Rendu TME Solo

Project UE LU3IN017: Web Technologies
KEDADRY Yannis 3808875

Serveur

On modifie l'entite user pour ajouter une liste de termes bloques initialement vide.

```
    //fichier birdy-server/src/entities/users.js
    //cree un utilisateur et l'ajoute dans la db et renvoie son id
    create(username, fullname, dateOfBirth, emailAddress, passwd) {
        return new Promise(async (resolve, reject) => {
            const exists = await this.exists(username, emailAddress);

            if(!exists){
                //console.log('try to insert');
                let newUser = {
                    username: username.toLowerCase(),
                    fullname: fullname.toLowerCase(),
                    dateOfBirth: new Date(dateOfBirth),
                    emailAddress: emailAddress.toLowerCase(),
                    passwd: passwd,
                    following: [],
                    followers: [],
                    tweets: [],
                    tweetsLiked: [],
                    tweetsRetweeted: [],
                    tweetsReplied: [],
                    profilePicture: "",
                    dateCreated: new Date(),
                    //on ajoute une liste de termes bloques
                    blockedTerms: []
                }
                this.db.insert(newUser);
            }

            let userid = await this.getUserId(username.toLowerCase());

            if(exists) {
                //erreur
                reject();
            } else {
                resolve(userid);
            }
        });
    }
```

On modifie le getter sur la lits de tweets pour ajouter le filtre

```
    //renvoie la liste des tweets d'un utilisateur
    getTweets(userid){
        return new Promise( (resolve, reject) => {
            //console.log('test get, id: ', userid);
            this.db.find({ _id: userid }, { _id: 0 }, function(err, docs) {
                if(err){
                    reject(err);
                }
                if(docs.length !== 1){
                    reject(null);
                }

                //on recupere la liste des termes bloques
                const blockedTerms = docs[0]['blockedTerms'];
                //on filtre les tweets
                const tweets = docs[0]['tweets'].filter((item) => {
                    for(let word of item['content']){
                        if(blockedTerms.includes(word))
                            return false;
                    }
                    return true;
                });


                if(!tweets) {
                    //erreur
                    reject(err);
                } else {
                    resolve(tweets);
                }
            });
        });
    }
```

On modifie l'api pour ajouter une nouvelle requete. On ajoute une route vers l'url /api/user/:id/blockedTerms et on ajoute une methode post pour l'ajout d'un terme bloque.

```
    //fichier birdy-server/src/api/apiUser.js
    //on ajoute une route vers l'url pour gerer les termes bloques d'un
utilisateur
    api
        .route('user/:id/blockedTerms')
        //on ajoute une methode post pour ajouter un nouveau mot bloque
        .post(async (req, res) => {
            try{
                const userId = req.params._id;
                //on recupere le nouveau mot bloque
                const { newBlockTerm } = req.body;
            }
            /* ... */
```

```
        })
        //on ajoute une methode delete pour supprimer un terme bloque
        .delete()
```

Pour la suppression, on modifie l'api pour ajouter une nouvelle requete. On ajoute une route vers l'url /api/user/:id/delBlockTerm et on ajoute une methode post pour la suppression d'un terme (on n'utilise pas de delete sur la route precedente car il n'est pas evident de passer un parametre a une methode delete).

Ajout de terme

- On test si l'utilisateur existe

```
    //fichier birdy-server/src/api/apiUser.js
    if(!await users.get(userId)){
        res.status(404).json({
            status: 404,
            message: "User not found"
        });
        return;
```

- On test si un nouveau mot bloque nous a bien ete fourni

```
    //fichier birdy-server/src/api/apiUser.js
    if(!newBlockTerm){
        res.status(400).json({
            status: 400,
            message: "Missing Field"
        });
        return;
    }
```

- On test si le mot bloque n'etait pas deja dans la liste des mots bloques de l'utilisateur

```
    //fichier birdy-server/src/api/apiUser.js
    if(await users.blockTermExists(userId, newBlockTerm)){
        res.status(409).json({
            status: 409,
            message: "Blocked Term already set"
        });
        return;
    }
```

```
    //fichier birdy-server/src/entities/users.js
    //verif qu'un mot bloque existe deja
    blockTermExists(userId, newBlockTerm){
```

```
        return new Promise( (resolve, reject) => {
            this.db.find( {_id: userId }, { _id: 0 }, function(err, docs) {
                if(err){
                    reject(err);
                }
                if(docs.length !== 1){
                    reject(null);
                }

                const blockedTerms = docs[0]['blockedTerms'];

                if(!blockedTerms) {
                    //erreur
                    reject(err);
                } else {
                    resolve(blockedTerms.includes(newBlockTerm));
                }
            });
        });
    }
```

- On appelle la db pour ajouter ce nouveau terme dans la liste des termes bloqes de l'utilisateur

```
    //fichier birdy-server/src/api/apiUser.js
    if(!await users.addBlockTerm(userId, newBlockTerm)){
        res.status(422).json({
            status: 422,
            message: "Can't add this block term"
        });
        return;
    }
```

```
    //fichier birdy-server/src/entities/users.js
    //ajoute un mot bloque dans la liste des mots bloques de l'utilisateur
    addBlockTerm(userId, newBlockTerm){
        return new Promise( (resolve, reject) => {
            this.db.update( {_id: userId }, { $push: { blockedTerms:
newBlockTerm } }, {}, function (err, numReplaced) {
                if(err){
                    reject(err);
                }
                if(numReplaced === 0){
                    //si le mot ne s'est pas ajoute correctement
                    resolve(false);
                } else {
                    resolve(true);
                }
            });
```

```
        });
    }
```

- Si tout s'est bien passe on renvoie le statut 201

```
    //fichier birdy-server/src/api/apiUser.js
    res.status(201).send('New blocked term added successfully');
```

- En cas d'erreur interne

```
    //fichier birdy-server/src/api/apiUser.js
    } catch(e) {
        //console.log('test like tweet in catch');
        // Exception
        res.status(500).json({
            status: 500,
            message: "Internal error",
            details: (e || "Unknown error").toString()
        });
    }
});
```

Supression d'un terme

```
    //fichier birdy-server/src/api/apiUser.js
    api
        .route('user/:id/removeBlockedTerms')
        .post(async (req, res) => {
            //console.log('test like tweet before try');
            try{
                const userId = req.params._id;
                const { newBlockTerm } = req.body;

                //console.log('check user exists');
                if(!await users.get(userId)){
                    res.status(404).json({
                        status: 404,
                        message: "User not found"
                    });
                    return;
                }

                // s'il n'y a pas le terme a supprimer
                if(!newBlockTerm){
                    res.status(400).json({
                        status: 400,
                        message: "Missing Field"
```

```
                    });
                    return;
                }

                //si le terme n'existait pas
                if(! await users.blockTermExists(userId, newBlockTerm)){
                    res.status(404).json({
                        status: 404,
                        message: "Blocked Term wasn't set"
                    });
                    return;
                }

                //supression du terme dans la db
                if(!await users.removeBlockTerm(userId, newBlockTerm)){
                    res.status(422).json({
                        status: 422,
                        message: "Can't remove this block term"
                    });
                    return;
                }

                res.status(201).send('New blocked term removed
successfully');

            } catch(e) {
                //console.log('test like tweet in catch');
                // Exception
                res.status(500).json({
                    status: 500,
                    message: "Internal error",
                    details: (e || "Unknown error").toString()
                });
            }
        });
```

- On enleve le mot de la base de donne des termes bloquant de l'utilisateur

```
    //fichier birdy-server/src/entities/users.js
    //enleve un mot bloque dans la liste des mots bloques de l'utilisateur
    removeBlockTerm(userId, newBlockTerm){
        return new Promise( (resolve, reject) => {
            this.db.update( {_id: userId }, { $pull: { blockedTerms:
newBlockTerm } }, {}, function (err, numReplaced) {
                if(err){
                    reject(err);
                }
                if(numReplaced === 0){
                    //si le mot ne s'est pas ajoute correctement
                    resolve(false);
                } else {
                    resolve(true);
```

```
                }
            });
        });
    }
```

Client

- On ajoute un formulaire pour ajouter ou enlever un mot a bloquer
- On ajoute un state pour chaque contenu de ces formulaires

```
    //fichier birdy-client/src/components/SideBsr.js
    const [formAddBlockTerm, setFormAddBlockTerm] = useState("");
    const [formDelBlockTerm, setFormDelBlockTerm] = useState("");
```

- On ajoute un handler les bouttons submit de ces formulaires pour faire un appel a axios

Handlers

```
    //fichier birdy-client/src/components/SideBsr.js
    async function handleAddNewBlockTerm(event, userId){
        if(!formAddBlockTerm!==""){
            axios
                .post(`/api/user/${props.connectedUser}/blockedTerms`, {
newBlockTerm: formAddBlockTerm })
                .then( () => {
                    setFormAddBlockTerm("");
                })
        }
    }

    async function handleDeleteNewBlockTerm(event, userId){
        if(!formDelBlockTerm!==""){
            axios

.post(`/api/user/${props.connectedUser}/removeBlockedTerms`, {
newBlockTerm: formDelBlockTerm })
                .then( () => {
                    setFormDelBlockTerm("");
                })
        }
    }
```

TODO

- Il reste a ajouter les formulaires et les boutons de soumission

- Il reste a modifier certains acces a la liste des tweets dans la partie client pour ne pas recuperer ceux qui contiennent les mots bloques quand on se ballade sur un autre profil.
- Il reste a faire des tests