Projet : Systèmes de vote

**> Consigne :**

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.

**> Liens GitHub qui a permis d’avoir la refonte :**

<https://github.com/pyrce/simplon_vote>

**> Lien GitHub :**

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

# **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

**Les modèles attendus : (Déploiement suivant mongoose)**

Database :

* simplonvote

3 collections :

* users
* votes
* usersVotes

Modèle de données attendu 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

    }

})

Modèle de données attendu pour la collection « vote » :

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 })

Modèle de données attendu 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' });

**Précisions liées aux modèles :**

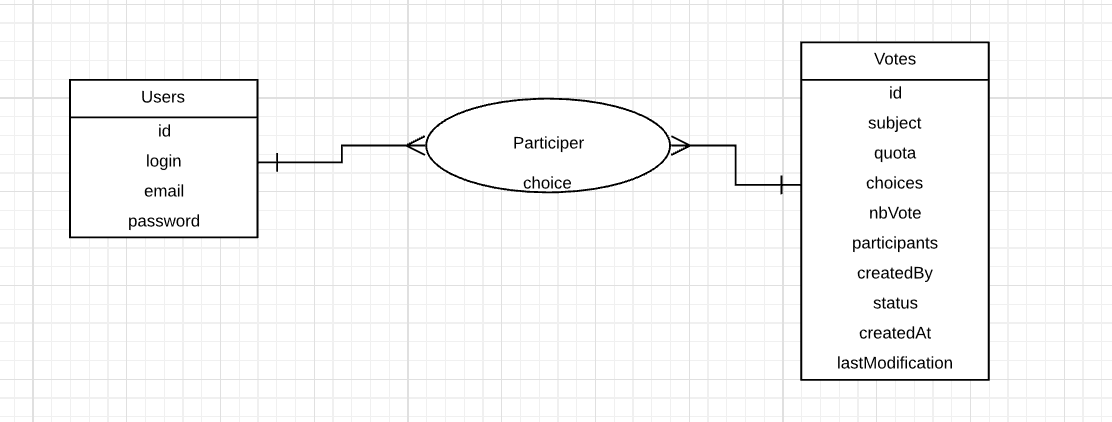
Obligation :

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.

**MCD :**



**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>

# **Validation des données**

*Validation des données 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'

                }

            }

        }

    }

})

*Validation des données 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'

                }

            }

        }

    }

})

*Validation des données pour la collection « usersVotes » :*

db.createCollection("usersVotes", {

    validator: {

        $jsonSchema: {

            bsonType: 'object',

            required: [

                'user',

                'vote'

            ],

            properties: {

                user: {

                    bsonType: 'objectId'

                },

                vote: {

                    bsonType: 'objectId'

                }

            }

        }

    }

})

# **Jeu de données**

Pour la collection users :

Pour la collection votes :

Pour la collection usersVotes :

# **Script de migration**

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;

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()

}

};

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()

  }

}

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) {

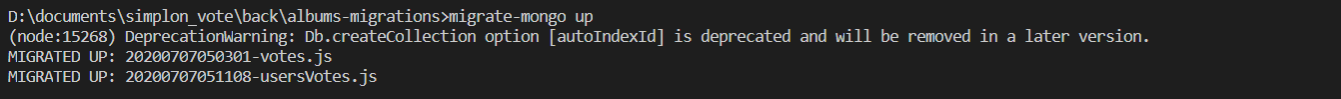
return db.collection('usersVotes').drop()

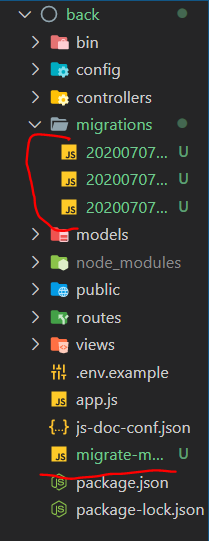
}

}

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

Pour générer la bdd





**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/>