Travaux accompagnés

Ionic React - Projet Writ'Pins

Introduction

L'objectif ici est d'utiliser dès maintenant vos connaissances en React pour créer un frontend avec la librairie Ionic. Celle-ci présente des avantages intéressants :

- Elle sert de "surcouche" à React, et ne change donc pas le framework et les concepts que vous avez appris
- Elle fournit une large bibliothèque de composants interactifs pour gagner du temps en ne réinventant pas la roue.
  De plus, ils sont conçus pour que leur apparence s'adapte aux chartes graphiques d'Android et Apple sans effort.
- Elle s'accompagne entre autres du module *Capacitor*, qui permet de faire tourner votre application en semi-natif sur iOS ou Android.

Vous pourrez suivre les étapes guidées de ce TP, ainsi que la documentation d'Ionic et toute autre ressource internet pour pratiquer React davantage, découvrir d'autres de ses fonctionnalités, ainsi que vous familiariser avec Ionic. Mais avant cela, commencez par prendre connaissance du cahier des charges ci-dessous.

Description Projet Writ'Pins

Cahier des charges

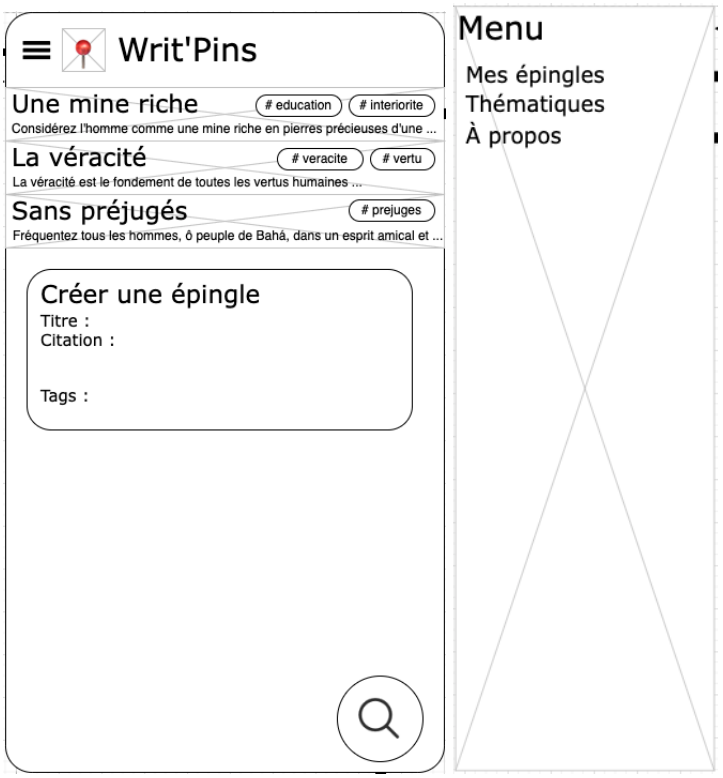
Writ'Pins est une application visant à permettre à toute personne de conserver un répertoire de citations ou de passages inspirants de différents auteurs sur différents thèmes. En voici les fonctionnalités principales que vous développerez.

Le concept de l'appli : les 'épingles'
L'utilisateur peut CRUD (Create Read Update Delete) des 'pins' (épingles) dont il est le propriétaire. Une épingle est constituée de :

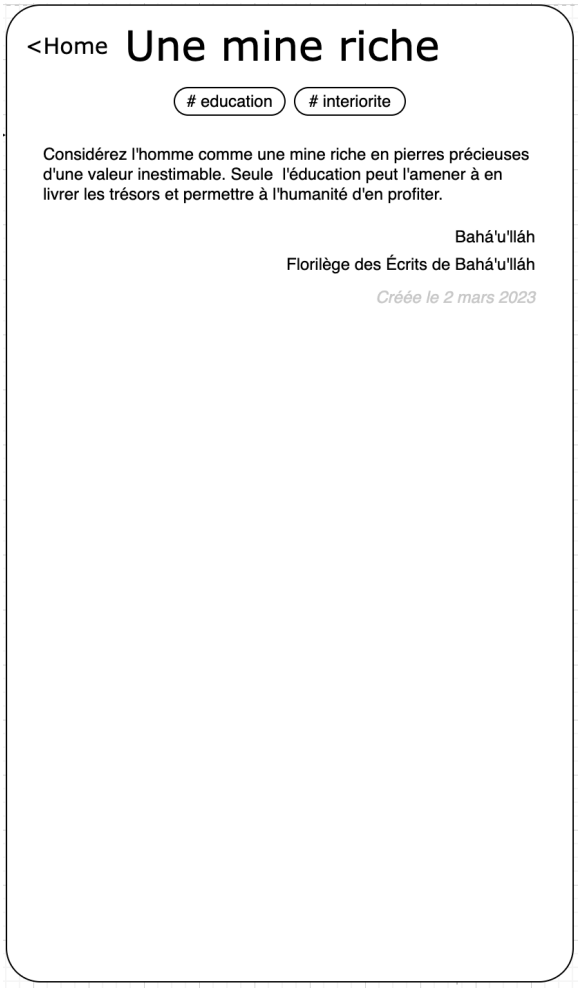
- Un titre, choisi par l'utilisateur, comme mémo
- Un passage de texte, écrit ou copié-collé depuis une source externe. Le texte peut être mis en forme (gras, italique, souligné)
- La source de la citation (nom d'ouvrage ou d'auteur)
- Des tags, composés d'un mot clé, servant à étiqueter l'épingle, par exemple pour la rattacher à certains thèmes (#éducation, #unité, #progrès…)
- Une date de création

La vue principale
Un utilisateur pourrait vouloir conserver des dizaines, voire des centaines de passages, et s'y référer très efficacement. On souhaite qu'un utilisateur puisse parcourir très rapidement toutes ses épingles, en ne voyant sur l'écran principal que les titres, avec les tags, suivis de la première ligne du texte (comme dans une boîte mail, qui affiche l'expéditeur, le sujet du mail et un petit aperçu du message).
Voir le mockup sur Diagrams.io

Ci-dessous se trouve les écrans de l'application en *wireframe* - il ne faut tenir compte que de la disposition et dimensionnement des éléments, mais pas le style, que vous pourrez choisir ou laisser par défaut.



La vue ‘épingle’
À partir de l’écran principal, en cliquant sur une épingle, il est possible d’ouvrir l’épingle en question, qui affichera tout le reste des informations associées à l’épingle.



Des commentaires entre utilisateurs
L'application est un mini-réseau social. Il est possible de rendre des épingles publiques pour partager avec d'autres utilisateurs des commentaires sur l'épingle.

Des utilisateurs, du coup !
Chaque utilisateur dispose d'un profil, comme sur tout site classique, avec les fonctions de base de profil utilisateur.

Partager une épingle
L'utilisateur peut rendre une épingle publique pour partager un fil de commentaires avec les autres utilisateurs. Il peut aussi envoyer/recevoir une invitation directe à partager une épingle avec un ou plusieurs utilisateurs pour échanger des commentaires.

Rechercher une épingle
L'avantage des épingles, c'est leur rapidité d'accès. Pour faciliter encore l'utilisateur, il existera des fonctions de recherche textuelle pour rechercher :
- Dans le titre des épingles seulement
- Dans le corps des textes seulement
- Dans les tags des épingles seulement
- Dans les épingles privées, les épingles publiques

Déroulement

Itération 1 - Vue principale

Vous allez découvrir comment installer Ionic et utiliser vos premiers composants.

- [ ] Installer la CLI Ionic

- [ ] Avec la CLI de Ionic, créez un nouveau projet avec `$ionic start` : ne sélectionnez pas le *creation wizard*, et choisissez le framework React. Quand l'invite de commande vous demande quel template utiliser, **choisissez le template *liste***. Vous aurez ainsi une base de projet qui présente un template d'application de mailing.

```
livingstonehgs@MBP-de-Quentin Desktop % ionic start
? Use the app creation wizard? No

Pick a framework! 😄

Please select the JavaScript framework to use for your new app. To bypass this prompt next time,
supply a value for the --type option.

? Framework: React

Every great app needs a name! 😍

Please enter the full name of your app. You can change this at any time. To bypass this prompt next
time, supply name, the first argument to ionic start.

? Project name: writpins-project

Let's pick the perfect starter template! 💪

Starter templates are ready-to-go Ionic apps that come packed with everything you need to build your
app. To bypass this prompt next time, supply template, the second argument to ionic start.

? Starter template:
  blank        | A blank starter project
❯ list         | A starting project with a list
  my-first-app | A template for the "Build Your First App" tutorial
  sidemenu     | A starting project with a side menu with navigation in the content area
  tabs         | A starting project with a simple tabbed interface
```

- [ ] Prenez le temps de découvrir la structure du projet : dossiers, fichiers, composants.

Vous vous êtes peut-être rendu compte que le code n'est pas écrit en simple JavaScript, mais en TypeScript. Commencez par découvrir TypeScript et ses principales fonctionnalités en lisant ces deux pages avant de continuer le TP :

- TS for the New Programmer
- TS for JS Programmers

Ce qui est suggéré dans un premier temps, c'est que vous conserviez le code du projet template tel quel, sans l'altérer, mais que vous créiez vos propres fichiers en vous inspirant du code template. Vous supprimerez plus tard le code initial.

- [ ] Dans le sous-dossier `data` du projet, ajoutez un fichier `pins.ts` qui contienne une première liste, hardcodée, de citations, en vous inspirant de `messages.ts`.

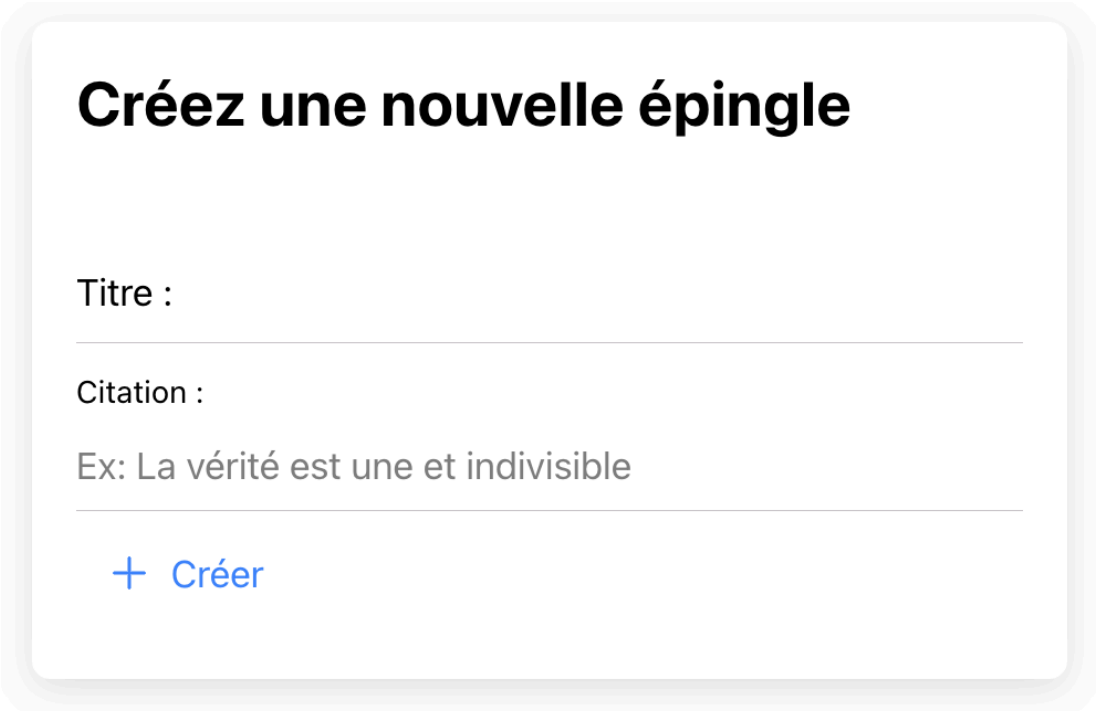
- [ ] Créez un composant `PinListItem` (dans un nouveau fichier) similaire au `MessageListItem` du template. Utilisez un `IonList` (dans le DOM, `Home/IonPage/IonContent`) pour y insérer votre liste de pins tirées du fichier `data/pins.ts`, comme sur le mockup de la Vue principale.

  Ce nouveau composant n'affiche pour le moment qu'un titre et la première ligne de la citation, imbriquée dans un `<span>` pour avoir trois points à la fin si la citation prend plus que la largeur de l'écran.

- [ ] Tout en bas de la liste des épingles, ajoutez un composant "carte" (ex. ci-dessous) avec un titre, deux champs texte et un bouton pour valider la création d'une nouvelle épingle qui apparaîtra ensuite dans la liste au-dessus. Elle sera créée dans l'état de la page, sans avoir besoin de l'enregistrer de manière permanente pour le moment.

  Vous chercherez vous mêmes les composants Ionic nécessaires (carte, champs de texte sur une ligne, champ sur plusieurs ligne, bouton..). Ne perdez pas trop de temps à rendre le graphisme parfaitement fidèle.

  Note : bien entendu, puisque les nouvelles citations seront ajoutées dans l'état d'un composant, il est normal que celles-ci disparaissent au rafraîchissement de la page.



Itération 2 - Écran de connexion

Vous allez maintenant découvrir comment utiliser Ionic pour naviguer d'une page à l'autre en implémentant un système de connexion, afin d'utiliser le localstorage et le hook `React.useEffect`.

- [ ] Dans le fichier App.ts, utilisez un `IonReactRouter` pour configurer les routes suivantes :
  - `/` (chemin vide) redirige vers `/home`, la vue principale.
  - `/login` dirige vers la page de connexion

  Quel est l'avantage d'utiliser un `IonReactRouter` ici, plutôt qu'un simple `ReactRouter` ?

- [ ] Sans vous occuper de la logique de connexion (i.e. sans implémenter d'action spécifique), créez d'abord la page qui contient le formulaire identifiant/mot de passe et un bouton "se connecter". Ce bouton a seulement pour effet de faire passer à la page `/home`.

Vous allez maintenant devoir conserver en mémoire, et de manière globale dans le projet, un "objet" (variable personnalisée avec propriétés, un peu comme un dictionnaire Python ou une structure en C) pour stocker l'état de connexion de votre utilisateur afin de savoir, à chaque fois que la page est chargée, si l'utilisateur est déjà connecté ou non. Vous allez pour cela suivre le guide donné plus bas, mais avant, il vous sera utile de découvrir les notions suivantes :

- Le mot-clé `interface` défini par TypeScript qui sert à déclarer une certaine structure de données (pour une variable par exemple). Lisez cette page, du début jusqu'à la fin de la section "readonly Properties" (soit juste avant le début de "Index Signatures").

- Le `localStorage`, qui est une API (= Application Programming Interface, au sens large, pas au sens d'API REST) du navigateur qui permet de stocker des données persistantes sous forme de paires clé-valeur. Ces données restent disponibles même après la fermeture du navigateur ou le redémarrage de l'ordinateur, jusqu'à ce qu'elles soient explicitement supprimées par le code ou l'utilisateur. Découvrez ici ce qu'est un localstorage.

- Enfin nous remarquons une chose : un certain nombre de pages de notre application devront être protégées, ou inaccessibles, pour l'utilisateur non connecté. Comment alors une page (composant React) peut-il savoir s'il lui est permis de s'afficher (quand l'utilisateur est connecté), ou bien de rediriger vers la page de connexion ? Et s'il peut s'afficher, comment afficher par exemple les données du profil utilisateur ?
  La première idée qui vient est d'utiliser des props, puisque c'est finalement le composant App gère toutes les pages. Mais cela serait très fastidieux de passer, par exemple, une `interface User` à tous les composants enfants, voire même leurs enfants à eux ! C'est pourquoi il existe ReactContext, une alternative excellente aux propriétés dans notre cas du login. Survolez cette page "Passing Data Deeply with Context" afin de comprendre le concept.

Une fois que vous aurez suffisamment compris ces trois concepts (interface, localStorage et ReactContext), inspirez-vous de ce tuto "Use React Context for Auth" pour créer un système d'authentification pour votre application. Pas besoin, dans un premier temps, de vérifier un couple utilisateur/mot de passe valide : un simple clic sur le bouton "se connecter" suffira et vous enregistrerez un booléen `loggedIn` dans le localStorage.
En lisant ce tuto, vous vous demanderez à quoi sert le `React.useEffect` qui est utilisé dans le hook useAuth.ts : C'est une excellente question que vous approfondirez plus loin. Pour le moment contentez-vous de cette simple explication : dans le fichier useAuth.ts (du tuto), le bloc `useEffect(...)` a pour but de mettre automatiquement à jour la valeur de `user` dans le localstorage dès que l'utilisateur change dans le navigateur — s'il se déconnecte, reconnecte, ou a changé.

- [ ] En vous appuyant sur le tuto *Use React Context for Auth*, implémentez un système d'authentification (à base d'un simple clic sur un bouton) pour authentifier un utilisateur et lui laisser l'accès à des pages privées.

- [ ] Firebase est un service Google très accessible et initialement gratuit pour commencer à utiliser des services Cloud pour vos applications. Créez une console Firebase pour votre projet et configurez un système d'authentification, cette fois, basée sur des couples identifiant/mot de passe. Vous trouverez aisément la documentation et des exemples de code pour cela, et vous utiliserez le code écrit dans l'étape précédente. Ne faites pas la fonctionnalité de signup (créer un compte), mais seulement de connexion - vous créerez manuellement les utilisateurs dans Firebase.

Itération 3 - Utiliser Firebase pour synchroniser les épingles

Dans le but de découvrir un peu plus le fonctionnement de `useEffect` avec React, un hook couramment utilisé, vous allez synchroniser les épingles de l'utilisateur sur Firebase. Pour cela, prenez vraiment le temps de comprendre cette doc : Synchronizing with Effects. Vous voudrez aussi vous intéresser au mot-clé `async` de Javascript

- [ ] Créez une table dans Firebase et enregistrez manuellement quelques épingles.
- [ ] Faites en sorte qu'au moment du montage de la page des épingles, un `useEffect` agisse pour charger les épingles stockées sur le Cloud, puis les afficher sur la page.
- [ ] Faîtes maintenant en sorte que lors de la création d'une épingle, celle-ci soit enregistrée sur Firebase.

Itération 4 - Navigation vers la vue épingle

- [ ] Ajouter un chemin vers `/pin/id` qui ouvre la nouvelle vue `ViewPin` que vous allez créer maintenant
- [ ] Créez la nouvelle page `ViewPin` qui, à la façon de `ViewMessage`, affiche le titre, la citation, les tags, la date de création de l'épingle.
- [ ] Utilisez cette nouvelle vue pour qu'en cliquant sur une épingle, la page glisse vers la Vue épingle. S'afficheront pour l'instant :
    - Le titre
    - La citation
    - La source et l'auteur
    - La date de création

Itération 5 - Compiler pour iOS et Android

L'équipe d'Ionic a également développé Capacitor, un module capable de transformer une webapp en code dédié à Android ou à iOS. C'est ce que vous allez faire dans cette itération : lancer votre appli sur les VM Android et iOS, ainsi que sur votre smartphone !

- [ ] Installez Capacitor dans votre répertoire projet : `npm install @capacitor/core && npm install @capacitor/cli`
- [ ] Ensuite, lancez-là sur votre appareil mobile ! Trouvez pour cela le tutoriel adapté. Notez que vous ne pourrez pas lancer votre application sur votre iPhone si vous n'avez pas un Mac… dans ce cas, faites seulement la compilation pour Android, et lancez l'appli sur la VM d'Android sur votre ordinateur.

Faîtes vos recherches ! (facultatif)

Produisez un document d'une page sur un sujet au choix en lien avec le développement mobile : librairies graphiques pour les jeux, réalité augmentée, utilisation d'algorithmes d'apprentissage automatique, ergonomie…
L'objectif de ce document est triple :

- Découvrir l'état de l'art du sujet (d'un point de vue technique)
- Faire vos propres recherches sur le sujet, afin d'être capable d'exposer un chemin d'apprentissage : par où commenceriez-vous pour commencer à développer des compétences sur ce sujet ?
- Présenter le résultat de vos recherches lors de l'entretien individuel pour la deuxième évaluation