

Notes - Projet

Front-end : ReactJS

Back-end : Firebase RTDB

Repo github

UX : Css léger ou Tailwind avec composants réutilisable Bootstrap

ChatGPT : Utilisation intelligente de l'IA

Objectif : Faire une application web permettant de créer des "quêtes" personnelles, chaque quête accomplie rapport des points. Les quêtes doivent être stockés sur Firebase RTDB et l'UI se met à jour instantanément avec même plusieurs onglets ouvert.

Brouillon de l'UI simpliste :

QuestBoard

Ajouter une quête

Listes des quêtes :

- Aller à la salle de sport ☐
- Lire un chapitre par jour ☐

Valider

Profile

points :

nom :

- **Optimistic Update** : c'est une approche de développement où l'interface utilisateur va être mis à jour avant le serveur, c'est le résultat d'une opération asynchrone. Cela permet aux utilisateurs d'avoir une UI plus responsive.

Schéma de l'architecture du projet :

```
questboard/
|
├─ src/
|   ├─ components/
|       ├─ QuestList.jsx (ou .tsx)
|       ├─ QuestItem.jsx
|       ├─ QuestProgress.jsx
|       └─ AddQuestForm.jsx
|   |
|   ├─ App.jsx
|   └─ index.jsx
|   └─ firebase.js ← (config Firebase)
|
├─ public/
|   └─ index.html
|
├─ package.json
└─ ...
```

A retenir :

- Un composant contrôlé est "esclave" du `state` React.
- Tu ne modifies jamais le champ directement. C'est React qui dit quoi afficher.
- Ça te donne un **contrôle total** : validation, formatage, blocage, etc.

Initialisation du projet :

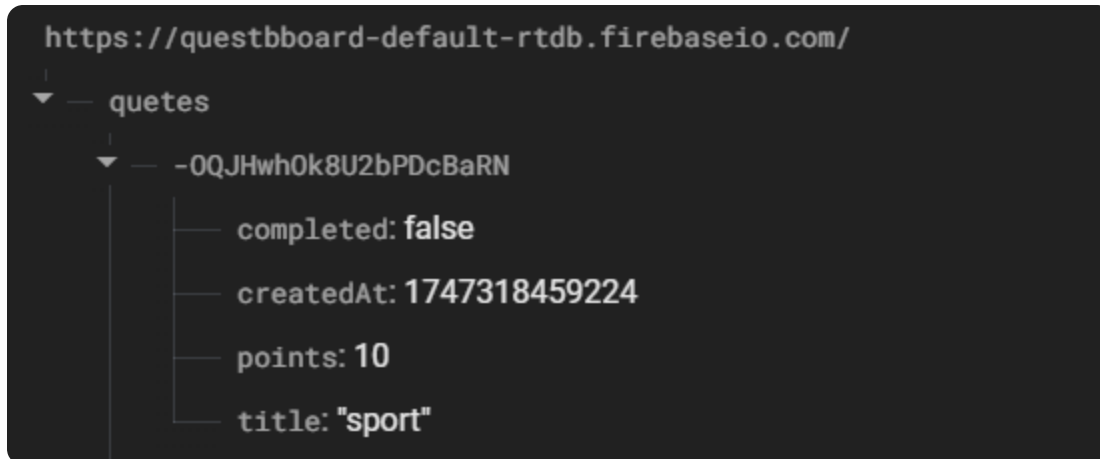
J'ai initialisé le projet avec l'aide de la formation de Mosh, j'ai fais un `npm create vite@latest`, j'ai choisi React et Typescript puis j'ai fais l'installation avec `npm i` et ensuite lancer l'app avec `npm run dev`.

1ère étape le Form :

J'ai refait le Form de la formation de Mosh, mais je dois pouvoir stocker l'ajout de la quête sur Firebase donc je vais demander à ChatGPT de m'expliquer chaque étape et le code si j'en utilise.

- Maintenant il manque le `useState` pour stocker la valeur de l'input.
- La fonction d'import dans Firebase est `addDoc`.
- Il faut envoyer les données dans Firebase dans `handleSubmit`.
- J'ai commenté quelques parties du code pour comprendre et en cas d'oubli m'en souvenir.
- Il ne faut pas mettre le `return` dans le `handleSubmit` car sinon il ne renvoie pas le JSX. Le `return` dans React correspond au rendu, donc il doit être à la racine du composant.

Le formulaire communique bien avec la db mais il faut régler l'heure (même si en soit l'heure n'est pas fausse car en général c'est en format unix) et peu être modifier des choses comme générer un id quête et pas un aléatoire :




Ajout d'un svg bolt avec un peu de css directement dans la `div` et la balise `img`

- `useState` sert à **créer une donnée dynamique** (comme un compteur ou une liste) dans un composant React.

Etape 2 la List :

- J'ai utilisé Bootstrap et adapté pour React une liste en checkbox pour valider les quêtes. J'ai rajouter un logo et un peu de CSS.
- Il faut maintenant afficher les quêtes qui sont dans la DB Firebase dans les checkboxes.
- Je défini ma quête je l'appelle `Item` car j'ai déjà utilisé quête (pour éviter de me perdre).
- Ensuite on récupère les quêtes de ma DB pour les afficher dans la liste.
- Je demande à chatGPT de m'expliquer ce que je ne comprends pas trop ou si je n'y arrive pas.
- Ensuite je fais le code pour générer une checkbox à chaque ajout de quête, ensuite je boucle la liste `quests` avec `.map()`.
- J'ajoute `key={quest.id}` à ma balise `` pour identifier chaque élément de la liste avec un id aléatoire que Firebase m'a donné.

- `htmlFor={checkboxStretched-${quest.id}}` : associe ce label à un `input` avec `id="checkboxStretched-..."` pour que cliquer sur le texte clique aussi sur la case à cocher.
- Pour le reste de la partie front je met du Bootstrap simple du margin ect.
- Maintenant il faut ajouter le `onChange` pour modifier l'état et mettre à jour Firebase.
- Copilot m'a bien aidé à debug quand il fallait placer du code dans une const ou non ça m'a permis de gagner du temps et comprendre l'erreur, ou encore ce genre d'erreur :

 You should replace `Object.entries` with just `Object.entries` (with a lowercase "O"), as `Object` (uppercase) is the global object, not the TypeScript type.

- **snapshot** : L'objet que Firebase donne dans le callback, il contient les données actuelles et on peut récupérer les données en faisant un `snapshot.val()`.
- **onValue** : Fonction de Firebase RTDB écoute en real time les changements à l'emplacement donnée. (C'est le listener)
- **unsubscribe** : Arrête l'écoute quand le composant est supprimé
- **||** : Exemple `val || 0` sert à dire si val existe prends-le sinon 0, utile quand la DB est vide au départ.
- **useEffect** : Hook React permet de lancer du code au moment où le composant s'affiche.
- **FC** : Signifie **Function Component**. C'est un **type générique** fourni par React pour **typer un composant fonctionnel** en TypeScript. C'est pratique car ça type automatiquement les props, reconnaît les children et une meilleure visibilité du projet. **(C'est optionnelle)**

J'ai **allégé** le code du fichier **QuestList** en mettant la partie type de Quest dans le **QuestItem**. QuestItem va afficher les **checkbox** avec cocher et décocher la quête et afficher le **bouton supprimer** relié à la DB pour **retirer** la quête de la DB.

Etape 3 ProgressBar :

J'ai trouvé sur react-bootstrap des barres de progressions donc pour le questboard c'est parfait, j'ai donc ajouté la barre et relié la progression aux points dans la DB, il a fallu faire :

- Un calcul du pourcentage en fonction de l'objectif, l'objectif quotidien est de 70 points donc j'ai demandé un peu d'aide : `Math.min((points / objectif) * 100, 100)`.
- Le reste était plutôt simple mettre un `useEffect` pour récupérer automatiquement les points dans la DB les enregistrer dans l'état points à jour et afficher la progression sur la barre.
- Je ferais le reset de toute les 24h plus tard, car j'aimerais voir les **cloud functions** quand l'ensemble du site sera ok.

Etape 4 Ajout bouton suppression :

Dans le fichier QuestItem, j'ai mis le bouton Supprimer à côté des quêtes et je les ai stylisés en utilisant Bootstrap. En suite j'ai fait la fonction `onDelete` qui va récupérer les infos de la DB et ensuite quand le bouton est cliqué ça va remove la quête (normalement). Mais avant de mettre ce bouton supprimer dans une balise label j'avais laissé dans le `className` un `stretched-link` qui faisait que quand on cliquait sur une quête même hors de la checkbox ça sélectionnait et ignorait le bouton supprimer. Je l'ai donc enlevé et ça marche maintenant !

Etape 5 Test avec Vitest :

Je vais utiliser `Vitest` car mon projet est un projet vite, donc je change et j'installe `jsdom` dans mon projet car c'est nécessaire pour simuler le DOM dans un env de test. Ca supporte `jsx` `tsx` sans problème et c'est directement intégré dans mon écosystème `Vite`, c'est beaucoup plus simple pour réaliser des tests. Je fais un fichier `setupTests.ts` ou j'importe les libs que je vais utilisé, dans le `vitest.config.ts` j'indique mon setup file. Comme ça dans mon fichier de test je peux utiliser :

- `toBeInTheDocument` : Qui va permettre de vérifier qu'un élément est affiché à l'écran et que l'état du DOM correspond à ce que l'utilisateur doit voir.

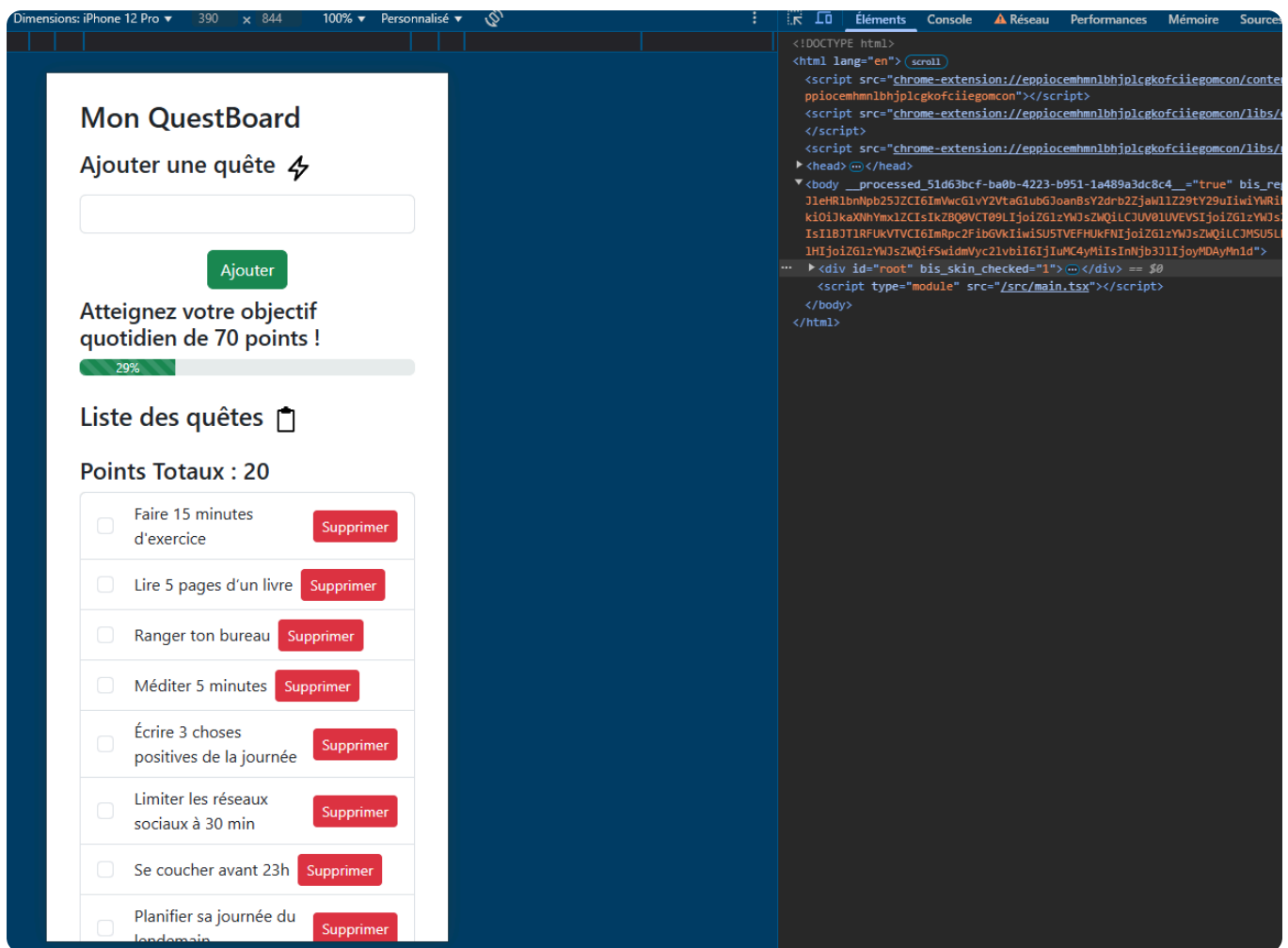
```
RERUN  src/setupTests.ts x2
```

```
✓ src/__test__/AddQuestForm.test.tsx (1 test) 324ms  
✓ affiche un champ texte et un bouton 322ms
```

```
Test Files  1 passed (1)  
Tests      1 passed (1)  
Start at   15:02:21  
Duration   3.94s
```

```
PASS  waiting for file changes...  
press h to show help, press q to quit
```

Responsivité :



Le site s'adapte automatiquement aux appareils mobiles et tablettes.

Résultat de la page & Rétrospective :

Mon QuestBoard

Ajouter une quête ⚡

Ajouter

Atteignez votre objectif quotidien de 70 points !

29%

Liste des quêtes 📅

Points Totaux : 20

<input type="checkbox"/>	Faire 15 minutes d'exercice	Supprimer
<input type="checkbox"/>	Lire 5 pages d'un livre	Supprimer
<input type="checkbox"/>	Ranger ton bureau	Supprimer
<input type="checkbox"/>	Méditer 5 minutes	Supprimer
<input type="checkbox"/>	Écrire 3 choses positives de la journée	Supprimer
<input type="checkbox"/>	Limiter les réseaux sociaux à 30 min	Supprimer
<input type="checkbox"/>	Se coucher avant 23h	Supprimer
<input type="checkbox"/>	Planifier sa journée du lendemain	Supprimer
<input type="checkbox"/>	Faire une bonne action (aider, remercier, etc.)	Supprimer
<input type="checkbox"/>	Boire 1,5L d'eau	Supprimer

Valider

J'ai plutôt bien aimé ce petit projet d'apprentissage, j'ai pu construire petit à petit mon app de Questboard, ce que j'ai préféré faire ce sont d'ajouter des features comme la barre de progression et aussi de réussir à faire une liste dynamique qui interagit avec la DB.

Il me reste encore à approfondir les logiques des composants comme par exemple pour ajouter une quête et les Hooks aussi. Je compte apprendre à mieux les utiliser en pratiquant de plus en plus et au fil du temps je serais en mesure de les utiliser sans problèmes. J'ai vu seulement à la fin que j'aurais dû faire un Hook personnalisé pour la logique Firebase et effectivement ça aurait évité la duplication de code et rendre plus lisible le tout. Donc je pense que pour les prochains projets je veillerais plus à mieux organiser mon code et qu'il soit plus lisible pour plus de compréhension.

Le projet est maintenant terminé, mais il pourrait y avoir de grosses améliorations comme ajouter une page de login et d'inscription pour que chaque utilisateur possède ses propres points et ses propres quêtes.