COM519 Assignment 1 CRUD Application

• Kacper Maselko (https://github.com/KMaselko)

Github repository: https://github.com/KMaselko/COM519

Hosted application (Heroku): https://boiling-fjord-53043.herokuapp.com/

This CRUD Application has been created for the module COM519 as an Assignment 1.

This application will help every gamer to look up the most important information about any video game that exists in a database of the application. The data that a user can look at are: - Title of a game - Genre - Price in \pounds - Publisher - Avarage amount of hours to finish the main plot

User can add, modify and delete data from the database that is connected to mongoDB using mongoose.

Running the project

To run the project using VS code, a user needs to clone this repository by using the command line such as cmd or Git Bash and inserting git clone https://github.com/KMaselko/COM519.

The user then needs to type npm install in their VS code terminal.

Rename config.en.example to config.env

After that, they need to change MONGO_URI URL into their own mongoDB database. This property can be find in config.env file.

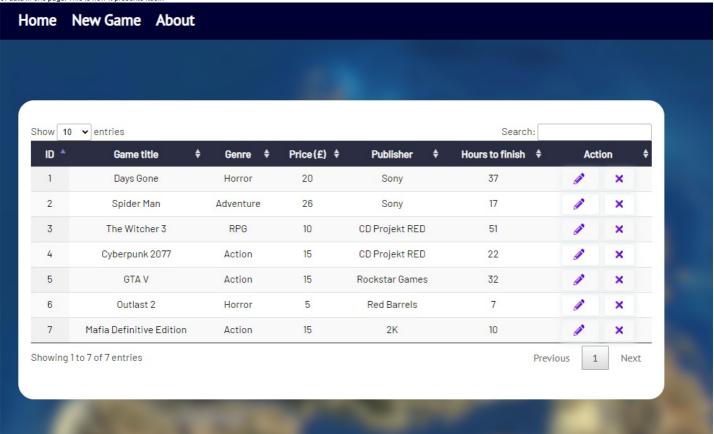
Finally, they need to type npm start in the VS code terminal

Alternatively, a user can access this application with this link: https://boiling-fjord-53043.herokuapp.com/.

If the provided link on Heroku does not send any requests (for example it does not delete or modify the entries) please run the project on VS Code by following the guide provided above and try again to send a

Usage

The application will load the main page as its index which will display all the data in the database. To make it more readable, I have implemented a sorting and filtering system so the user would not see hundreds of data in one page. This is how it presents itself:



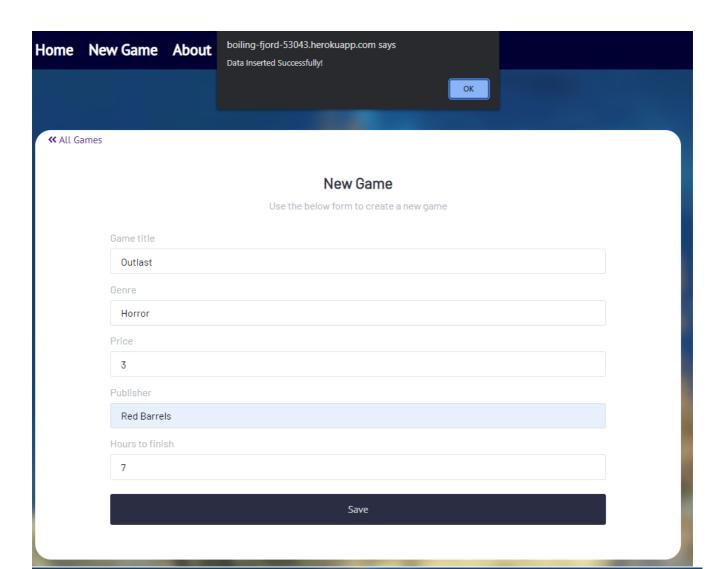
As we can see, the main page has only 7 entries

They can do multiple things to display the data in more readable way or to find something that the user is looking for, for instance the user can: - Sort the data by categories (ascending and descending) - Display a specified amount of entires in one page - Show next page for more entries - Search for any category in a search bar

The database is connected to the application so all entries that are displayed on the website are shown from MongoDB:

videoGames.games COLLECTION SIZE: 826B TOTAL DOCUMENTS: 7 INDEXES TOTAL SIZE: 36KB Find Indexes Schema Anti-Patterns (1) Aggregation FILTER { field: 'value' } QUERY RESULTS 1-7 OF 7 _id: ObjectId("61a3b94f420400a89c973bbf") name: "Days Gone" genre: "Horror" price: 20 publisher: "Sony" time: 37 _id:ObjectId("61a75a16a8d2e10f836f11d3") name: "Spider Man" genre: "Adventure" price: 26 publisher: "Sony" time: 17 _v:0 _id:ObjectId("61a91d0f0225433f65a62b70") name: "The Witcher 3" genre: "RPG" price: 10 publisher: "CD Projekt RED" time: 51 _v:0 _id: ObjectId("61a91d4e0225433f65a62b73") name: "Cyberpunk 2077" genre: "Action" price: 15 publisher: "CD Projekt RED" time: 22 _v:0 _id: ObjectId("61a91d810225433f65a62b75") name: "GTA V" genre: "Action" price: 15 publisher: "Rockstar Games" time: 32 _v:0 _id: ObjectId("61a91dc70225433f65a62b78") name: "Outlast 2" genre: "Horror" price: 5 publisher: "Red Barrels" time: 7 _id:ObjectId("61a91e050225433f65a62b7a") name: "Mafia Definitive Edition" genre: "Action" price: 15 publisher: "2K" time: 10 __v:0

If we modify any entry from the website, it would also save those changes in the database.





```
_id:ObjectId("61a91e050225433f65a62b7a")

name: "Mafia Definitive Edition"

genre: "Action"

price: 15

publisher: "2K"

time: 10

_v: 0

_id:ObjectId("61acf27f7cb5c994bd6315e5")

name: "Outlast"

genre: "Horror"

price: 3

publisher: "Red Barrels"

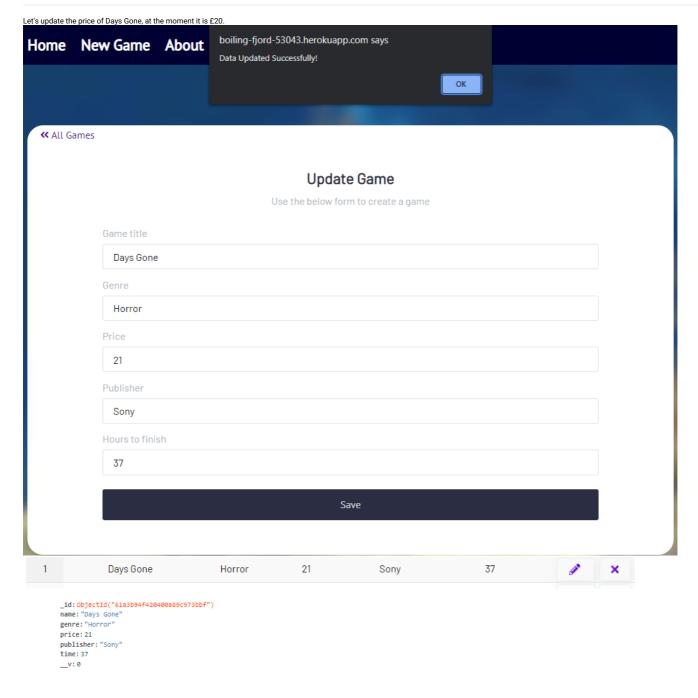
time: 7

_v: 0
```

it is being read from that database to display on the website.

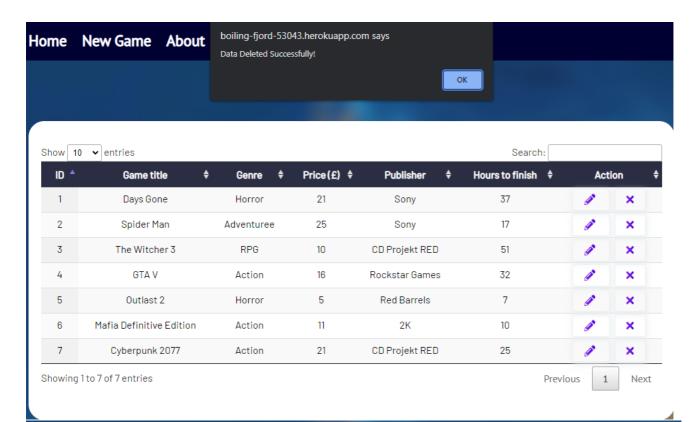
As shown above, it is clearly seen that an added entry has been inserted into the database and

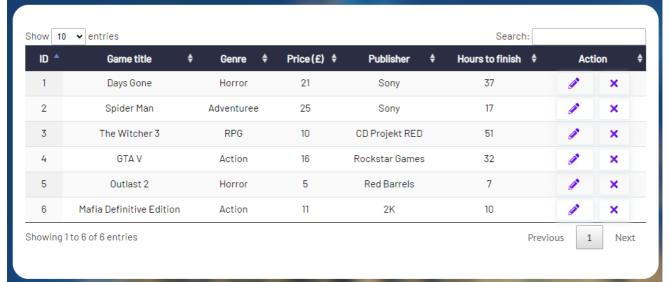
Modify



As you can see, the price has been increased to £21 $\,$

Delete





Problems solved

This simple application can solve multiple problems such as: - Checking how long does it take to beat a given game - Looking for a game by its genre (if someone is interested in a specific genre but does not know any games) - Checking the price - Checking the publisher - Checking which game is the longest or the shortest (if someone does not have much time to play for example)

Problems not solved

Unfortunately, there is one main function that has not been implemented due to lack of knowledge. It is **Login as Administrator** so everyone has the same permissions meaning everyone can edit the database. It took me so long to figure out how to implement it but unfortunately failed to do so, therefore I decided to leave it as it is.

Structure

This website application has been built fairly simple. It uses app.js as its index when npm start is typed. That application sets up all the necessary libraries, sets up a connection, defines a port, sets a server (EJS), uses different assets such as images, JavaScript and CSS, loads routers from the router file that will define different routes and finally runs a server.

Except of having different views as EJS that are just webpages, I have also provided different files that are seperated in different folders. The main folder is called server. This server contains: - Controller.js which will create every request - Connection.js which will connect to a server - Model.js which will create a schema containing the dataset - Router.js which will retrieve GET request when loading any webpage - And finally render.js which will render these webapges for the router

I am using API to render the database as JSON from MongoDB so the application will get information from this route. It looks like this: //API route.post('/api/games', controller.create); route.get('/api/games', controller.find); route.put('/api/games/:id', controller.update); route.delete('/api/games/:id', controller.delete); A user can access /api/games/route to see the database but it is not very readable.

When it comes to PUT and DELETE requests, the application will render a route with its original ID so that is how the application will know which exactly entry will be deleted or modified.

Security

It is also using MongoDB connection that needs a password, this connection is set up in config.env so only people with an access to the application files will be able to access that database.

Unfortunately as I mentioned earlier, it does not have login system implemented meaning that everyone could modify the database through the application, in these terms the website is not very secure.

Conclusion

In conclusion, I think that the project has been done fairly good. I am only missing couple of features where the login feature is the main one that is also missing.

I had lots of fun working on this project, I have learned a lot about Express, Node JS, MongoDB and couple new libraries such as axios and cors. I had an opportunity to polish my JavaScript skills as I have a feeling it was lacking a bit before that module appeared.

I have been using several references to help me find solutions to some problems such as YouTube, Stackoverflow and SOL Learning.