige vers la page et “name” qui est le texte affiché).

Dans les captures d'écran ci-dessous, on peut voir que les éléments de la barre de navigation "Sign In" et “My Stuff” possèdent une syntaxe différente. Ces lignes utilisent une syntaxe appelée "ternary operator" (opérateur ternaire) qui permet de créer une condition et d'exécuter une expression en fonction de cette condition. Dans ce cas précis, si l'utilisateur de l'application n'est pas connecté, la barre "Sign In" sera visible dans la barre de navigation, et si l'utilisateur est connecté, la ligne en dessous sera affichée, car elle possède une condition opposée (cf. capture d'écran).

```jsx
export default function Navbar({islog}) {
    return (
        <nav className="header_nav">
            <ul>
                <CustomLink to='./' name='Home' />
                <CustomLink to='./collections' name='Collections' />
                <CustomLink to='./about' name='About' />
                {/* Show "Sign In" link only if user is not logged in */}
                {!islog &&<CustomLink to='./signin' name='Sign In' />}
                {/* Show "My Stuff" link only if user is logged in */}
                {islog &&<CustomLink to='./mystuff' name='My Stuff' />}
            </ul>
        </nav>
    )
}

function CustomLink({ to, name }) {
    const path = useResolvedPath(to)
    // Allows to know if we are currently on this page (active page)
    const isActive = useMatch({path: path.pathname, end: true})
    return (
        <li className={isActive ? "active" : ""}>
            <Link to={to}>
                {name}
            </Link>
        </li>
    )

}
```

Figure 16 : Structure de la barre de navigation de mon front-end

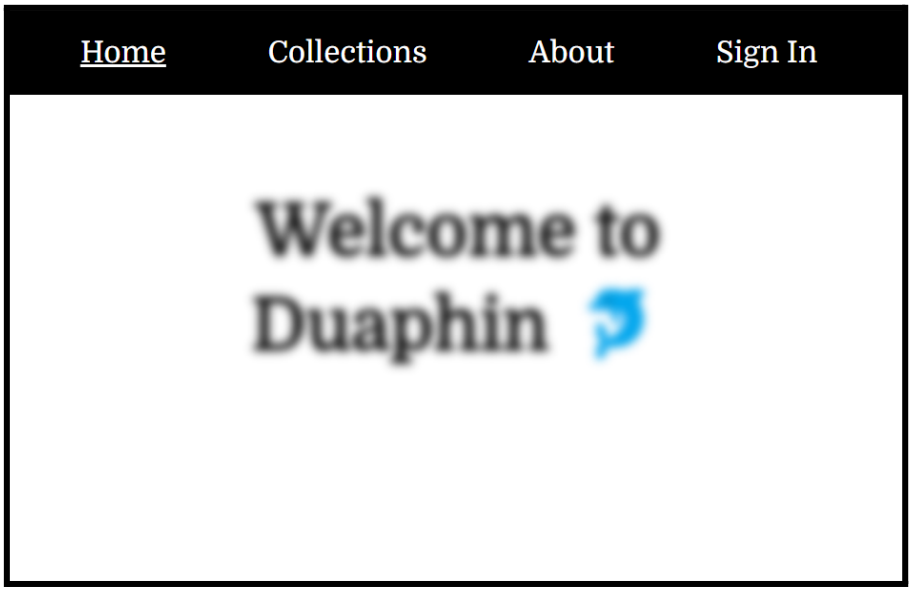


Figure 17 : Page d'accueil de mon application non connecté

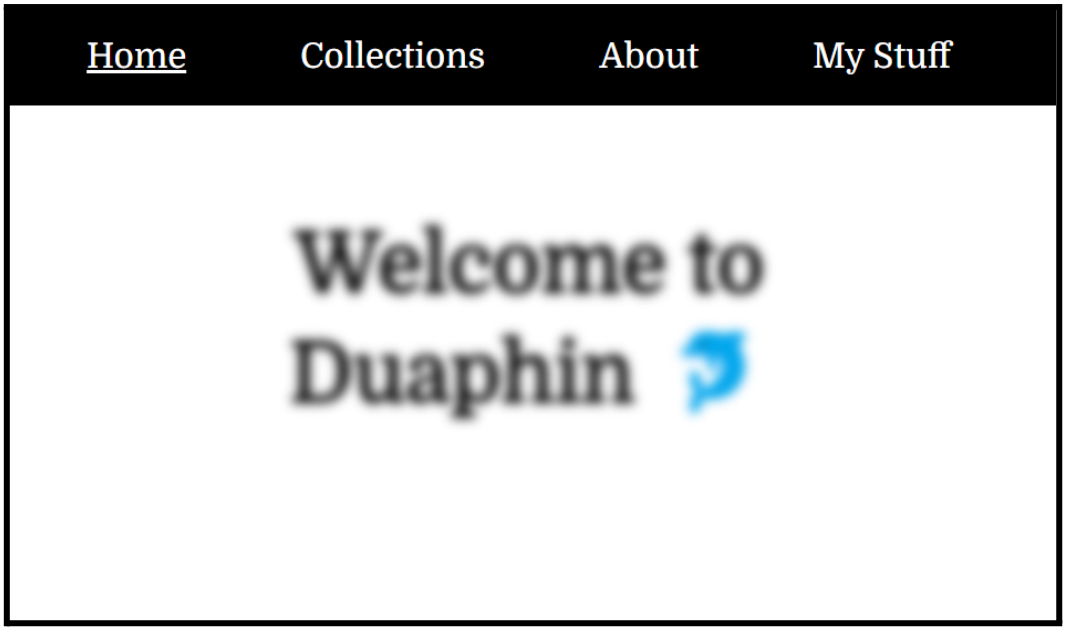


Figure 18 : Page d'accueil de mon application connecté

La page Collections regroupe tous les objets qu'il est possible de collectionner. Sur la capture d'écran du site ci-dessous (ou la seconde, où j'ai dézoomé pour que l'on puisse voir plus de cartes), on peut voir comment sont affichés les objets.

J'ai cherché à respecter la maquette en utilisant le style des photos d'appareils Polaroid pour l'affichage des objets.

La différence de couleur des cartes est liée à leur rareté. Pour l'instant, j'ai utilisé la même photo pour toutes les cartes, car c'est une étape de création de modèles qui sera ajustée ultérieurement.

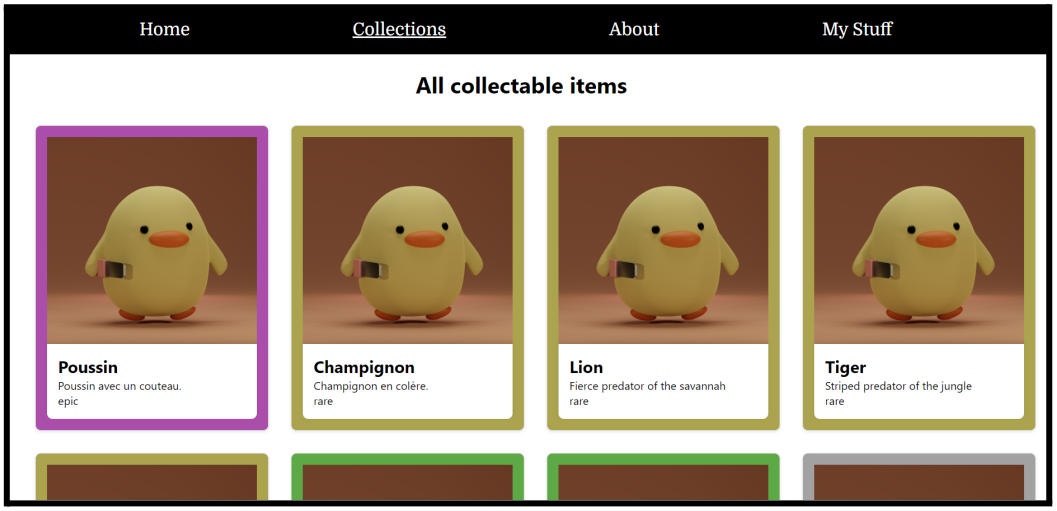


Figure 19 : Page de collection de mon application

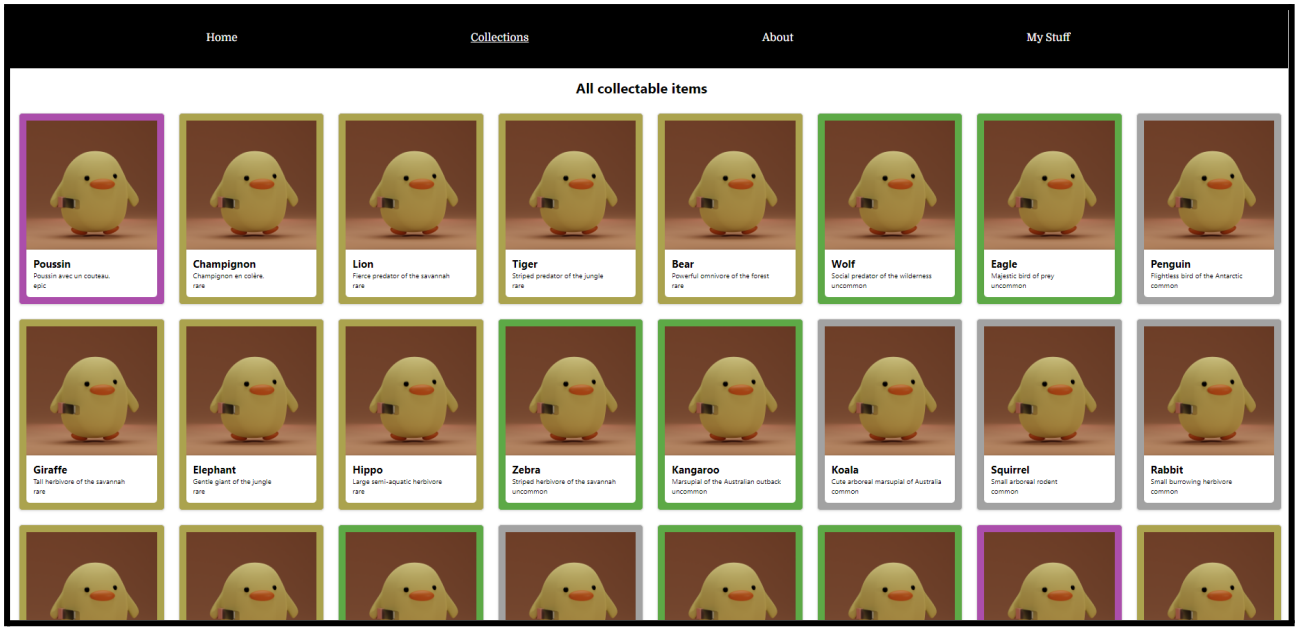


Figure 20 : Page de collection de mon application dézoomé

La page "About" permet à l'utilisateur de connaître la version actuelle de l'application et les modifications qui ont été apportées au fil des versions, notamment le numéro de version, la date de sortie, les nouvelles fonctionnalités et les changements à venir. Cette page inclut également les coordonnées du support du site, afin que les utilisateurs puissent contacter l'équipe de développement en cas de problème et obtenir une assistance efficace.

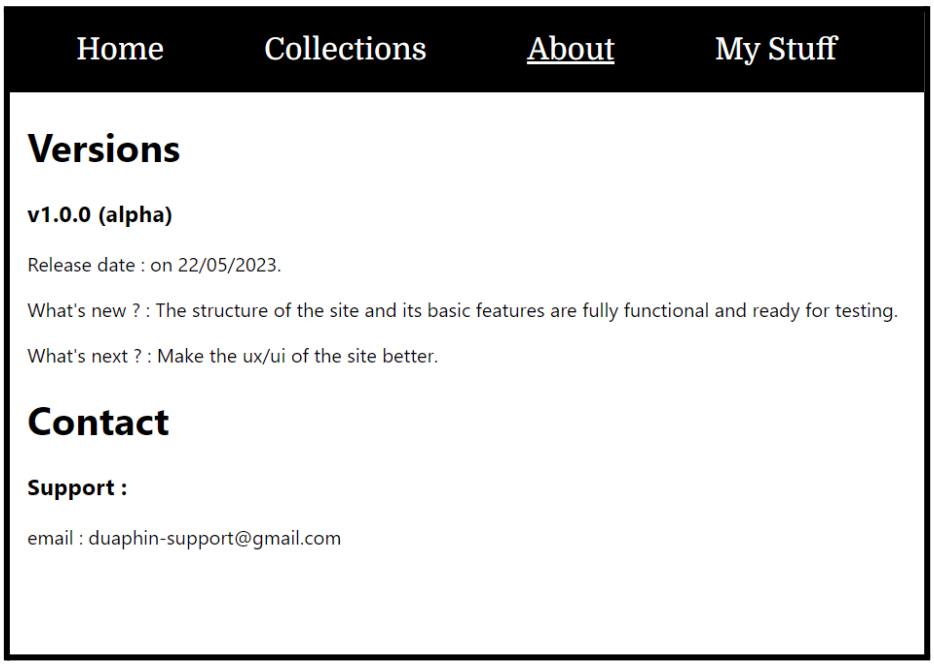


Figure 21 : Page à propos de mon application

La page "Sign in" permet aux utilisateurs de l'application de s'inscrire ou de se connecter à leur compte, en saisissant leur nom d'utilisateur et leur mot de passe. Cette page est essentielle pour accéder aux fonctionnalités de l'application et pour garantir la sécurité des données de l'utilisateur.

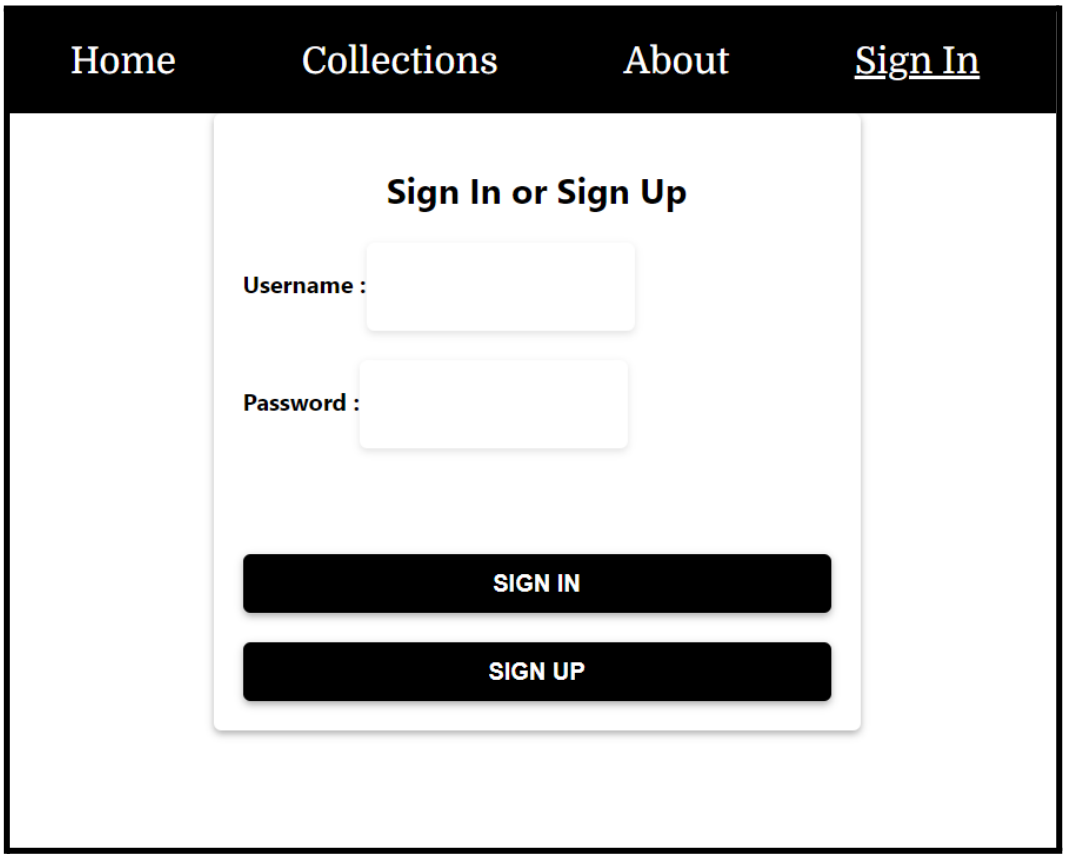


Figure 22 : Page de connexion/inscription de mon application

Lors de l'inscription, l'utilisateur doit fournir un nom d'utilisateur unique. Si le nom d'utilisateur est déjà pris, un message d'erreur s'affiche pour informer l'utilisateur que le nom d'utilisateur n'est pas disponible.

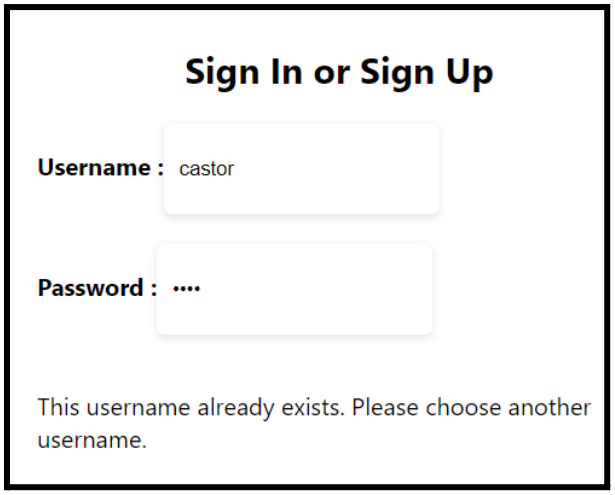


Figure 23 : Tentative d'inscription avec un nom d'utilisateur déjà utilisé

Lors de l'inscription, l'utilisateur ne doit utiliser que des caractères alphanumériques pour son nom d'utilisateur afin d'éviter les attaques XSS*. Si l'utilisateur saisit un nom d'utilisateur contenant des balises de script, telles que <script>, un message d'erreur s'affiche pour l'informer que le nom d'utilisateur n'est pas valide et doit être modifié.

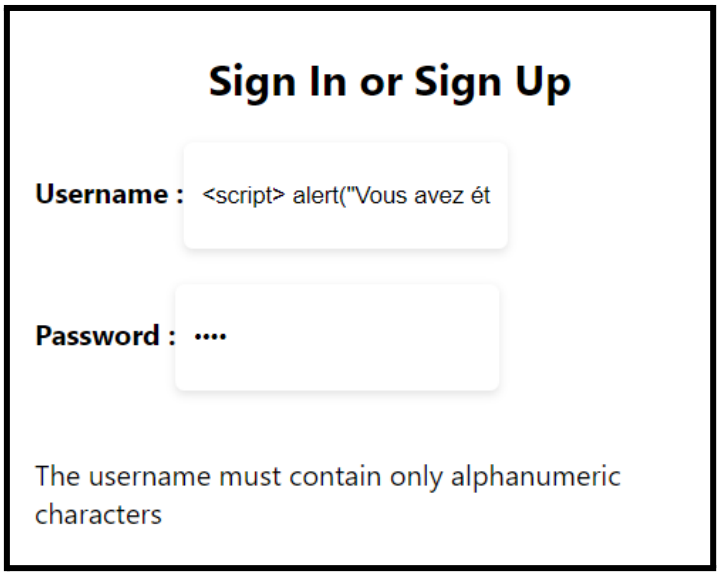


Figure 24 : Tentative d'inscription/connexion avec un nom d'utilisateur qui contient des caractères non valides

Lors de la connexion, des messages d'erreur s'affichent si l'utilisateur oublie de saisir son nom d'utilisateur et/ou son mot de passe, s'il essaie de se connecter à un compte qui n'existe pas ou s'il saisit un mot de passe incorrect. Ces messages d'erreur aident l'utilisateur à comprendre ce qui s'est mal passé et à corriger les erreurs pour pouvoir se connecter avec succès.

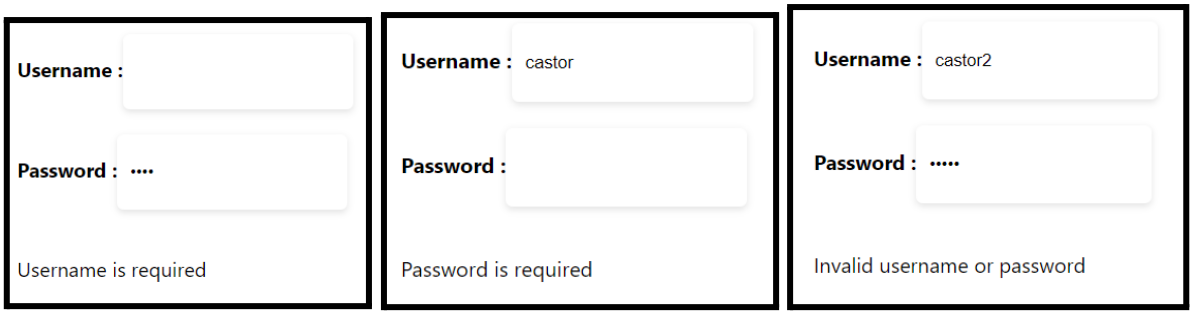


Figure 25 : Tentatives d'inscription/connexion avec un oubli de champs ou avec des informations non valides

Une fois que l'utilisateur s'est connecté avec succès en saisissant son nom d'utilisateur et son mot de passe, le back-end génère un token* unique qui possède une durée de validité limitée. Ce token est une chaîne de caractères qui permet d'identifier de manière unique l'utilisateur et de vérifier ses droits pour chaque action qu'il souhaite effectuer. Par exemple, un utilisateur ne peut pas modifier le compte d'un autre utilisateur, car cela nécessite des droits spécifiques.

Ce token est stocké dans la session de l'utilisateur, ce qui permet de maintenir sa connexion active pendant une période déterminée. Si l'utilisateur se déconnecte ou supprime le cache de son navigateur, la session est automatiquement supprimée, ce qui invalide le token et oblige l'utilisateur à se reconnecter pour accéder à nouveau à son compte.

En résumé, le token permet d'assurer la sécurité de l'application en limitant l'accès aux utilisateurs autorisés et en garantissant la confidentialité des données. La session permet quant à elle de maintenir la connexion de l'utilisateur active pendant une durée limitée et d'assurer que les actions effectuées par cet utilisateur sont bien autorisées.

La page "My Stuff" est l'endroit où les utilisateurs peuvent consulter des statistiques sur leur compte, telles que la date de création du compte et la quantité d'or qu'ils possèdent. Ils peuvent également voir toutes les cartes qu'ils possèdent.

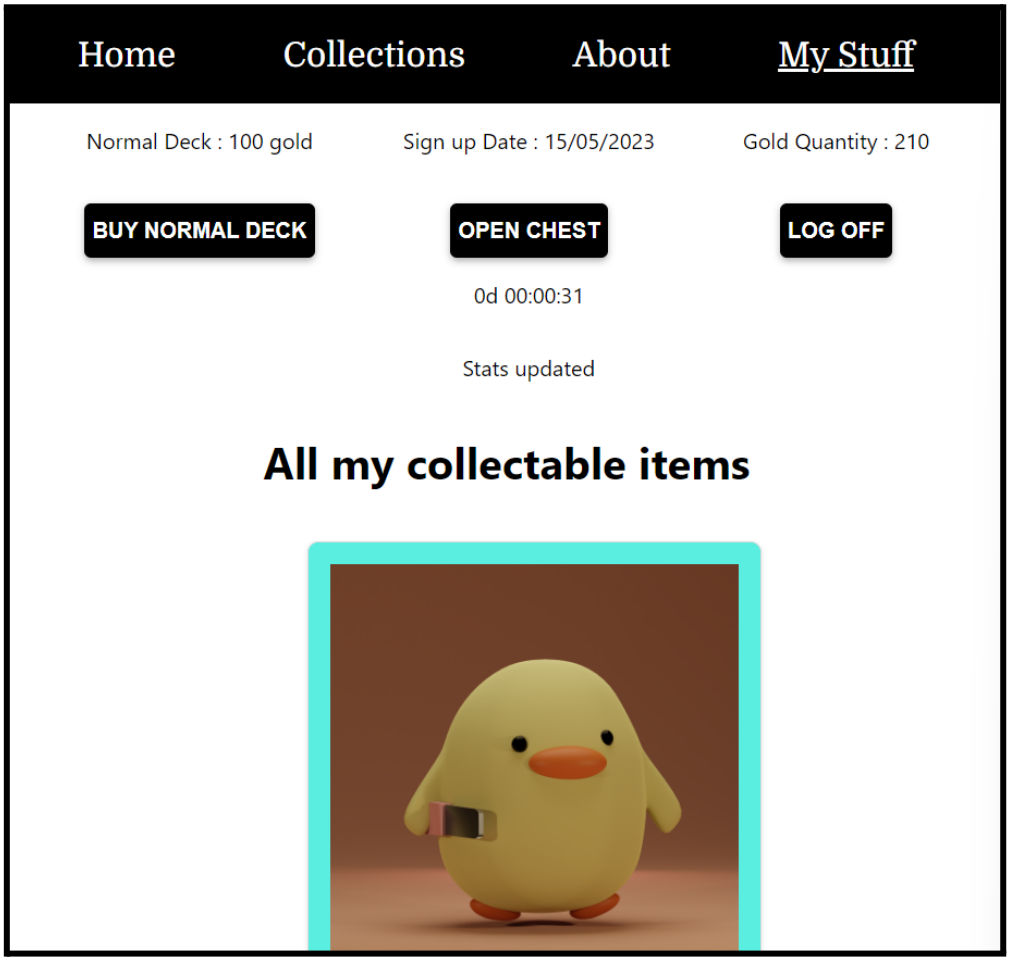


Figure 26 : Page d'information du compte utilisateur

De plus, il y a trois boutons sur la page. Le premier bouton, "Acheter un deck normal", permet aux utilisateurs d'acheter un deck qui contient une carte. Lorsque l'utilisateur clique dessus, le système vérifie d'abord s'il dispose de suffisamment d'or pour effectuer l'achat. Si l'utilisateur peut se le permettre, le système sélectionne de manière aléatoire une rareté à l'aide de la fonction "selectRarity", qui choisit une rareté en fonction du pourcentage d'apparition (rarityWeight).

```
function selectRarity() {
    const rarity = ["common", "uncommon", "rare", "epic", "legendary"]
    //const rarityWeights = [52,26,13,6,3]
    const rarityWeights = [52, 26, 13, 6, 1]
    // const rarityWeightsluxe = [0,0,50,30,20]

    let rd = 0
    // sumArray just make sum of an Array
    rd = Math.floor(Math.random() * sumArray(rarityWeights))

    let index = 0
    while (rd >= 0) {
        rd -= rarityWeights[index]
        index++
    }
    return rarity[index - 1]
}
```

Figure 27 : Fonction permettant de choisir la rareté en fonction de probabilité

Il n'est pas nécessaire que la somme des poids soit égale à 100. Dans ce cas, la somme des poids est de 98, donc par exemple, 52/98 = 53,06 %. J'ai testé cette fonction pour m'assurer qu'elle fonctionne correctement. Voici un exemple du pourcentage d'apparition de chaque rareté après avoir appelé cette fonction 100 millions de fois.

```
common: 53.06%
uncommon: 26.53%
rare: 13.27%
legendary: 1.02%
epic: 6.12%
```

Figure 28 : Pourcentage d'apparition des cartes après 100 millions tirages

Une fois que la rareté est déterminée, le système sélectionne une carte au hasard parmi toutes les cartes qui ont cette rareté. Dans la version alpha actuelle, le système vérifie si l'utilisateur possède déjà cette carte. Si ce n'est pas le cas, la carte est ajoutée à sa collection. Si l'utilisateur possède déjà la carte, il reçoit une compensation en or en fonction de la rareté de la carte.

Toutes ces étapes sont résumées dans le diagramme de séquence que j’ai fait ci-dessous.

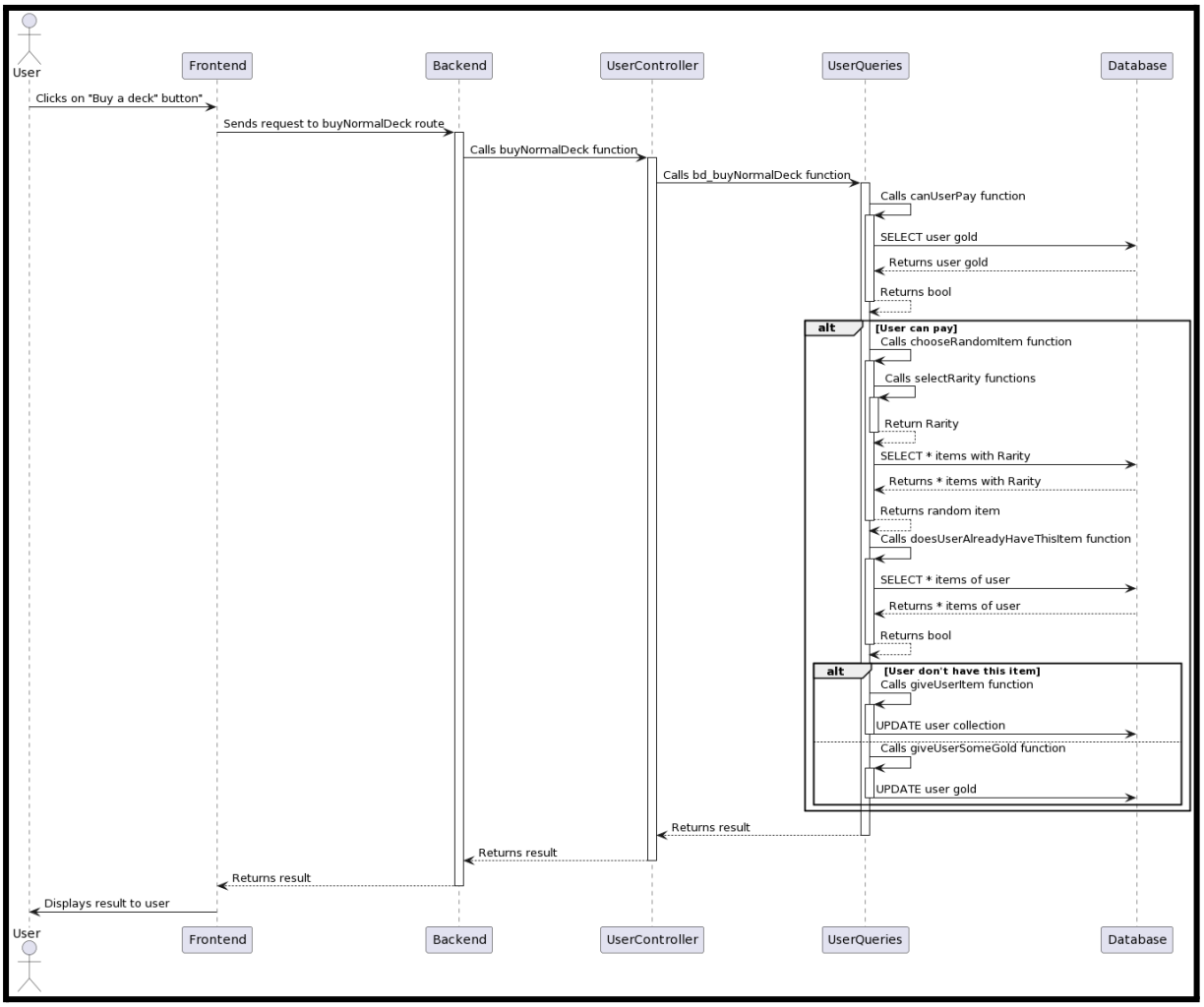


Figure 29 : Diagramme de séquence de la fonctionnalité d’achat de deck

Voyons maintenant une démonstration de ce qu'il se passe pour l'utilisateur avant et après avoir appuyé sur le bouton "Buy Normal Deck".

Sur la première capture d'écran, nous pouvons voir que l'utilisateur possède 220 pièces d'or et qu'il n'a qu'une seule carte dans sa collection.

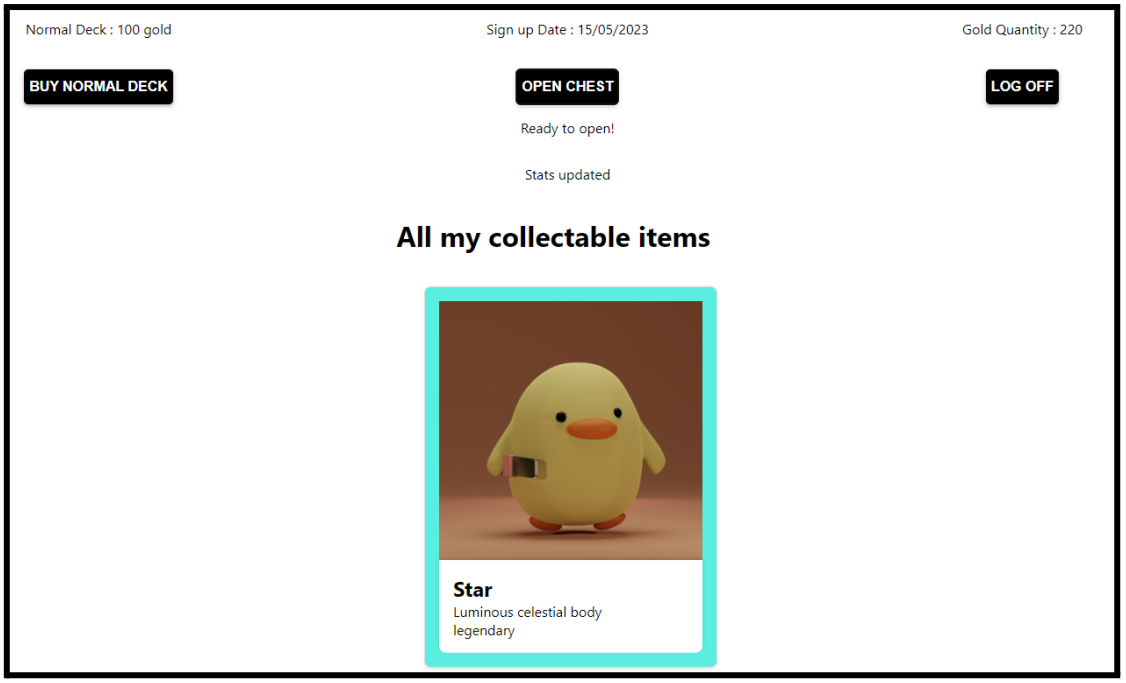


Figure 30 : Page d'information de l'utilisateur possédant une carte dans sa collection

Une fois que l'utilisateur appuie sur le bouton pour acheter un deck normal, nous pouvons constater qu'il a bien dépensé 100 pièces d'or, qui est le prix d'achat d'un deck normal, et qu'il a maintenant une carte supplémentaire dans sa collection.

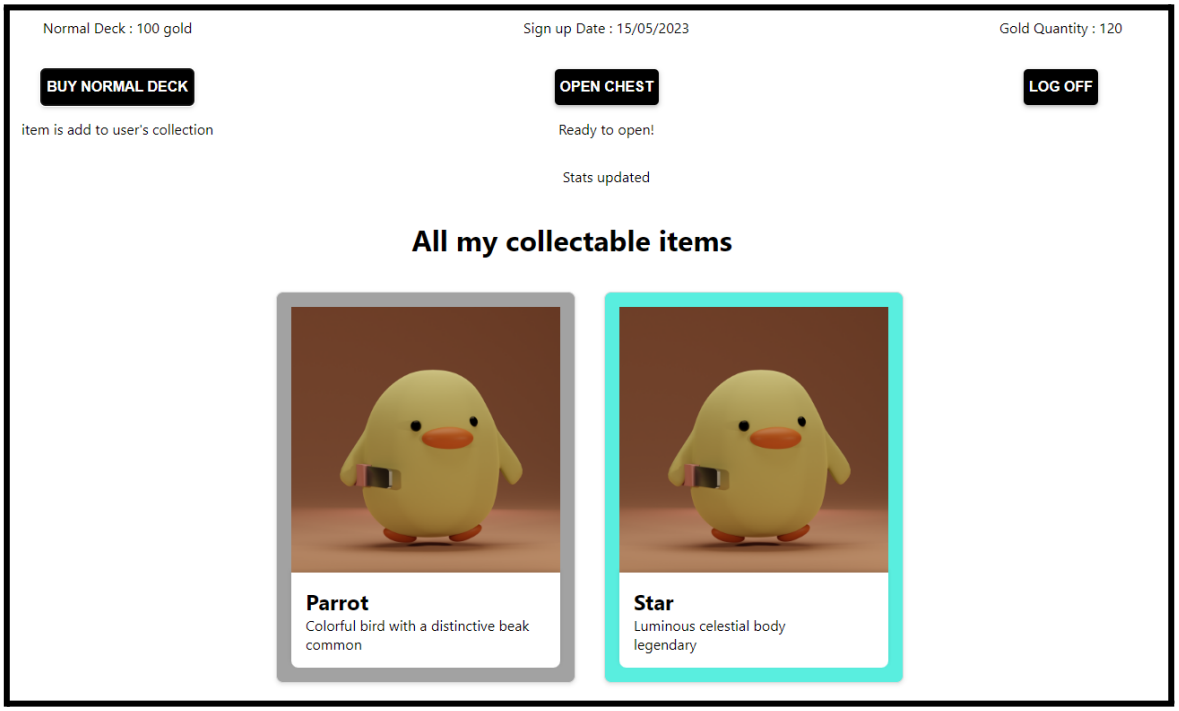


Figure 31 : Page d'information de l'utilisateur possédant deux cartes dans sa collection

Le deuxième bouton, "Open Chest", permet aux utilisateurs de récupérer de l'or une fois que le compte à rebours est terminé. Lorsque le temps d'attente est écoulé, la date à laquelle l'utilisateur pourra ouvrir le coffre à nouveau est mise à jour dans la base de données. L'utilisateur reçoit alors de l'or et le compte à rebours est mis à jour pour indiquer combien de temps, il lui reste à attendre.

Par défaut, le temps d'attente est de 24 heures, mais pour faciliter les tests, il est réduit à une minute. Comme on peut le voir sur les deux captures d'écran ci-dessous, l'utilisateur possède 210 pièces d'or et peut ouvrir son coffre. Une fois qu'il appuie sur le bouton, il reçoit 10 pièces d'or supplémentaires et il lui reste 57 secondes à attendre avant de pouvoir ouvrir le coffre à nouveau.

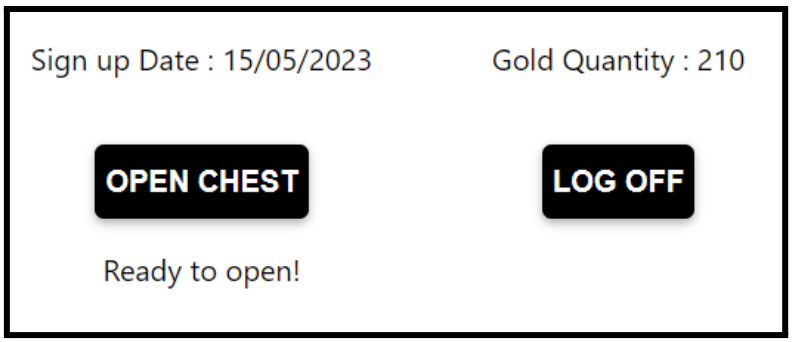


Figure 32: Page d'information de l'utilisateur pouvant ouvrir son coffre

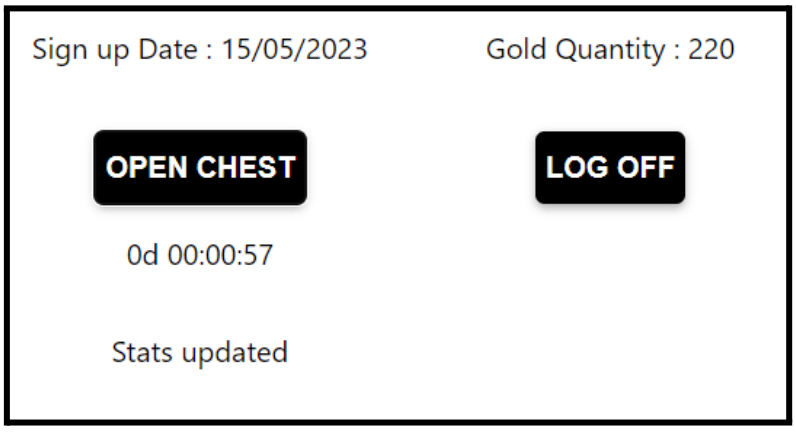


Figure 33 : Page d'information de l'utilisateur devant attendre encore 57 secondes avant de pouvoir ouvrir de nouveau son coffre

Cette fonction permet de mettre à jour le compte à rebours toutes les secondes en prenant en paramètre la date à laquelle l'utilisateur pourra ouvrir le coffre. Ainsi, l'utilisateur peut voir en temps réel combien de temps, il lui reste à attendre avant de pouvoir ouvrir le coffre.

Il est important de noter que même si l'utilisateur tente d'ouvrir le coffre avant la fin du compte à rebours, il y a une vérification supplémentaire dans le back-end qui empêche toute tentative de contournement. Cela garantit que l'utilisateur ne peut pas ouvrir le coffre avant que le temps imparti ne soit écoulé.

```
function useCountdown(lastChestOpened) {
    // Initialize countdown state to empty string
    const [countdown, setCountdown] = useState('');···
            // Update countdown state with new time left value
            setCountdown(timeLeft);
        }, 1000);
        // Clean up interval when component unmounts
        return () => {
            clearInterval(interval);
        };
    }, [lastChestOpened]);
    // Define function to calculate time left until next chest can be opened
    const calculateTimeLeft = useCallback((date) => {
        // Calculate time remaining until next chest can be opened
        const timeRemaining = new Date(date) - Date.now();
        if (timeRemaining <= 0) {
            // If time remaining is less than or equal to zero, return "Ready to open!"
            return 'Ready to open!';
        } else {
            // Otherwise, calculate days, hours, minutes, and seconds left and return formatted string
            const days = Math.floor(timeRemaining / (1000 * 60 * 60 * 24));
            const hours = Math.floor((timeRemaining / (1000 * 60 * 60)) % 24);
            const minutes = Math.floor((timeRemaining / (1000 * 60)) % 60);
            const seconds = Math.floor((timeRemaining / 1000) % 60);
            return `${days}d ${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2, '0')}:${seconds.toString().padStart(2, '0')}`;
        }
    }, []);

    // Return countdown state
    return countdown;
}
```

Figure 34 : Fonction qui permet de calculer le temps qu'il reste avant de pouvoir ouvrir un coffre

Le dernier bouton, "Log Off", permet à l'utilisateur de se déconnecter de son compte et de supprimer sa session. En appuyant sur ce bouton, l'utilisateur sera renvoyé à l'écran d'accueil où il pourra se reconnecter plus tard s'il le souhaite. Cette fonctionnalité est utile pour garantir la sécurité de ses données et pour éviter que d'autres personnes accèdent à son compte s'il partage son appareil avec d'autres utilisateurs.

Il est important de noter que pour assurer la sécurité de l'application, le token qui permet de faire les appels aux fonctions à une durée de validité limitée. Même si l'utilisateur laisse sa session ouverte pendant une période prolongée, le token deviendra inutilisable après un certain temps, ce qui empêchera toute personne non autorisée d'accéder à son compte. Par conséquent, il est recommandé aux utilisateurs de se déconnecter régulièrement de leur

compte, même s'ils quittent l'application pendant une courte période, afin de garantir la sécurité de leurs données.

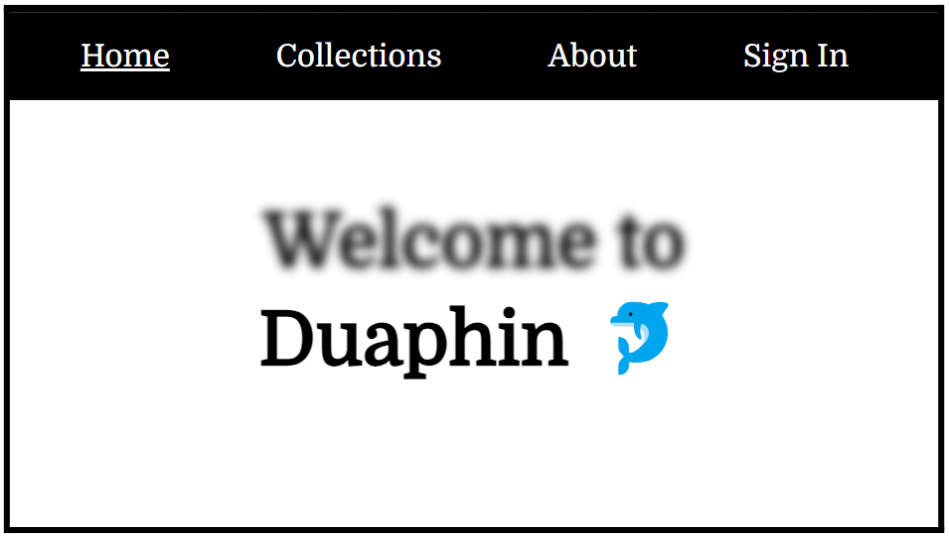


Figure 35 : Page d'accueil après que l'utilisateur se soit déconnecté

3.3 Déploiement

Pour déployer mon application en production, j'ai utilisé deux outils : Docker et Vercel.

Docker m'a permis de créer des images, qui sont des fichiers contenant tout le nécessaire pour faire tourner mon application. Ces images peuvent ensuite être déployées sur des serveurs, tels que des instances EC2, pour que l'application soit accessible en ligne.

Quant à Vercel, il m'a permis de mettre en ligne à la fois le front-end et le back-end de mon application directement depuis mon dépôt Github.

Dans la suite de ce texte, je vais détailler le processus de création d'image Docker que j'ai utilisé pour déployer mon application en production.

Du projet à Docker Desktop :
Pour créer une image Docker de notre projet, nous devons rédiger un fichier nommé Dockerfile à la racine de celui-ci.

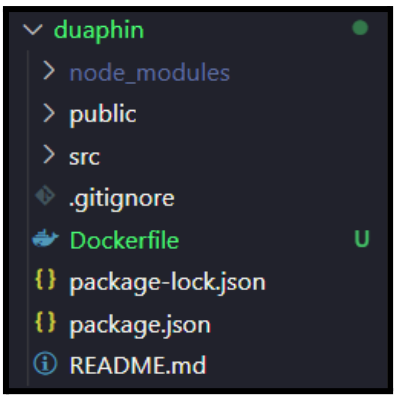


Figure 36 : Structure de mon code avec le Dockerfile

Le Dockerfile contient les instructions nécessaires à la construction de l'image. Voici le contenu du Dockerfile que j'ai utilisé pour créer l'image de mon application :

```
duaphin > Dockerfile
1   # Set the base image for this Docker image
2   FROM nginx:1.21-alpine
3
4   # Copy all files from the current local directory to the root directory of the Docker image
5   COPY . .
6
7   # Expose port 80 to be able to access the application via web browser
8   EXPOSE 80
9
10  # Set the default command to run when the container is started
11  CMD ["nginx", "-g", "daemon off;"]
```

Figure 37 : Contenu du Dockerfile

Une fois que le fichier Dockerfile est créé, nous pouvons créer l'image Docker de notre application en exécutant la commande suivante dans le terminal :

docker build -t image-duaphin-front-v1 .

Cette commande permet de construire une nouvelle image Docker nommée "image-duaphin-front-v1" à partir des instructions contenues dans le fichier Dockerfile.

Une fois que l'image est créée, elle est visible dans l'application Docker Desktop, où nous pouvons la gérer et la déployer selon nos besoins.

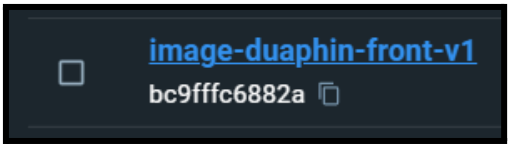


Figure 38 : Image Docker visible sur Docker Desktop

De Docker Desktop à Docker Hub :

Une fois que j'ai créé mon image, je peux me connecter à Docker Hub en utilisant la commande "**docker login**". Une fois connecté, je dois renommer mon image en ajoutant mon nom d'utilisateur. Pour cela, j'utilise la commande "**docker tag**" :

docker tag image-duaphin-front-v1 pampx/image-duaphin-front-v1

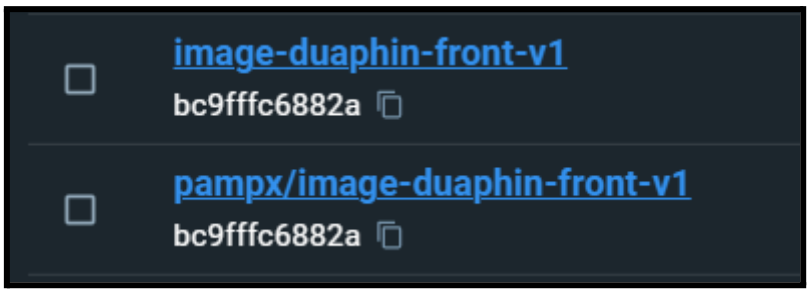


Figure 39 : Image Docker renommé visible sur Docker Desktop

Ensuite, je dois publier mon image sur Docker Hub en utilisant la commande "**docker push**". Cela permettra à d'autres utilisateurs de Docker de télécharger et d'utiliser mon image :

docker push pampx/image-duaphin-front-v1

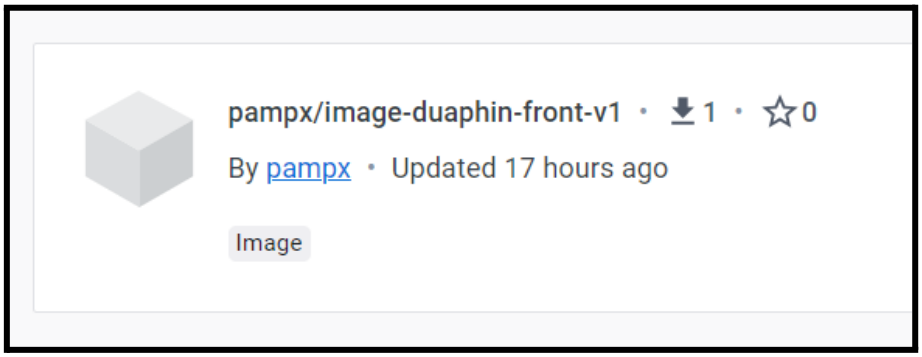


Figure 40 : Image Docker renommé visible sur Docker hub

De Docker Hub au server :
Imaginons que je possède un serveur EC2. Une fois connecté au serveur et après avoir installé Docker, je peux me connecter à Docker Hub en utilisant la commande "docker login". Ensuite, je peux copier l'image Docker depuis Docker Hub sur mon serveur en utilisant la commande suivante :

docker pull pampx/image-duaphin-front-v1

Une fois l'image copiée, je peux l'exécuter en utilisant la commande "**docker run**". Dans cette commande, le paramètre "-p" permet de mapper le port de l'image Docker avec celui de mon serveur. L'ID de l'image que je veux exécuter peut être obtenu en utilisant la commande "**docker images**". Voici un exemple de commande pour exécuter l'image :

docker run -p 8080:8080 a8811995519d

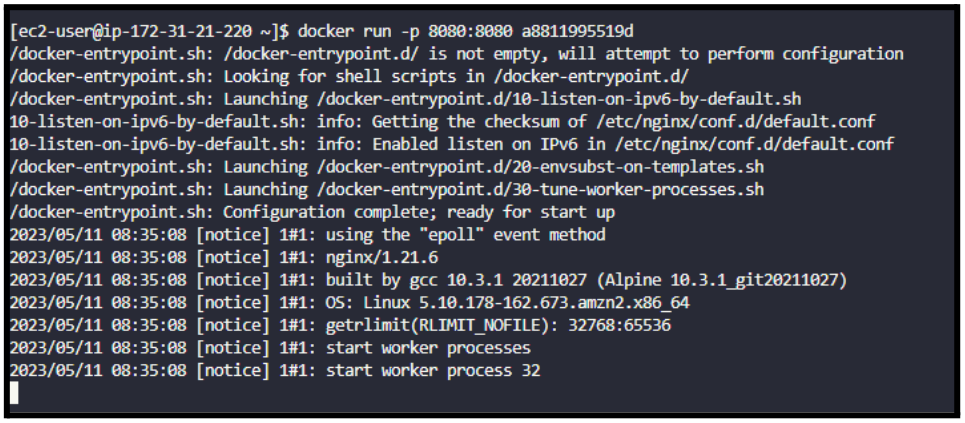


Figure 41 : Lancement de l'image Docker sur le serveur EC2

Enfin, pour tester l'application, je peux accéder à l'adresse IP publique de mon serveur avec le port choisi (dans cet exemple, le port est 8080) en utilisant un navigateur web. Par exemple :

http://15.237.124.135:8080/

4. Gestion de projet

Conformément aux exigences énoncées dans le cahier des charges, la planification initiale pour la livraison de la version alpha de ce projet était de deux mois. Ce délai a été respecté grâce à une répartition efficace des tâches d'apprentissage et de mise en pratique des différentes technologies nécessaires pour le projet.

Le processus de développement a débuté par la rédaction du cahier des charges détaillant toutes les idées que j'avais pour ce projet. Ensuite, j'ai recherché les technologies les plus appropriées pour la réalisation de ces idées.
Afin de versionner mon code, j'ai utilisé GitHub, une plateforme de gestion de code source très populaire. J'ai créé un référentiel pour le back-end et un pour le front-end du projet et j'ai régulièrement sauvegardé mon code pour pouvoir y accéder à tout moment et suivre son évolution au fil du temps.
Une fois que j'ai identifié les outils nécessaires ou des pistes à explorer, j'ai créé des maquettes et effectué une conception préliminaire pour clarifier la structure du projet.

J'ai ensuite commencé par le développement du back-end et de la base de données, avant de passer au front-end et enfin au déploiement. Ce processus a été méthodiquement planifié et exécuté dans les délais impartis pour assurer la livraison réussie de la version alpha du projet.

Tout au long du processus de développement, j'ai effectué des tests sur mon code et j'ai utilisé Jest, un framework de test très populaire, pour tester l'API du projet. Cela m'a permis de m'assurer que le code était fonctionnel, d'identifier rapidement les erreurs et de les corriger avant leur déploiement. De plus, les tests ont permis d'améliorer la qualité du code et de s'assurer que l'API répondait correctement aux différentes requêtes.
Puisque je n'avais pas d'autres membres dans mon équipe, la communication s'est avérée relativement aisée.

5. Cahier des charges

5.1 Introduction

Afin de répondre à la dernière compétence requise pour l'obtention du diplôme de CDA, j'ai choisi de présenter un projet que j'ai réalisé : une application mobile d'informations météorologiques.

Ce projet a été réalisé dans le cadre de mes cours de programmation mobile, dans le but de mettre en pratique mes connaissances et compétences en développement d'applications pour appareils mobiles.

5.2 Contexte et objectif

Ce projet d'application météorologique vise à fournir à l'utilisateur des informations en temps réel pour une ville spécifique. Développée en utilisant Java, cette application intègre l'API de OpenWeatherMap. Une fois que l'utilisateur saisit la ville souhaitée, les informations telles que la température, l'état du ciel et la vitesse du vent sont affichées instantanément, offrant ainsi une expérience météorologique précise et pratique.

6. Rapport technique

6.1 Réalisation

La première page de l'application comporte un champ de texte (voir figure 42) et un bouton. Lorsque l'utilisateur saisit le nom d'une ville et appuie sur le bouton "Voir Météo", l'application effectue un appel à l'API d'OpenWeatherMap (voir figure 43) afin d'obtenir les informations météorologiques correspondantes.

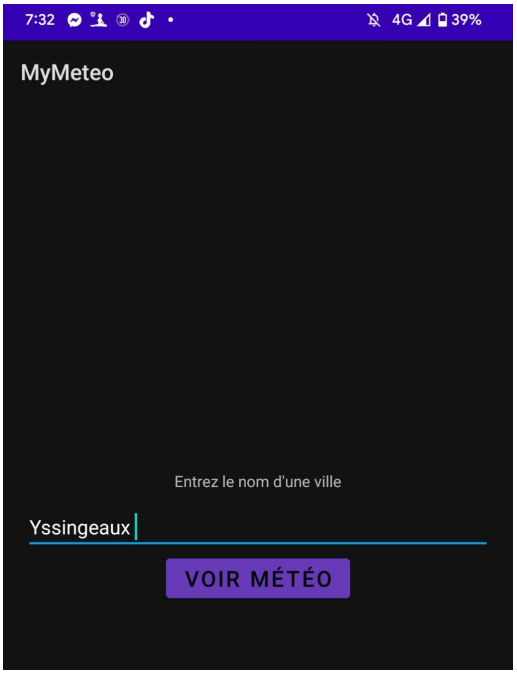


Figure 42: Page d'accueil de l'application météo

```java
public class MeteoAPIConnector {

    1 usage
    private static AsyncHttpClient client = new AsyncHttpClient();

    1 usage
    public static void getMeteo(String ville, Consumer<JSONObject> callback) {
        String url = "http://api.openweathermap.org/data/2.5/weather?q=" + ville + "&APPID=8fe644634cf77a9cc9b6698ec75c2f6b&lang=fr&units=metric";
        client.get(url, (JsonHttpResponseHandler) onSuccess(statusCode, headers, response) -> {
                callback.accept(response);
        });
    }
}
```

Figure 43 : Classe MeteoApiConnector qui permet de faire l'appel à l'API

Une fois que l'utilisateur appuie sur le bouton "Voir Météo", la page d'affichage des informations météo est chargée (voir figure 44). La fonction "updateMeteo" (voir figure 45) est alors utilisée comme rappel (callback) pour traiter et afficher le résultat de l'appel à l'API.

```
public class MeteoActivity extends AppCompatActivity {

    2 usages
    private TextView villeName;
    2 usages
    private TextView vent;
    2 usages
    private TextView temps;
    2 usages
    private TextView degres;
    4 usages
    private String ville;
    2 usages
    private Button boutonRetour;
    private MeteoAPIConnector meteo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_meteo);
        Intent intent = getIntent();
        if (intent.hasExtra( name: "villeName")){
            ville = intent.getStringExtra( name: "villeName");
            System.out.println(ville);
        }
        villeName = findViewById(R.id.textville);
        villeName.setText(ville);
        boutonRetour = findViewById(R.id.boutonRetour);
        boutonRetour.setOnClickListener(this::retourAuMenu);
        vent = findViewById(R.id.textvents);
        temps = findViewById(R.id.textweather);
        degres = findViewById(R.id.texttemp);
        MeteoAPIConnector.getMeteo(ville, this::updateMeteo);
    }

    1 usage
    private void retourAuMenu(View view) {
        Intent intentExplicite = new Intent( packageContext: this, MainActivity.class
        this.startActivity(intentExplicite);
    }
```

Figure 44 : Classe MeteoActivity qui permet d'afficher les informations météo d'une ville

```
1 usage
public void updateMeteo(JSONObject map) {
    try {
        //Temperature de Montpellier
        //La temperature est dans un sous-objet "main"
        JSONObject main = map.getJSONObject( name: "main");
        //La température est un nombre à virgules
        double temperature = main.getDouble( name: "temp");
        JSONArray weather = map.getJSONArray( name: "weather");
        JSONObject wind = map.getJSONObject( name: "wind");
        JSONObject objWeather = weather.getJSONObject( index: 0);
        String vent = wind.getString( name: "speed");
        this.vent.setText(vent);
        String temps = objWeather.getString( name: "description");
        this.degres.setText(Double.toString(temperature) + "°C");
        this.temps.setText(temps);
    }
    catch(JSONException e) {
        e.printStackTrace();
    }
}
```

Figure 45 : Fonction UpdateMeteo qui permet de traiter les informations de l'API

La fonction "updateMeteo" récupère les identifiants des éléments de la page et affiche les informations météorologiques correspondantes, y compris le nom de la ville, la température actuelle, l'état du ciel et la vitesse du vent (voir figure 46).

Figure 46 : Page d'affichage des informations météo de la ville

7. Conclusion

Le cahier des charges initial prévoyait la création d'une version alpha fonctionnelle d'une application de collection de modèles 3D. Aujourd'hui, cette application est opérationnelle et permet à un visiteur de découvrir tous les modèles disponibles dans la collection, ainsi que de créer un compte pour commencer sa propre collection. Le projet a été divisé en trois couches distinctes : la base de données, le back-end et le front-end, chacune étant développée dans un langage de programmation spécifique.

La base de données a été développée en utilisant MySQL et l'interface PhpMyAdmin avec l'aide de MAMP. Elle respecte les règles de sécurité et les formes normales pour garantir une stabilité et une sécurité optimale.

Le back-end a été développé en utilisant Node.js et suit le modèle MVC pour dissocier les différentes couches. L'API développée permet de relier l'interface utilisateur à la base de données. Des tests ont été effectués pour assurer une structure stable et une expérience utilisateur fluide.

Enfin, le front-end a été développé en utilisant React.js pour offrir une expérience dynamique et agréable aux utilisateurs.

Des améliorations seront apportées à la version alpha de ce projet, car j'ai découvert lors de sa création Node Js et React Js et le code va devoir être factorisé pour être plus clair et optimisé. Pour la suite de ce projet, je dois modéliser un premier échantillon de modèle, puis ajouter les fonctionnalités manquantes à l'application, telles que l'échange de cartes et l'achat de différents decks.

Une fois toutes les fonctionnalités ajoutées, l'application sera un lieu de rencontre et de création pour toute personne s'intéressant à la 3D ou souhaitant simplement collectionner avec ses amis. Elle offrira une plateforme conviviale et intuitive pour échanger, acheter et collectionner différents modèles 3D, et permettra aux utilisateurs de découvrir de nouveaux designs et de partager leur passion pour la création 3D avec une communauté en ligne dynamique.

Le projet d’application mobile m'a permis de développer une application en Java et d'utiliser une API pour fournir aux utilisateurs des informations météorologiques sur la ville de leur choix. Bien que ce soit un projet de petite envergure, il m'a offert une expérience précieuse dans la création d'applications et l'intégration d'API, renforçant ainsi mes compétences en développement mobile.