Projet Web Avancé 4A CFA FSI2

Membres

- Alexandre Cimmonard
- Corentin Ferrier
- Etienne Amiche

Lien du projet

> https://parc-jurassique.glitch.me

Respect des contraintes techniques

Front End

- VueJs et Vue CLI
- Vuetify
- Vue Router pour la navigation
- VueX pour la gestion des variables dans un store
- Axios
- Système d'authentification d'utilisateurs

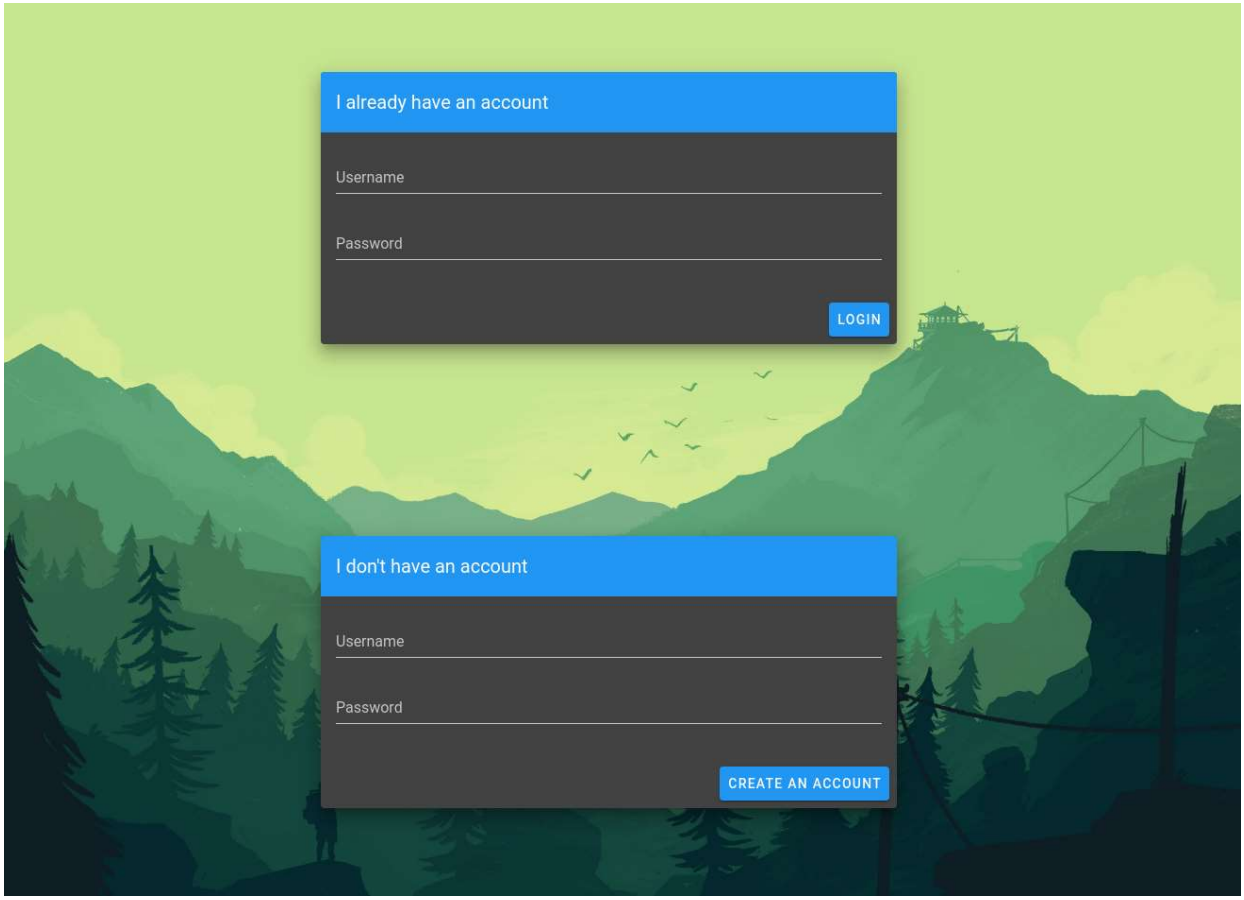
Backend

- Serveur Express hebergé sur glitch
- CRUD
- Interaction front-back

Description du projet

Ce projet de programmation WEB est un jeu de gestion sur le thème du jurassique et inspiré du film Jurassic Parc. Le but de ce jeu est de faire évoluer ses ressources en gérant plusieurs biens, sans objectifs précis à atteindre.

Après l'ouverture du lien, une page de connexion s'affiche avec deux formulaires. L'un servant à se connecter, l'autre pour la création d'un compte. Ainsi l'exigence « Authentification de l'utilisateur » est satisfaite.



Des identifiants par défaut sont disponibles au cas où.

utilisateur: **admin**

mot de passe: **pass**

Des resources

- Argent

> L'argent permet d'acheter des biens

- Visiteurs

> Plus le nombre de visiteurs est grand plus vos magasins feront de bénéfice et apporterons de l'argent

- Niveau de danger

> Pourcentage indiquant le risque dans le parc (si le danger est à 5%, toute les 10 secondes, il y a 5% de chance pour qu'un accident arrive et qu'un nombre aléatoire de visiteurs quitte le parc)

Des biens

- Les dinosaures

> Vous pouvez acheter différents types de dinosaures plus ou moins cher. Plus le dinosaure est cher, plus il attire de visiteurs, mais en contre partie plus le danger augmente.

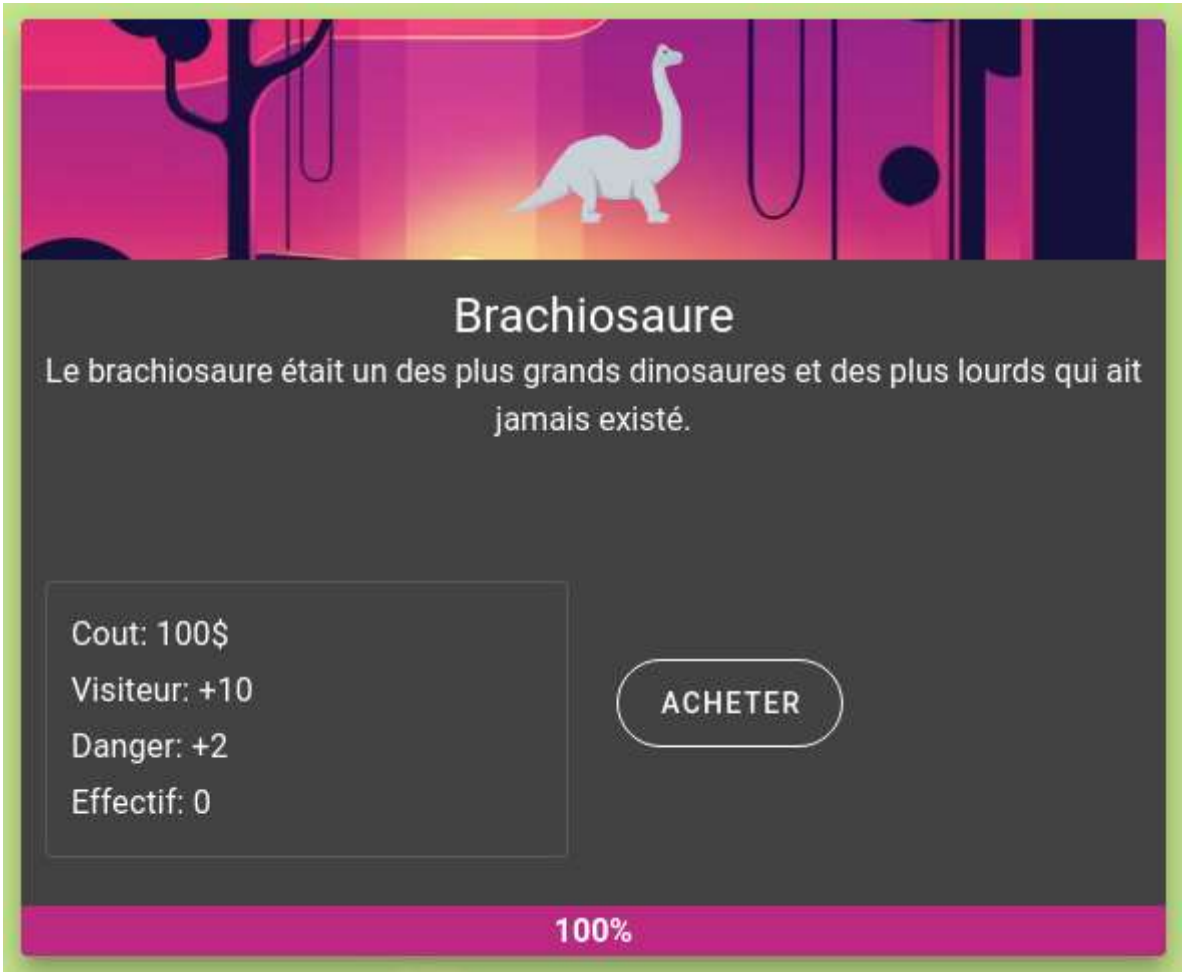
- Les magasins

> Les magasin permet d'apporter de l'agent au parc toute les seconde en fonction du nombre de visiteurs. Plus un magasin est cher et plus il rapport d'argent.

- Le personnel

> Le personel permet de faire baisser le niveau de danger dans le parc. Embauchez du personel qualifié pour assurer la sécurité des visiteurs.

Par exemple:



Chaque bien est composé des mêmes éléments.

- Une image
- Le nom
- Une description
- Le prix (le prix augmente en fonction de l'effectif)
- l'impact (ici +2% de danger dans le parc)
- l'effectif (le nombre possedé)
- Un bouton *acheter*
- une barre de poucentage qui indique le prix en fonction de l'argent possedé (si il y a assez d'argent le pourcentage indique 100%)

Information pour le joueur

Lorsque l'on n'a plus suffisamment de monnaie pour acheter ou quand un incident arrive à cause du danger, il faut avertir l'utilisateur. Pour cela, un composant générateur d'alerte lancera un message d'alerte en fonction de l'évènement (perte de visiteurs, manque de ressources). Si l'événement est une perte de visiteurs, une fonction choisira la cible des dinosaures.

Composant UI utilisés

- Bouton
- Cards
- List
- Image
- Progress Bar
- NavBar
- Alert

Étapes et difficultés

Nous avons commencé par concevoir le jeu sur papier afin d'avoir une première idée du résultat final. Ensuite, nous avons mis en place Le dépôt git et créer l'application vue a l'aide de la CLI.

Nous avons ensuite mis en place le routeur VueRouter ainsi que le store VueX. Par la suite, nous avons créer les différentes cartes a l'aide de composants et créer des fonction pour modifier les valeur du store. Nous avons ensuite créer un fichier game_data qui contient toutes les constantes du jeu comme les prix de base, les descriptions, les images… afin des les injecter dans les trois listes de composant et pouvoir par la suite ajuster le jeu plus facilement.

Nous avons choisi de respecter la méthode gitflow. Chaque branche correspond à une fonctionnalité, une modification ou un debuggage. La branche est mergée sur master quand la modification apportée fonctionne.

Après avoir réalisé plusieurs test de design et réaliser les ajustement du jeu, nous avons mis en place le serveur et l'authentification des utilisateur.

Enfin pour respecter le CRUD nous avons mis en place un système de récupération et de sauvegarde des données de l'utilisateur. Cette étape a été compliquée car il a fallu créer un utilisateur avec un objet contenant toutes les variables du jeu propre au joueur, injecter ces données dans le store et pour la sauvegarde, transférer les données sur le serveur.

Enfin la mise en production a été légèrement compliquée à cause de problème d'arborescence et certain oublis (par exemple les lien d'api, la session…) Nous avons cependant réussi à mettre en production.