**1. Introduction (1 minute)**

**Contenu :** Présentez le projet Kasa, une application web construite avec React qui permet de visualiser des appartements à louer. Expliquez l'objectif de la présentation : donner un aperçu de l'architecture du projet, des technologies utilisées, et des fonctionnalités clés.

**2. Initialisation et Configuration du Projet (2 minutes)**

**Contenu :** Expliquez que le projet est initialisé avec Create React App pour simplifier la configuration initiale. Mentionnez l'utilisation de React Router pour la gestion des routes et de Sass pour les styles. Vous pouvez décrire brièvement le processus d'installation des librairies via Yarn ou npm, en soulignant l'importance du fichier package.json pour gérer les dépendances.

**Mise en place de l'environnement React**

1. **Initialisation du projet :**
   * Création du projet avec create-react-app pour une configuration initiale rapide.
   * Organisation des fichiers et des dossiers selon une structure logique et modulaire.
2. **Installation des dépendances :**
   * Utilisation de axios pour la gestion des requêtes HTTP.
   * Installation de sass pour permettre l'écriture de styles en SASS.

**3. Structure du Projet (3 minutes)**

**Contenu :** Détaillez la structure du dossier, en expliquant la séparation entre les composants réutilisables dans src/components et les vues spécifiques dans src/pages. Mentionnez le dossier utils pour les fonctions auxiliaires comme les appels API. Expliquez comment cette organisation favorise la maintenance et l'évolutivité de l'application.

**4. Utilisation des Hooks et Props (4 minutes)**

**Contenu :**

**useState et useEffect**

* **useState** : Ce hook permet de gérer l'état local dans un composant fonctionnel. Par exemple, dans Home.js, nous utilisons useState pour créer une variable d'état flats qui stocke la liste des appartements récupérés depuis l'API. Voici comment cela est mis en œuvre :

javascript

Copier le code

const [flats, setFlats] = useState([]); // Initialisation de l'état avec un tableau vide

Cet état est ensuite mis à jour avec les données récupérées de l'API, permettant de rendre dynamiquement une liste de FlatCard basée sur les données chargées.

* **useEffect** : Ce hook est utilisé pour exécuter des effets de bord (comme des appels API) après le rendu du composant. Dans Flat.js, useEffect est utilisé pour charger les détails d'un appartement spécifique, basé sur un id récupéré de l'URL. L'exemple suivant montre comment cet effet est configuré pour s'exécuter une seule fois à l'initialisation du composant, ou à chaque fois que l'id change :

javascript

Copier le code

useEffect(() => {

const fetchFlats = async () => {

const data = await getFlats();

const flat = data.find(flat => flat.id === id);

setFlat(flat);

};

fetchFlats();

}, [id]); // Dépendance sur l'ID pour recharger les données si l'ID change

**useParams**

* **useParams** : Ce hook de react-router-dom permet de récupérer les paramètres de l'URL. Il est crucial pour des pages qui dépendent de paramètres dynamiques, comme la page Flat. Dans Flat.js, useParams est utilisé pour extraire l'id de l'appartement directement de l'URL :

javascript

Copier le code

const { id } = useParams(); // Extraction de l'ID de l'appartement depuis l'URL

**Props**

* **Props** : Les props sont utilisées pour passer des données et des gestionnaires d'événements entre composants. Dans Home.js, les props sont passées au composant FlatCard pour afficher chaque appartement. Chaque FlatCard reçoit un id, un title et une image de couverture (cover) pour afficher correctement les informations de l'appartement :

javascript

Copier le code

flats.map(flat => (

<FlatCard key={flat.id} id={flat.id} title={flat.title} cover={flat.cover} />

))

Cette méthode permet de rendre chaque carte d'appartement de manière dynamique en fonction des données récupérées, en utilisant les informations spécifiques passées via les props.

### Cette section montre comment les hooks et les props sont essentiels pour la gestion des états, des effets de bord, et le passage de données dans votre application React, facilitant la création d'une interface utilisateur interactive et réactive. **Utilisation de Props pour Personnaliser Accordion**

**1. Props de Personnalisation**

Vous avez utilisé des props spécifiques telles que origin et alignment pour adapter l'apparence et la configuration de l'**Accordion** selon la page où il est utilisé. Ces props sont passées au composant **Accordion** et sont utilisées pour appliquer des classes CSS conditionnelles qui modifient le style de l'**Accordion**.

**Exemple de Code**

**Dans la page About.js**

jsx

Copier le code

<Accordion origin="about" items={items} />

* **origin="about"**: Cette prop pourrait être utilisée pour appliquer un style spécifique lorsque l'**Accordion** est utilisé dans la page About. Par exemple, cela peut affecter la mise en page, les couleurs ou d'autres éléments de style qui correspondent au thème de la page About.

**Dans la page Flat.js**

jsx

Copier le code

<Accordion origin="flat" items={items} alignment="flat" />

* **origin="flat"**: Similaire à ci-dessus, mais spécifique à la page Flat. Cela indique que l'**Accordion** doit adopter un style ou une configuration différente adaptée à la page Flat.
* **alignment="flat"**: Cette prop supplémentaire pourrait être utilisée pour ajuster l'alignement ou la disposition des éléments de l'**Accordion** spécifiquement pour la page Flat, influençant comment les éléments sont visualisés en termes de disposition spatiale.

**5. Responsive Design et Version Mobile (2 minutes)**

**Contenu :** Parlez de l'approche mobile-first ou responsive design adoptée dans le projet. Expliquez comment les styles Sass sont utilisés pour adapter l'application aux différents appareils, en assurant une expérience utilisateur cohérente sur desktop et mobile.

**6. Réutilisabilité et Maintenance (2 minutes)**

**Contenu :** Mettez en avant comment la création de composants réutilisables comme Banner et Accordion aide à réduire la redondance du code et améliore la maintenance. Expliquez également comment les hooks personnalisés (potentiellement dans src/hooks) pourraient être utilisés à l'avenir pour encapsuler et réutiliser la logique d'état ou d'effets.

**7. Conclusion (1 minute)**

**Contenu :** Résumez les points clés abordés et soulignez les bénéfices de l'architecture choisie pour Kasa, en particulier en termes de maintenabilité, évolutivité et expérience utilisateur. Invitez votre audience à poser des questions ou à explorer le code plus en détail si possible.

En structurant votre présentation autour de ces points, vous pourrez couvrir efficacement tous les aspects importants du projet Kasa en 15 minutes, tout en laissant de la place pour des explications détaillées et des interactions avec l'audience.