Task 6. Use mongodb to store notes

Task descrition

In this task we will start to use mongodb to work with data.

Time

2 hours

Detailed desctiption

6.1 Working with mongodb from NodeJS

1) Edit package.json

Add dependency on last mongodb package:

```
"mongodb": "*"
```

Then execute **npm install** to install necessary modules.

2) Define variables in server.js

```
var Db = require('mongodb').Db;
var Server = require('mongodb').Server;
```

3) Open mongodb connection

```
var db = new Db('tutor',
new Server("localhost", 27017, {safe: true},
               {auto_reconnect: true}, {}));

db.open(function(){
        console.log("mongo db is opened!");
});
```

4) Add collection notes

Add these lines inside db.open callback function

```
db.collection('notes', function(error, notes) {
        db.notes = notes;
});
```

5) Allow notes to be loaded from database

```
app.get("/notes", function(req,res) {
db.notes.find(req.query).toArray(function(err, items) {
        res.send(items);
});
});
```

6) Allow notes to be saved in database

```
app.post("/notes", function(req,res) {
```

```
db.notes.insert(req.body);
res.end();
});
```

7) Go to db folder and execute mongodb:

```
<MONGODB_PATH>/bin/mongod --dbpath .
```

Now you can restart server and add some notes.

6.2 Using mongo console to find/update/insert notes

1) Start mongo client:

```
<MONGODB_PATH>/bin/mongo
```

2) Change database to tutor:

```
use tutor
```

3) Find all notes:

```
db.notes.find()
```

4) Insert note:

```
db.notes.insert({text:"my note"})
```

5) Find note by query:

```
db.notes.find({text:"my note"})
```

6) Change note text:

```
db.notes.update({text:"my note"}, { $set:{text:"his note"} })
```

7) Change or insert note:

```
db.notes.update({text:"my note"}, { $set:{text:"her note"}})
```

This command will not do anything because there's no note with text "my note".

```
db.notes.update({text:"my note"}, { $set:{text:"her note"}},
    {upsert:true})
```

This will find for note "my note", and if it was not found, insert the new note with the text "her note".

8) Change multiple notes

Usually update changes only one record. If we need to change all records, we have to add parameter **{multi:true}**.

Let’s add property lastUpdated to the notes.
Execute this command:

```
db.notes.update({}, {$set:{lastUpdated:new Date().getTime()}})
```

You will see that only one item is updated. To update all items, you need to add {multi:true} parameter:

```
db.notes.update({}, {$set:{lastUpdated:new Date().getTime()}},
{multi:true})
```

It will add lastUpdated to all notes.

9) Ordering items

By default items in collection are ordered as they were added. You can reorder it by using orderBy. For example to order by text (alphabetically) use this command:

```
db.notes.find().sort( { text: 1 } )
```

This command will order by text in decreasing order:

```
db.notes.find().sort( { text: -1 } )
```

You also can use several fields for ordering:

```
db.notes.find().sort( { lastUpdate:1, text: 1 } )
```

It will order by lastUpdate, and then by text.

9.1) Removing item from collection

```
db.notes.remove({text:"my text"})
```

Remove collection with indexes

```
db.notes.drop();
```

10) Remove field from the items

To remove field lastUpdated from notes, execute this command:

```
db.notes.update({}, {$unset: {lastUpdated:""}}, {multi:true})
```

6.3 Delete notes from database by id

1) Add ObjectID variable to server.js:

```
var ObjectID = require('mongodb').ObjectID;
```

2) Implement delete in server.js

```
app.delete("/notes", function(req,res) {
var id = new ObjectID(req.query.id);
db.notes.remove({_id: id}, function(err){
        if (err) {
                console.log(err);
                res.send("Failed");
        } else {
                res.send("Success");
        }
})
});
```

3) Add remove button to NotesComponent template

```
<ul>
   <li *ngFor="let note of notes">
      {{note.text}} <button
(click)="remove(note._id)">remove</button>
   </li>
</ul>
```

4) Extract reading notes from constructor:

```
constructor(private http: Http) {
   this.readNotes();
}

readNotes() {
   this.getNotes().then(notes=>{
      this.notes=notes
      console.log(notes);
   });
}
```

5) Define remove function in NotesComponent

```
remove(id:string) {
   let params: URLSearchParams = new URLSearchParams();
   params.set('id', id);
   this.http.delete(this.notesUrl, { search: params })
      .toPromise()
      .then(response => {
         console.log(
            `note with id ${id} removed, response`, response);
         this.readNotes();
      });
}
```

6) Also update addNote() to retrieve notes with their ids:

```
addNote(note:Note) {
   this.http.post(this.notesUrl, note).toPromise()
      .then(response => {
         console.log("note sent, response", response);
         this.readNotes();
      } );
}
```

Now you can restart server and refresh page, and delete notes by clicking remove button.

Additional tasks

1) Save date and time of adding note in date field. Print it in hh:mm dd.mm.yyyy format (use filter to format the date):

```
{{date | date: 'HH:mm dd.MM.yyyy'}}
```

2) Add possibility to reorder notes (add "send to top" button)
For this:

- add order property to the note
- find the note with minimal order number by using query

```
db.notes.find().sort( { order: 1 } ).limit(1)
```

- read the minimal order value and decrement it
- update the note by setting order to decremented value
- use ordering by order when loading the notes from mongodb

3) Add possibility to edit notes