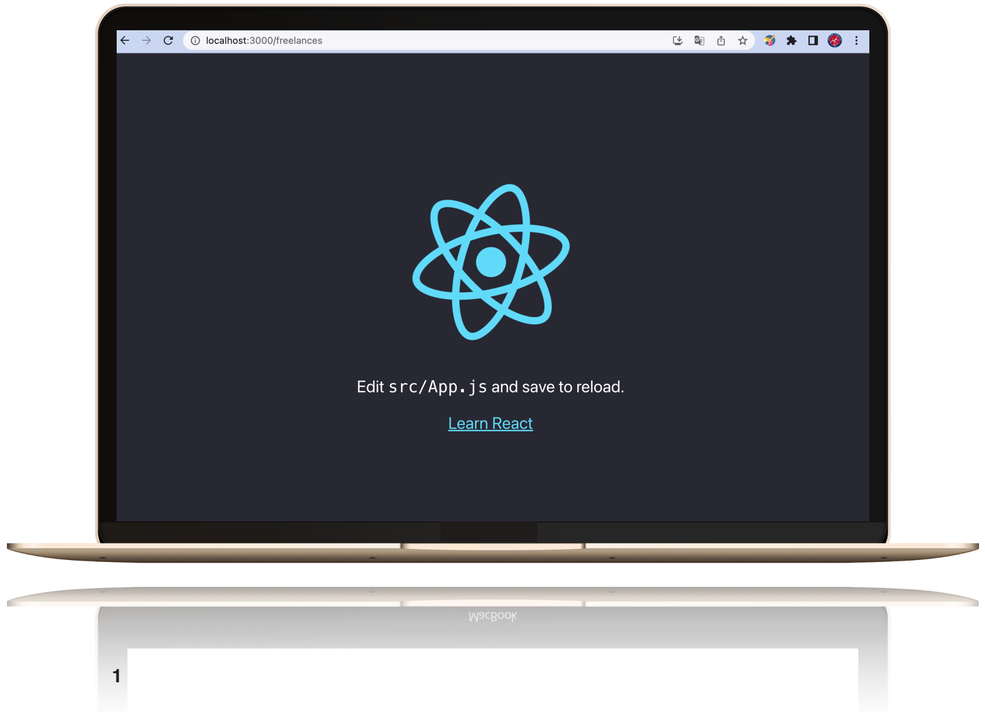
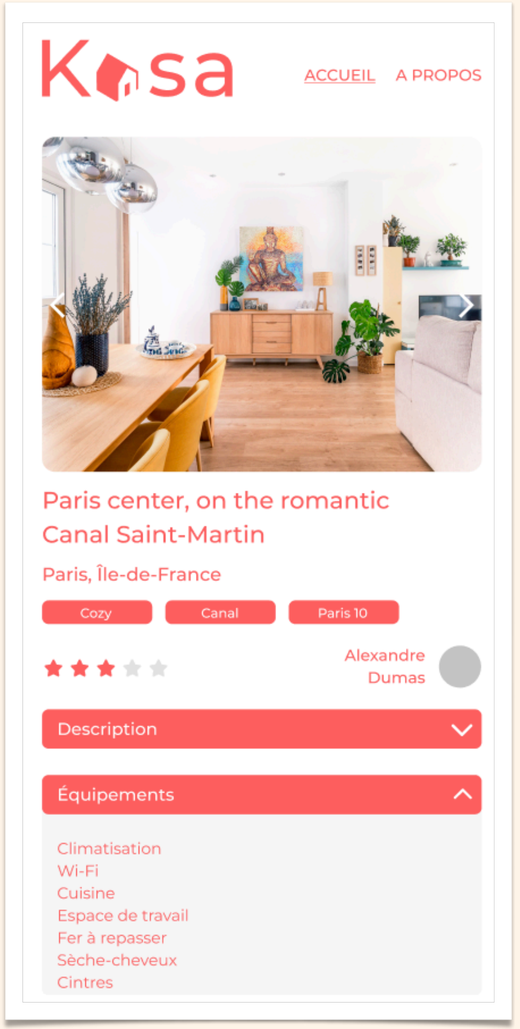
Créez une application web de location immobilière avec React

Livrable

1. Un lien vers le dépôt GitHub contenant l'ensemble du code de l'application, ainsi que le code React Router pour les routes dans un fichier dédié.

https://github.com/GuillaumePevrier/OC_Projet6





Créez une application web de location immobilière avec React

Introduction (1 minute)

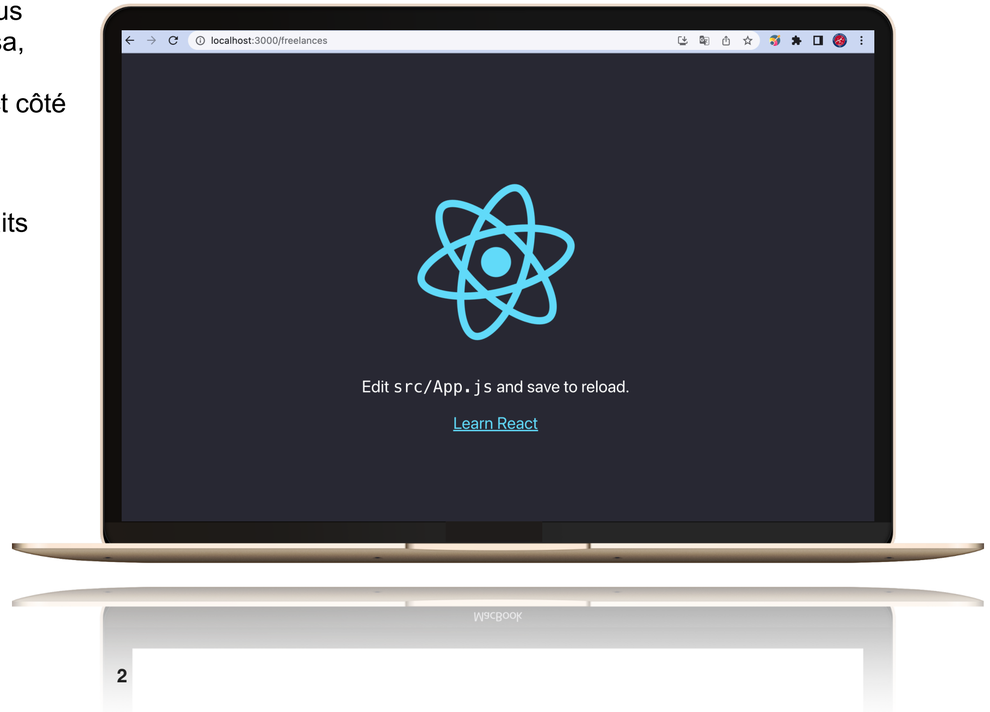
Bienvenue,

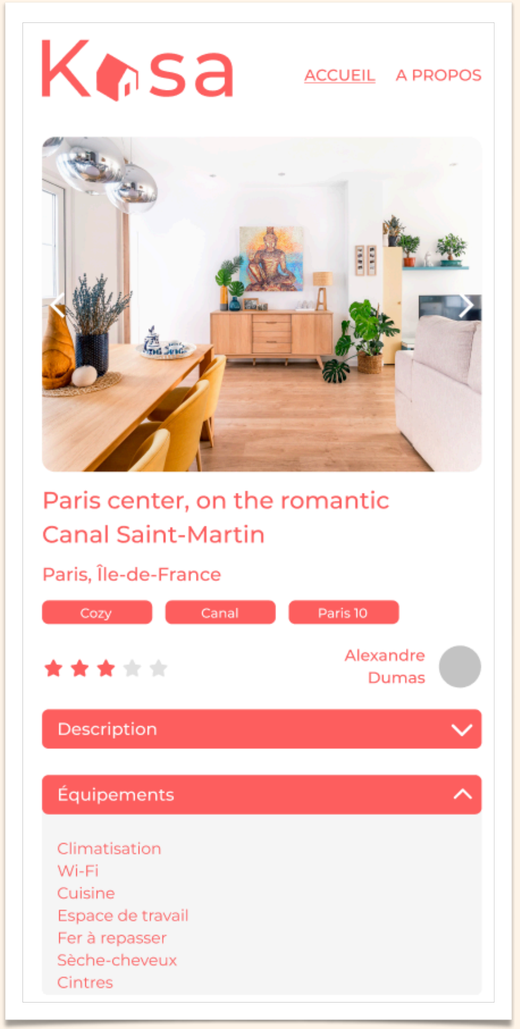
Aujourd'hui, je suis ravi de vous présenter notre projet de création d'une application web de location immobilière avec React pour la société Kasa. Le but de cette application est de faciliter la location d'appartements entre particuliers en proposant une plateforme simple, intuitive et sécurisée. Notre public cible est donc constitué de personnes qui cherchent à louer un logement en toute confiance.

Ce projet a été développé dans le contexte de la refonte totale de la plateforme web de Kasa, qui avait été codée il y a plus de 10 ans en ASP.NET. En tant que développeur front-end en freelance, nous avons eu l'opportunité de travailler avec Laura, la CTO de Kasa, pour créer une nouvelle application web en utilisant une stack complète en JavaScript, avec NodeJS côté back-end, et React côté front-end.

Nous allons vous présenter les différentes fonctionnalités de l'application, ainsi que les choix techniques que nous avons faits pour la mettre en place. Nous allons également répondre aux questions que Laura, notre CTO pour l'occasion, pourra nous poser sur notre méthodologie et sur les livrables que nous allons présenter.

Nous sommes fiers de vous présenter notre travail, et nous espérons que vous apprécierez notre présentation.





Créez une application web de location immobilière avec React

Présentation des livrables (5 minutes)

Le projet utilise les composants React suivants :

Header : affiche la barre de navigation en haut de la page. Il contient le logo et la barre de navigation qui est stylisée à l'aide de styled-components.

Footer : affiche le bas de la page avec un logo et un texte de copyright. Les styles CSS sont définis dans le composant StyledFooter, qui utilise un fond sombre et du texte blanc.

Gallery : affiche une galerie d'images de logements à partir du fichier "logements.json". Chaque image est affichée dans une carte cliquable qui redirige vers la page du logement correspondant. Si une carte est cliquée, un composant "Fiche" s'affiche avec plus d'informations sur le logement sélectionné. La galerie est stylisée avec une mise en page de grille flexible qui utilise les propriétés CSS "display: grid" et "grid-template-columns".

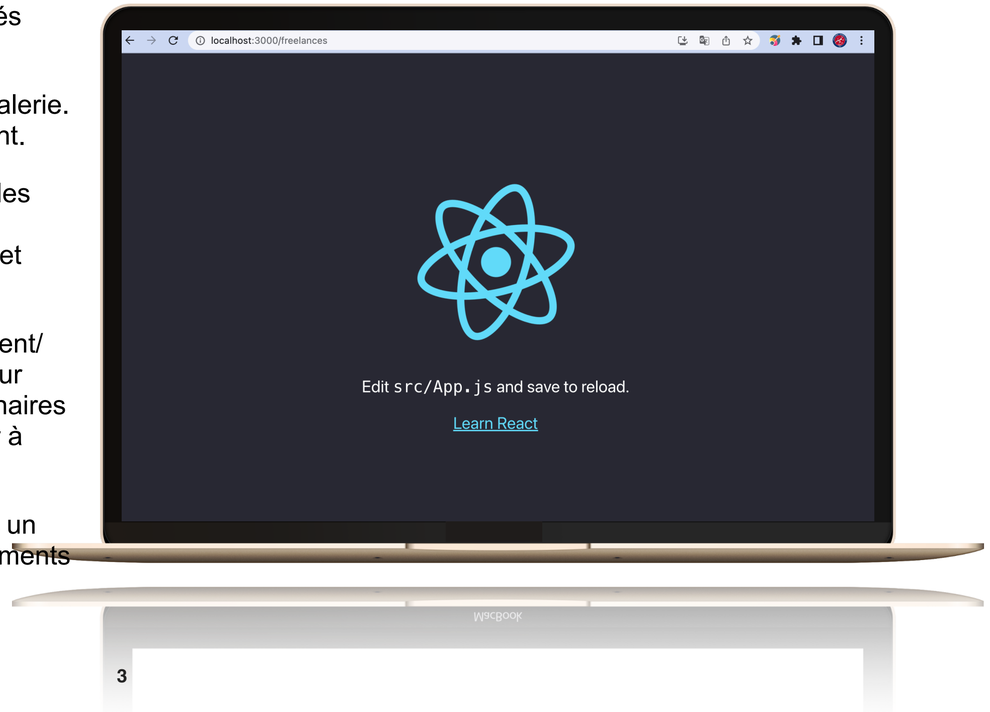
Fiche : affiche les détails d'un logement sélectionné dans la galerie. Il utilise le composant Rating pour afficher le score du logement.

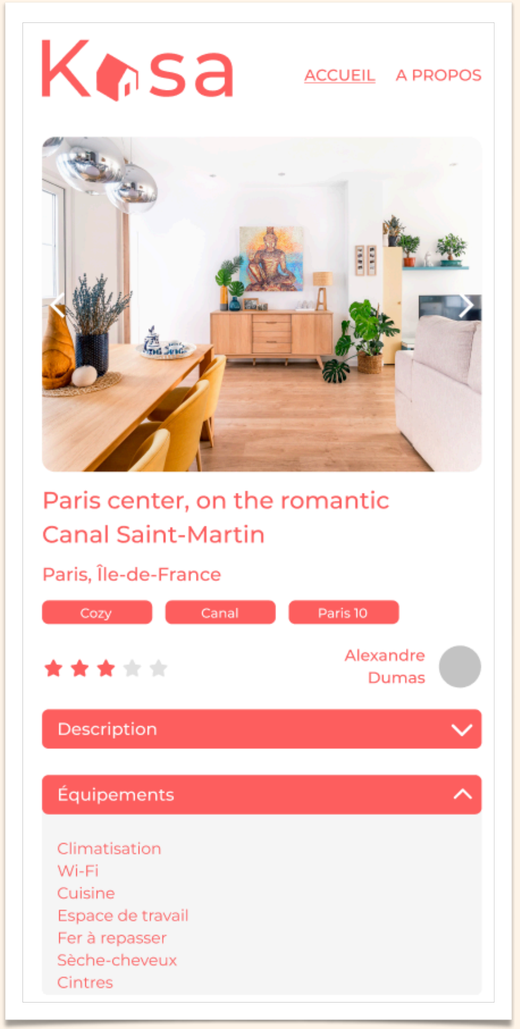
Rating : récupère un score et affiche un certain nombre d'étoiles pleines et un certain nombre d'étoiles vides pour un total de 5 étoiles. Les étoiles sont représentées par les icônes solidStar et regularStar de la bibliothèque FontAwesome.

Carousel est un carrousel d'images avec des boutons précédent/suivant et un compteur d'images. Il utilise le hook useState pour suivre l'index de l'image active actuelle et définit deux gestionnaires d'événements de clic pour permettre à l'utilisateur de naviguer à travers les images.

Accordion est une liste déroulante de contenus stockés dans un fichier JSON. Il utilise styled-components pour le style des éléments et crée un conteneur d'accordéon avec un en-tête et un contenu pour chaque élément dans le tableau "accordionData".
Lorsque l'utilisateur clique sur l'en-tête, l'état de l'élément est modifié pour afficher ou masquer le contenu.
La fonction utilise useState() pour créer un tableau "activeIndexes" rempli de "true" (Accordion ouvert) avec une longueur égale à la longueur de "accordionData", et utilise également la méthode "handleClick" pour inverser les valeurs booléennes (true ou false) du tableau.

Tous les composants utilisent des **hooks** d'état, tels que **useState**, pour gérer l'état et afficher le contenu en conséquence. Les styles sont définis à l'aide de **styled-components** et appliqués aux éléments **JSX** dans les composants correspondants.





Créez une application web de location immobilière avec React

Définition

- **Hooks d'état** : Les hooks d'état sont des fonctions spéciales en React qui vous permettent de stocker et de gérer les données dans votre application. Ils vous permettent également de mettre à jour ces données et de les réutiliser facilement dans différents composants. Par exemple, si vous voulez stocker le nombre de clics sur un bouton dans votre application, vous pouvez utiliser le hook d'état "useState" pour stocker cette valeur et la mettre à jour à chaque clic.

- **useState** : "useState" est un hook d'état en React qui vous permet de stocker des données et de les mettre à jour dans votre application. Par exemple, vous pouvez utiliser "useState" pour stocker un nombre de clics sur un bouton et mettre à jour ce nombre chaque fois que le bouton est cliqué.

- **Fichier .JSX** : Un fichier .JSX est un fichier JavaScript qui contient du code en JSX, une syntaxe spéciale qui permet d'écrire du code HTML dans votre JavaScript. Les fichiers .JSX sont utilisés en React pour créer des composants qui contiennent à la fois du code HTML et du code JavaScript. Par exemple, vous pouvez créer un composant "Button" en utilisant un fichier "Button.jsx" qui contient du code HTML pour un bouton ainsi que du code JavaScript pour ajouter une fonctionnalité de clic.

- **Navigation ROOT** : La navigation ROOT est le point d'entrée principal d'une application. C'est là que votre application est chargée et où vous pouvez naviguer vers d'autres pages ou composants. Dans une application React, la navigation ROOT est généralement définie dans un composant "App" qui est chargé en premier. Par exemple, vous pouvez définir la navigation ROOT de votre application dans le composant "App.jsx" et naviguer vers d'autres pages ou composants à partir de là.