

Architecture de dossiers



Write date here

Information available in audio.



Pourquoi une bonne architecture de dossiers est essentielle ?

1. Lisibilité et compréhension du projet :

Un projet avec des dossiers bien nommés (e.g., components, services, utils) est immédiatement compréhensible.

2. Facilité de maintenance et de refactoring :

Facilité de maintenance et de refactoring, modifier une fonctionnalité dans un projet bien structuré est plus facile

3. Collaboration efficace :

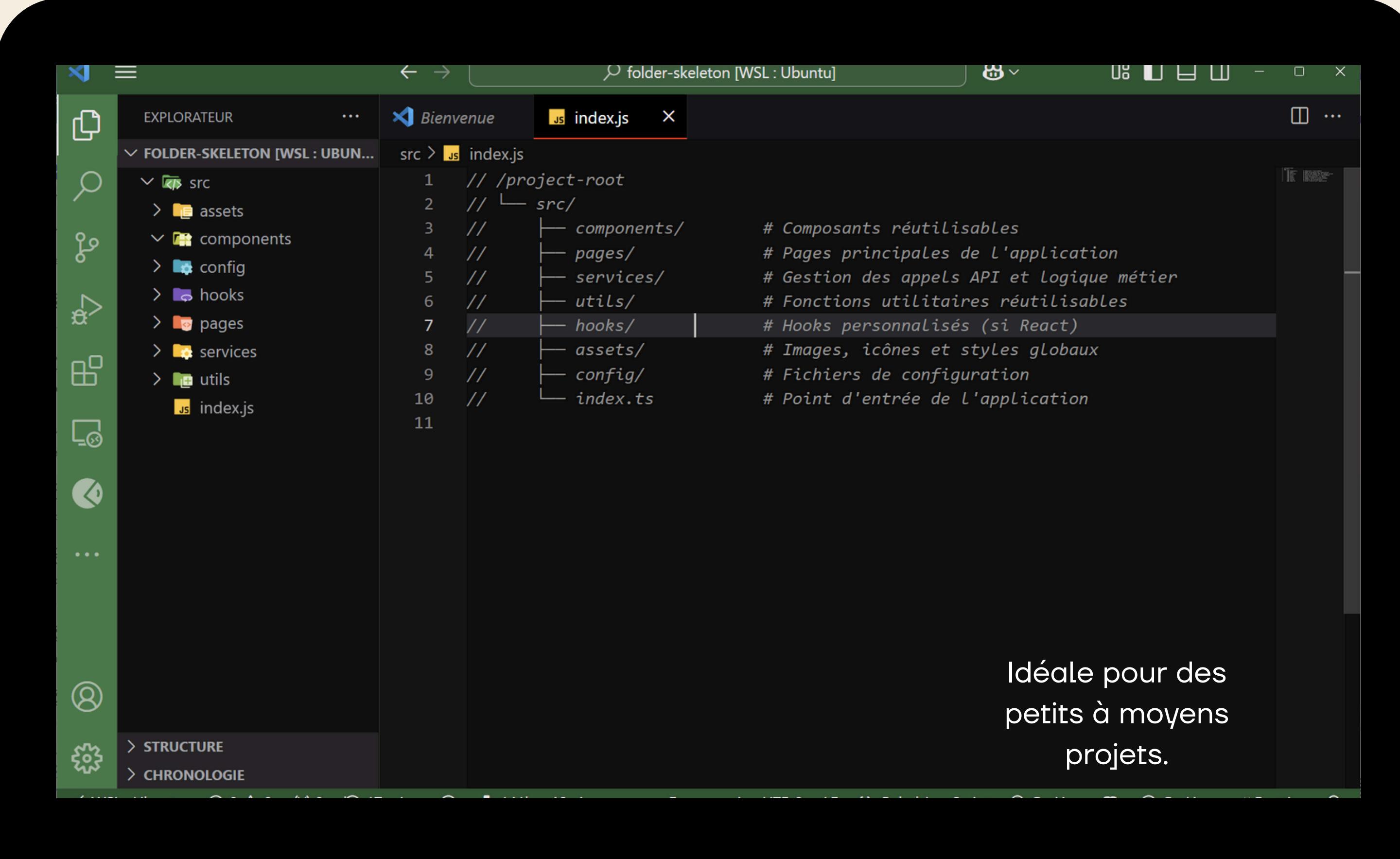
Un développeur entrant dans un projet peut se concentrer sur une fonctionnalité spécifique sans avoir à rechercher dans tout le code.

4. Évolutivité et modularité :

Ajouter un module dans une application sans avoir à réorganiser tout le projet.



Architecture basée sur les types de fichiers



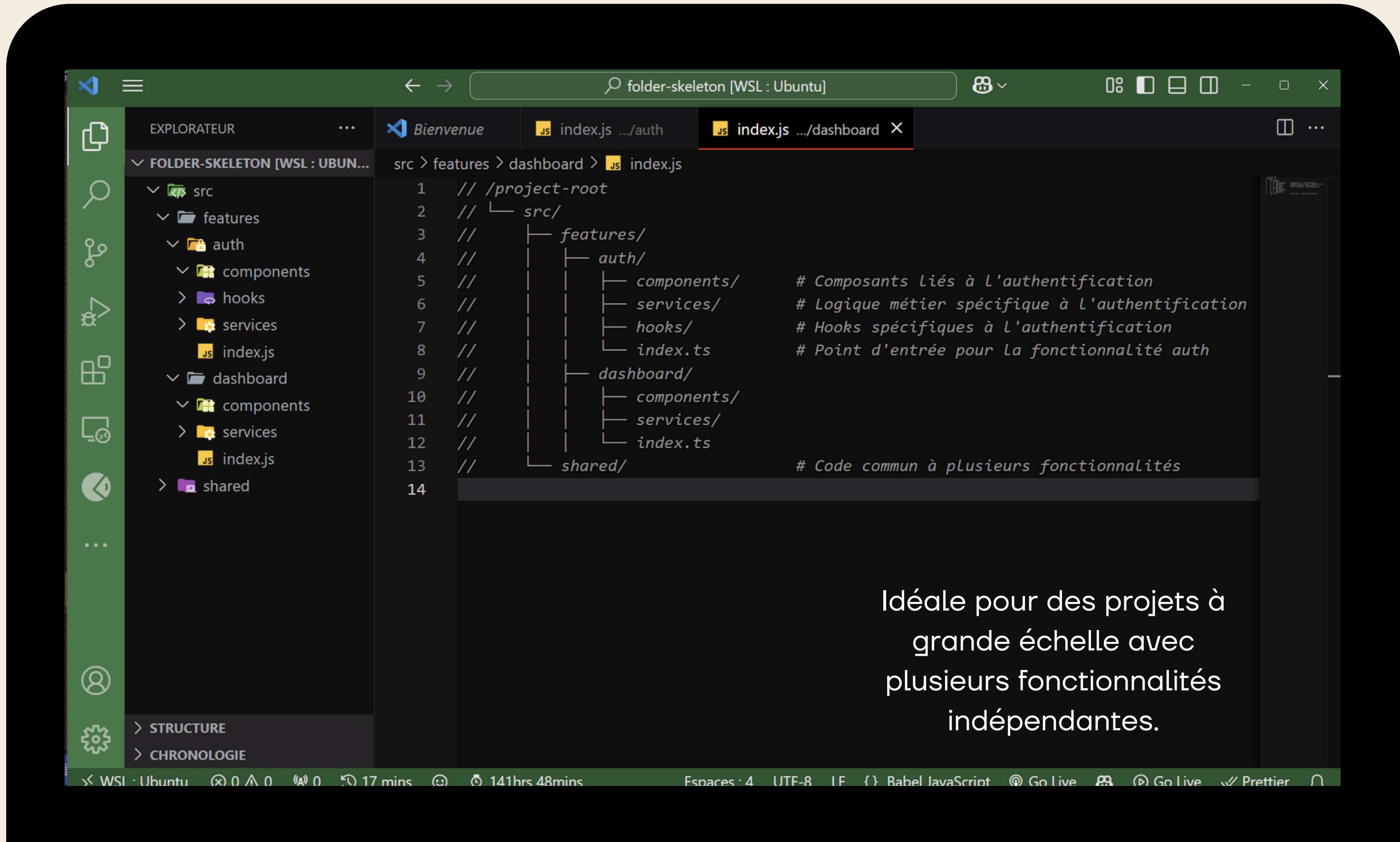
The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) running in WSL (Ubuntu). The left sidebar displays a file tree for a project named "FOLDER-SKELETON [WSL : UBUN...]" under the "src" directory. The tree includes "assets", "components", "config", "hooks", "pages", "services", "utils", and two "index.js" files. The main editor area shows the content of "index.js". The code is a multi-line comment defining a project structure:

```
// /project-root
//   src/
//     components/      # Composants réutilisables
//     pages/           # Pages principales de l'application
//     services/        # Gestion des appels API et logique métier
//     utils/           # Fonctions utilitaires réutilisables
//     hooks/           # Hooks personnalisés (si React)
//     assets/          # Images, icônes et styles globaux
//     config/          # Fichiers de configuration
//     index.ts         # Point d'entrée de l'application
```

A pink ribbon graphic is visible on the right side of the slide.

Idéale pour des
petits à moyens
projets.

Architecture feature-based



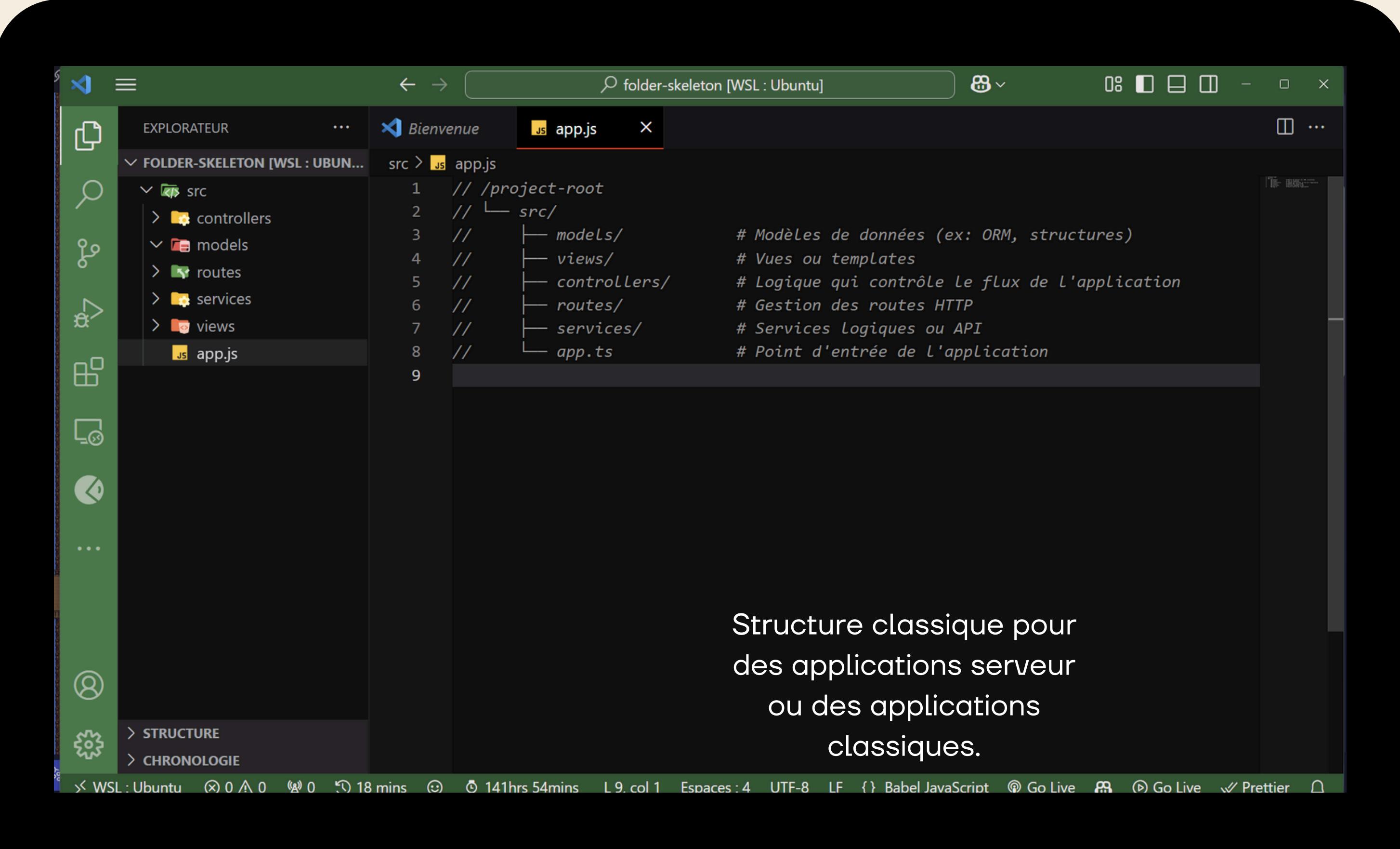
The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the title bar "folder-skeleton [WSL : Ubuntu]". The left sidebar is titled "EXPLORATEUR" and shows a file tree for "FOLDER-SKELETON [WSL : UBUN...]" containing a "src" directory with "features" and "shared" sub-directories, and "auth" and "dashboard" functional areas. The "auth" area contains "components", "hooks", "services", and an "index.js" file. The "dashboard" area also contains "components", "services", and an "index.js" file. The "shared" directory is empty. The right pane displays the contents of the "index.js" file under "auth". The code is as follows:

```
// /project-root
// └── src/
//     ├── features/
//     |   ├── auth/
//     |   |   ├── components/
//     |   |   ├── services/
//     |   |   ├── hooks/
//     |   |   └── index.ts
//     |   └── dashboard/
//     |       ├── components/
//     |       ├── services/
//     |       └── index.ts
//     └── shared/           # Code commun à plusieurs fonctionnalités
```

A callout box from the bottom-left corner points to the "STRUCTURE" button in the Explorer sidebar, which is highlighted in grey. A pink wavy line graphic is visible on the right side of the slide.

Idéale pour des projets à grande échelle avec plusieurs fonctionnalités indépendantes.

Architecture MVC



The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) running in WSL (Ubuntu). The title bar indicates the workspace is named "folder-skeleton [WSL : Ubuntu]". The left sidebar features a green "Explorateur" (File Explorer) icon and a vertical toolbar with icons for search, file operations, and more. The main editor area displays the "app.js" file from the "src" directory. The code is a comment-based project structure:

```
// /project-root
//   src/
//     models/          # Modèles de données (ex: ORM, structures)
//     views/           # Vues ou templates
//     controllers/    # Logique qui contrôle le flux de l'application
//     routes/          # Gestion des routes HTTP
//     services/        # Services logiques ou API
//     app.ts           # Point d'entrée de l'application
```

Below the code editor, a status bar shows "WSL : Ubuntu" and various system metrics like CPU usage (0%), memory (0%), disk (0%), and network (18 mins, 141hrs 54mins).

On the right side of the slide, there is a decorative graphic element consisting of a thick pink ribbon-like shape.

Structure classique pour
des applications serveur
ou des applications
classiques.

Bonne pratiques de nomage

camelCase	fichiers utilitaires : Exemple : fetchData.js	
PascalCase	composants et les classes : Exemple : UserCard.tsx	
kebab-case	noms de dossiers : Exemple : user-profile/	

Pour terminer

- Ne pas mélanger la logique métier et l'interface utilisateur.
- Diviser les fichiers volumineux en plus petites parties
- Documenter la structure des dossiers dans un fichier README.md

