

Routing front-end

Ajoutons des pages à notre
projet front-end





Rappel router NextJS

Un routeur intégré à NextJS



Router intégré

Le route intégré dans NextJS s'appuie sur l'arborescence de fichiers, ainsi pour la gestion des routes :

1 page = 1 composant React = 1 fichier

Arborescence

→ `/index.tsx`

→ `/offers`

→ `/index.tsx`

→ `/[offerId].tsx`

Routes générées

→ GET `/`

→ -

→ GET `/offers/`

→ GET `/offers/[offerId]`



Liste des pages à créer

- Une page statique “About”
- Page dynamique des détails d’une annonce



Layout

Certains éléments, tels que le **Header**, vont se retrouver sur toutes ou du moins plusieurs pages du site.

Nous allons donc l'intégrer à un **layout**

<https://nextjs.org/docs/pages/building-your-application/routing/pages-and-layouts#single-shared-layout-with-custom-app>



Layout

Crée un composant Layout qui prend une props children.

<https://react.dev/learn/passing-props-to-a-component#passing-jsx-as-children>

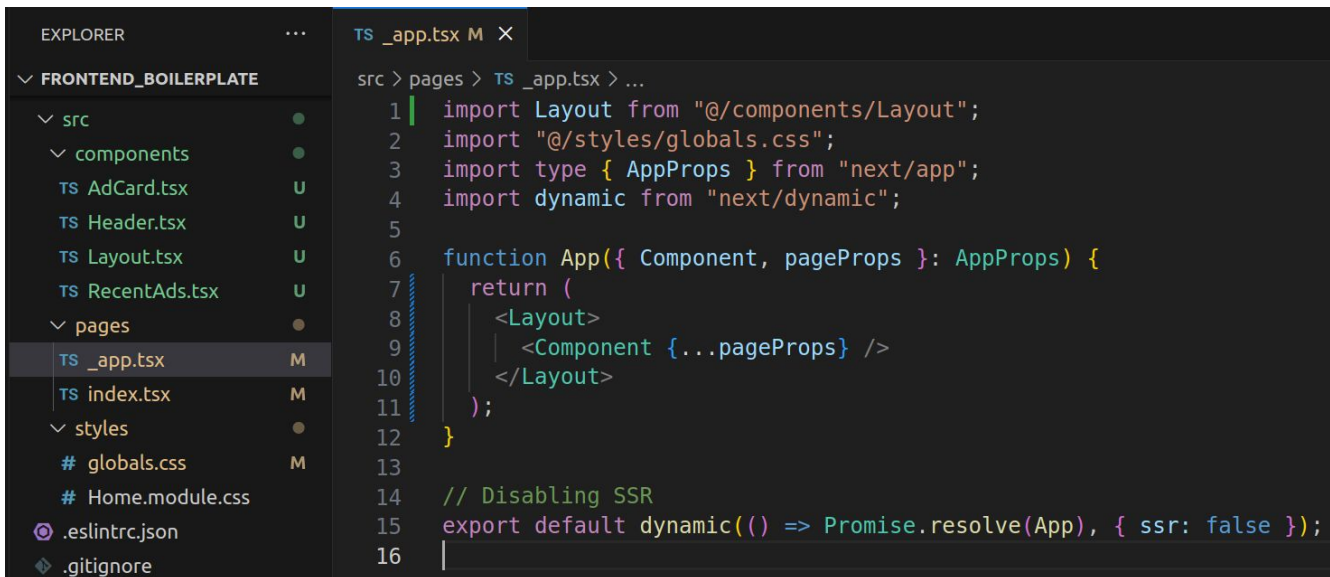
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'src' directory containing 'components' and 'pages'. The 'components' directory contains 'AdCard.tsx', 'Header.tsx', 'Layout.tsx', and 'RecentAds.tsx'. The 'pages' directory contains '_app.tsx' and 'index.tsx'. The 'styles' directory contains 'globals.css' and 'Home.module.css'. The code editor shows the implementation of the 'Layout' component in 'src > components > TS Layout.tsx > ...'. The code is as follows:

```
1 import Head from "next/head";
2 import Header from "../Header";
3 import { ReactNode } from "react";
4
5 const Layout = ({ children }: { children: ReactNode }) => {
6   return (
7     <>
8       <Head>
9         <title>Create Next App</title>
10        <meta name="description" content="Generated by create next app" />
11        <meta name="viewport" content="width=device-width, initial-scale=1" />
12        <link rel="icon" href="/favicon.ico" />
13      </Head>
14      <Header />
15      <main className="main-content">{children}</main>
16    </>
17  );
18 };
19
20 export default Layout;
```



Layout

Utilise le composant **Layout** dans **_app.tsx** pour qu'il soit présent dans toutes les pages de l'application:



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays the project structure under 'FRONTEND_BOILERPLATE'. The 'pages' folder is expanded, showing 'TS _app.tsx' (marked with an 'M' icon) and 'TS index.tsx' (marked with an 'M' icon). The main editor area shows the code for 'TS _app.tsx'. The code imports the 'Layout' component from '@components/Layout', imports global styles, and defines an 'App' function that wraps the page component in the 'Layout' component. The 'App' function is then exported as the default export, wrapped in a 'dynamic' function to disable SSR.

```
1 | import Layout from "@components/Layout";
2 | import "@styles/globals.css";
3 | import type { AppProps } from "next/app";
4 | import dynamic from "next/dynamic";
5 |
6 | function App({ Component, pageProps }: AppProps) {
7 |   return (
8 |     <Layout>
9 |       <Component {...pageProps} />
10 |    </Layout>
11 |  );
12 | }
13 |
14 | // Disabling SSR
15 | export default dynamic(() => Promise.resolve(App), { ssr: false });
16 |
```



Layout

Retire le composant **Header** du fichier **index.tsx** et constate que le Header apparaît toujours dans le navigateur car il fait maintenant partie du layout

The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for 'FRONTEND_BOILERPLATE'. The tree includes a 'src' directory with subdirectories 'components' and 'pages'. Under 'components', there are files 'AdCard.tsx', 'Header.tsx', 'Layout.tsx', and 'RecentAds.tsx', all marked with a green dot and a 'U' icon. Under 'pages', there are files '_app.tsx' and 'index.tsx', both marked with a green dot and an 'M' icon. The 'index.tsx' file is selected. On the right, the editor shows the code for 'index.tsx'. The code starts with an import statement for 'RecentAds' from '@components/RecentAds'. It then defines a default export function 'Home()' which returns a JSX element. The JSX element consists of a root element with a single child, a 'body' element. The 'body' element has a single child, a 'main' element with the class 'main-content'. The 'main' element has a single child, a 'RecentAds' component. The code ends with a closing tag for the root element. The line numbers 1 through 14 are visible on the left side of the code editor.

```
1 import RecentAds from "@components/RecentAds";
2
3 export default function Home() {
4   return (
5     <>
6       <body>
7         <main className="main-content">
8           <RecentAds />
9         </main>
10      </body>
11    </>
12  );
13 }
14
```




About

Crée une nouvelle page About et vérifie que tu y accèdes bien via l'url /about

The image shows a web browser on the left and a code editor on the right, illustrating the process of creating an 'About' page.

Browser View (Left): The browser is at `http://localhost:3000/about`. The page has a header with 'THE GOOD CORNER', a search bar, and a 'Publier une annonce' button. Below the header is a horizontal list of categories: Ameublement, Électroménager, Photographie, Informatique, Téléphonie, Vélos, Véhicules, Sport, Habillement, Bébé, and Outillage. The main content area displays the text: 'This project was made with love, Typescript and NextJS for educational purposes.'

Code Editor View (Right): The code editor shows the file explorer with the 'FRONTEND_BOILERPLATE' project structure. The 'pages' directory is expanded, showing 'TS _app.tsx', 'TS about.tsx' (selected), and 'TS index.tsx'. The 'TS about.tsx' file is open in the editor, showing the following code:

```
1 const about = () => {
2   return (
3     <p>
4       This project was made with
5       purposes.
6     </p>
7   );
8 };
9
10 export default about;
11
```



AdDetails

Nous allons maintenant créer une page permettant d'afficher les détails d'une annonce en particulier

L'id de l'annonce sera contenue dans l'url

Nous allons pour cela utiliser la fonctionnalité de routing dynamique de NextJS

<https://nextjs.org/docs/pages/building-your-application/routing/dynamic-routes>



AdDetails

Crée un dossier **ad** avec un fichier **[id].tsx** à l'intérieur

Utilise ensuite le **hook useRouter** pour afficher dans la console l'**url**, celle-ci nous servira plus tard à identifier l'annonce dont on souhaite afficher le détail

<https://nextjs.org/docs/pages/api-reference/functions/use-router>



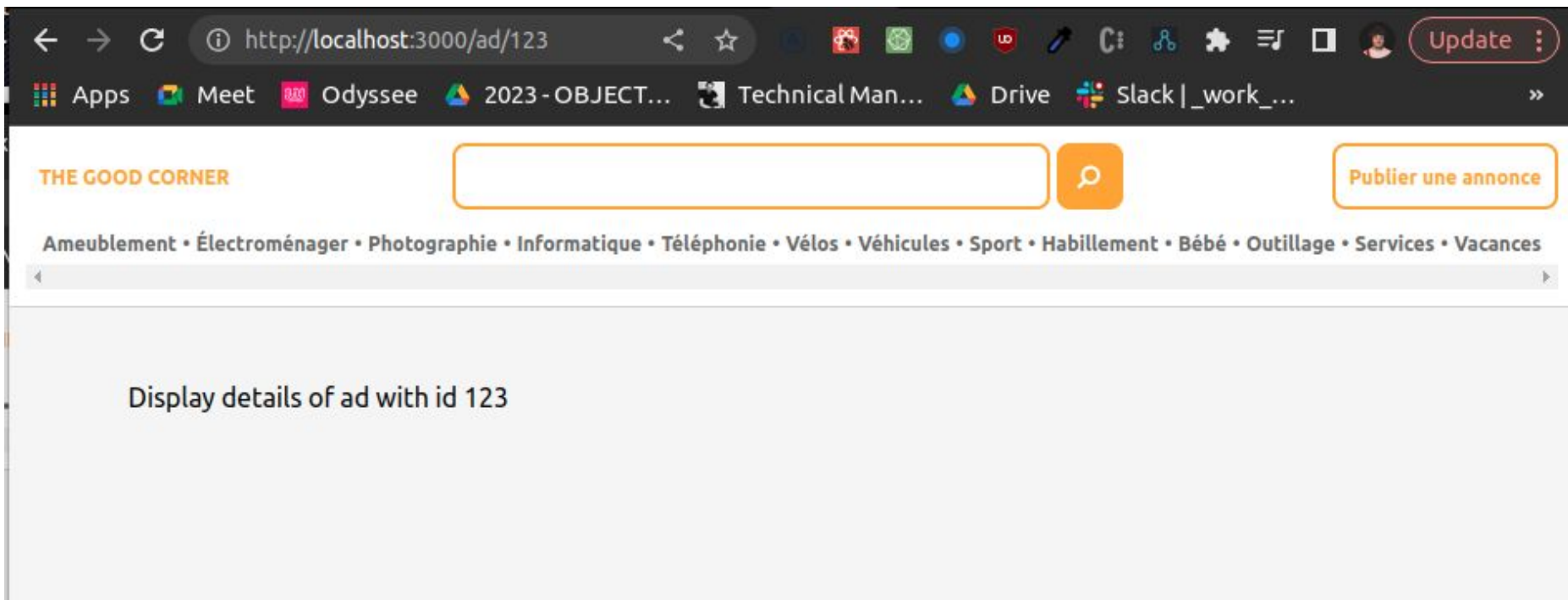
AdDetails

[id].tsx - Frontend_boilerplate - Visual Studio Code

EXPLORER	...	TS [id].tsx U X
▼ FRONTEND_BOILERPLATE		src > pages > ad > TS [id].tsx > ...
TS Header.tsx	U	1 import { useRouter } from "next/router";
TS Layout.tsx	U	2
TS RecentAds.tsx	U	3 const AdDetailComponent = () => {
▼ pages	●	4 const router = useRouter();
▼ ad	●	5 console.log(router);
TS [id].tsx	U	6 return <p>Display details of ad with id {router.query.id}</p>;
TS _app.tsx	M	7 };
TS about.tsx	U	8
TS index.tsx	M	9 export default AdDetailComponent;
▼ styles	●	10
# globals.css	M	

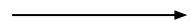


AdDetails





A toi de jouer !



- Crée le layout
- Crée les différentes pages du projet frontend et vérifie le fonctionnement de l'url dynamique