PROJET

SIMPLONVOT

Table des matières

1. Générale

1.1. Contexte du projet

Nous allons travailler sur une application de vote. Le but de l'application et de pouvoir créer des sujets de votes. Les utilisateurs s'affectent à un vote. Le créateur du sujet de vote lance le vote quand le nombre d'utilisateur "inscrit" pour le vote atteint le nombre fixer par le créateur.

1.2. Ressources GitHub

Ressource GitHub qui a permis d'avoir une refonte

- https://github.com/pyrce/simplon_vote

Ressource GitHub qui a permis d'avoir une refonte

- https://github.com/Darylabrador/Simplon-vote

2. Conception

2.1. User story

En tant qu'utilisateur je veux créer un sujet de vote afin d'avoir l'avis des autres

En tant qu'utilisateur je veux saisir autant de choix (options) à un sujet de vote afin d'avoir un résultat

En tant qu'utilisateur je peux participer à un vote afin de donner mon avis

En tant qu'utilisateur je veux connaitre le nombre d'utilisateur afin de lancer le vote

En tant qu'utilisateur je veux pouvoir modifier mon sujet de vote afin de corriger mes erreurs

En tant qu'utilisateur, je veux pouvoir fixer un quota afin d'activer le vote

En tant qu'utilisateur je veux pouvoir m'authentifier afin de voir le quota de vote à un sujet particulier.

En tant qu'utilisateur je veux pouvoir m'inscrire afin de lancer un sujet de vote

En tant qu'utilisateur je peux voter 1 fois à un sujet de vote

En tant qu'utilisateur je veux voir les résultats des votes

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 : simplonvote
- 3 collections : Users, Votes, usersVotes

3.1. Modèles attendus :

3.1.1. Pour la collection Users :

```
const userSchema = mongoose.Schema({
    login: {
        type: String,
        required: true,
        unique: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    }
})
```

3.1.2. Pour la collection Votes :

```
var ObjectId = mongoose.Types.ObjectId;

const voteSchema = mongoose.Schema({
    subject: {
        type: String,
        required: true
    },
    quota: {
        type: Number,
        required: true
    },
    choices: {
        type: Array
    },
```

```
        nbVote: {
            type: Number
        },
        createdBy: {
            type: ObjectId,
            ref: 'user'
        },
        participants: {
            type: Array,
        },
        visibility: {
            type: String,
            required: true,
            enum: ['public', 'private'],
            default: "public"
        },
        status: {
            type: String,
            required: true,
            enum: ['created', 'inprogress', 'finished'],
            default: "created"
        }
    },{ timestamps: true })
```

3.1.3. Pour la collection usersVotes :

```
    var ObjectId = mongoose.Types.ObjectId;

    const userVoteSchema = mongoose.Schema({
        user: {
            type: ObjectId,
            required: true,
            ref: 'user'
        },
        vote: {
            type: ObjectId,
            required: true,
            ref: 'vote'
        },
        choice: {
            type: Number,
            default: null
        },
    },{ collection: 'usersVotes' });
```

3.1.4. Précisions liées aux modèles

Obligation :

- 1 sujet de vote doit avoir au minimum 2 options, sinon pas de création.

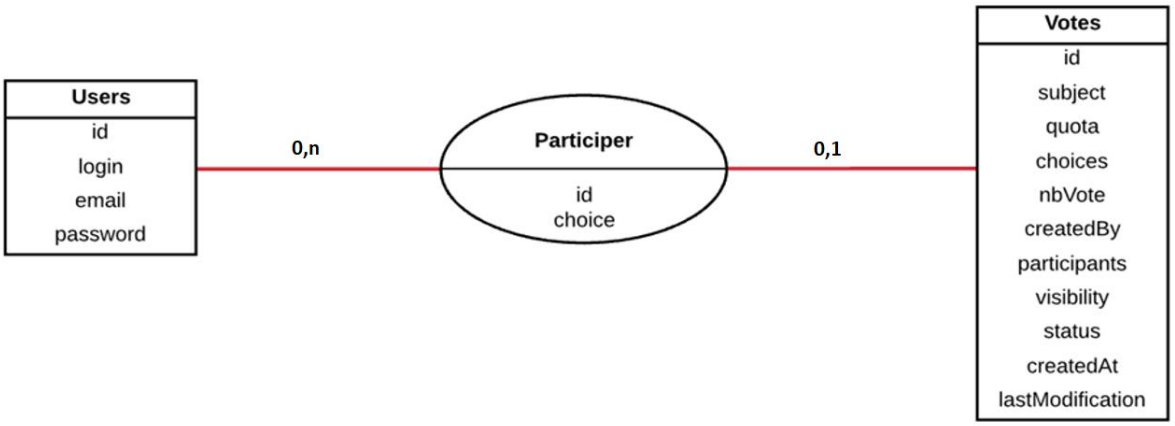
Note concernant le statut d'un vote :

- Pour l'état "created" : c'est l'état par défaut lors de la création d'un sujet de vote avec au minimum ses deux options.

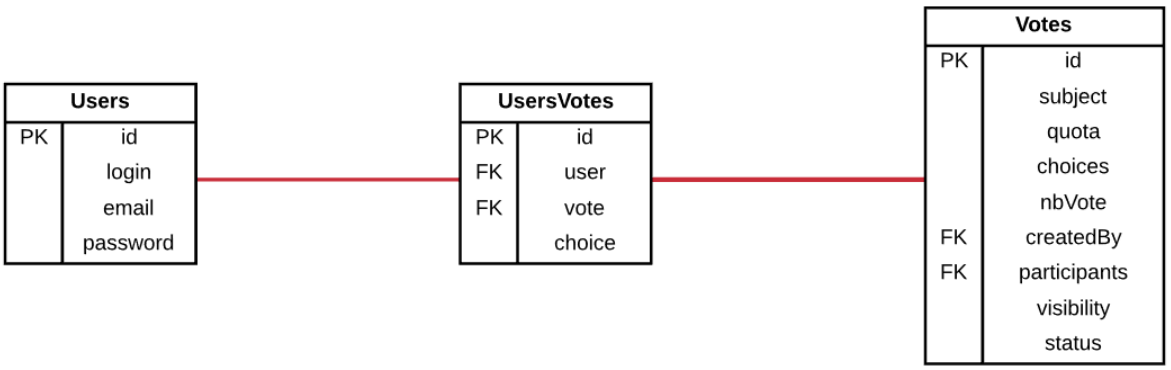
- Pour l'état "inprogress": c'est l'état qui sera défini lorsque le nombre de participant est égale à celui du quota qui est défini par l'utilisateur.

- Pour l'état "finished": c'est l'état qui sera défini lorsque le nombre de vote "nbVote" est égale à celui du quota.

3.2.MCD (modèle conceptuel des données) :



3.3.MPD (modèle physique des données) :



3.4.Pour avoir accès à la base de données :

Voir la vidéo d'explication dans le dossier ressource du GitHub.

https://github.com/Darylabrador/Simplon-vote/tree/master/Ressources

4. Validation des données :

4.1.Pour la collection Users :

```
db.createCollection("users", {
    validator: {
        $jsonSchema: {
            bsonType: 'object',
            required: [
                'login',
                'email',
                'password'
            ],
            properties: {
                pseudo: {
                    bsonType: 'string'
                },
                email: {
                    bsonType: 'string'
                },
                password: {
                    bsonType: 'string'
                }
            }
        }
    }
})
```

4.2.Pour la collection Votes :

```
db.createCollection("votes", {
    validator: {
        $jsonSchema: {
            bsonType: 'object',
            required: [
                'subject',
                'quota',
                'choices',
                'nbVote',
                'participants',
                'createdBy',
                'visibility',
                'status'
            ],
            properties: {
                subject: {
                    bsonType: 'string'
                },
                quota: {
                    bsonType: 'int'
                },
                choices: {
                    bsonType: 'array'
                },
                nbVote: {
                    bsonType: 'int'
                },
                participants: {
                    bsonType: 'array'
                },
                createdBy: {
                    bsonType: 'objectId'
                },
                visibility: {
                    bsonType: 'string'
                },
                status: {
                    bsonType: 'string'
                }
            }
        }
    }
})
```

4.3.Pour la collection usersVotes :

```
        validator: {
            $jsonSchema: {
                bsonType: 'object',
                required: [
                    'user',
                    'vote'
                ],
                properties: {
                    user: {
                        bsonType: 'objectId'
                    },
                    vote: {
                        bsonType: 'objectId'
                    }
                }
            }
        }
    })
```

5. Script de migration

5.1. Fichier de configuration

Utilisation de l'outil de migration migrate-mongo.

Installation de migrate-mongo : **npm install -g migrate-mongo**

Création du dossier de migration et son fichier conf : **migrate-mongo init**

- **Fichier migrate-mongo-config.js :**

```
    const config = {
      mongodb: {
        // TODO Change (or review) the url to your MongoDB:
    url:"mongodb+srv://simplon974:simplon974@cluster0-
2q4j1.azure.mongodb.net/simplonvote?retryWrites=true&w=majority",

        // Création de la database if not exist:
        databaseName: "simplonvote",

        options: {
          useNewUrlParser: true, // removes a deprecation warning when connecting
          useUnifiedTopology: true, // removes a deprecating warning when
connecting
          //   connectTimeoutMS: 3600000, // increase connection timeout to 1 hour
          //   socketTimeoutMS: 3600000, // increase socket timeout to 1 hour
        }
      },
      // The migrations dir, can be an relative or absolute path. Only edit this
when really necessary.
      migrationsDir: "migrations",

      // The mongodb collection where the applied changes are stored. Only edit
this when really necessary.
      changelogCollectionName: "changelog",
      // The file extension to create migrations and search for in migration dir
      migrationFileExtension: ".js"
    };
    // Return the config as a promise
    module.exports = config;
```

5.2. Fichier de migration pour la collection users

Dans le dossier de migration fait une commande : **migrate-mongo create users**

Ajouter ces scripts dans votre dossier de migration

- **Fichier users.js**

```
    module.exports = {
      async up(db, client) {
        // TODO write your migration here.
        //      See      https://github.com/seppevs/migrate-mongo/#creating-a-new-migration-script
        // Example:
        // await db.collection('albums').updateOne({artist: 'The Beatles'}, {$set: {blacklisted: true}});
        return db.createCollection('users', {
          capped: false,
          autoIndexId: true,
          validator: {
            $jsonSchema: {
              bsonType: 'object',
              required: ['login', 'email', 'password'],
              properties: {
                login: {
                  bsonType: 'string'
                },
                email: {
                  bsonType: 'string'
                },
                password: {
                  bsonType: 'string'
                }
              },
            },
          },
          validationLevel: 'strict',
          validationAction: 'error',
        })
      },

      async down(db, client) {
        // TODO write the statements to rollback your migration (if possible)
```

```
        // Example:
        // await db.collection('albums').updateOne({artist: 'The Beatles'}, {$set:
{blacklisted: false}});
      return await db.collection('users').drop()
       }
     };
```

5.3. Fichier de migration pour la collection votes

Dans le dossier de migration fait une commande : **migrate-mongo create votes**

Ajouter ces scripts dans votre dossier de migration

- **Fichier votes.js**

```
    module.exports = {
      up(db) {
        return db.createCollection('votes', {
          capped: false,
          autoIndexId: true,
          validator: {
            $jsonSchema: {
              bsonType: 'object',
              required: ['subject', 'quota', 'choices', 'nbVote', 'participants',
 'createdBy', 'visibility', 'status'],
              properties: {
                subject: {
                  bsonType: 'string'
                },
                quota: {
                  bsonType: 'int'
                },
                choices: {
                  // array string ex: ['oui', 'non', 'peut être']
                  bsonType: 'array'
                },
                nbVote: {
                  bsonType: 'int'
                },
                participants: {
                  // Array des Object ID des utilisateurs qui participent au vote
                  bsonType: 'array',
                },
                createdBy: {
                  // Object ID de l'utilisateur qui a créer le sujet de vote
                  bsonType: 'objectId',
                },
```

```
            visibility: {
              // 2 status possibles : public / private
              bsonType: 'string'
            },
            status: {
              // On peut avoir 3 valeurs : created,inprogress,finished
              bsonType: 'string'
            }
          }
        },
      },
      validationLevel: 'strict',
      validationAction: 'error',
    })
  },
  down(db) {
    return db.collection('votes').drop()
  }
}
```

5.4. Fichier de migration pour la collection usersVotes

Dans le dossier de migration fait une commande : **migrate-mongo create usersVotes**

Ajouter ces scripts dans votre dossier de migration

- **Fichier usersVotes.js**

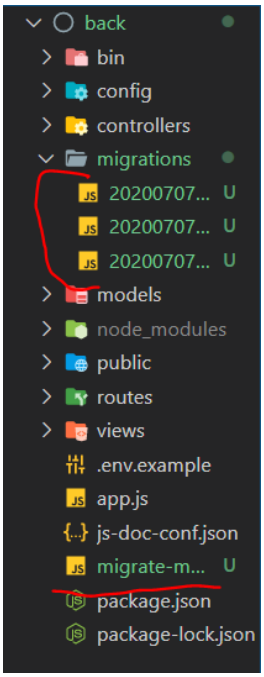
```
module.exports = {
  up(db) {
    return db.createCollection('usersVotes', {
      capped: false,
      autoIndexId: true,
      validator: {
        $jsonSchema: {
          bsonType: "object",
          required: ["user", "vote"],
          properties: {
            user: {
              bsonType: 'objectId'
            },
            vote: {
              bsonType: 'objectId'
            }
          }
        }
```

```
        },
        validationLevel: 'strict',
        validationAction: 'error',
      })
    },
    down(db) {
      return db.collection('usersVotes').drop()
    }
  }
```

Dans votre dossier migration fait une commande :**migrate-mongo up**

Vous aurez le résultat ci-dessous :

```
D:\documents\simplon_vote\back\albums-migrations>migrate-mongo up
(node:15268) DeprecationWarning: Db.createCollection option [autoIndexId] is deprecated and will be removed in a later version.
MIGRATED UP: 20200707050301-votes.js
MIGRATED UP: 20200707051108-usersVotes.js
```



6. Annexes :

Gestion des sessions
https://blog.jscrambler.com/best-practices-for-secure-session-management-in-node/

Validation de données :
https://docs.mongodb.com/manual/core/schema-validation/

Socket.io :
https://socket.io/docs/

Express.js :
https://expressjs.com/fr/

Mongoose :
https://mongoosejs.com/docs/

MongoDB :
https://www.mongodb.com/fr

Node JS :
https://nodejs.org/en/