Projet - Application frontale

Sur les 2 prochaines séances de TP, vous allez travailler sur le projet, binôme ou trinôme. Vous allez développer une application Vue.js du jeu “c’est plus / c’est moins” en utilisant l’ensemble des connaissances acquises durant les précédents TPs.

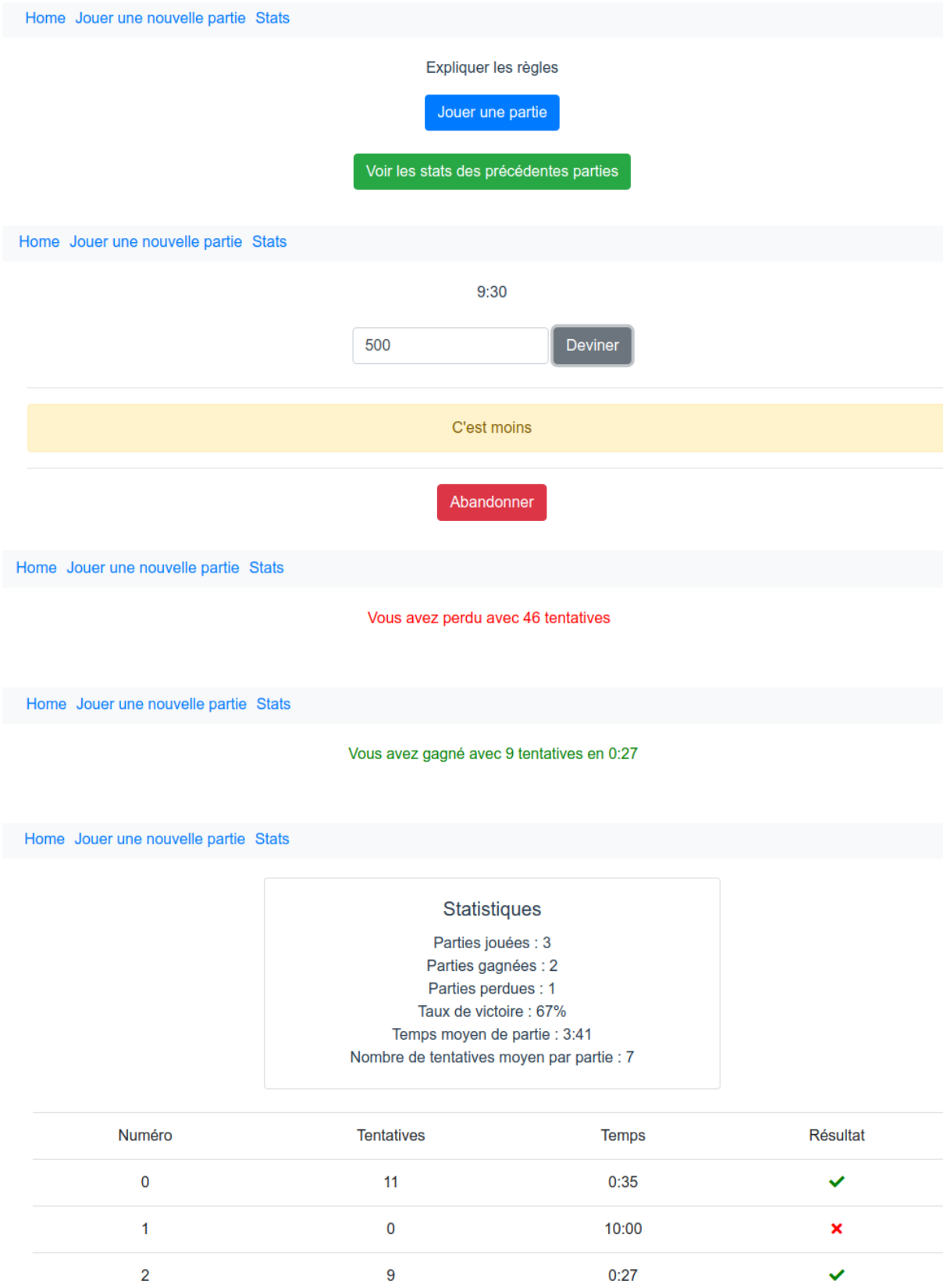
Note : pour rappel, ce projet comptera pour 30% de la note du module application frontale.

Démarrage du projet avec Git

- Utilisez le lien fourni sur Moodle pour générer un dépôt privé correspondant au TP, hébergé dans l'organisation https://github.com/UE-FRONTEND ;
- Clonez le dépôt GitHub ainsi obtenu ;
- Ajoutez les noms et prénoms des membres de votre projet ainsi que le groupe de TP dans le fichier *README.md* ;
- Créez un commit dans *master* pour ce changement et poussez le vers le dépôt distant.
- Prenez connaissance de la structure du projet avec IntellIJ ;
- Faites la commande `npm install` à la racine du dossier
- Lancez votre application à l’aide de la commande `npm run serve`

Note : il y a du livre reload, ce qui signifie que vous n’avez pas à redémarrer votre application à chaque modification : vous les verrez directement apparaître dans votre navigateur.

Voici un exemple de rendu possible à la fin du projet :



Documentation technique

Ce jeu utilisera un web service, exposé sous forme d'API Rest. Ce service est joignable à l'url suivante : https://vuejs-rest-challenge.herokuapp.com/
C'est un service hébergé sur la plate-forme Heroku ; il se peut que la première requête soit un peu longue car il faut le temps que l'application démarre. Pour les curieux, vous pourrez trouver le code source du serveur ici.

Voici une description de l'API exposée par l'API Rest au format JSON :
Vous avez 2 URI possibles :

- **/token (Méthode GET) :**

  Permet de récupérer un **token**.

  Exemple : `{"id":1,"token":"31357799-5c97-422a-8168-13a50fbab74f"}`

  Le token a une durée de vie de 10 minutes et permet d'identifier le nombre à deviner. Si vous souhaitez changer de nombre à deviner, il vous faudra demander un nouveau token.

- **/try (Méthode POST) :**
  - Body de la requête (format JSON) :
    - **token** : le token précédemment récupéré
    - **guess** : le nombre proposé par l'utilisateur, au format integer (number sans virgules pour du Javascript). Ce nombre doit être compris entre 0 et 1000.

    Exemple : `{token: "31357799-5c97-422a-8168-13a50fbab74f", guess: 10}`

  - Body de la réponse (format JSON) :
    - **code** :
      - 0 => Bonne valeur
      - -1 => C'est moins
      - 1 => C'est plus

    Exemple: `{"code":1}`

Pour faire jouer l'utilisateur, il faudra donc appeler **/token** en 1er pour récupérer un token permettant d'identifier le nombre à deviner. Ensuite, pour chaque tentative de l'utilisateur, il faudra faire un appel à **/try** en renseignant le token précédemment récupéré.
Si le token n'existe pas (ou plus si ça fait plus de 10 minutes) lors de l'appel à **/try**, alors le serveur retournera une erreur 400 de type BAD_REQUEST.

Exigences fonctionnelles

1) Lorsqu'on arrive sur la page principale (path = /), vous devez présenter une page indiquant :
    a) Vos noms et prénoms + groupe de TP
    b) Un bouton démarrer une partie
    c) Un bouton consulter les précédentes parties

2) Lorsque l'utilisateur clic sur le bouton "démarrer une partie" (exigence 1b), alors changer l'affichage pour afficher l'écran de jeu. Sur celui-ci, nous pouvons y voir :
    a) Un chronomètre qui indique le temps restant
    b) Un champ texte permettant à l'utilisateur de saisir un nombre
    c) Un bouton permettant à l'utilisateur de valider sa réponse
    d) Un bouton permettant à l'utilisateur d'abandonner

    Aide : Dès que l'écran est chargé, faites une requête pour récupérer un token de jeu et lancez le chronomètre (méthode `setInterval` avec un pas de 1 seconde, pensez à bien arrêter ce setInterval).
    Utilisez l'action sur l'état "mounted" d'un composant pour effectuer ces opérations.

3) Lorsque l'utilisateur clic sur le bouton valider sa réponse (exigence 2c), alors faire une requête à l'API Rest pour savoir si c'est le bon nombre.
    a) Si c'est le bon nombre, alors afficher une page de "victoire" indiquant le nombre de tentatives et le temps écoulé pour trouver la bonne réponse. Sur cette page, afficher un bouton permettant de revenir à la page de l'exigence 1.
    b) Si c'est le mauvais nombre, alors affichez une bulle d'aide permettant de guider l'utilisateur, de type "C'est plus" ou "C'est moins"

4) Lorsque le chronomètre atteint 0 secondes restantes ou que vous rencontrez une erreur 400 BAD_REQUEST, ou que l'utilisateur clic sur le bouton "abandonner la partie" (exigence 2d), alors affichez une page de "défaite" indiquant le nombre de tentatives. Sur cette page, faire un bouton permettant de revenir à la page de l'exigence 1.

5) Lorsque l'utilisateur clic sur le bouton de l'exigence 1c, affichez une page récapitulant les différentes parties jouées. Cette page doit avoir une partie statistiques, avec :
    a) Le temps moyen de jeu
    b) Le nombre de tentatives moyennes
    c) Le % de victoire

6) La page de l'exigence 5 doit également afficher l'ensemble des parties jouées. Chaque partie jouée peut être vue comme une ligne d'un tableau, avec les informations suivantes :
    a) Le numéro de la partie
    b) Le nombre de tentatives
    c) Le temps de jeu
    d) Un indicateur pour savoir si l'utilisateur a réussi ou non

Exigences techniques

L'application doit être développée en Vue.js.
Vous devez utiliser les dépendances suivantes :

- vue-router
- vuex
- axios

Votre application doit pouvoir se lancer à l'aide de la commande `npm run serve`. Votre application doit être du type SPA (single page app), c'est à dire que vous ne devez pas changer de page : vous devez donc utiliser vue-router. Vous devrez utiliser le web service mis à votre disposition pour faire jouer l'utilisateur.

Afin de pouvoir restituer les parties jouées (exigences 5 et 6), vous devez utiliser `vuex` : vous stockerez toutes les parties jouées ainsi que toutes les informations nécessaires à l'affichage dans un état global.

Vous devez tester votre application avec Cypress. Voici la liste des tests attendus :

1) Vérifier que l'application se charge bien
2) Vérifier que lorsque vous cliquez sur le bouton "lancer une partie", l'affichage propose bien à l'utilisateur de saisir une valeur
3) Vérifiez que lorsque vous saisissez "1001", vous avez le message d'aide "C'est moins" qui apparaît
4) Vérifiez que lorsque vous saisissez -1, vous avez le message d'aide "C'est plus" qui apparaît
5) Vérifiez que lorsque vous appuyez sur le bouton "abandonner la partie", vous arrivez bien sur une page de défaite
6) Lorsque vous accédez aux statistiques, vérifiez bien que vous avez la page des statistiques qui s'affiche (inutile que votre test valide que vos statistiques sont bonnes)

Vous pouvez vous servir des issues Github pour gérer votre projet et suivre un minimum les tâches que vous devez faire.

Rendu du projet

Le projet est à rendre pour le 22/12/2021.
Pour rendre votre projet, vous devrez :

1) Compléter le README avec l'ensemble des informations nécessaires
    a) Le nom et prénom de chaque participant
    b) Le groupe de TP
2) Commiter et pusher votre code sur Github
   Attention à bien embarquer les bons fichiers dans votre commit. Vérifiez bien que votre code est présent sur Github.
3) Créer un tag (= release sur Github) nommé 1.0.0

Ce tag devra être posé sur Github avant la date de rendu de projet. Tout tag posé après cette date limite entraînera un malus de 1 point / jour de retard.

Bonus

Vous pouvez faire plus de tests, comme en faire un qui va trouver tout seul la bonne valeur pour vérifier que vous affichez bien l'écran de victoire.

Le style graphique de votre application comptera uniquement comme du bonus.