DiRVED - Tâches et responsabilités actuelles

> Personnes impliquées: Aira TRIBORD

Programmation du logiciel de gestion conventions & prestations

Objectifs

Cet outil a été conçu pour aider la DiRVED à gérer efficacement leurs activités liées aux conventions et prestations. Il offre une interface intuitive et des fonctionnalités pour suivre les conventions, planifier les prestations et gérer les données clients. Ce logiciel utilise des technologies telles qu'Electron, React, Express et MongoDB pour offrir une expérience utilisateur fluide et une gestion des données fiable.

Logiciel de gestion des conventions et des prestations

La template utilisé est la suivante : Electron React Boilerplate

> Documentation : Electron React Boilerplate

La base de données utilisée est MongoDB.

Prérequis

Les prérequis pour exécuter ce logiciel sont les suivants :

- **Node.js v16.13.1** : ce logiciel nécessite une installation de Node.js sur votre ordinateur. Vous pouvez télécharger et installer Node.js depuis le site web officiel de Node.js.
- **npm** : npm est le gestionnaire de paquets intégré à Node.js. Il est inclus avec l'installation de Node.js.
- **MongoDB** : ce logiciel utilise MongoDB comme base de données. Vous devez donc avoir une installation de MongoDB en cours d'exécution sur votre ordinateur. Vous pouvez télécharger et installer MongoDB depuis le site web officiel de MongoDB.

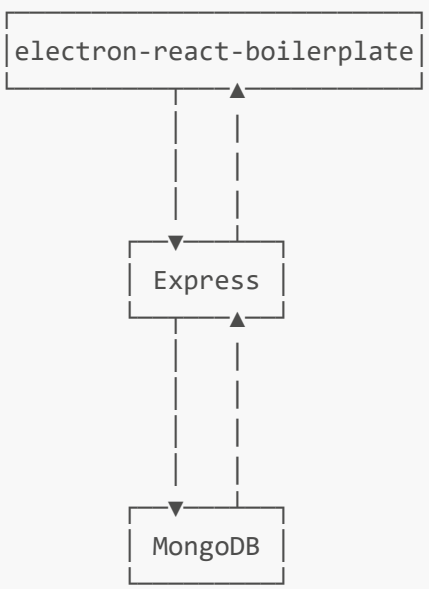
Pour installer ces prérequis, suivez les instructions de leurs sites web respectifs. Une fois les prérequis installés, vous pouvez cloner le référentiel Github pour le logiciel et installer les dépendances en utilisant npm, comme décrit dans la section Environnement de développement.

Architecture

L'architecture du logiciel est basée sur la combinaison de différents composants, notamment electron-react-boilerplate, Express et MongoDB. Voici une brève description de comment ces composants interagissent les uns avec les autres :

- **electron-react-boilerplate** : C'est un modèle de projet qui permet de démarrer rapidement un projet Electron en utilisant React comme bibliothèque pour le développement d'interface utilisateur. Il comprend également des outils pour faciliter le développement et le déploiement d'une application Electron-React.
- **Express** : Express est un framework pour développer des applications web à l'aide de Node.js. Il est utilisé pour mettre en place le serveur web et les API qui permettront à l'application Electron d'interagir avec la base de données MongoDB.
- **MongoDB** : MongoDB est une base de données NoSQL qui stocke les données de l'application. Les données sont accessibles via les API Express et utilisées pour alimenter les composants React dans l'application Electron.

En résumé, electron-react-boilerplate fournit la structure de base pour l'application Electron-React, tandis qu'Express permet de gérer les communications entre l'application et la base de données MongoDB. MongoDB stocke les données qui seront utilisées par l'application pour fonctionner. Chacun de ces composants joue un rôle clé dans la mise en œuvre de la solution de gestion de conventions et de prestations.



Environnement de développement

Pour configurer l'environnement de développement pour ce logiciel, vous devrez suivre les étapes suivantes :

1. Installez les outils nécessaires

Pour développer ce logiciel, vous aurez besoin d'avoir installé Node.js v16.13.1, npm et Git sur votre ordinateur.

Clonez le référentiel Github : pour obtenir le code source du logiciel, vous pouvez cloner le référentiel Github à l'adresse `https://github.com/NicoLarson/gestion-dirved`. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
git clone https://github.com/NicoLarson/gestion-dirved.git
```

2. Installez les dépendances

Une fois le code source cloné, accédez au répertoire du projet et exécutez la commande suivante pour installer toutes les dépendances nécessaires :

```
npm install
```

3. Mettre en place MongoDB

Pour utiliser MongoDB avec ce logiciel, vous devez avoir une installation de MongoDB en cours d'exécution sur votre ordinateur. Vous pouvez télécharger et installer MongoDB depuis le site web officiel de MongoDB.

Une fois MongoDB installé, vous pouvez créer une base de données pour le logiciel en exécutant les commandes MongoDB appropriées. Vous pouvez trouver des informations sur les commandes MongoDB dans la documentation officielle de MongoDB.

4. Démarrez l'application

Une fois toutes les dépendances installées et MongoDB configuré, vous pouvez démarrer l'application en exécutant la commande suivante :

Pour lancer le développement, il faut installer les dépendances du projet avec la commande suivante :

```
npm run install:backend
npm run install:frontend
```

Ensuite, il faut lancer et le serveur de développement avec la commande suivante :

```
npm run dev
```

Maintenant, l'application devrait être en cours d'exécution et vous pourrez démarrer le développement.

Structure du code

Fichiers et dossiers clés

Dossier frontend

Le dossier frontend contient le code de l'application Electron-React. Il est composé des dossiers suivants :

- Dossier `src` : ce dossier contient le code source de l'application Electron-React.
    - Le dossier `main` contient le code source du processus principal de l'application Electron.
    - Le dossier `renderer` contient le code source du processus de rendu de l'application Electron.
    - Le dossier `components` contient les composants React qui seront utilisés dans l'application.
    - Le dossier `pages` contient les pages React qui seront utilisées dans l'application.
- Dossier `assets` : ce dossier contient les fichiers statiques tels que les images, les polices et les fichiers HTML.
- Fichier `package.json` : ce fichier contient les informations sur le projet et les dépendances du projet.
- Fichier `App.js` : ce fichier est l'entrée de l'application Electron-React et il charge les composants React dans le rendu HTML. Il est également responsable de la création de la fenêtre de l'application Electron.

Dossier backend

Le dossier `backend` contient le code du serveur Express qui gère les communications via requêtes HTTP entre l'application Electron-React et la base de données MongoDB. Il est composé des dossiers suivants :

- Le dossier `routes` les contrôleurs qui seront utilisés dans l'application.
- Le dossier `models` contient les modèles Mongoose qui seront utilisés dans l'application.
- Le dossier `files` contient les fichiers téléchargé par l'application.
- Le fichier `package.json` contient les informations sur le projet et les dépendances du projet.
- Le fichier `.env` contient les informations de connexion à la base de données.
- Le fichier `con.js` contient les informations de connexion à la base de données.
- Le fichier `server.js` est l'entrée du serveur Express et il configure les routes, les middlewares et les interactions avec la base de données.
- Le fichier `dirved.route.js` contient les routes.

Les composants React

L'application Electron-React est composée de plusieurs composants React qui sont organisés en pages. Les pages sont organisées en fonction de leur fonctionnalité. Les pages sont les suivantes :

CreateConvention

Ce code crée un formulaire React pour la création d'une convention. Il utilise les hooks `useState` et `useEffect` pour gérer l'état du formulaire, et `useNavigate` pour naviguer vers une autre page.

Le hook `useState` est utilisé pour définir l'état initial du formulaire, qui contient des informations telles que le nom et le prénom du responsable, les dates de début et de fin, le montant, les pièces jointes, etc.

Les méthodes `updateForm` et onSubmit sont utilisées pour mettre à jour les propriétés d'état et soumettre le formulaire respectivement. La méthode `updateForm` permet de mettre à jour les propriétés d'état en utilisant la fonction setForm qui modifie la valeur de form dans l'état. La méthode onSubmit envoie une requête POST à l'URL http://localhost:5000/create/convention avec les données du formulaire.

Le rendu du formulaire se fait en utilisant des éléments HTML tels que des entrées, des boutons, des champs de formulaire, etc. Les entrées sont mises à jour à chaque fois que l'utilisateur modifie leur valeur, ce qui permet de mettre à jour les propriétés d'état correspondantes.

Enfin, lorsque le formulaire est soumis avec succès, l'application navigue vers la page d'accueil en utilisant le hook `useNavigate`.

En conclusion, la structure du code est divisée en deux parties principales, l'interface utilisateur gérée par React et le serveur Express qui gère les requêtes et les interactions avec la base de données. Il est important de comprendre cette structure pour pouvoir maintenir et développer efficacement le logiciel.

CreatePrestataire

Ce code crée un prestataire.

L'application envoie des données via une requête HTTP POST à l'URL http://localhost:5000/create/prestataire. Les données sont envoyées sous forme de JSON avec le corps de la requête. Les données proviennent du formulaire HTML dans le code, qui inclut des informations telles que le nom, le type, l'adresse, le téléphone, l'email, le RIB et le KBIS du prestataire.

Lorsque la requête a réussi, l'application utilise la fonction navigate pour rediriger l'utilisateur vers la page d'accueil /. Si la requête échoue, une fenêtre d'alerte est affichée avec le message d'erreur.

CreatePrestation

Ce code crée une prestation.
Il utilise les Hooks de React `useEffect` et `useState` ainsi que le Hook de navigation `useNavigate` de la bibliothèque react-router.

Lorsque la fonction est exécutée, `useEffect` appelle une fonction getPrestataires qui fait une requête à une API en utilisant fetch pour obtenir une liste de prestataires. Si la requête échoue, une alerte est affichée à l'utilisateur. Si la requête réussit, les prestataires sont stockés dans l'état prestataires en utilisant la fonction setPrestataires.

La fonction handleFormSubmit est déclenchée lorsque le formulaire est soumis et prévient le comportement par défaut en utilisant e.preventDefault(). Il récupère les données du formulaire en utilisant new FormData (form) et les stocke dans des variables distinctes. Ensuite, il assemble ces données en un objet JSON data et l'envoie à une autre API en utilisant fetch et JSON.stringify. Si la requête échoue, une erreur est affichée à l'utilisateur. Si la requête réussit, la navigation de l'utilisateur est redirigée vers la page d'accueil.

Enfin, le code retourne un composant React qui affiche un formulaire contenant des champs pour entrer les données de la prestation et une liste déroulante pour sélectionner un prestataire à partir de la liste prestataires.

NavBar

Ce code crée une barre de navigation. Il importe la bibliothèque React ainsi que NavLink du package `react-router-dom`.

La fonction Navbar retourne une structure de composants React qui définit une barre de navigation. Elle utilise NavLink pour définir des liens vers différentes pages de l'application. Elle utilise également FontAwesomeIcon pour afficher des icônes à côté de chaque élément de menu.

La structure HTML est composée d'un lien vers le logo, d'une section nav qui contient une liste ul d'éléments de menu avec des sous-éléments li et enfin d'un élément div avec la classe invisible-block.

ShowConventionDetail

Ce composant affiche les détails d'une convention en utilisant les données obtenues à partir de API REST. Il utilise les hooks `useState`, `useEffect` et `useParams` pour gérer les états et les paramètres de l'URL. L'utilisation de `useParams` permet de récupérer l'identifiant de la convention à afficher depuis l'URL. Le hook `useEffect` est utilisé pour appeler l'API pour obtenir les détails de la convention et les stocker dans l'état local product. Enfin, le composant affiche les détails de la convention en utilisant les données stockées dans product.

ShowConventionsByCategory

Ce composant utilise la bibliothèque React pour créer une page d'affichage de conventions. Il utilise les Hooks `useEffect` et `useState` pour gérer les données de l'état et la mise à jour des conventions lorsqu'elles sont modifiées. La fonction ConventionList s'occupe de récupérer les conventions depuis une API (à l'URL http://localhost:5000/show/conventions/), en utilisant la fonction getConventions. La fonction conventionList est ensuite utilisée pour afficher les conventions dans un tableau, en filtrant celles qui ont la bonne catégorie à l'aide du paramètre de route params.category. Il existe également une fonction deleteConvention pour supprimer une convention, en appelant une API pour la supprimer du serveur.

ShowConventionsCountCategory

Le code que vous avez fourni est un composant React qui affiche les conventions en utilisant les fonctions de hook React `useEffect` et `useState`. Le composant utilise l'effet `useEffect` pour appeler l'API http://localhost:5000/show/conventions/ et définit l'état de la convention en utilisant setConventions. Le composant utilise également une fonction deleteConvention pour supprimer une convention en appelant l'API http://localhost:5000/delete/convention/:id en utilisant la méthode HTTP DELETE. Enfin, le composant

affiche les statistiques des catégories de convention en utilisant la propriété con_categories. Les catégories sont affichées avec un lien vers une URL spécifique à la catégorie.

ShowPrestataires

Le code importe React et les hook `useEffect` et `useState` depuis le module React, ainsi que le composant Link de `react-router-dom`.

Le composant Prestataire affiche les informations d'un prestataire dans un tr (table row) d'un tableau, en utilisant les propriétés du prestataire passées via les props. Il contient également des boutons pour afficher les RIB et KBIS du prestataire, ainsi que des boutons pour le modifier et le supprimer.

Le composant ShowPrestataires utilise le hook `useState` pour gérer un tableau de prestataires et le hook `useEffect` pour récupérer les prestataires à partir de l'API. Il contient également une fonction deletePrestataire pour supprimer un prestataire, et une fonction prestataireList pour afficher les prestataires en utilisant le composant Prestataire.

Enfin, le composant ShowPrestataires retourne un tableau avec les prestataires, en utilisant la fonction prestataireList.

ShowPrestations

Le code utilise la bibliothèque React pour créer une composante "Prestation". Cette composante affiche les informations sur une prestation, telles que le numéro d'opération, le nom du prestataire, le montant, la date de mise à jour et le statut.

Il y a également d'autres fonctions définies telles que "dateIsDefined", "timeLeft", "statusPrestation" et "displayPrestataireName" qui sont utilisées pour traiter les données de la prestation.

La fonction "PrestationList" utilise l'effet "`useEffect`" pour récupérer les données des prestations à partir d'une API en utilisant "fetch". Les données sont ensuite mises à jour dans l'état local "prestations" en utilisant "setPrestations". Les données de chaque prestation sont ensuite passées à la composante "Prestation" pour les afficher.

Il y a également des liens et des boutons pour modifier et supprimer une prestation.

UpdateConvention

Ce code crée une page web React qui met à jour des données de convention. Il utilise les Hooks `useState` et `useEffect` pour gérer l'état du formulaire et charger les données de la convention à partir d'une API. Lorsque le formulaire est soumis, le code envoie une requête POST à l'API pour mettre à jour les données de la convention. Les données sont ensuite affichées dans un formulaire qui peut être rempli par l'utilisateur.

UpdatePrestataire

Il s'agit d'un composant React pour mettre à jour les informations d'un prestataire dans une base de données. Il utilise les fonctions `useState` et `useEffect` de React pour gérer l'état local et les effets de l'application. Le composant utilise également les fonctions `useParams` et `useNavigate` de la bibliothèque de navigation React pour obtenir les paramètres de l'URL et naviguer vers d'autres pages.

Le composant utilise une fonction `useEffect` pour obtenir les informations sur le prestataire à partir de l'API en utilisant l'ID fourni dans les paramètres d'URL. La fonction `updateForm` est utilisée pour mettre à jour l'état local lorsque les entrées de l'utilisateur sont modifiées. La fonction onSubmit envoie une demande POST à l'API pour mettre à jour les informations du prestataire dans la base de données. Enfin, le composant rend un formulaire HTML qui affiche les informations du prestataire et les permet de les mettre à jour.

UpdatePrestation

Le code ci-dessous montre une fonction "UpdatePrestation" qui gère la mise à jour des informations sur une prestation. La fonction utilise les composants React suivants : `useState`, `useEffect`, `useParams`, et `useNavigate`.

`useState` est utilisé pour définir l'état local des données prestataires et form, qui est un objet représentant les informations sur la prestation.

`useEffect` est utilisé pour effectuer une requête à l'API pour obtenir les informations sur les prestataires et sur la prestation en question.

`useParams` et `useNavigate` sont utilisés pour obtenir les paramètres de l'URL et pour naviguer vers une autre page.

Lors de la soumission du formulaire, la fonction onSubmit envoie une requête HTTP POST à l'API avec les données modifiées pour mettre à jour les informations sur la prestation.

Fonctionnalités actuelles

Le logiciel permet de gérer les prestataires et les prestations. Il est possible de créer, modifier et supprimer des prestataires et des prestations. Il est également possible de consulter les prestataires et les prestations.

Tâches en suspens

Il reste à implémenter :

- Un système d'alert par email
- Un système de login
- Un système de gestion des utilisateurs

Conventions de codage

Le code est écrit en anglais.