|  |
| --- |
| **Advanced Database Systems (COM519) Rodrigo Queiroga Amaral** |
|  |
|  |
| Link to local web application: <http://localhost:3000/home>Link to Github repository: [https://github.com/RQAmaral/ADVANCED\_DATABASE\_SYSTEMS\_COM519\_ASSESSMENT](https://github.com/RQAmaral/ADVANCED_DATABASE_SYSTEMS_COM519_ASSESSMENTt) |

# 

# **Introduction**

Hospitality is important in every country, it is an industry that allows for other people to feel comfortable when they are not in their usual area. It spans many industries, from restaurants to hotels. Restaurants need to make people feel satisfied and hotels strive to be like a home away from home.

Almost every industry uses technology to aid their workers in their tasks, allowing the service to be more efficient. Hotels are one such industry, using technology to keep track of everything from payments to room service.

The goal of this app is for the users to be able to see the rooms, add new room reservations and just provide information on the rooms in general.

The ultimate goal for the app is to be used in hotels, by housekeeping staff and managers alike, in order to keep track of the status of each room in real time, as a way to make the housekeeping process as efficient as possible, and keep its cost controlled.

***System Overview***

The flow of the system is simple: It brings forth information from the database and allows the user to interact with the information in the database.

On a deeper level, the application connects to a MongoDB Atlas database through the mongo.js file, which is then imported into index.js, and used in every exchange of information between the database and the website.

On startup, index.js will check the views folder and render the index.ejs page by using the app.get() function, which renders the page and sends a list of rooms fom the database to the ejs file, which, just like every other page consists of a header, a body and a footer, in which the header and the footer are located in the “common” folder within the “views” folder.

The body of index.ejs is also located in “common”. It is called “search.ejs”, and it uses javascript to render the database entries on the main page.

Uma imagem com texto

Descrição gerada automaticamente

Index.ejs

Uma imagem com texto

Descrição gerada automaticamente

Database

Pressing the “New Reservation” button will lead to the add.ejs page, which allows the user to create another entry on the database. It presents the user with a form that allows them to enter a room number, a name for the guest staying at the room and then, from a set group of options, to also pick a price(300£ or 600£) and a type of stay(Stayover or Departure).

When the user finishes adding all the information to the form, the user can then press the “add reservation” button to add the new entry to the database using a post function to retrieve the data from the ejs file and save the entry to the database.

Uma imagem com texto

Descrição gerada automaticamente

Add.ejs

Uma imagem com texto

Descrição gerada automaticamente

Index.ejs

Uma imagem com texto

Descrição gerada automaticamente

Database

If the user thinks they have made a mistake, or they just want to update a rooms status, they can head to the “Update reservation” tab, which allows them to pick from the entries on the database which one they want to update, then allowing them to change input the new information they need, much like the process to create the new reservation, but instead of allowing the user to input any room number, it then allows them to pick only from the rooms in the database.

Uma imagem com texto

Descrição gerada automaticamente

Update.ejs

Uma imagem com texto

Descrição gerada automaticamente

Index.js

Uma imagem com texto

Descrição gerada automaticamente

Database

Finally, if a room is unavailable, the user should be able to go to Delete Reservation and choose one the available rooms in the database and delete it from the database.

Uma imagem com texto

Descrição gerada automaticamente

Delete.ejs

Uma imagem com texto

Descrição gerada automaticamente

Index.js

Uma imagem com texto

Descrição gerada automaticamente

Database

***Key Design Decisions***

Database Design

The Database connection is defined in the mongo.js file, and the function is then exported to index.js, so we can use index.js to connect to the database.

Uma imagem com texto

Descrição gerada automaticamente

Mongo.js

Uma imagem com texto

Descrição gerada automaticamente

Index.js

As for what goes into the database, the schema.js file is responsible for structuring the object that gets sent to the database, as well as all its required fields. If a field is not included, the new reservation will not get sent to the database.

Uma imagem com texto

Descrição gerada automaticamente

Room-schema.js

Uma imagem com texto

Descrição gerada automaticamente

Index.js

Security and Scalability

For security, the project could be improved by storing the mongoDB details in a JSON file, so that it would only be called when necessary. Different types of accounts were considered, and could be implemented in the future, so that different users have access to specific functions instead of access to the whole program.

As for scalability, the project is scalable in the way that it would be very easy to add new features, such as more fields to the objects sent to the database, by updating schema, index and the ejs files in order to include more fields, or different types of accounts, such as user or admin, with different levels of permissions.

# **Conclusion**

Due to time and personal/professional constraints, the application implements only a small amount of the features that I would like to include in it.

Given more time, a user/admin system could have been implemented, wich would allow for the restricted use of features, with the option to delete the rooms being one of them, as right now, any person can delete all entries to the database.

There is a possibility that injection attacks would work on the database, but it is highly likely that MongoDB would handle some of those attacks. More testing on this would be required, which would take even more time.

Another feature that could be implemented is more information regarding the rooms, as right now the information that is provided about the rooms is very small. Things like if the room is VIP, If there are any special requests, or what rooms are available to be selected. A room limit should also be implemented,

When all is said and done, this project was very interesting to develop, I learned new ways to access and structure data using MongoDB and Mongoose, developed my skills with ejs and javascript, and managed to learn new ways of building a fullstack application, and I now have even more tools to become a better developer and a better professional.