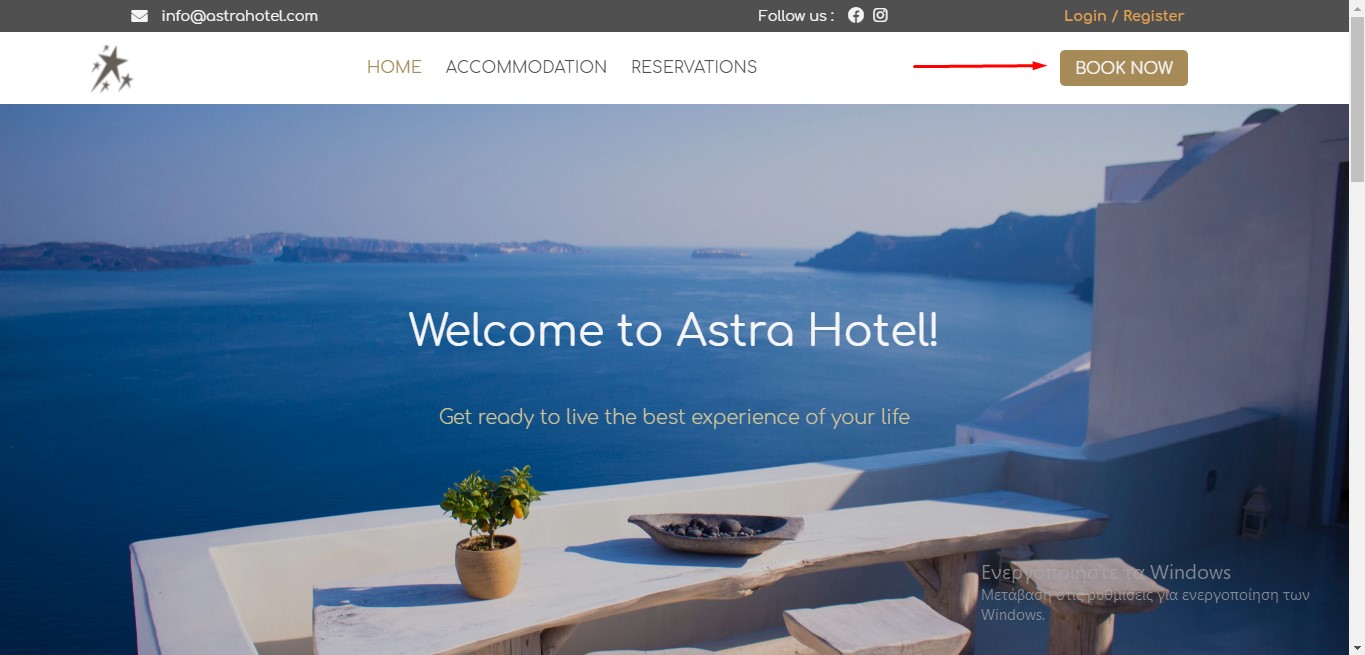
Hotel reservation web application

**Full Name**: Panagiotis Sakellaropoulos

In this project, the development of a full stack web application was implemented, which was about the reservation of accommodation in a hotel. The main goal of the application was the user-friendliness and the optimal user experience, who would be able to complete their task easily and at the same time with the smallest possible cognitive process. For this reason, special emphasis was placed on the responsiveness of the application, as well as on the flow of screens through which users were required to pass in order to complete their task successfully. The tools that were used are the back-end JavaScript runtime environment "Node.js", the "Express.js" framework, the NoSQL "MongoDB", the front-end design tools "CSS", "vanilla JavaScript" and "pug" template engine for the html page rendering, as well as the “Bootstrap” framework, which was very helpful for the responsiveness of the application.

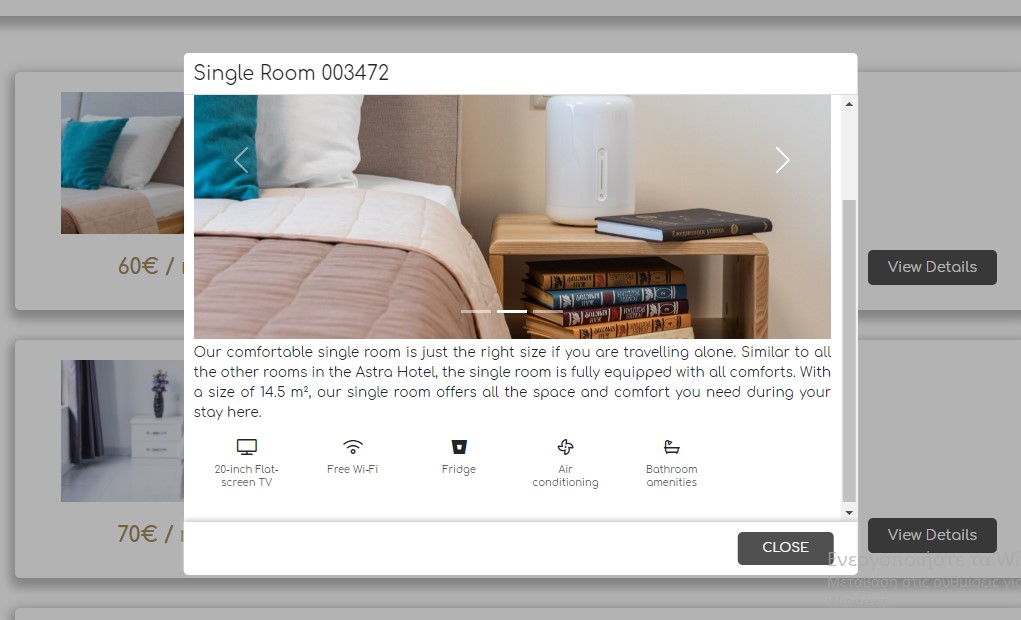
Initially, after entering on first page of the website, user should proceed by pressing the "BOOKNOW" button at the top right of the page (Image 1). It is worth noting that this button was separated from the menu with background color, to pique the user's interest and make it noticeable.

Image 1

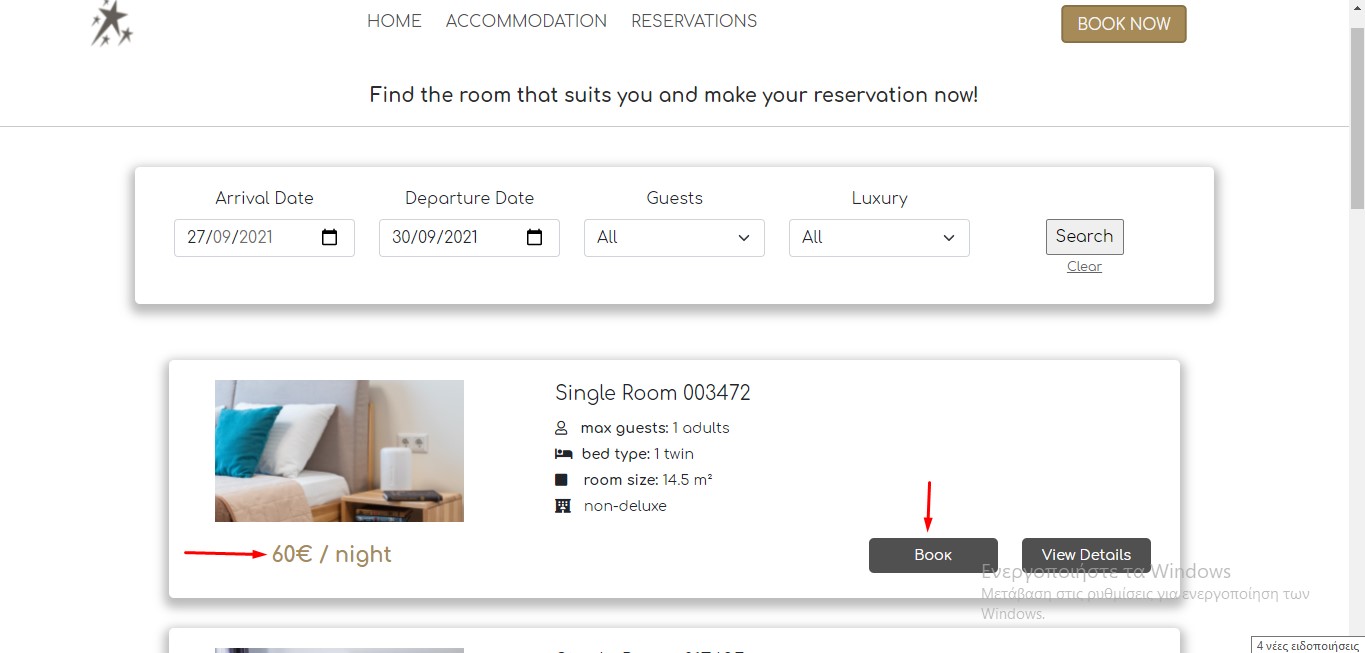
User was then taken to the rooms page (Image 2.1) where she/he had the ability to read more information about each room, via a pop-up popping up by clicking the "View Details" button (Image 2.2), while at the top of the page there was a "Search Bar". With this, the user had the opportunity to select the desired arrival and departure date, as well as two optional filters related to the number of people in each room and whether she/he wanted to be deluxe or not.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματαImage 2.1

Image 2.2

Then, by pressing the "Search" button, the application was able to find all the accommodations that met the above conditions and that were available during the period which was determined by the arrival date and departure date entered by the user (Image 3). For arrival and departure dates, there was a check that the arrival date is always shorter than the departure date. If a room was already booked at that time, it could not be found through the search. In addition, the user at this stage, had the opportunity to see the prices of each available room for each night - which varied according to the dates he wished to make his reservation - and to proceed to the reservation of the room he wished through the “Book” button. Finally, he could remove the filters by pressing "Clear" under the "Search" button.

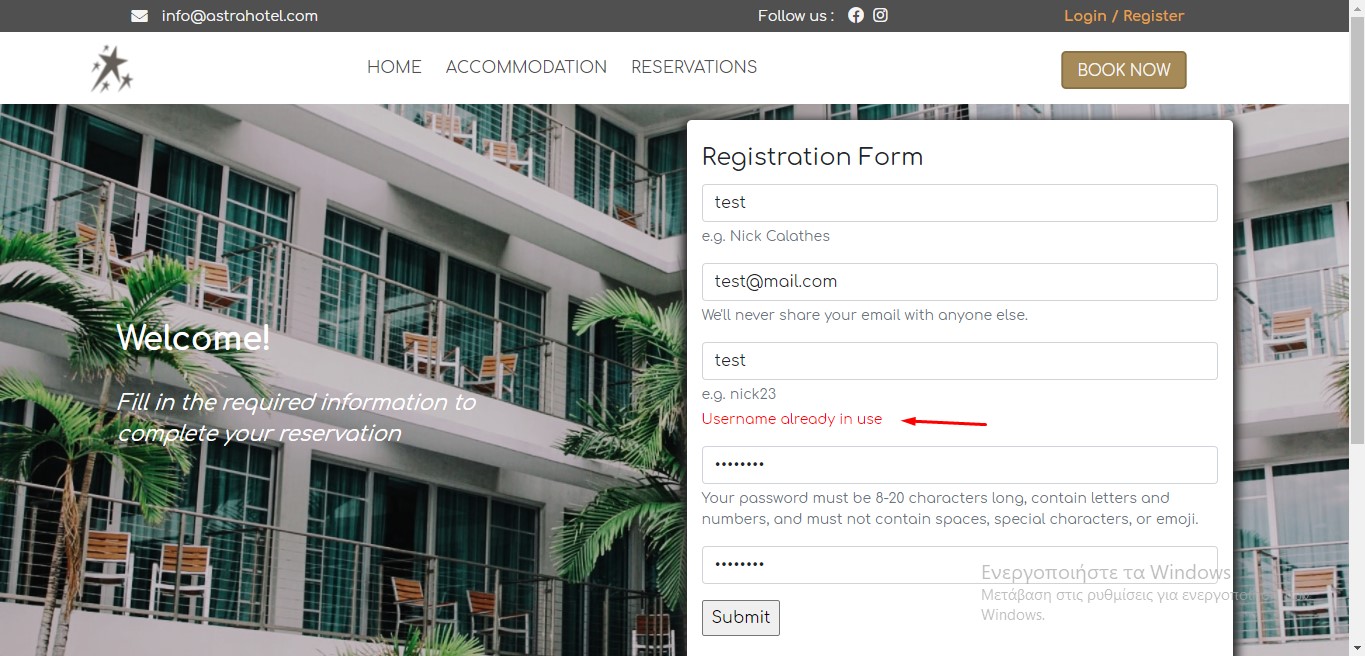
Image 3

By pressing the "Book" button, the user went to the next page (Image 4) where she/he could get informed about the details of the room he wanted to book. These included the desired arrival and departure dates assigned by the previous step, as well as the total cost of the room he had to pay to complete her/his booking. Finally, with the "Confirm" button the user was transported to the last step of her/his reservation.

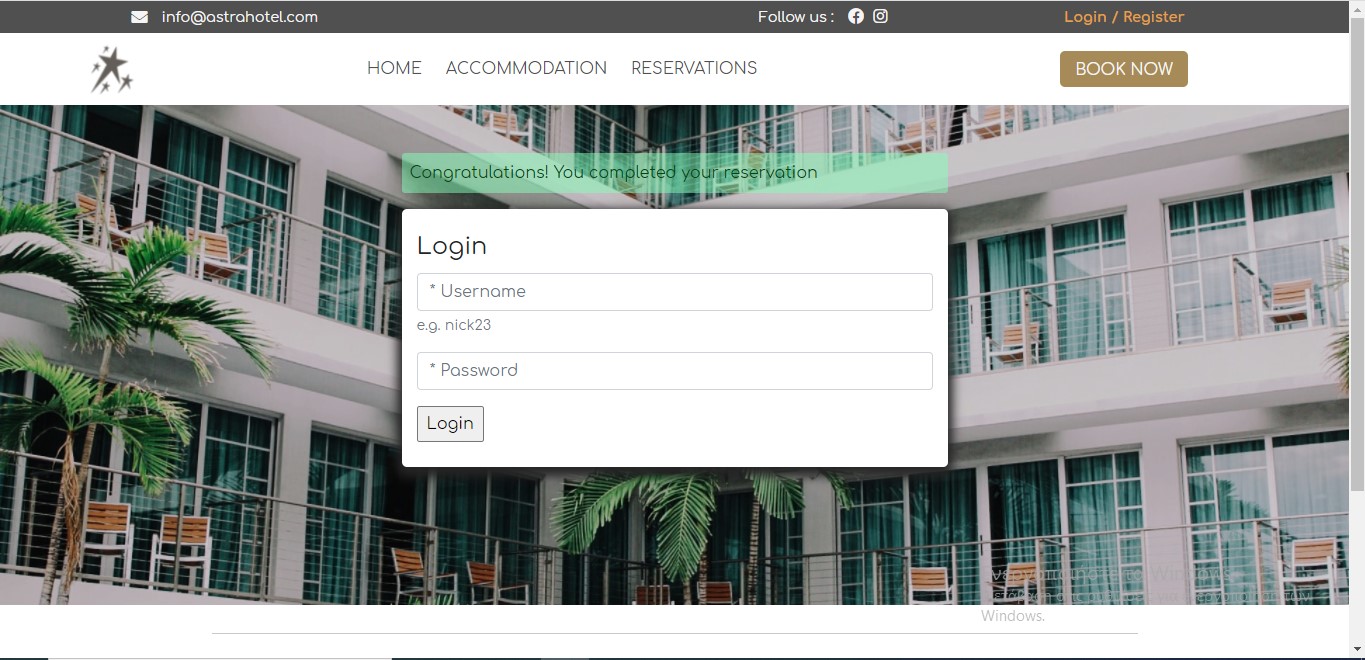
Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματαImage 4

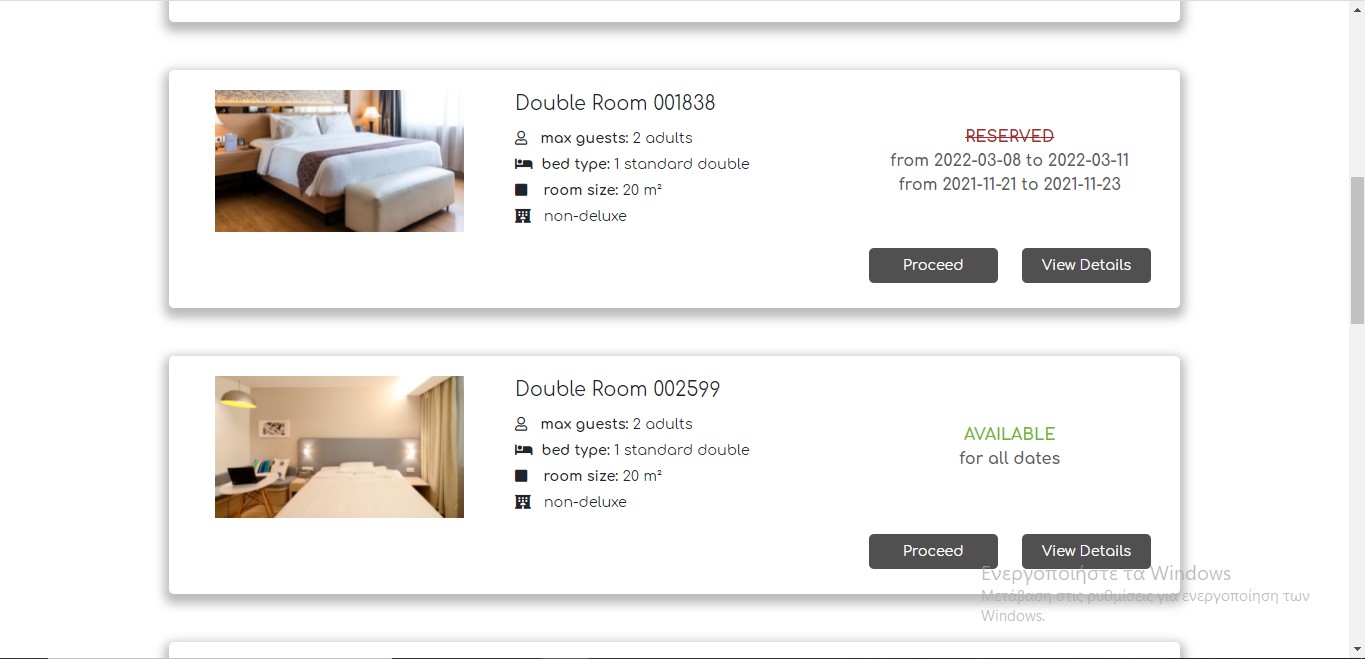
This page (Image 5) contained a form with the details that user had to enter and submit to complete the booking. For the submission of the form, there were various constraints which concerned that all the fields of the form were required. In addition, the user was not allowed to enter the same username as a previous user, and the verification codes he provided had to be the same and contain from 8-20 characters.

Image 5

Pressing the "Submit" button, successfully completed the user's task, and confirmed the reservation of an informative pop-up message (Image 6).

Image 6

In addition to the task of booking a room, the user had the opportunity to check for the availability of accommodation by selecting from the menu "RESERVATIONS". This page (Image 7) listed for each room all the dates that had already been booked and was not available.

Image 7

**Technical parts of the project**

The technical parts of this project will be briefly explained below.

Firstly, it is worth mentioning that all the pages have been created using the Bootstrap’s front end framework grid-system. In addition, this framework has been used for the carousels that exist both on the home page and in the pop-up with the photos for each room. The pop-ups are made with css and vanilla javascript.

The design of the database has been implemented with four models.

The first model was about the collection "**rooms**", which includes all the information about each room as shown on the booking page, to which the user first goes after clicking on the "**BOOK NOW**" button, as described above. In addition, this model includes some variables, which are "**updated**" with the "**set**" command of mongo. Specifically, "**pricePerNight**" and "**totalPrice**" are referred to the variables in which the average price of each room for each night and the total price of each room are stored, respectively. In addition, the boolean variable “**available**” is also "**updated**" and is false if the corresponding room is reserved for the dates entered by the user. When the user presses the “**Search**” button, the variables “**possibleArrivalDates**” and “**possibleDepartureDates**” which are part of the “**temporaryBooking**” collection are updated too.

In addition, when the user select the desired room and goes from the “**booking**” page to the “**booking/:id**” page, where the selected **arrival** and **departure** dates are displayed as well as the room **name** she/he wishes to reserve, the “**temporaryBooking**” collection is updated, which concerns the temporary reservations and includes 3 values ​​regarding the possible **arrival** and **departure** dates of the room, as well as the **name** of the room. This temporary storage is necessary so that as soon as the user fills the final “**booking form**”, the room **name**, **arrival** date and **departure** date are updated and stored.

Once the user chooses to submit her/his details in order to complete the booking, the room **name**, the **arrival** and **departure** dates, which are taken from the collection "**temporaryBooking**", as well as the **username** of the user, which is taken from the form submission (with the “POST” request) are stored in the “**booking**” collection.

Finally, there is the model of the collection "**users**", which allows to check some of the form inputs. More specifically, the "**username**" variable for the user's username has the property of “**unique”**, so that no user can have the same username. In addition, the variables "**password**" and "**passwordConfirmation**" check if the passwords submitted by users are the same.

It is worth mentioning that for the page "**reservations**", where users could be informed about the availability of accommodation, a new object has been created in the reservation route, which had the following structure:

{

room\_name1: [

{ arrivalDate: arrivalDate1, departureDate: departureDate1 },

{arrivalDate: arrivalDate2, departureDate: departureDate2 }, …

],

room\_name2: [

{ arrivalDate: arrivalDate1, departureDate: departureDate1 },

{arrivalDate: arrivalDate2, departureDate: departureDate2 }, …

],

…

}

This object thus informs us about each room in a list of objects for the **arrival** and **departure** dates of each saved booking. This information is taken from the collection "**bookings**" and "**rooms**".

A lot of promises “**Promises”** were used in the route files, so that the collections of the database were updated before the pages were rendering through the pug template.

Finally, all the photos used were free and came from the website "unsplash.com".