Marvel - Version 0.4.0

Alexandre GIRARD - Conseiller en Nouvelles Technologies - alexandre.girard@maif.fr

Gestion de la navigation dans l'application

- Installation et configuration de **React Router**, librairie tierce permettant de gérer la navigation
- Mise en place de la navigation dans l'application Marvel
  - définir les routes
  - définir les composants à afficher en fonction de la route
- Définir le layout de l'application
  - header, footer, menu, sidebar, contenu principal...

git

A vous de faire le nécessaire pour travailler correctement avec git et générer la version 0.4.0 de l'application.

- Rappel:
  - On ne travaille pas directement sur la branche principale.
  - On utilise des branches de fonctionnalités pour travailler sur des fonctionnalités spécifiques.
  - On commit régulièrement de manière atomique et avec des messages de commit explicites.
  - On ne commit pas du code rendant l'application inutilisable.
  - On ne merge pas une branche sans avoir testé le code.

git

Nous verrons plus tard comment sécuriser nos branches de développement et de production. Pour l'instant c'est à nous de faire attention à ne pas commettre d'erreurs.

Création des pages de l'application

- Création des pages de l'application
  - Home
  - Characters
  - Contact
  - About

Création des pages de l'application (suite)

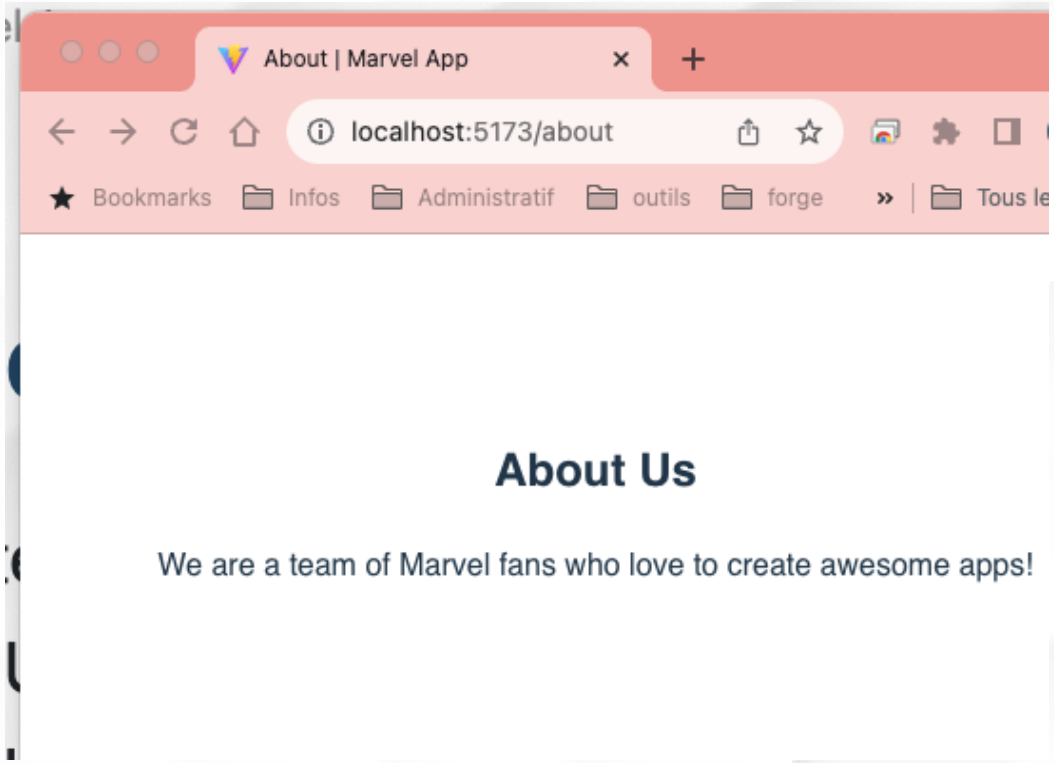
- Les pages sont des composants React
  - définis dans le dossier `src/pages`
  - importés dans le composant principal `App`

Page About

Créez le composant `AboutPage` dans le dossier `src/pages` et ajoutez le contenu suivant:

- Utilisation de **document.title** permettant de redéfinir le titre de la page
- Un en-tête **h2**
- Un paragraphe

Pour tester cette page, modifier le fichier App.jsx pour utiliser le composant **AboutPage** (et supprimer l'affichage actuel).

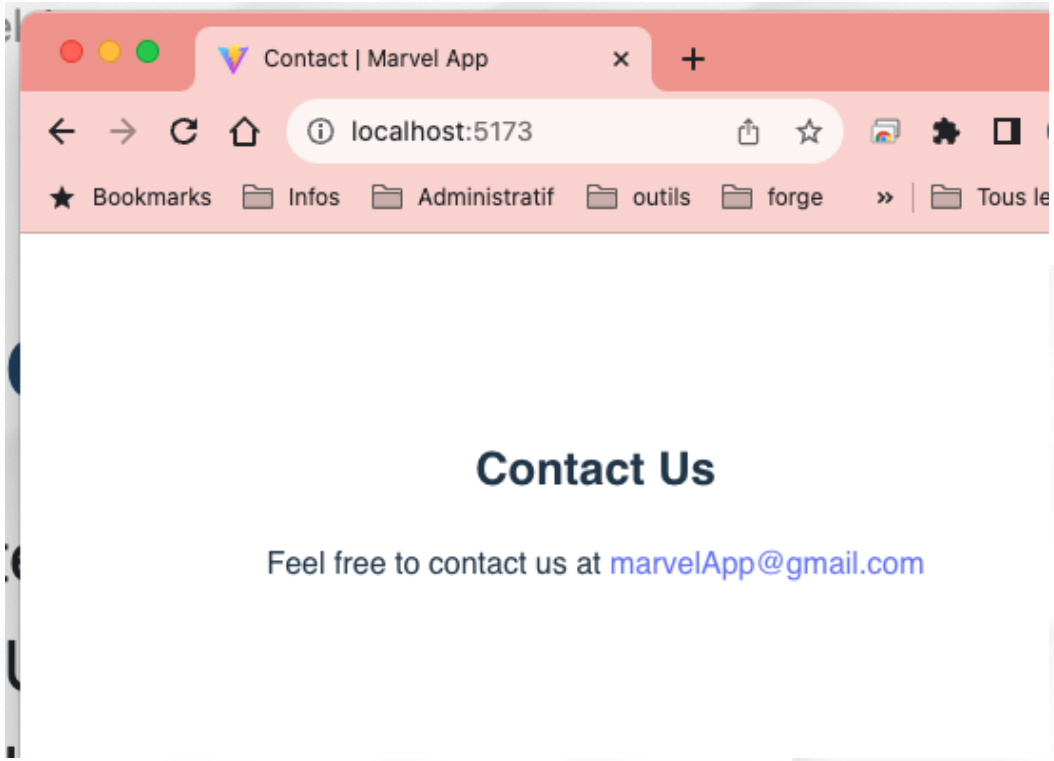


Page Contact

Créez le composant `ContactPage` dans le dossier `src/pages` et ajoutez le contenu suivant:

- Utilisation de **document.title** permettant de redéfinir le titre de la page
- Un en-tête **h2**
- Un message de contact avec un lien de type **mailto**

Pour tester cette page, modifier le fichier App.jsx pour afficher la page **Contact**.

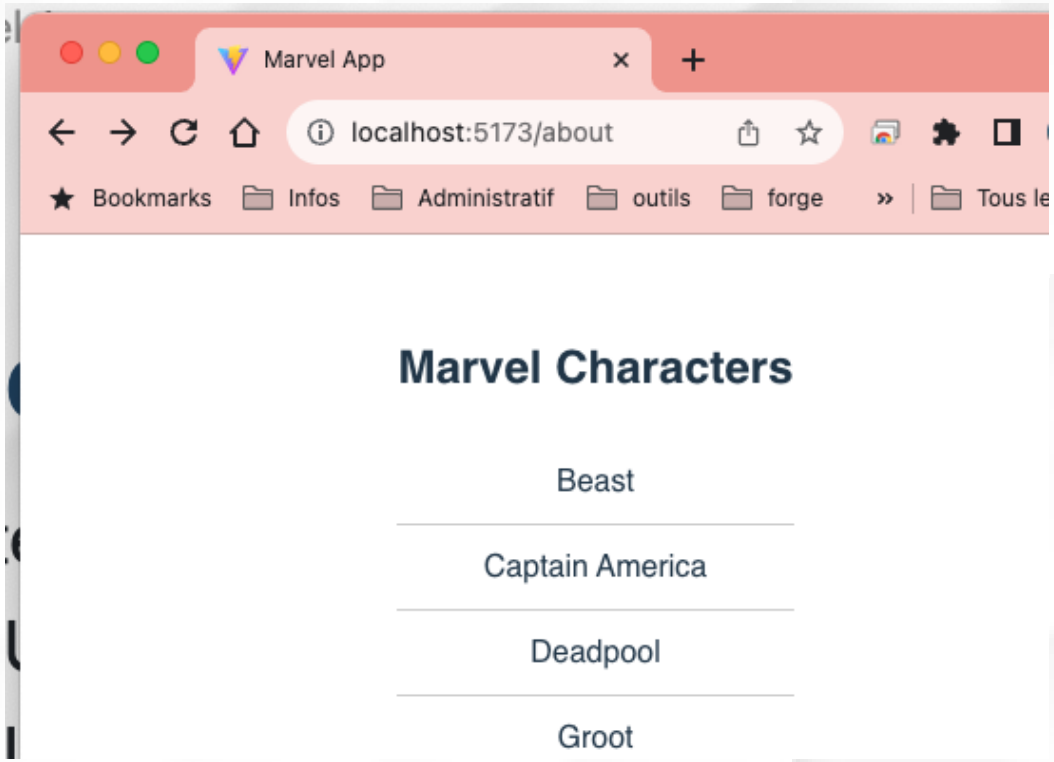


Page Characters

Créez le composant `CharactersPage` dans le dossier `src/pages` et ajoutez le contenu suivant:

- Utilisation de **document.title** permettant de redéfinir le titre de la page
- Le contenu qui était affiché sur la page d'accueil, attention au chemin d'accès aux imports

Pour tester cette page, modifier le fichier App.jsx pour afficher la page **Characters**.



Page Home

Nous n'avons pas de contenu spécifique pour la page Home, nous afficherons le contenu de la page Characters. Nous traiterons la page Home plus tard.

Problématique - Gestions des routes

- Comment gérer la navigation dans une application React ?
- Comment définir les routes de l'application ?
- Comment afficher les composants en fonction de la route ?

React Router

React Router est une librairie tierce permettant de gérer la navigation dans une application React, voir React Router

Son objectif est de permettre une navigation fluide et dynamique entre les différentes pages de l'application sans recharger la page.

React Router - routes

Nous allons utiliser **React Router** pour définir les routes de l'application et ainsi gérer la navigation entre les pages.

Pour chaque page de l'application, nous allons définir une route correspondante dans le routeur et définir le composant à afficher en fonction de la route.

```
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const router = createBrowserRouter([
  {
    path: "/",
    element: <CharactersPage />,
  },
  {
    path: "/about",
    element: <AboutPage />,
  },
  {
    path: "/contact",
    element: <ContactPage />,
  },
]);

function App() {
  return (
    <>
      <RouterProvider router={router} />
    </>
  );
}
```

React Router - routes (suite)

Nous pouvons maintenant accèder à chaque page de l'application en fonction de l'URL.

Nous n'avons pas encore géré le cas d'une page inexistante, mais cela peut se faire facilement en ajoutant une route correspondante.

La route répondra à toutes les URL qui ne correspondent à aucune des routes définies précédemment.

```
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

const router = createBrowserRouter([
  {
    path: "/",
    element: <CharactersPage />,
  },
  {
    path: "/about",
    element: <AboutPage />,
  },
  {
    path: "/contact",
    element: <ContactPage />,
  },
  {
    path: "*",
    element: <div>404 Not Found</div>,
  },
]);

function App() {
  return (
    <>
      <RouterProvider router={router} />
    </>
  );
}
```

React Router - Layout

Le layout de l'application permet de définir la structure de l'application, c'est-à-dire les éléments qui seront affichés sur toutes les pages de l'application et ainsi ne pas dupliquer le code dans chaque page.

Une application est généralement composée de trois parties :

- **Header** : en-tête de l'application
  - navigation entre les pages
- **Main** : contenu principal de l'application
- **Footer** : pied de page de l'application

React Router - Layout (suite)

- Création du composant `Layout` dans le dossier `src` (nommé `Layout.jsx`)
- Utilisation du composant `Outlet` de `react-router-dom` dans le composant `Layout`
  - `<Outlet />` permet d'afficher le contenu de la page en fonction de la route

```
import { Outlet } from "react-router";

const Layout = () => {
    return (
        <>
            <header>
                <h1>Marvel App</h1>
                <nav>
                    <a href="/">Home</a>-
                    <a href="/about">About</a>-
                    <a href="/contact">Contact</a>-
                </nav>
            </header>
            <main>
                <Outlet />
            </main>
            <footer>
                <p>Marvel App - 2025</p>
            </footer>
        </>
    );
};

export default Layout;
```

Router - Layout (suite)

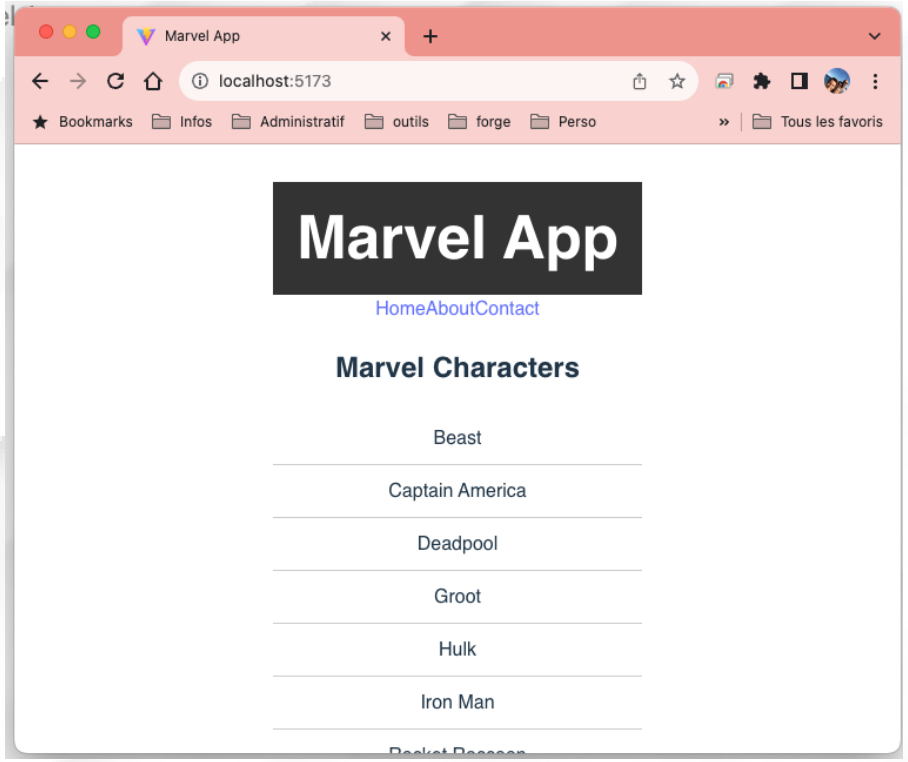
Le composant `Layout` devient le composant principal de l'application. Les pages seront affichées dans le composant `Outlet` de `react-router-dom` grâce à la prop `children`.

```
// routes of the application
const routes = [
  {
    path: "/",
    Component: Layout,
    children: [
      {
        // main page
        index: true,
        element: <CharactersPage />,
      },
      {
        path: "/about",
        element: <AboutPage />,
      },
      {
        path: "/contact",
        element: <ContactPage />,
      },
      {
        path: "*",
        element: <div>Page not found</div>,
      },
    ],
  },
];

export default routes;
```

React Router

Nous avons maintenant une application avec une gestion de la navigation grâce à **React Router**.

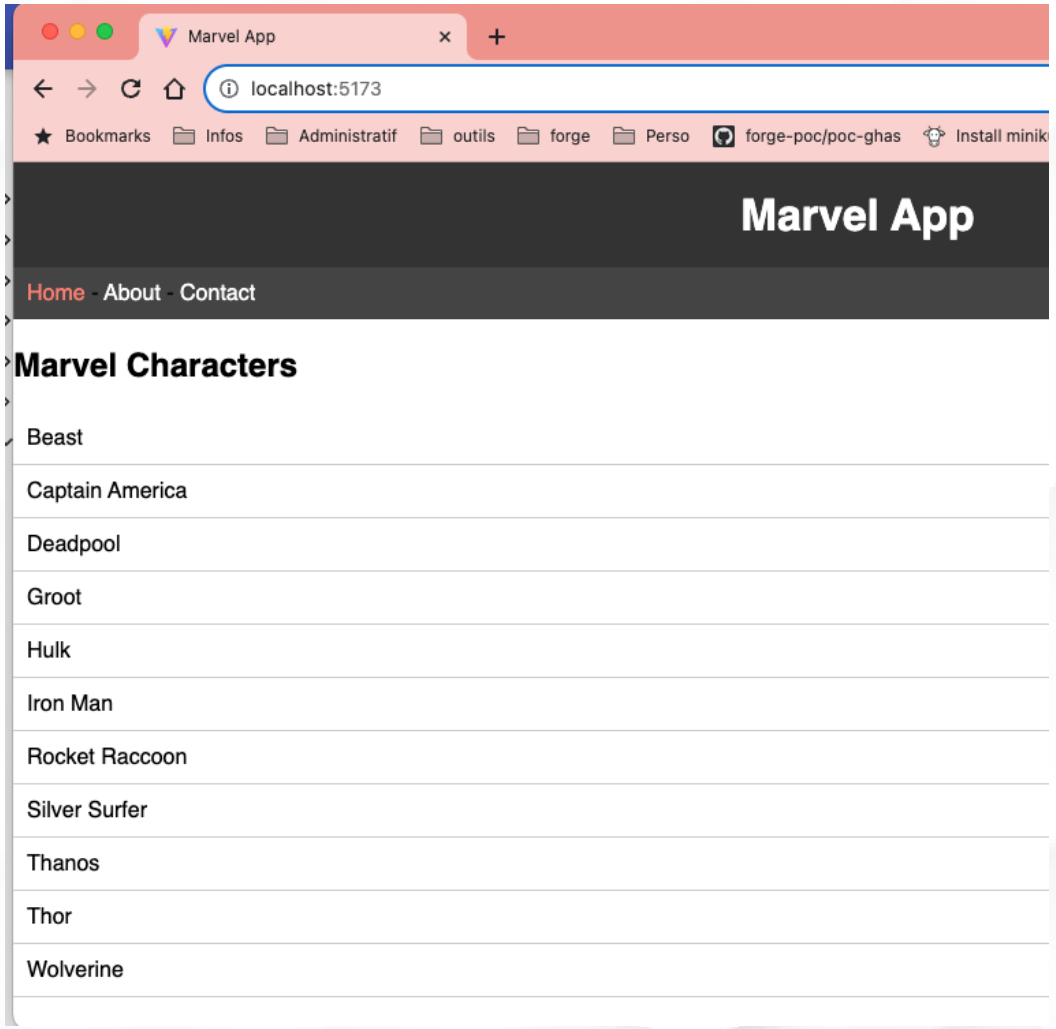


React Router - NavLinks

- Nous avons utilisé des balises `a` pour les liens de navigation, ce qui n'est pas optimal, on perd le fonctionnement de **React Router** et on recharge la page à chaque clic.
- Nous allons remplacer ces balises par des composants `NavLink` de **React Router**
  - Permet de gérer la navigation sans recharger la page
  - Permet de gérer les classes CSS pour les liens actifs

Ajout de css pour améliorer l'affichage

- Remplacer le contenu du fichier `App.css` afin d'améliorer l'affichage de l'application
- Télécharger le fichier App.css



Fin de la version 0.4.0

- Nous avons ajouté la gestion de la navigation dans l'application
- Nous avons créé les pages de l'application
- Nous avons créé le layout de l'application
- Nous avons utilisé **React Router** pour gérer la navigation
- Nous avons utilisé le composant `Outlet` pour afficher les composants correspondant à la route
- Nous avons utilisé les composants `NavLink` pour gérer la navigation sans recharger la page

Fin de la version 0.4.0 (suite)

L'application est maintenant fonctionnelle, nous avons ajouté les fonctionnalités de base pour une application web permettant de naviguer entre les différentes pages de l'application de manière fluide.

Il est maintenant temps de générer une nouvelle version de l'application et de l'emmener jusqu'à la production. Faites le nécessaire pour générer la version 0.4.0 de l'application.

git - Etat final