PROJET EMARGEN

Table des matières

1. Générale

1.1. Contexte du projet

La dématérialisation des documents est de plus en plus présente dans notre quotidien. Aujourd'hui, dans le contexte de la crise sanitaire de la Covid-19, les échanges de document physique deviennent restreints.

Dans le cadre de la formation, Simplon utilise le format papier des feuilles d'émargement et une interaction physique pour apposer nos signatures. Je suis donc confronté à la problématique où je dois créer une application qui puisse limiter les échanges et faire gagner du temps à l'administration.

Cette application doit répondre aux besoins qui s'appuient sur les données d'un Google Sheets :

- Permettre de générer une feuille d'émargement PDF ;
- Permettre de synchroniser notre feuille si une modification est effectué sur le Sheets ;
- Permettre aux apprenants de signé aux emplacements qui leurs sont attribués ;
- Régénérer la feuille d'émargement avec les signatures ;
- Avoir une vision sur les personnes qui ont signés ou non.

Les apprenants peuvent se connecter uniquement à l'application mobile pour signer. Ils n'ont pas les droits nécessaires pour s'authentifier sur l'application web.

1.2. Ressources conception / développement

Ci-dessous le lien vers le code source du projet :

https://github.com/Darylabrador/simplon-emargen

Vous pouvez retrouver la maquette ci-dessous :

https://github.com/Darylabrador/simplon-emargen/tree/master/ressources

2. User story

2.1.1. Dans le cadre d’un espace administrateur :

En tant qu’administrateur, je veux m’authentifier afin de gérer les feuilles d’émargement.

En tant qu’administrateur, je veux créer des Template pour l’en-tête de la feuille d’émargement.

En tant qu’administrateur, je veux modifier des Template.

En tant qu’administrateur, je veux supprimer des Template.

En tant qu’administrateur, je veux créer une feuille d’émargement avec l’utilisation du Template.

En tant qu’administrateur, je veux créer une feuille d’émargement suivant le numéro de la feuille du Google Sheets.

En tant qu’administrateur, je veux voir une feuille d’émargement sous format PDF.

En tant qu’administrateur, je peux synchroniser les données de l’application et du Google Sheets.

En tant qu’administrateur, je veux savoir le nombre de signature manquante une feuille d’émargement.

En tant qu'administrateur je veux être capable de gérer une liste d'apprenants.

En tant qu'administrateur je veux être capable de gérer une liste de promotion.

En tant qu'administrateur je veux être capable d'affecté une promotion à un apprenant.

En tant qu'administrateur je veux être capable de signaler à un apprenant de la présence d'une feuille d'émargement le concernant.

2.1.2. Dans le cadre d’un espace pour les apprenants

En tant qu’apprenant, je veux m’authentifier afin de consulter la feuille d’émargement de la semaine.

En tant qu’apprenant, je veux créer une signature.

En tant qu’apprenant, je veux signer une feuille d’émargement.

3. Modèles de données

La création des composantes du backend doit se faire par le biais de l'utilisation de mongoose. Ceci dans le but de gagner du temps dans les échanges entre Node JS et MongoDB.

Nous avons les éléments suivants :

- 1 base de données : simplonemargen
- 3 collections : Users, Yeargroups, Templates, Signoffsheets, Assigns

3.1.Modèles attendus :

3.1.1. Modèle : la collection Users :

```
const userSchema = mongoose.Schema({
    promoId: {
        type: ObjectId,
        default: null,
        ref: 'yeargroups'
    },
    name: {
        type: String,
        required: true
    },
    surname: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    role: {
        type: String,
        required: true,
        default: "apprenant"
    },
    firstConnection: {
        type: Boolean,
        default: true
    },
```

```
    signImage: {
        type: String,
        default: null
    },
    resetToken: {
        type: String,
        default: null
    }
}, {timestamps: true})
```

3.1.2. Modèle : la collection Yeargroups :

```
const yeargroupSchema = mongoose.Schema({
    label: {
        type: String,
        required: true
    }
}, { timestamps: true })
```

3.1.3. Modèle : la collection Templates :

```
const templateSchema = mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    intitule: {
        type: String,
        required: true
    },
    organisme: {
        type: String,
        required: true
    },
    logo: {
        type: String,
        required: true
    }
}, { timestamps: true })
```

3.1.4. Modèle : la collection Signoffsheets :

```
const signoffsheetSchema = mongoose.Schema({
    templateId: {
        type: ObjectId,
        ref: "templates",
        required: true
    },
    promoId: {
        type: ObjectId,
        ref: 'yeargroups',
        required: true
    },
    urlSheet: {
        type: String,
        required: true
    },
    name: {
        type: String,
        required: true
    },
    learners: {
        type: Array
    },
    days: {
        type: Array
    },
    trainers: {
        type: Array
    },
    timeStart: {
        type: String
    },
    timeEnd: {
        type: String
    },
    version: {
        type: Number,
        default: 1
    },
    versionningHistory: {
        type: Array,
        default: null
    },
    fileExist: {
        type: Boolean,
        default: false
    },
    signLocation: {
```

```
        type: Array
    }
}, { timestamps: true })
```

3.1.5. Modèle : la collection Assigns :

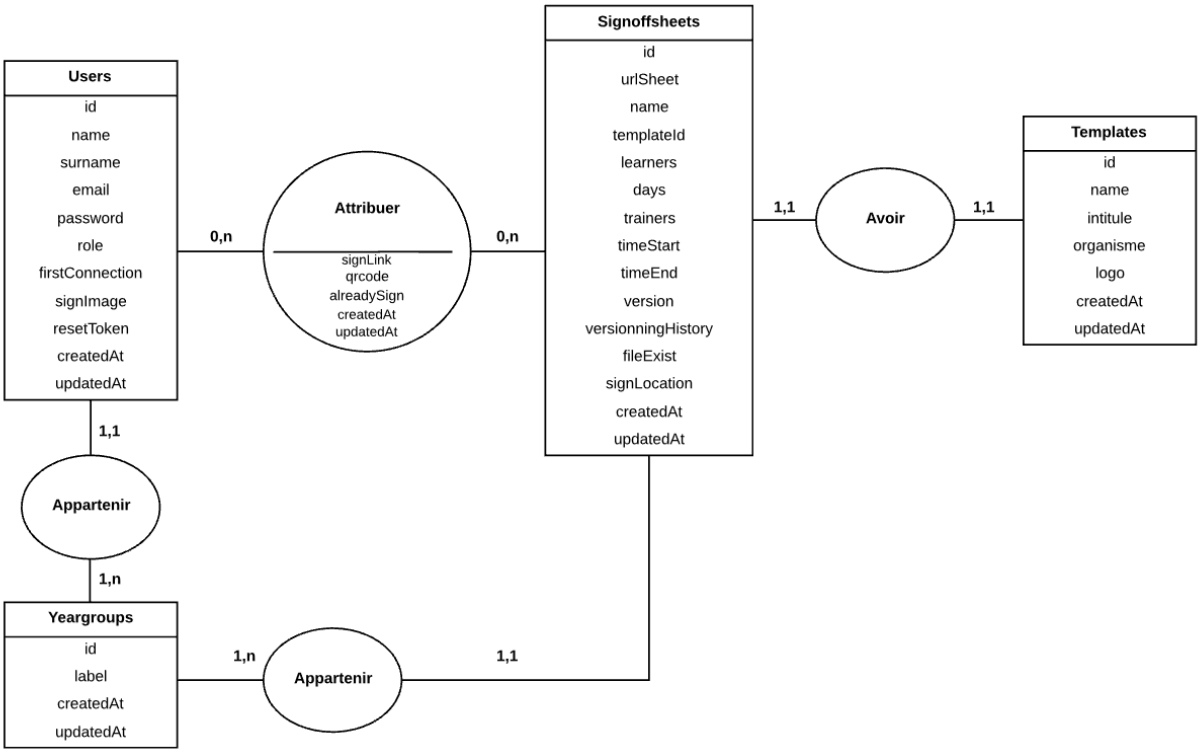
```
const assignSchema = mongoose.Schema({
    userId: {
        type: ObjectId,
        ref: 'users'
    },
    signoffsheetId: {
        type: ObjectId,
        ref: 'signoffsheets'
    },
    signLink: {
        type: String,
        required: true
    },
    qrcode: {
        type: String,
        required: true
    },
    alreadySign: {
        type: Boolean,
        default: false
    }
}, { timestamps: true })
```

3.1.6. Précisions liées aux modèles

ObjectId provient de la ligne suivante :

```
const ObjectId = mongoose.Types.ObjectId;
```

3.2.MCD (modèle conceptuel des données) :



3.3.MPD : Modèles physique des données