Web Development

Koffee Addikts

Semester Project

Jose Carlos Esparza Martí
Frontend/Backend Courses

Index

Koffee Addikts

Koffee Addikts is a web application where its main purpose is that businesses advertise their coffees and the user can know where to find their ideal coffee. In addition to being able to give their opinion on the site and help the user from their experience.
In this application we also listen to the opinion of our users about it, through the function of contacting us, which we will explain later.

Technologies employed

To create this web app I've chosen three different technologies, for the server NodeJS, as database MongoDB and in the frontend I use EJS templates.

- NodeJS: I'm using this runtime environment because it's based in JavaScript and has one of the biggest developer community in the server area. Also, I've chosen it because it's made to create scalable applications, feature that will be important for the future development of my app.
- MongoDB: I've decided to use MongoDB because I've never worked with a non-relational database before, so it was a great opportunity to learn and understand the structure and way to work of this databases.
- EJS: it's a template language that allows you to generate HTML markup with vanilla JavaScript. Like NodeJS, I've chosen it because it has a large community of active users and it's always under development. The best feature is that to use my JavaScript code we just have to add this tags: <% %>. Also, with EJS I can render directly dynamic pages with the combination of Express and I just have to pass the JSON data params and later manage the values using the EJS data binding functionality.

Inside NodeJS

NodeJS uses a module manager, NPM. This manager is already installed in the environment when we install NodeJS.

The most used packages in my project have been express and mongoose. Express has been used for the creation of the server in a routing structure. Mongoose has been used for the manipulation of my MongoDB database with a route structure. The next schema will be attached as an extra pdf document for a better view.

app.js
/

notes.js
/notes
/notes/new
/notes/:id
/notes/:id/edit
/notes/images/:filename

comments.ejs
/notes/:id/comments
/notes/:id/comments/new
/notes/:id/comments/:comment_id/edit
/notes/:id/comments/:comment_id/edit

users.js
/user/registes
/user/login
/user/logout
/user/:id/edit
/user/:id
/user/:id/follow-user

contact.js
/contact

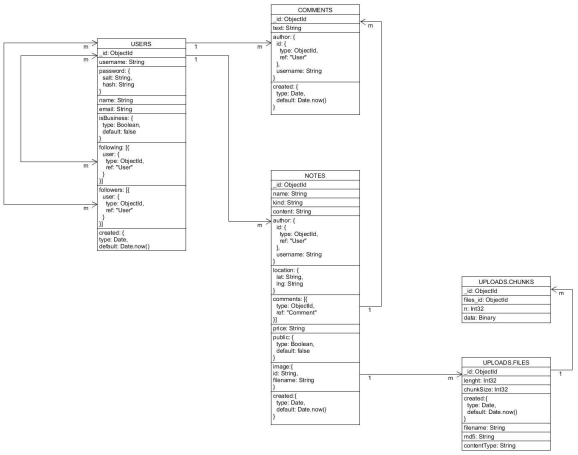
api.js
/api
/api/note
/api/note/:note_id
/api/comment/:comment_id
/api/user
/api/user/:user_id

T

Route squema

App Functionality:

We've created a complete CRUD functionality for my big three collections coffees(notes), comments and users. As well, we have two more collections to store the images, uploads.files and uploads.chunks.



UML diagram

The UML diagram will be attached as an extra pdf document for a better view.

As we can see in the previous diagram, I am using both embedded and normalized data. This is because the relations such as user-comments or users-coffees(notes) are one-to-many relationships. In these cases I am duplicating the username because it gives us more reading speed.

Instead in the comments-coffees(notes) relation, I use a normalized design model, since in this case we can avoid the duplication of information with the use of the reference to comments.

Now are going to explain more deeply the functionality and characteristics of each of these cases.

Coffees(Notes):

We can create, read, update and delete (CRUD) coffees. Only the business users are allowed to create, update and delete their own coffees, we will explain later the difference between the two types of users. Everyone can read the coffees, even if you don't have a user.

The coffees can be public or private, when a coffee is private, only the owner can see it. Also in the notes we can store images. For that process like we have mentioned before we are using crypto, path, multer and multer-gridfs-storage npm packages. We are saving the images in two collections in the database, uploads.files and uploads.chunks. Each image has one file and two chunks, in this way we are saving the images using less space and more security.

In the creation of the coffees we are saving automatically the author of them. Also we are saving the coordinates of where you can get the coffee in two strings, referring the latitude

and the longitude. In this function the interesting part is how we use the data for the view and use of the Google Maps JavaScript API. We are calling the Map API and then we are creating a map using the position that we have saved in my database.

Users:

Like we have mentioned before, we can create two types of user accounts, a business account or a personal account.

First of all, to create a business account we have to provide you a code to create this type of user, after verifying that you are a company interested in publish your products in my web app. In other words, if you don't have this secret code the users are just allowed to create personal accounts. Secret code: secretcode123.

With a business account a user is allowed to create new coffees, update and delete his own coffees and see all the public coffees of the rest of business users. Also, this type of user can do a full CRUD with comments.

The difference with a personal account is that they are not allowed to create and manage coffees, but the can comment to give their opinion in the coffees.

Also both accounts can use the contact us function, we will explain it later in the extra functionalities part.

To save the password of the users we are using two npm packages, passport and passport-local. In this way we are not storing the password as plain text, we are saving it in a salt and hash string.

Without an account, the user can only view the coffees, comments and the user profiles.

Comments:

As stated above, the comments can be created by both types of users. But only the owner can edit and delete them.

General Stuff:

In this section we will describe the common characteristics of the big three collections

These collections are defined in my app following the structure of MongoDB models, we can see all the attributes with more detail in the UML diagram that we have placed after.

Also, all my routes are protected by a middleware that we have created checking the ownership of the data and if the user is logged in.

Extra Functionalities:

We have extra functionalities outside the requirements like the option of create a HTTPS server, a contact us page working SMTP server, the creation of an API for developer purposes, flash pop ups, coffee searcher and the use of GitHub as a control version tool.

HTTPS:
We have the option to run my server in localhost guaranteeing my users a secure communication. Also we are using two files to get this function, a certificate.pem and a key.pem.

This code is commented, since my default option is http. To enable it just eat the code http and decomment the https part.

Contact Us:
To implement this functionality we are using the NPM module nodemailer and mailtrap.io as a SMTP server.

The contact us function can be used just by user with a created account.

Developer API:
We have created a route /api where we can find the GET methods to obtain the public information of my database. This API can be used by everyone and for free, so another developers can use my information for their web apps.

At the moment we have just implemented the GET method, since this API is only to see the data and not to manipulate it.

Pop Ups:
The NPM module connect-flash was used to create the notifications inside the app and with a pop up view. This pop ups are a help for the user, as they guide him through my app. Not only if there is an error, also when you have created a coffee, edited a comment and much more.

Coffee Searcher:
Regular expressions have been used to create the coffee searcher engine. In that engine we are looking for all the coffees and types of coffees that match the word searched.

Seed File:
Koffee Addikts has the option to create data in the database for development purposes, using the seeds.js file. This function deletes all the old data and creates basic data in the database. This file by default is deactivated, to make it work we just have to uncomment the 14th line in app.js. And after commented again to don't delete the new information.

GitHub:
GitHub is the biggest version control and we have considered a great chance to use it in my project. We are not just using it to keep the code save, also we are using the new project area in GitHub. In this area, we are creating a Kanban board to follow the project issues, improvements and to dos. Click in this link to view the project.

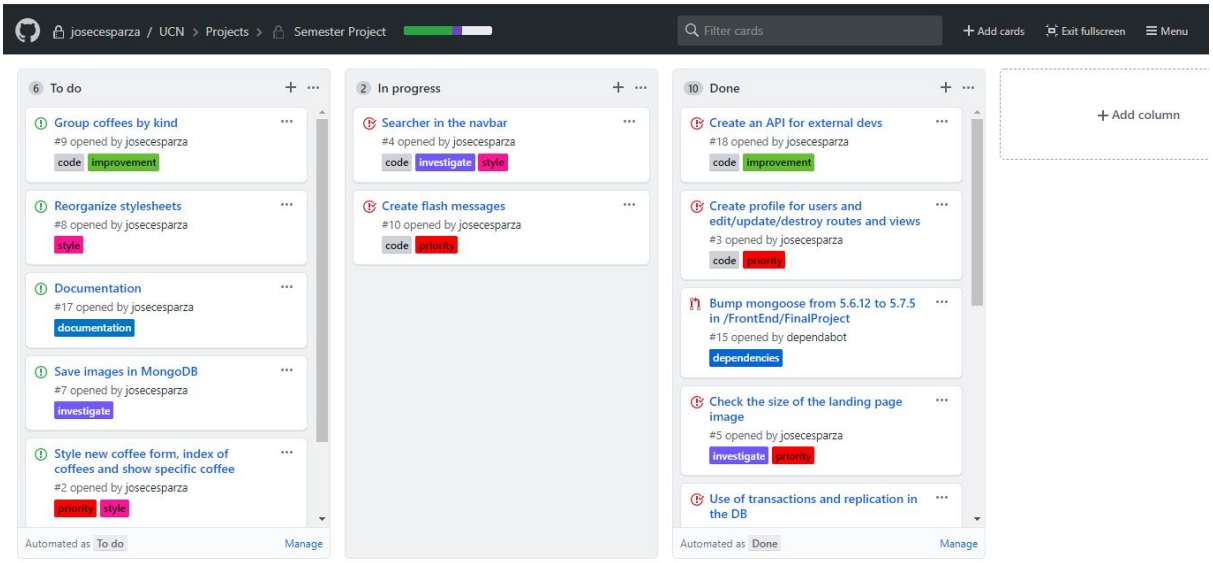


Image showing the use of GitHub

Conclusion:

As a conclusion I would like to say that this project is in its first phase of development. Since there are many ideas for its growth and improvement. From the types of data stored in a cafe to being able to share a coffee on social networks.

Also, we have deployed the app and can be tested by the following link:
https://koffee-adikts.herokuapp.com/

For the testing we created the user starcoffee as a business account and the user jose with a personal account.

CREDENTIALS	
USERNAME	PASSWORD
starcoffee	starcoffee
jose	jose

MongoDB Atlas link:
https://cloud.mongodb.com/

MONGODB ATLAS CREDENTIALS	
Cluster	Cluster0
Collection	koffee_app
User	josecarlosesparza@gmail.com
Password	m0ng0DB!

I hope you find your ideal coffee.