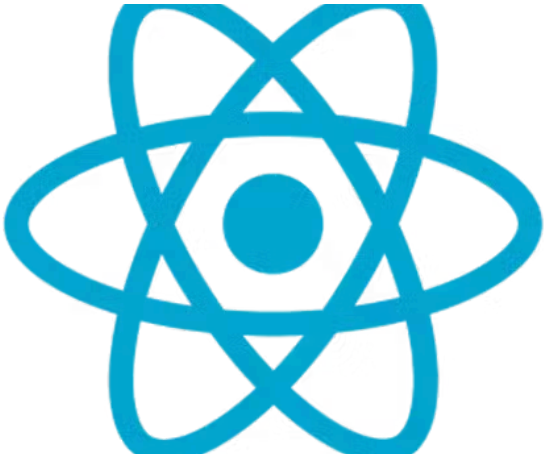

Documentation du Projet

Objectif du Projet

Ce projet vise à créer une application React en TypeScript utilisant **JSONForms** pour afficher dynamiquement des formulaires basés sur des schémas JSON. L'interface utilisateur est construite avec **Ant Design**, et les formulaires sont organisés dans des **cartes (cards)** pour une meilleure expérience utilisateur.

Structure du Projet

```
my-jsonforms-app/
├── public/                 # Fichiers publics (HTML, images, etc.)
├── src/
│   ├── assets/             # Ressources statiques (images, icônes, etc.)
│   ├── components/         # Composants réutilisables
│   │   ├── CardLayout/     # Composant pour afficher les formulaires dans des cartes
│   │   ├── FormView/       # Composant principal pour afficher les formulaires
│   │   └── CustomRenderers/ # Composants personnalisés pour JSONForms
│   ├── data/               # Schémas JSON, UI Schemas et données
│   │   ├── schema.json     # Schéma JSON pour la structure des données
│   │   ├── uischema.json   # UI Schema pour la disposition des formulaires
│   │   └── data.json       # Données par défaut pour les formulaires
│   ├── pages/              # Pages de l'application (si nécessaire)
│   ├── App.tsx             # Composant principal de l'application
```

```
│   ├── main.tsx            # Point d'entrée de l'application
│   └── styles/             # Fichiers de style globaux
├── .env                    # Variables d'environnement
├── .gitignore              # Fichiers ignorés par Git
├── package.json            # Dépendances et scripts
├── tsconfig.json           # Configuration TypeScript
└── README.md               # Documentation du projet
```

Fonctionnement du Projet

1. Schéma JSON

Le schéma JSON définit la structure des données du formulaire. Il décrit les champs disponibles, leurs types et leurs contraintes.

Exemple :

```
{
  "type": "object",
  "properties": {
    "firstName": { "type": "string", "minLength": 2 },
    "lastName": { "type": "string", "minLength": 2 }
  },
  "required": ["firstName", "lastName"]
}
```

2. UI Schema

Le UI Schema définit comment les champs du formulaire sont affichés. Il permet de regrouper les champs dans des cartes et de personnaliser leur disposition.

Exemple :

```
{
  "type": "VerticalLayout",
  "elements": [
    {
```

```
      "type": "VerticalLayout",
      "label": "Informations Personnelles",
      "elements": [
        { "type": "Control", "scope": "#/properties/firstName", "label": "Prénom" },
        { "type": "Control", "scope": "#/properties/lastName", "label": "Nom" }
      ]
    }
  ]
}
```

3. Data

Les données contiennent les valeurs par défaut des champs du formulaire. Elles peuvent être modifiées dynamiquement par l'utilisateur.

Exemple :

```
{
  "firstName": "John",
  "lastName": "Doe"
}
```

4. Composants Personnalisés

Le projet utilise des composants personnalisés pour JSONForms, comme `CardLayout`, qui encapsule les champs dans des cartes Ant Design.

Exemple de `CardLayout` :

```
import React from 'react';
import { Card } from 'antd';
import { LayoutProps } from '@jsonforms/core';

const CardLayout: React.FC<LayoutProps> = (props) => {
  return (
    <Card title={props.uischema.label} style={{ marginBottom: 16 }}>
      {props.uischema.elements.map((element, index) =>
        props.renderers.find((r) => r.tester(element, props.schema))?.renderer(
```

```
            element,
            props.schema,
            props.path,
            props.enabled,
            props.visible,
            props.renderers,
            props.cells
          )
        )}
      </Card>
    );
  };

  export default CardLayout;
```

5. Affichage des Formulaires

Le composant `FormView` utilise JSONForms pour afficher les formulaires en fonction du schéma, du UI Schema et des données fournis.

Exemple :

```
  import React from 'react';
  import { JsonForms } from '@jsonforms/react';
  import { materialRenderers, materialCells } from '@jsonforms/material-renderers';
  import CardLayout from './CardLayout/CardLayout';

  const renderers = [
    ...materialRenderers,
    {
      tester: (uischema, schema) => uischema.type === 'VerticalLayout' && uischema.label ? 1 : -1,
      renderer: CardLayout,
    },
  ];

  const FormView: React.FC = ({ schema, uischema, data, onChange }) => {
    return (
      <JsonForms
        schema={schema}
        uischema={uischema}
```

```
        data={data}
        renderers={renderers}
        cells={materialCells}
        onChange={({ data }) => onChange(data)}
      />
    );
  };

  export default FormView;
```

Comment Utiliser le Projet

Ref in Readme

Personnalisation

- **Ajouter des formulaires** : Modifiez les fichiers `schema.json`, `uischema.json` et `data.json` dans le dossier `src/data/`.
- **Modifier le style** : Utilisez les classes CSS ou les propriétés `style` d'Ant Design pour personnaliser l'apparence.
- **Ajouter des composants personnalisés** : Créez de nouveaux composants dans `src/components/CustomRenderers/`.

Technologies Utilisées

- **React** avec **TypeScript**
- **JSONForms** pour la gestion des formulaires
- **Ant Design** pour l'interface utilisateur
- **Pro Layout** (optionnel) pour la mise en page