



DEVELOPING OUTSYSTEMS WEB APPLICATIONS

Bookings Assignment



Table of Contents

| | |
|--|----|
| Assignment 1: Bookings Data Model..... | 3 |
| Assignment 2: Bookings Widgets..... | 7 |
| Assignment 3: Bookings Data Model II..... | 10 |
| Assignment 4: Screen Lifecycle | 11 |
| Assignment 5: Data Queries & Widgets | 14 |
| Assignment 6: Encapsulate Bookings Logic | 17 |
| Assignment 7: Bookings Input Validation..... | 20 |
| Assignment 8: Improve Bookings User Experience | 23 |
| Assignment 9: Reuse Bookings Page Elements | 24 |
| Assignment 10: Secure Bookings..... | 25 |
| Assignment 11: Improve Bookings UI | 27 |
| Assignment 12: RichWidgets | 32 |
| Assignment: Bookings Extra..... | 36 |
| List of Figures | 38 |

Copyright

This material is owned by OutSystems and may only be used in the ways described in this Copyright Notice:

- You may take the temporary copies necessary to read this document
- You may print a single copy of this material for personal use
- You must not change any of this material or remove any part of any copyright notice
- You must not distribute this material in any shape or form

Assignment 1: Bookings Data Model

The goal for this Project is to implement the **Bookings** application to be used by the **hotel staff** to book rooms for hotel guests. Each hotel room accommodates a number of adults and a number of children, includes some amenities (Television, Internet Access, etc.) and has a price per night.

A hotel guest calls in to make a room reservation and the clerk at the reception checks for available rooms. If there is a room available, the clerk registers the reservation in the system with the guest's name, the arrival and leaving dates, alongside the number of adults and children staying in the room.

Upon arrival, the guest confirms the booking at the reception and is checked in. Upon departure, the guest checks out at the reception.

1. Create the Application

- a) Inside Service Studio, and after successfully connecting to the Server, select the **New Application** option.



Figure 1. Create a New Application

- b) This is going to be a **Web Application**, so select that type and click on the **Next** button.

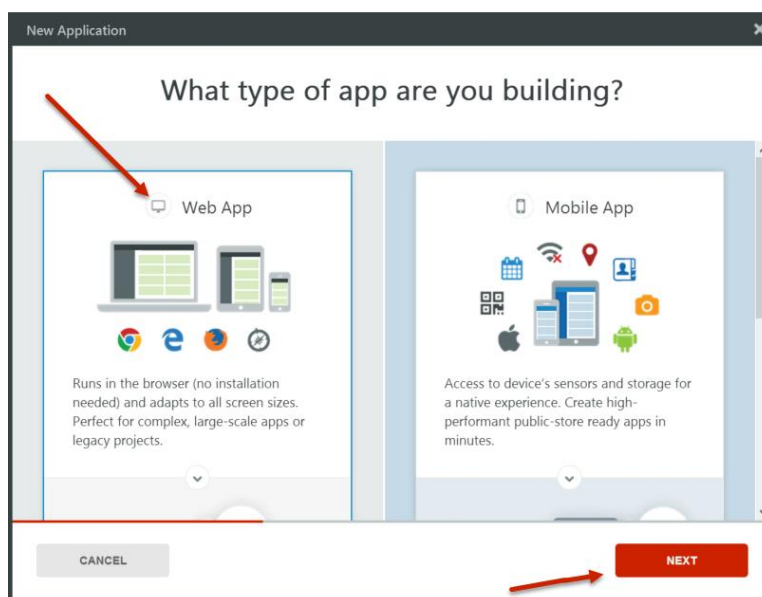


Figure 2. Select the Web App option

c) Select the **Web Application** template.

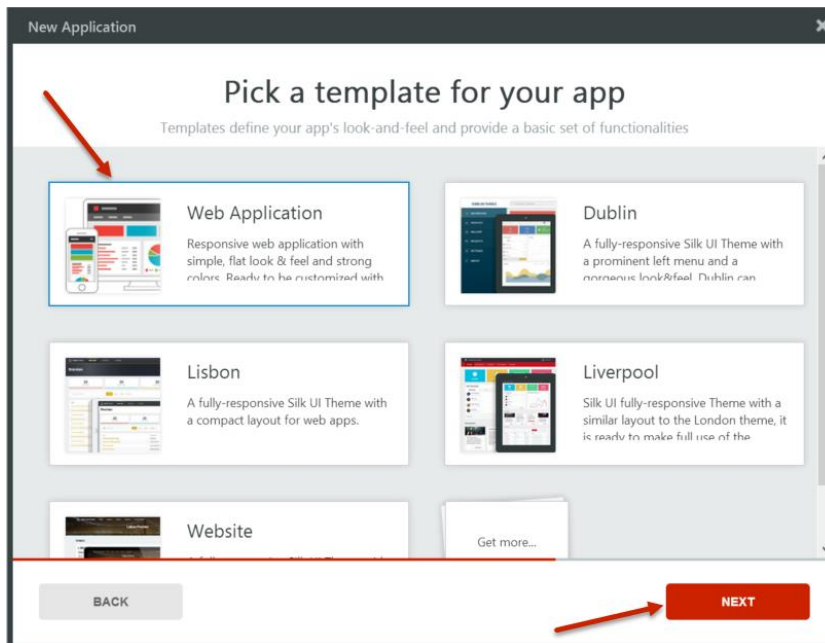


Figure 3. Select the Web App template

d) Name the Application '**Bookings_<your initials>**', upload the Bookings logo as the Icon of the application and create it. You can find the logo in the **Resources** folder of the materials.

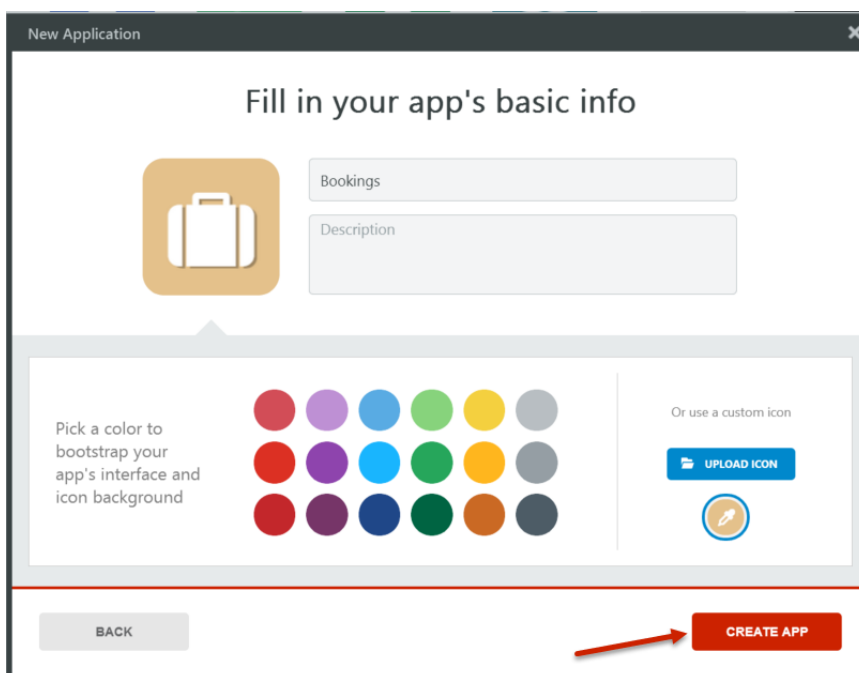


Figure 4. Give a name and logo to the application

e) Select the Module Type to be **Blank** and create a Core Module to keep the Data Model of your Application. Name it '**BookingsCore_<your initials>**'.

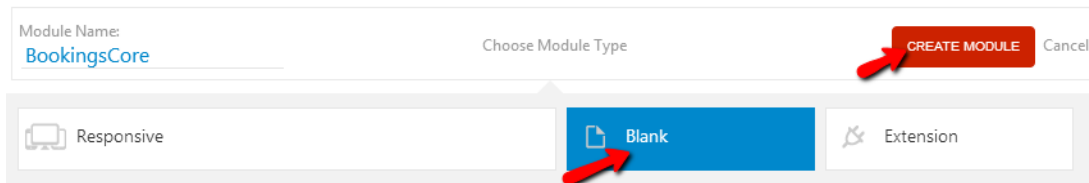


Figure 5. Creating a BookingsCore Blank module

2. Data Modeling

Create the application's **data model** according to the following definition:

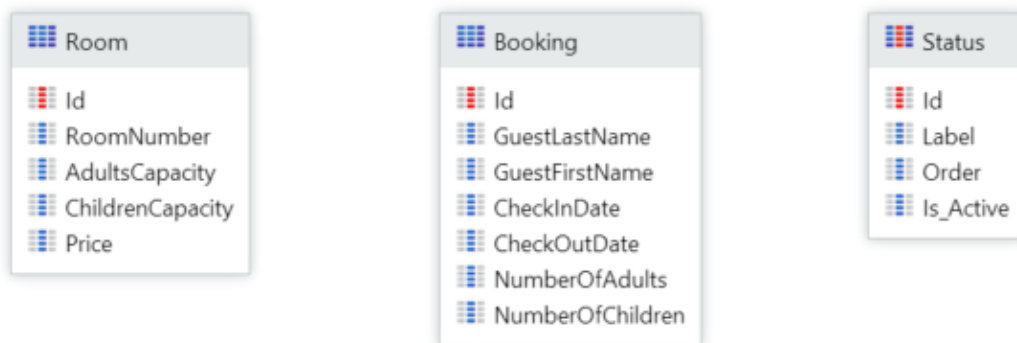


Figure 6. Room, Booking and Status Entities

Notes:

- **Room Entity**
 - RoomNumber is a **Text(3)**;
 - AdultsCapacity and ChildrenCapacity are **Integers**;
 - All fields are mandatory.
- **Status Static Entity**
 - Create the following records:
 - Booked
 - CheckedIn
 - CheckedOut
 - Canceled
 - Don't forget to validate if the **Label** property is correct for each record created.
- **Booking Entity**
 - GuestLastName, GuestFirstName are **Text** mandatory fields;
 - CheckInDate and CheckOutDate are of data type **Date** and also mandatory;
 - NumberOfAdults and NumberOfChildren are of data type **Integer**, with only the first being mandatory;
- Don't forget to set the **Public** property of all the Entities to 'Yes' and the **Expose Read Only** flag to 'No', of the non-Static Entities, so they can be reused.

3. Bootstrap Data for the Hotel Rooms

Bootstrap data for the **Room** Entity using the Excel file available in the **Resources** folder.

1

Publish your eSpace (F5)

Assignment 2: Bookings Widgets

1. Create a New Module

Go back to your previously created Application. Before creating the new Module, make sure Core Module is **not set** as **Home**. Since it was created as a **Blank** Module, it doesn't have an associated Theme.

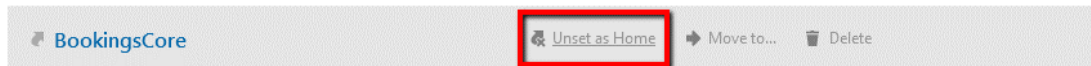


Figure 7. Unset the BookingsCore as the Home Module

- a) Select the **New Module** option.

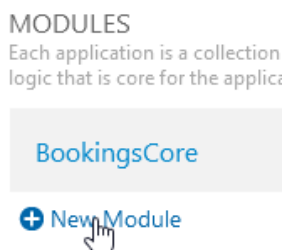


Figure 8. Create a new Module

- b) Name it '**Bookings_<your initials>**' and this time select the Module Type to be **Responsive**.

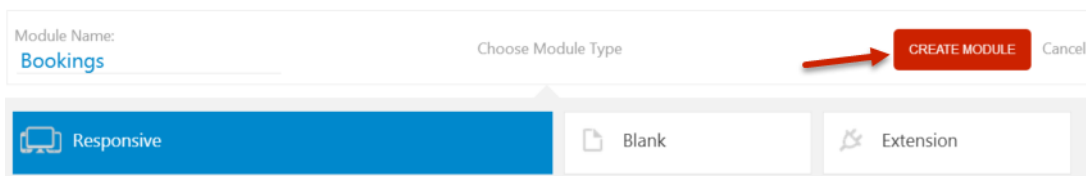


Figure 9. Choosing the Module Type to be Responsive

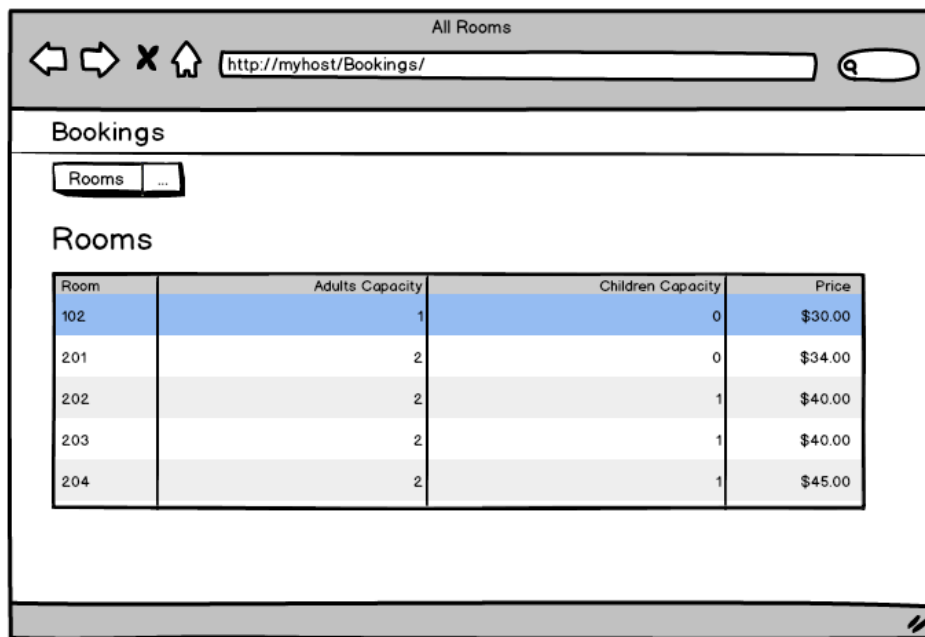
2. Create a Page to Lists Rooms

Add references to the Entities you created in the Core Module and create the **Rooms** page based on the mockup in Figure 10.

This Screen will list the Rooms in the Database, with the four columns displayed in Figure 10 (RoomNumber, AdultsCapacity, ChildrenCapacity and Price).

In Figure 10, right above the 'Rooms' title, there is the Menu of the application, with the **Rooms** Screen in it. To have the same behavior, add the **Rooms** Screen to the Menu, after creating it.

NOTE: Do not use Scaffolding patterns, including drag-and-drop of the Entities to the MainFlow, to create the Screens. This Assignment will also be helpful to gain experience on developing the application UI, without using accelerators.



The mockup shows a web browser window titled "All Rooms" with the URL "http://myhost/Bookings/". Below the browser, there's a "Bookings" section with a "Rooms" link. The main content is a table titled "Rooms" with the following data:

| Room | Adults Capacity | Children Capacity | Price |
|------|-----------------|-------------------|---------|
| 102 | 1 | 0 | \$30.00 |
| 201 | 2 | 0 | \$34.00 |
| 202 | 2 | 1 | \$40.00 |
| 203 | 2 | 1 | \$40.00 |
| 204 | 2 | 1 | \$45.00 |

Figure 10. Mockup of the Rooms Screen

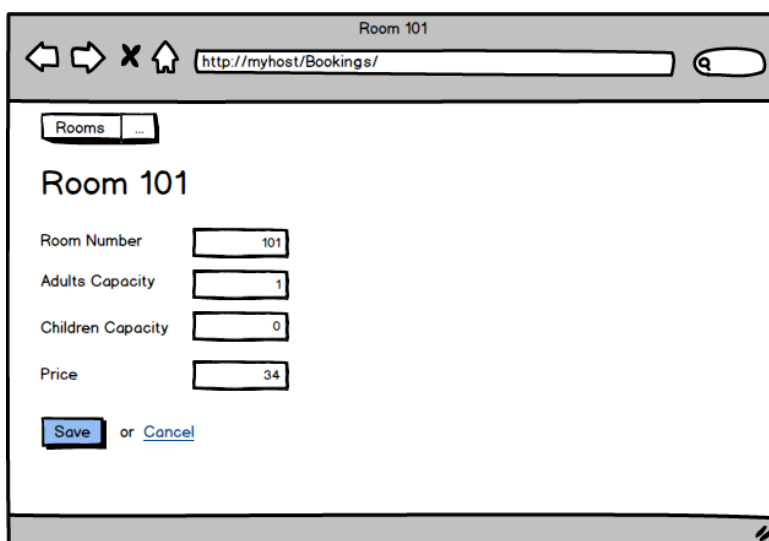
Note: Don't forget to tick the Anonymous role in the Rooms Screen.

3. Create a Page to Edit Rooms

Create a new **RoomDetail** Screen to allow you to edit an existing Room, based on the mockup displayed in Figure 11.

In this Screen, you need to fetch the Room by its identifier and then create a Form to enable editing the Room information.

At the bottom of the Screen, add a **Save** Button to save the changes to the database, and a **Cancel** Link to return to the Rooms Screen, without saving the changes.



The mockup shows a web browser window titled "Room 101" with the URL "http://myhost/Bookings/". Below the browser, there's a "Rooms" link. The main content is a form titled "Room 101" with the following fields:

Room Number:

Adults Capacity:

Children Capacity:

Price:

At the bottom, there are two buttons: **Save** and **Cancel**.

Figure 11. Mockup of the RoomDetail Screen

Notes:

- Once again, don't forget to tick the Anonymous role;
- Don't forget to add a **RoomId** as an Input Parameter to the Screen;
- Use the **CreateOrUpdateRoom** Entity Action to save the data in the Database, to enable using this Screen to create new Rooms as well, instead of just editing a Room information;
- In the **Rooms** Screen, add a Link on the **RoomNumber** in the Table, to enable opening the details of that Room, by clicking on its number. In this process, don't forget to use the appropriate **RoomId** as Input Parameter.

1**Publish** your eSpace (F5)

Assignment 3: Bookings Data Model II

1. Data Modeling

We now want to create relationships in the **Database**. As you may have noticed, we still don't have any way to relate the **Bookings** Entity with the **Room** or the **Status** Entities.

Reopen the **BookingsCore** Module. In the **Bookings** Entity, add a **RoomId** Attribute. Make sure its **Data Type** is set to Room Identifier. Repeat the process to add a new **StatusId** Attribute and make sure it is set to Status Identifier.

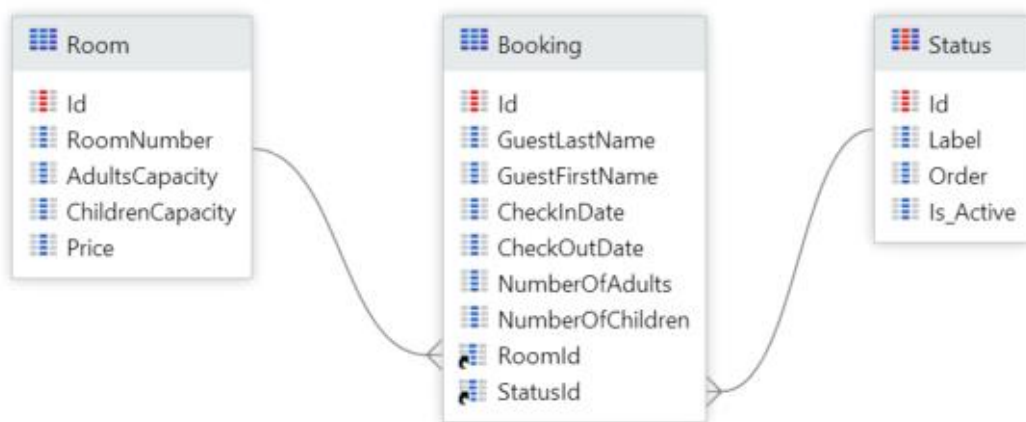


Figure 12. Bookings Data Model, with the relations between Entities

1

Publish your eSpace (F5)

Assignment 4: Screen Lifecycle

To book a room, the clerk at the reception fills in the guest's name, the check-in and check-out dates, the number of adults and children, and then checks for available rooms that match these criteria. The application assigns the cheapest room option and the clerk confirms the booking.

1. Create a Page to List Bookings

Create the '**Bookings**' page based on the following mockup. Make sure you refresh the references you have with the **BookingsCore** Module, so that the newly published version can be used.

Don't forget to add a reference to the **Bookings** Screen to the Menu of the application, as seen in the mockup in Figure 13.

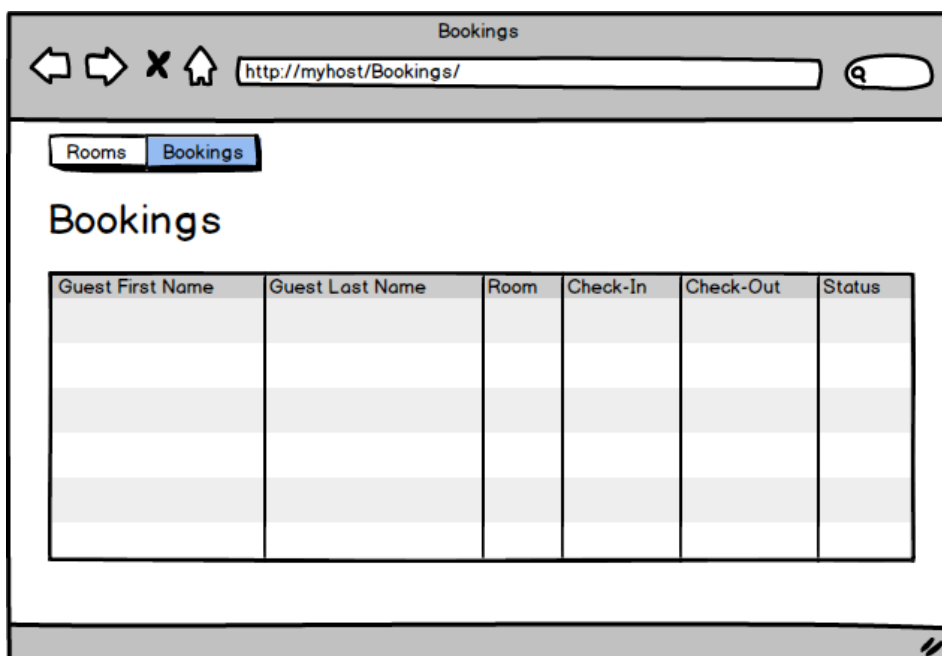


Figure 13. Mockup of the Bookings Screen

In a minute, you'll create a page to create new Bookings. For now, the list of Bookings will be empty.

2. Create the Detail Page to Book a Room

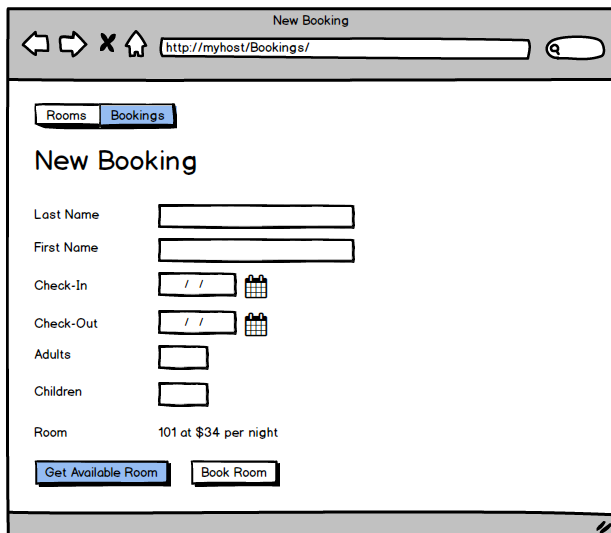
Create the **BookingDetail** Web Screen, to allow creating a new Booking for a particular Room. This new Screen will be used to create new Bookings, but also to change existing ones, so don't forget to add the **BookingId** as an Input Parameter of the Screen.

To add or edit information about a Booking, add a Form to the Screen with the fields presented in Figure 14.

At the end of the Screen, add two Buttons: **Get Available Room** and **Book Room**. The first Button will trigger an Action that will query for the cheaper available room, for the dates and number of

adults and children selected in the Form. The second Button will also trigger an action that actually creates the Booking in the Database, with the Room chosen.

Note: Since the Booking has a Room associated, don't forget to add the Room to the Aggregate in the Preparation.



The mockup shows a web browser window titled "New Booking" with the URL "http://myhost/Bookings/". It features two tabs: "Rooms" and "Bookings", with "Bookings" being the active tab. The form contains the following fields and controls:

- Last Name: Text input field
- First Name: Text input field
- Check-In: Date input field with a calendar icon
- Check-Out: Date input field with a calendar icon
- Adults: Text input field
- Children: Text input field
- Room: Display field showing "101 at \$34 per night"
- Buttons: "Get Available Room" and "Book Room"

Figure 14. Mockup of the BookingDetail Screen

The **Get Available Room** Action, associated with the respective Button, tries to get the cheapest available Room, for the Booking information added to the Form. To get the Room, you can add a SQL Query statement to the Action, with the following SQL:

```
SELECT {Room}.* FROM {Room} WHERE @NumberOfAdults > 0
AND {Room}.[AdultsCapacity] >= @NumberOfAdults
AND
{Room}.[AdultsCapacity]+{Room}.[ChildrenCapacity]>=@NumberOfAdults+@NumberOfChildren
AND NOT EXISTS
(SELECT 1 FROM {Booking} WHERE {Booking}.[RoomId] = {Room}.[Id]
AND (@CheckInDate BETWEEN {Booking}.[CheckInDate] AND {Booking}.[CheckOutDate] - 1 OR
@CheckOutDate BETWEEN {Booking}.[CheckInDate] + 1 AND {Booking}.[CheckOutDate])
AND {Booking}.[StatusId] <> @CanceledStatus)
ORDER BY {Room}.[Price] ASC
```

To properly create the SQL statement, don't forget to:

- Add the appropriate Input Parameters to the SQL statement to pass in the values from the Form to the Query, e.g: the NumberOfAdults, the NumberOfChildren, the CheckInDate and the CheckOutDate to the Form. Also add an addition Parameter with the Canceled Status.
- After fetching the cheapest Room from the Query:
 - Save the room information returned by the Aggregate in the Form variables. Set the **Room Number** and the **Room Price**, using the info returned by the Query.

Note: The last Form field in Figure 14, with the details about the Room, should use the information of the cheapest Room returned by the SQL Query.

The **Book Room** Button will trigger an Action that will create the new Booking in the Database. Before doing so, don't forget to set the **RoomId** of the Booking to the Room selected, and the **StatusId** of the Booking to the status 'Booked'.

Create the Links in the Bookings Screen to navigate to the BookingDetail Screen, to create new Bookings and to edit existing ones.

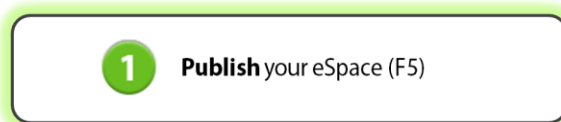


Assignment 5: Data Queries & Widgets

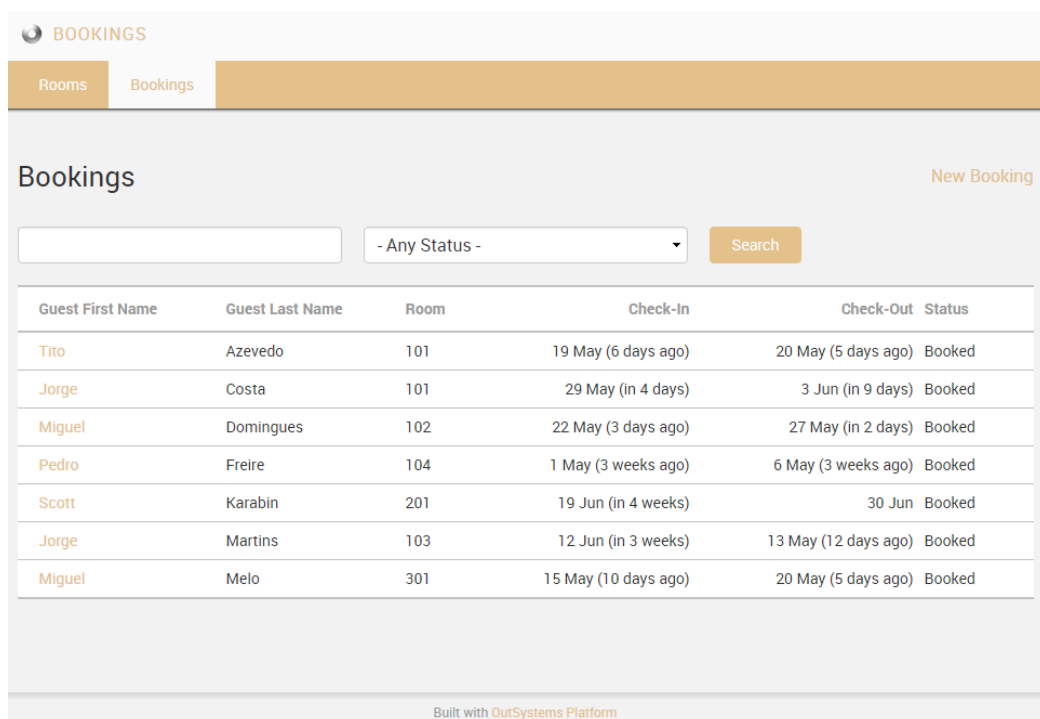
The goal of this assignment is to improve the application by using interface widgets, such as filters.

1. Add More Functionality to the List of Bookings

In the **Bookings** Screen, allow searching by the guest's last name and by the Booking status. When there are no filters applied, all Bookings should be displayed in the Table Records.



You should have something like this:



The screenshot shows a web application interface for 'BOOKINGS'. It has a navigation bar with 'Rooms' and 'Bookings' tabs. Below the tabs, there's a 'Bookings' section with a 'New Booking' link. A search bar is present with a text input field, a dropdown menu set to '- Any Status -', and a 'Search' button. Below the search bar is a table with the following data:

| Guest First Name | Guest Last Name | Room | Check-In | Check-Out | Status |
|------------------|-----------------|------|----------------------|----------------------|--------|
| Tito | Azevedo | 101 | 19 May (6 days ago) | 20 May (5 days ago) | Booked |
| Jorge | Costa | 101 | 29 May (in 4 days) | 3 Jun (in 9 days) | Booked |
| Miguel | Domingues | 102 | 22 May (3 days ago) | 27 May (in 2 days) | Booked |
| Pedro | Freire | 104 | 1 May (3 weeks ago) | 6 May (3 weeks ago) | Booked |
| Scott | Karabin | 201 | 19 Jun (in 4 weeks) | 30 Jun | Booked |
| Jorge | Martins | 103 | 12 Jun (in 3 weeks) | 13 May (12 days ago) | Booked |
| Miguel | Melo | 301 | 15 May (10 days ago) | 20 May (5 days ago) | Booked |

At the bottom of the screen, it says 'Built with OutSystems Platform'.

Figure 15. BookingDetail Screen, after adding the search functionality

2. Control the visibility of elements in the BookingDetail Screen

In the **BookingDetail** Screen control the information that is displayed during the flow of creation of a new Booking:

- Make sure that the line that displays the Room Number and the Room Price in the Form, is only displayed after clicking on the **Get Available Room** Button and successfully finding the cheapest Room available.
- Make sure that the **Book Room** Button is only shown after clicking on the **Get Available Room** Button.

3. Add Amenities to Hotel Rooms

In this Section of the Assignment, you will create the logic to allow the association of certain **Amenities** to the Rooms.

To do this, go back to your **BookingsCore** Module and create a new Static Entity called **Amenity**. Add the following records to the Static Entity:

- Television;
- Internet Access;
- Hair dryer;
- Premium Towels;
- Crib;
- Safe.

Note: Don't forget to define the **Label Property** of each one of the Records.

After having this Static Entity created, we now need a mechanism to relate the Amenities with a given Room. For that, create a new Entity and call it **RoomAmenity**. In this Entity, add a **RoomId** attribute with the **Data Type** set to Room Identifier, and an **AmenityId** of type Amenity Identifier.

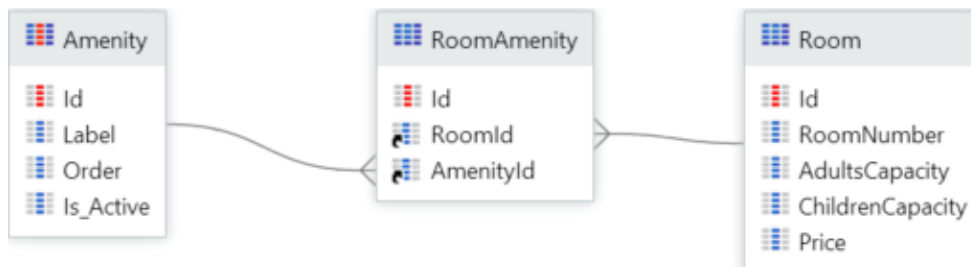
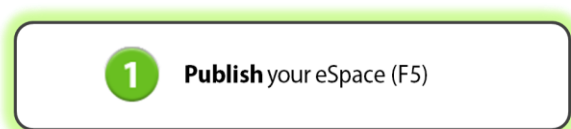


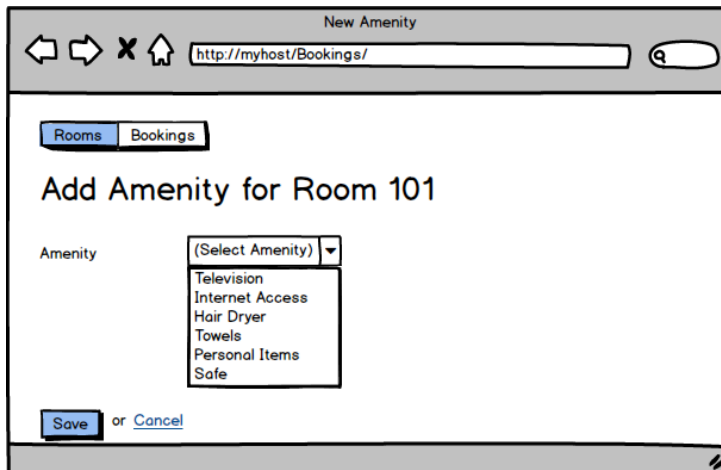
Figure 16. Part of the Data Model with Amenity and RoomAmenity Entities

Create a **Unique Index** in the **RoomAmenity** Entity, with the RoomId and the AmenityId attributes. Make these Entities **Public** and set the RoomAmenity's **Expose Read Only** property to 'No'.



Go back to your **Bookings** Module and add the new Entity references to the Core Module. In the **RoomDetail** Screen, add a new Link to a new Screen called **RoomAmenities**. Pass a RoomId to the new Screen as Input Parameter.

In the **RoomAmenities** Screen, implement the behavior that allows you to add Amenities to the specified Room, according to the following mockup:

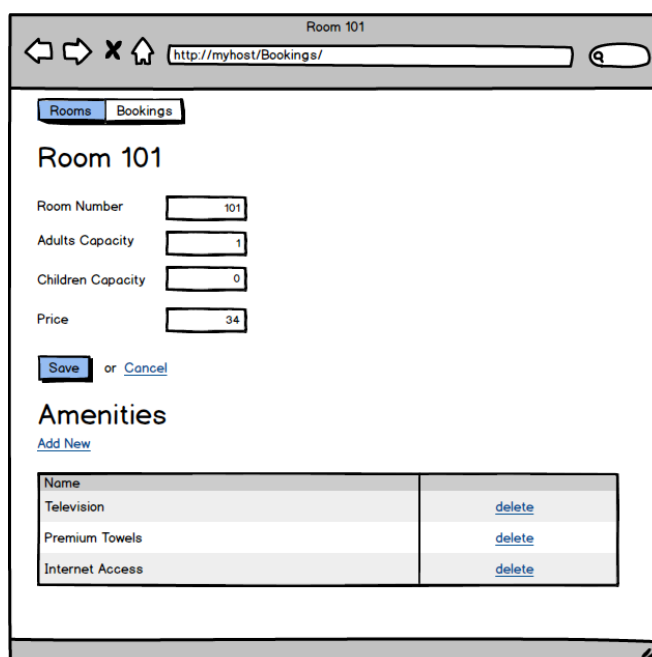


The mockup shows a web browser window titled "New Amenity" with the URL "http://myhost/Bookings/". It features a tabbed interface with "Rooms" and "Bookings" tabs. The main heading is "Add Amenity for Room 101". Below this, there is a label "Amenity" followed by a dropdown menu currently showing "(Select Amenity)". The dropdown list includes: Television, Internet Access, Hair Dryer, Towels, Personal Items, and Safe. At the bottom, there are "Save" and "Cancel" buttons.

Figure 17. Mockup of the RoomAmenities Screen

In the original **RoomDetail** Screen, add a Table to list all the Amenities for the current Room, below the Form. Also add a Link to delete one Amenity. An example of this scenario can be found in the following mockup, with a Delete Link in every row to allow removing an Amenity at any time.

Note: When adding a new Room, instead of editing one, make sure that the Add Amenities option is not available.



The mockup shows a web browser window titled "Room 101" with the URL "http://myhost/Bookings/". It features a tabbed interface with "Rooms" and "Bookings" tabs. The main heading is "Room 101". Below this, there are four input fields: "Room Number" (101), "Adults Capacity" (1), "Children Capacity" (0), and "Price" (.34). At the bottom of this section are "Save" and "Cancel" buttons. Below the form is a section titled "Amenities" with a link "Add New". Underneath is a table listing amenities for Room 101.

| Name | |
|-----------------|------------------------|
| Television | delete |
| Premium Towels | delete |
| Internet Access | delete |

Figure 18. Mockup of the RoomDetail Screen with the Amenities Table

Extra Challenge: As you probably realized by now, to add several Amenities to a Room you're forced to navigate multiple times between the **RoomAmenities** Screen and the **RoomDetail** Screen. To change this, create an additional **Save & New** Button in the **RoomAmenities** Screen. This way, instead of going back to the **RoomDetail** Screen, simply clear the input to allow you to add a new Amenity to the Room, without closing the Screen.

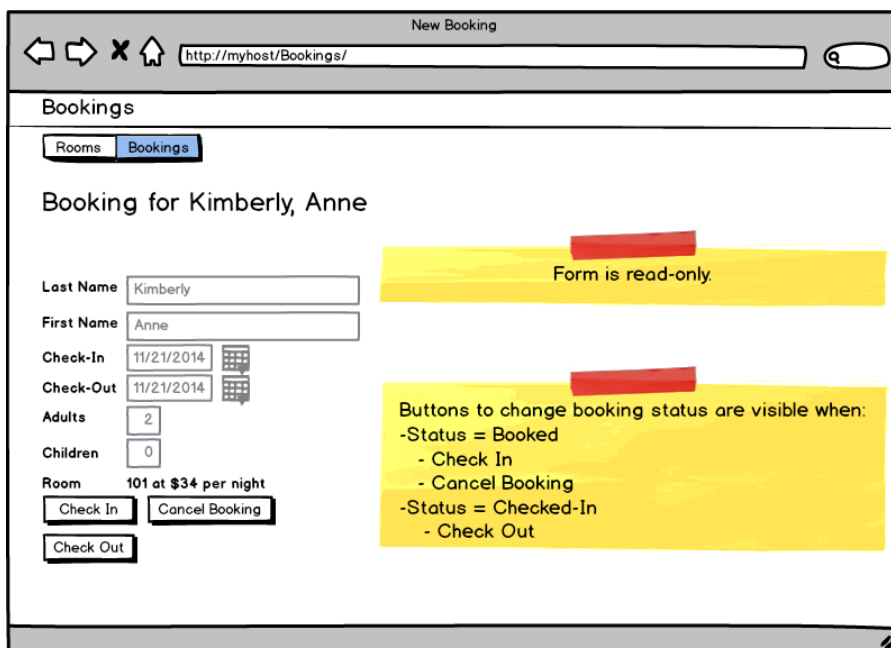
Assignment 6: Encapsulate Bookings Logic

The goal of this Assignment is to structure the application, by encapsulating its logic in a generic Action. That logic can then be used multiple times throughout the application.

1. Show a Booking

Change the functionality of the BookingDetail Screen, when you are editing an existing Booking, like changing its Status for instance.

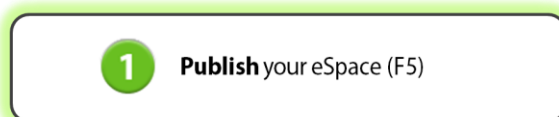
- a) Change the **BookingDetail** Screen to look just like it is shown in the following mockup.



The mockup shows a web browser window titled "New Booking" with the URL "http://myhost/Bookings/". The page has a "Bookings" tab selected. The main heading is "Booking for Kimberly, Anne". Below this, there are input fields for "Last Name" (Kimberly), "First Name" (Anne), "Check-In" (11/21/2014), "Check-Out" (11/21/2014), "Adults" (2), and "Children" (0). The room information is "Room 101 at \$34 per night". There are three buttons: "Check In", "Cancel Booking", and "Check Out". A yellow box with a red highlight indicates "Form is read-only." Another yellow box with a red highlight lists conditions for when buttons to change booking status are visible: "-Status = Booked" (Check In, Cancel Booking) and "-Status = Checked-In" (Check Out).

Figure 19. Mockup of the BookingDetail Screen with the Check In, Check Out and Cancel Bookings functionality

- b) Implement a generic Server Action to change the Status of a Booking.
 c) Implement the logic of the Buttons to change the status of the current Booking accordingly, reusing the Action created in the previous step.
 d) Link the Guest First Name, in the Table Records of the **Bookings** Screen, to edit existing Bookings.



2. Check In, Check Out, and Cancel Bookings

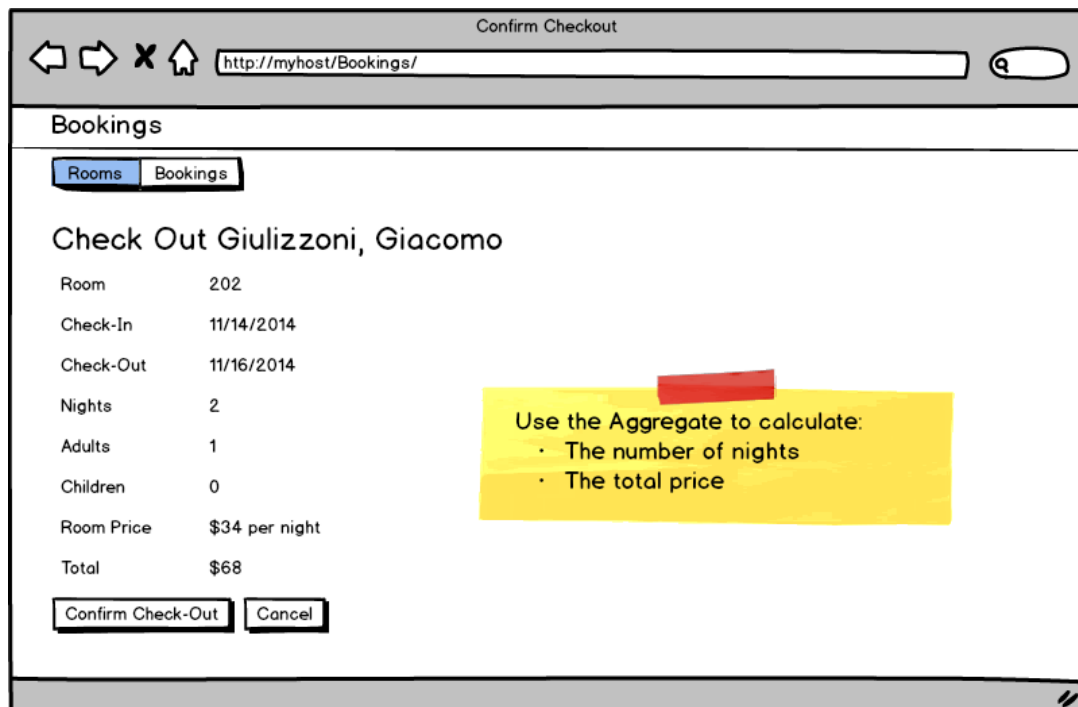
Follow the next steps to test if your application is behaving properly.

- a) Add three Bookings for the guests Jack Daniels, John Doe and Mary Jane.

- b) Check in Jack Daniels and John Doe.
- c) Did it work as expected? Both Bookings should be checked in.
- d) Cancel Mary Jane's Booking.
- e) Can you check in Mary Jane now?
- f) Try to cancel John Doe's Booking. Since John Doe has already checked in, the **Cancel** Button should not be visible, thus you should not be able to cancel the Booking.
- g) Check out Jack Daniels.

3. Add Confirmation Before Checking Out

In the **BookingDetail** Screen, change the Check-Out Action to go to a new **BookingCheckout** Screen, where the number of nights and the total payment for the Booking are determined.



The mockup shows a web browser window titled "Confirm Checkout" with the URL "http://myhost/Bookings/". The page has a "Bookings" header and two tabs: "Rooms" and "Bookings". The main content area is titled "Check Out Giulizzoni, Giacomo" and displays a table of booking details:

| | |
|------------|----------------|
| Room | 202 |
| Check-In | 11/14/2014 |
| Check-Out | 11/16/2014 |
| Nights | 2 |
| Adults | 1 |
| Children | 0 |
| Room Price | \$34 per night |
| Total | \$68 |

Below the table are two buttons: "Confirm Check-Out" and "Cancel". A yellow callout box with a red arrow points to the "Nights" and "Total" rows, containing the text: "Use the Aggregate to calculate: The number of nights, The total price".

Figure 20. Mockup of the Booking_Checkout Screen

The **number of nights** is calculated using the CheckOutDate and the CheckInDate. The **total price** considers the number of nights and the price of the room per night. Figure 20 shows an example where the guest stayed for two nights, in a Room that costs 34\$ per night, thus resulting in the total price for the entire Booking of 68\$.

4. Implement the Homepage

Create the **Homepage** Screen like the mockup in Figure 21. Don't forget to add the Homepage to the Menu of the application.

Make the **Homepage** Screen as the Entry Point of your application.

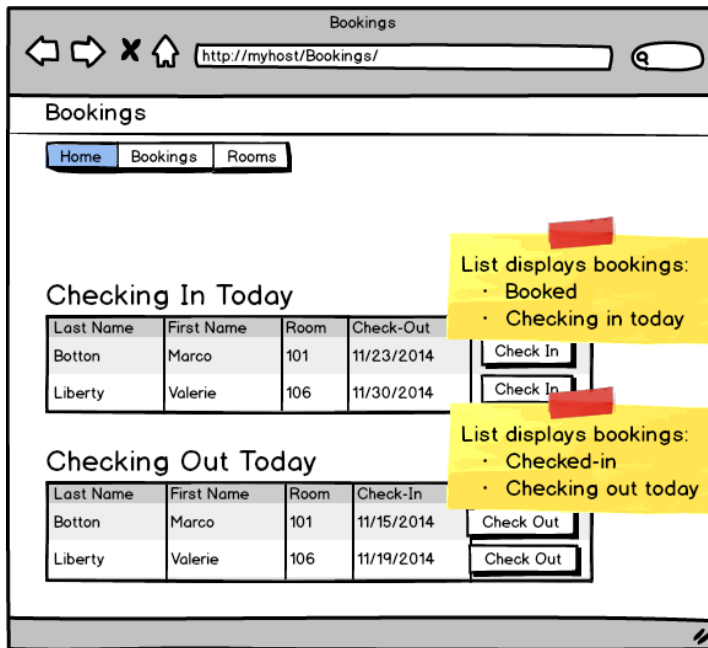


Figure 21. Mockup of the Homepage Screen

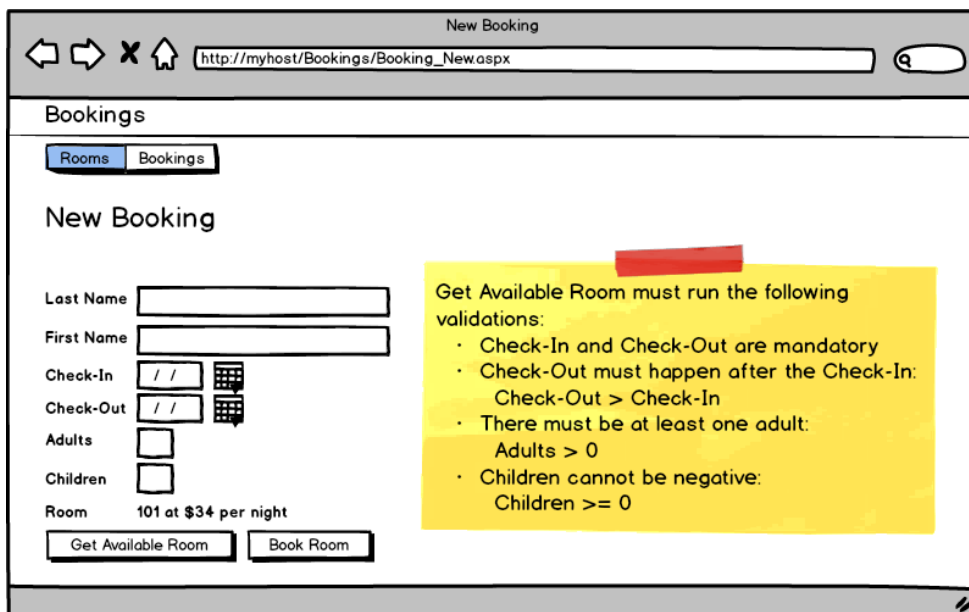
Did you have to implement logic for the **Check In** and **Check Out** Buttons, inside the Table Records, or can you reuse any of the existing logic?

Assignment 7: Bookings Input Validation

The goal of this Assignment is to avoid sending incorrect data to the Database, when filling information for new Bookings. You need to ensure that when checking for an available room, the user provides valid information according with some specific business rules.

1. Validate User Inputs Using Server-Side validations

The following mockup specifies some validations that your application needs to check, to ensure that those scenarios are applied.

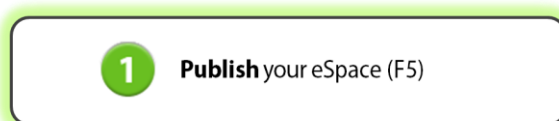


The mockup shows a web browser window titled "New Booking" with the URL "http://myhost/Bookings/Booking_New.aspx". The page has a "Bookings" header with "Rooms" and "Bookings" tabs. The "New Booking" form includes fields for "Last Name", "First Name", "Check-In", "Check-Out", "Adults", and "Children". The "Room" field is pre-filled with "101 at \$34 per night". There are "Get Available Room" and "Book Room" buttons. A yellow box on the right lists the validations for the "Get Available Room" button:

- Check-In and Check-Out are mandatory
- Check-Out must happen after the Check-In:
Check-Out > Check-In
- There must be at least one adult:
Adults > 0
- Children cannot be negative:
Children >= 0

Figure 22. Mockup of the validations in the BookingDetail Screen

Note: It is possible to create Bookings in the past.



2. Test the Validations

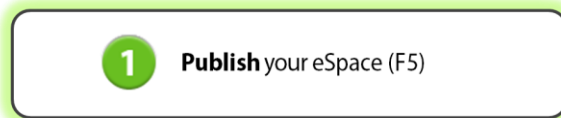
Follow the next steps to test the validations that you defined in the previous step:

- Try to Book a Room for a Guest:
 - Check-in:** today
 - Check-out:** yesterday
 - Adults:** 1
- Are the validations working? Great!
- Try to book a room:

- **Adults:** 0
 - **Children:** 3
- d) Are the validations working? Great!

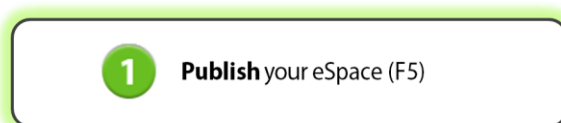
Now, try the Client & Server Validations

- a) Change your validations to Client & Server.



- b) Try to book a room:
- Don't fill-in the Check In field
 - Don't fill-in the Check Out field
- c) Are client-side validations working? Great!
- d) Change the Check In date to "Jack Daniels".
- e) Are client-side validations working? Great!

3. Create Bookings for Testing



Follow the next steps to create some Bookings in your Database.

- a) Book a room for John Doe:
- **Check-in:** today
 - **Check-out:** a week from today
 - **Adults:** 2
- b) Was the booking successful? Great!
- c) Were you directed to the **Bookings** Screen after creating the booking? Great!
- d) Book a room for Mary Jane:
- **Check-in:** tomorrow
 - **Check-out:** the day after tomorrow
 - **Adults:** 2

What has happened? Mary Jane had a new Room assigned to its Booking, different from the one assigned to John Doe.

- e) Book a room for Jack Daniels:
- **Check-in:** today
 - **Check-out:** tomorrow

- **Adults:** 1
- **Children:** 2

4. Clean invalid Bookings

At this point of the exercise, and since we just added the input validations, you might come to the conclusion that some data that you previously created is incorrect or doesn't fulfill all the necessary conditions. To overcome this problem, a new functionality to delete **Booking** should be added:

- a) In the **BookingDetail** Screen, add a new **Delete** Link to the **Actions** Placeholder;
- b) Add the following **Confirmation Message** to this Link, to make sure the User is certain of what will happen: "Are you sure you want to permanently delete this Booking?"
- c) In the Screen Action associated with this Link, simply delete the current Booking from the Database.

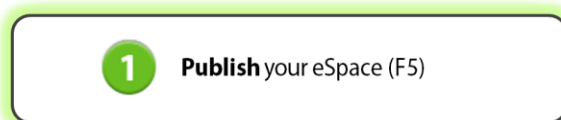
Assignment 8: Improve Bookings User Experience

The goal of this Assignment is to use Ajax patterns, to avoid refreshing the whole Screen when executing application logic. This improves the user experience, since the user is able to continue navigating in the Screen, while the server processes the request.

1. Refresh Screen Elements with Ajax

Follow the next steps to add Ajax behavior to your application:

- a) In the **BookingDetail** Screen, refresh the room information with Ajax, when the **Get Available Room Button** is pressed. Use the 'Highlight' **Animation** effect.
- b) In the **Homepage** Screen, review the logic of the **Check In** Button to only remove the respective line from the Table Records. Use the 'Highlight' **Animation** effect.
- c) In the **RoomDetail** Screen, change the logic of the **Delete** Link to use Ajax.



2. Test your Application

- a) Find a Room for Lewis Nash:
 - **Check-in:** today
 - **Check-out:** the day after tomorrow
 - **Adults:** 2
 - **Children:** 1
- b) Only the room information was refreshed? Good!
- c) Did the **Book Room** Button appear? Why not? Go back and fix it.
- d) Go to the **Homepage** and check in Lewis Nash.
- e) Was he removed from the list with Ajax? Nice!
- f) Find a Room for Amelia Jacobson:
 - **Check-in:** tomorrow
 - **Check-out:** today
 - **Adults:** 1
 - **Children:** -1
- g) Did the validations appear on the page? What happened for this to change? Go back and fix it.

Assignment 9: Reuse Bookings Page Elements

The goal of this Assignment is to improve the Bookings user interface (UI), by refactoring a part of a Screen, and customizing its look & feel.

1. Create Reusable UI Elements

Create a **Web Block** to display the Guest's names with the same format all over the application: last name first, followed by a comma, and then the first name, e.g: 'Doe, John'.

Enclose the elements in a **Container** and highlight the last name by displaying it in bold. The resulting names should be displayed as follows:



| |
|------------------|
| Montgomery, Jane |
| Millard, Anthony |
| Stuart, Pamela |
| Baker, John |

Figure 23. Example of the usage of the Web Block for the Guest Name

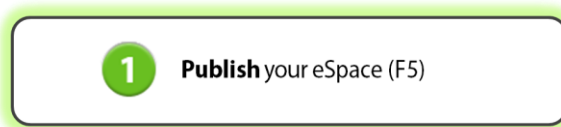
Replace the Guest Name Expressions, for the Last Name and First Name, with this Web Block on the **Homepage** and **Bookings** Screens. Instead of having two columns in the Table Records, you should just have one with the Web Block.

Assignment 10: Secure Bookings

The goal of this Assignment is to implement security mechanisms in the application, to restrict access to the application itself and to some of its functionality.

1. Restrict Access to the Application

Untick the **Anonymous** and **Registered** Roles from all the Web Screens of the application.



What happens when you run the application?

2. Create a User for the Application

Go to the Users application on your server i.e. http://<your_server>/Users/:

- Create a new user.
- You can login with the user, but still have no access. Why? [It's all about Roles](#).
- Grant the user the **Bookings_<yourinitials>User** Role.
- Can you now access the application with the user? Great!

3. Restrict Access to Create New Bookings

Restrict the creation of new Bookings to some users only.

- Create the **BookingsResponsible** Role.
- Leave only this Role checked in the **BookingDetail** Screen.
- In the **Bookings** Screen, make the **New Bookings** Link visible, only if the current user has the new **BookingsResponsible** Role.
- Create a new user and grant it the **BookingsResponsible** and the **Bookings_<yourinitials>User** Role.
- Can the first user that you created access the **BookingDetail** Screen? It shouldn't since it does not have the Role associated

4. Restrict Access to Delete Bookings

Use the previously created Role to control who can delete Bookings.

- Make the **BookingDetail** Screen accessible again to a user with the **Bookings_<yourinitials>User** Role.
- Since we granted access to the users with the **Bookings_<yourinitials>User** Role, make sure that only users with the **BookingsResponsible** Role can access the **Delete** Action, in the **BookingDetail** Screen, and **Cancel** Bookings.



Publish your eSpace (F5)

What is the difference between logging in with the first user you have created and with the second one, after these changes?

5. Use Session Variables to keep information related with the User

Session Variables can be used to store information that is valid while a user is logged in the Application.

In this example, go to the **Bookings** Screen, and convert the Variables you associated with the search input fields, the Input Widget for the Guest Name and the Combo Box for the Booking Status, to Session Variables.



Publish your eSpace (F5)

Assignment 11: Improve Bookings UI

The goal of this Assignment is to improve the user interface (UI), by customizing the look & feel of the Bookings application. By the end of this Assignment, your application should look like this:

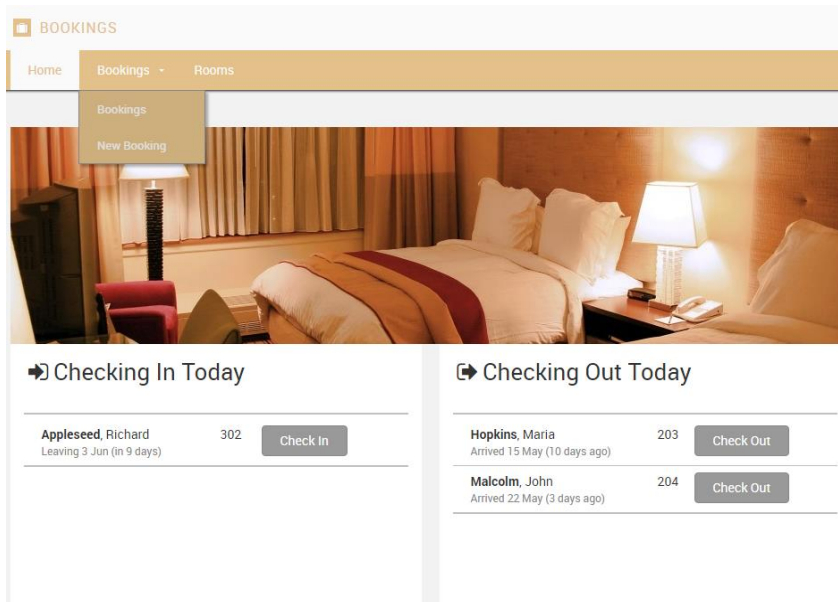


Figure 24. Example of the usage of the Web Block for the Guest Name

1. Customize the Header Icon and Preview on several devices

Start by replacing the OutSystems logo by the Bookings logo in the **Header** of your Application. Use [the preview buttons](#) to see how your application looks in a smartphone, tablet, and desktop. While in tablet preview mode, open the application in the browser, to get a result like this:

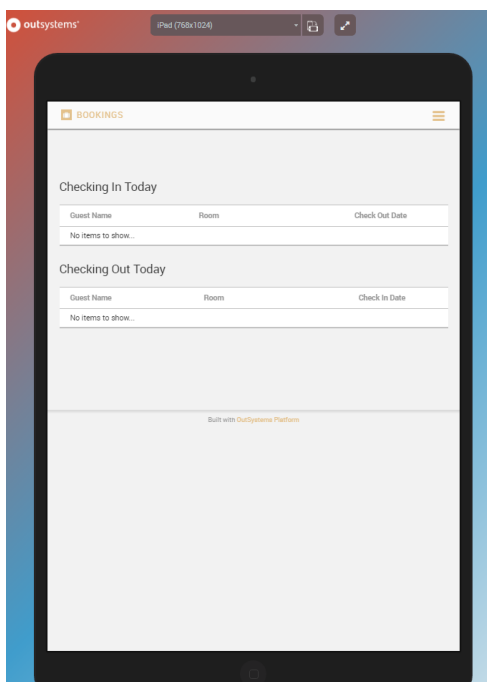


Figure 25. Tablet Mode Preview

2. Show the Check-In and Check-Out lists side by side

In this section, you will improve the **Homepage** Screen to display the Check-In and Check-Out Table Records side by side.

- a) Make sure your **Homepage** has a hierarchy like this:

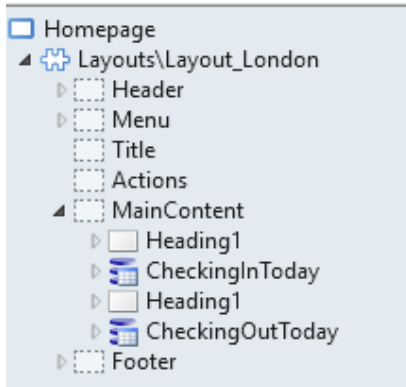


Figure 26. Widget Tree for the Homepage Screen

- b) Enclose each combination of Heading + Table Records in a Container, in this case the first Heading1 with the **CheckingInToday** Table and the second Heading1 with the **CheckingOutToday**.

- Resize the new Containers to '6 col'.
- Are the **CheckingInToday** and **CheckingOutToday** Table Records displayed side by side? Great!
- Limit the height of both lists, by enclosing each Table Records in a **Container**, and apply the following style:

```

.GuestList {
    height: 270px;
    overflow: auto;
}
  
```

This style class need to be added to the **Homepage** Screen Style Sheet and then applied to the mentioned **Container**.

3. Work the Homepage Headings

In this section, you will improve the **Homepage** headings, Checking In Today and Checking Out Today, to use Icons, making them easier to read.

For the icons, use the Icon Web Block from Rich Widgets. You can find it in the Elements Tree in Service Studio, as displayed in Figure 27, and drag it to the Screen. Set their **Name** property to 'sign_in' (for check-in) and 'sign_out' (for check-out).

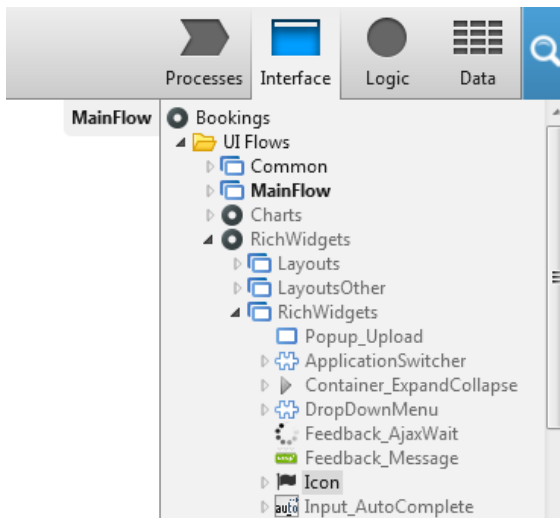


Figure 27. Icon Widget in the elements tree

The Homepage Screen Title is not being very useful, but is still occupying some space. We can hide it, by overriding a style defined in the base Theme. To accomplish that, add the following style to the **Homepage** Style Sheet:

```
.Title_Section {
    display: none;
}
```

4. Create Contrast Using Cards

Make your **Checking In Today** and **Checking Out Today** columns stand out. To accomplish that, apply the **CardWhite** Style to the Containers added in step 2 b), that enclose the Titles and the Tables.

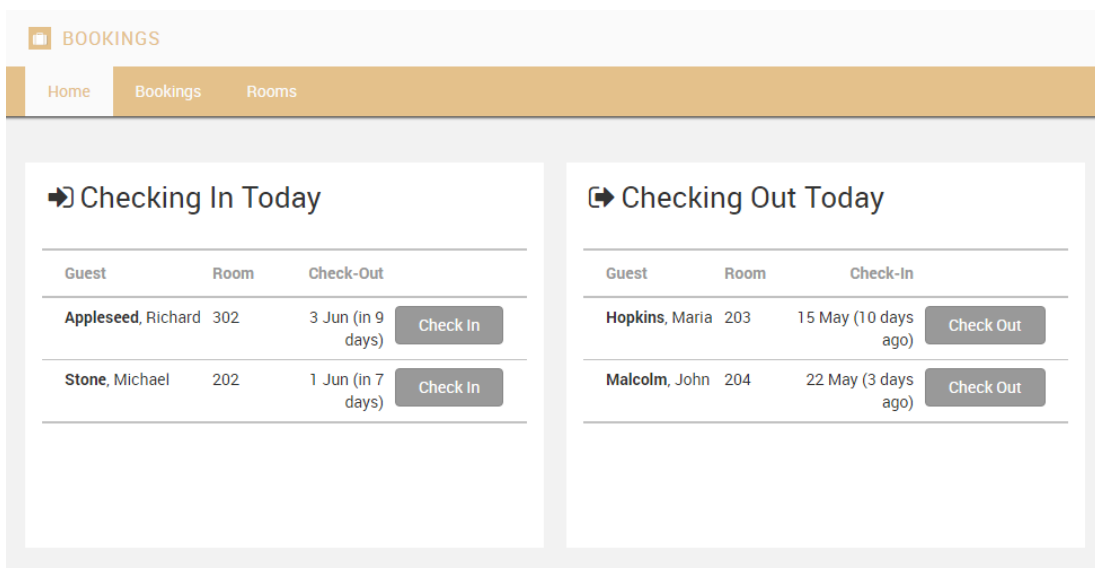


Figure 28. Homepage after applying the CardWhite style

Is the application looking great? Let's improve it even more.

5. Work the Table Records Content to Improve Readability

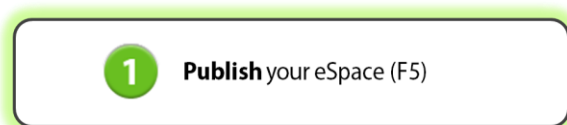
Do the following steps for both Check In and Check Out Tables:

- Hide the Table Records header.
- Drag the Date below the Guest Name:
 - Align it left, apply the 'Text_Note' Style, and remove the margin top;
 - Delete the Date column that was left empty.
- Add 'Leaving' or 'Arrived' before the Date, on the Checking In Today and Checking Out Today respectively.

By the end of this section your lists should look like this:

| ➔ Checking In Today | | | ➔ Checking Out Today | | |
|---------------------------|-----|----------|------------------------------|-----|-----------|
| Appleseed, Richard | 302 | Check In | Hopkins, Maria | 203 | Check Out |
| Leaving 3 Jun (in 9 days) | | | Arrived 15 May (10 days ago) | | |
| Stone, Michael | 202 | Check In | Malcolm, John | 204 | Check Out |
| Leaving 1 Jun (in 7 days) | | | Arrived 22 May (3 days ago) | | |

Figure 29. Final look and feel of the guest lists in the Homepage Screen



6. Make the Application Come Alive Using an Image

Add one **Container** above the Check In and Check Out columns and add the 'room_banner' image inside it. You can find the image in the **Resources** folder. Figure 30 shows an example of the Screen, with the image.

Note: Group the Checking In and Checking Out white cards in a **Container**. This allows you to control the space from the tables to the image.

7. Promote the Most Common Operations

Improve the application by adding Styles to the Screens and improving the Menu.

- In **Bookings**, validate that the **New Booking** Link is inside the Actions placeholder and apply the 'ActionAdd' Style to the Link.
- In **BookingDetail** do the same thing for the **Delete** Link, but now apply the 'ActionDelete' Style;
- Creating a new Booking is one of the most common operations, as such, add it under the Bookings menu option. i.e., as a submenu option.

Remember: use the Web Block options to see all placeholders. If you need assistance,

[check the reference help.](#)



If you want to know how you can further customize your Application's Theme, check London's [Theme Generator](#).

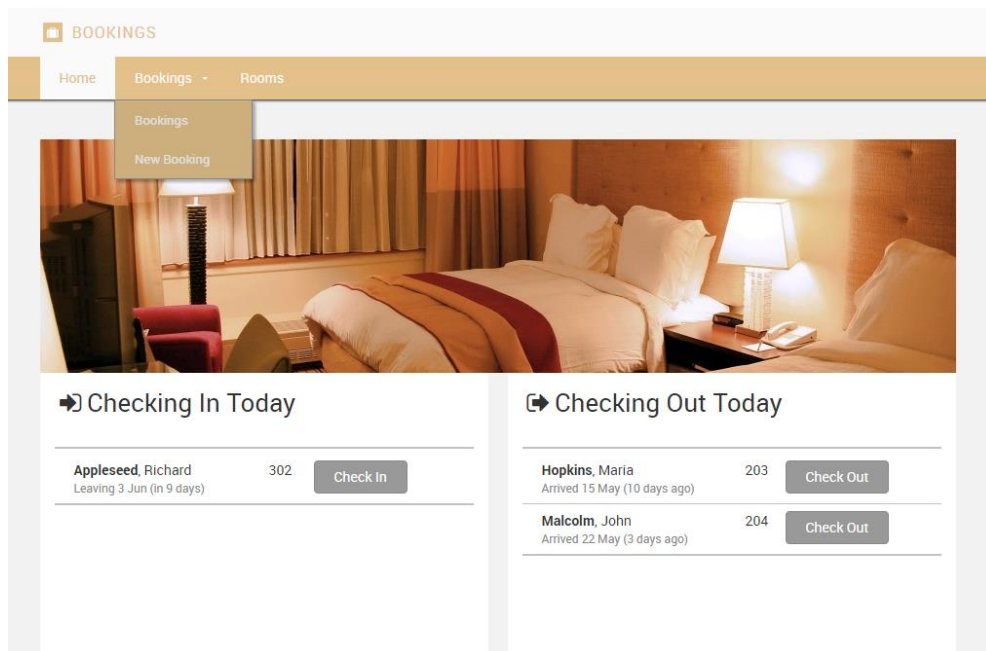
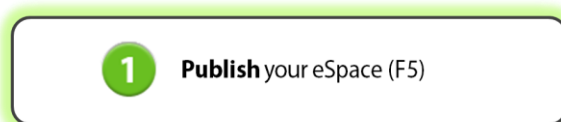


Figure 30. Final look and feel of the guest lists in the Homepage Screen

Assignment 12: RichWidgets

1. Use Popups to Perform Tasks in Context

- Change the **BookingCheckout** Screen to use the 'Layout_Popup' instead of the 'Layout_London'.
- Use the **Popup_Editor** Web Block, from Rich Widgets, to open the **BookingCheckout** Screen in a Popup window, both on the **Homepage** and on **BookingDetail**. Make sure your Buttons are set to the **Navigate Method**.
- When the Check-Out is confirmed, or you Cancel the operation, the Popup should close.
- Implement the **OnNotify** behavior accordingly.



You should have something like this:

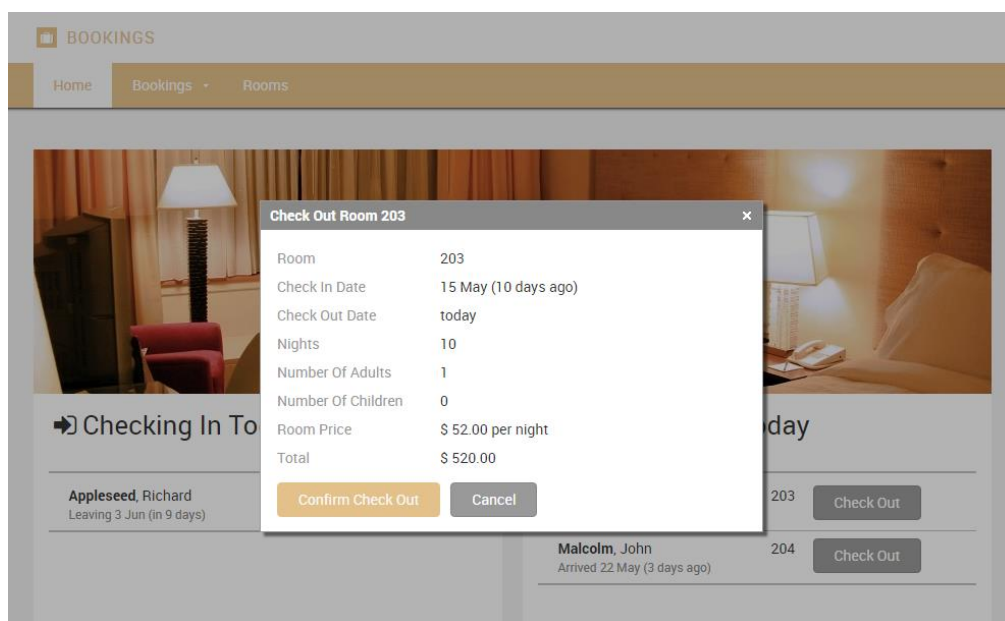
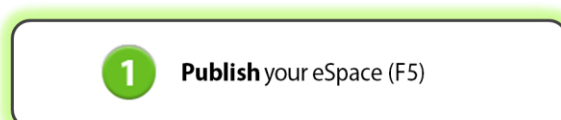


Figure 31. Booking Checkout Popup Screen.

2. Add More Functionality to the List of Bookings

- Add the pagination to the list of Bookings.
- Allow sorting that list by each column.



You should have something like this:

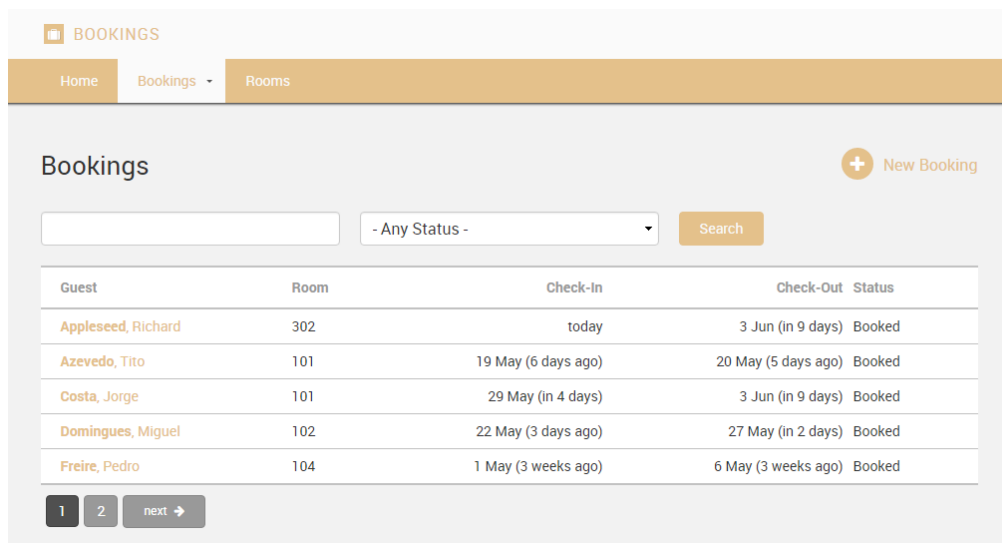
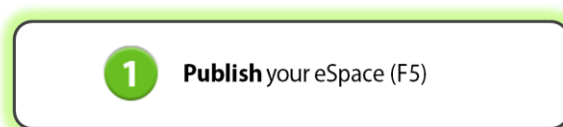


Figure 32. Bookings Screen with List_Navigation.

3. Provide Feedback to the User

Use the **Feedback_Message** Web Block, from RichWidgets, to give feedback to the user:

- When creating a new Booking;
- When Checking In a guest;
- When Checking Out a guest;
- When canceling a booking.



You should have something like this:

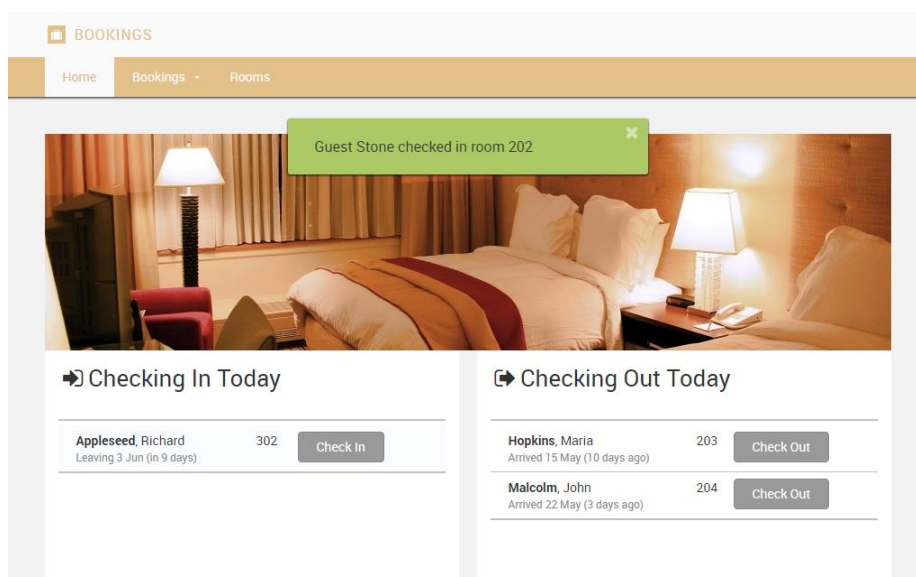


Figure 33. Homepage with Feedback_Message.

4. Display the Occupancy for the Next 7 Days

In the **Homepage** Screen, display the hotel occupancy. This will allow the hotel clerk to plan the next days, and better allocate the available resources. The Chart should display **the hotel occupancy**, in percentage (Y-axis), **for the next 7 days** (X-axis).

By the end of this exercise, your application should look like this:

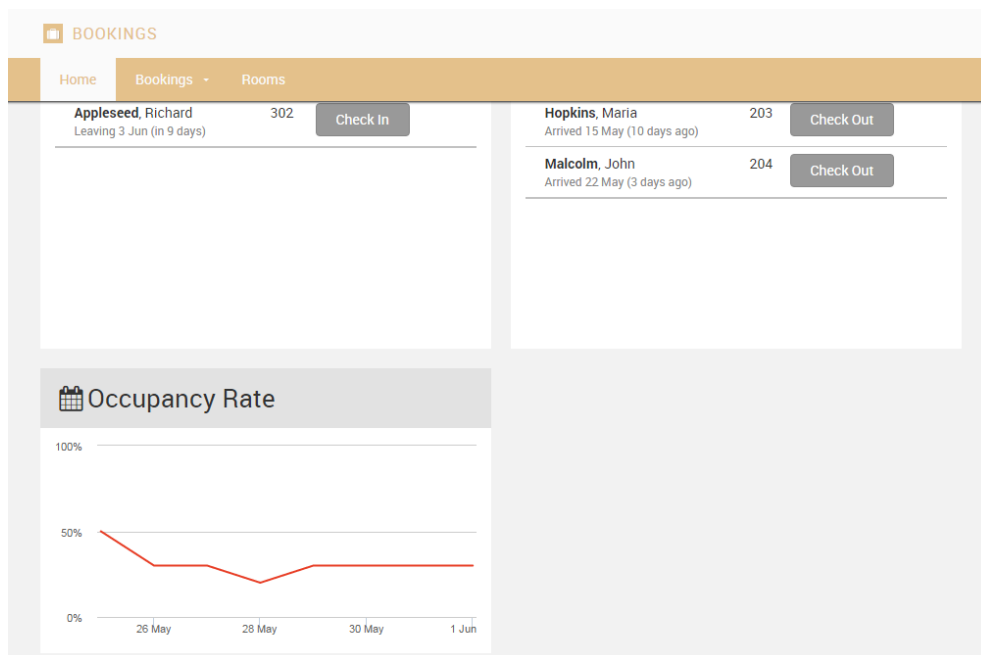


Figure 34. Homepage with Occupancy Rate Chart.

- Use the Grid, Containers, Styles and the calendar icon to create the layout for the **Occupancy Rate** title.
- Use a **Line Chart**, and set its height to '200'.
- In the Preparation, create the logic to build the data to be plotted on the chart:
 - Get all Rooms in the hotel, to calculate the percentage of occupancy.
 - Use a **Local Variable** to hold the Date to which the occupancy is calculated. The Variable should start on today's date.
 - If the Date in the Local Variable is within the next 7 days:
 - Get all Rooms Booked or Checked In on the date.
 - Build a **DataPoint** with the percentage of occupancy for the date.
 - Append the **DataPoint** to a List of Data Points.
 - Add one day to the date on the Local Variable.
 - Define the flow to return to the If.
- Pass the List of Data Points to the Chart.
- Change the look of your Chart so that the Y-axis always displays values between 0 and 100 and has its labels suffixed with '%'. You can use the **YAxisFormat** property of the chart. To

learn how to specify the **YAxisFormat**, checkout the **YAxisFormat_Init** Action online documentation [here](#).

1 Publish your eSpace (F5)

5. Test Your Chart

Make sure the logic is working and the chart is displaying correct information. Don't forget to use the Debugger to see what is going on in the server side.

You should have something like this:

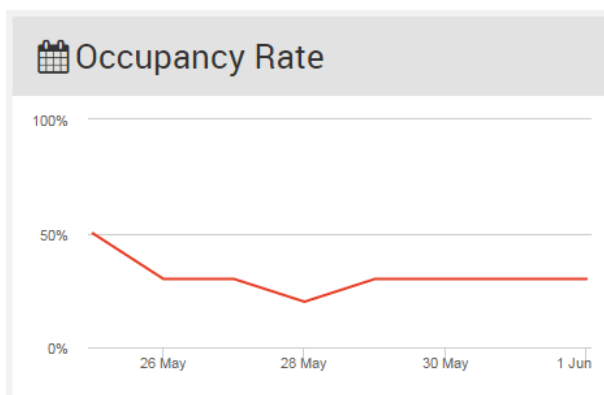


Figure 35. Occupancy Rate Chart.

Assignment: Bookings Extra

1. I'd like to Order Some Room Service Please!

In the **BookingsCore** Module, create the data model to support registering room services for a Booking:

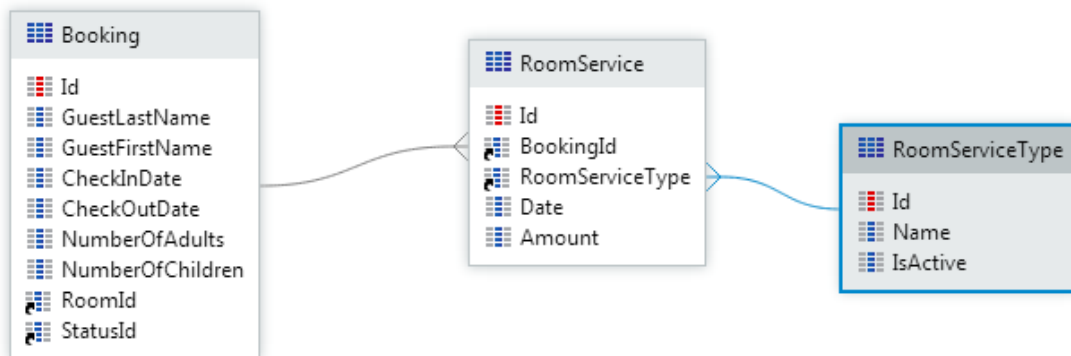


Figure 36. Fragment of the data model with RoomService and RoomServiceType Entities

Note: All attributes in **RoomService** and **RoomServiceType** Entities are mandatory.

- Create an Action to bootstrap data for the **RoomServiceType** Entity, from the Excel available in the **Resources** folder.
- On the **Homepage** Screen, allow the hotel clerks to register Room Services, right next the Occupancy Rate Chart.

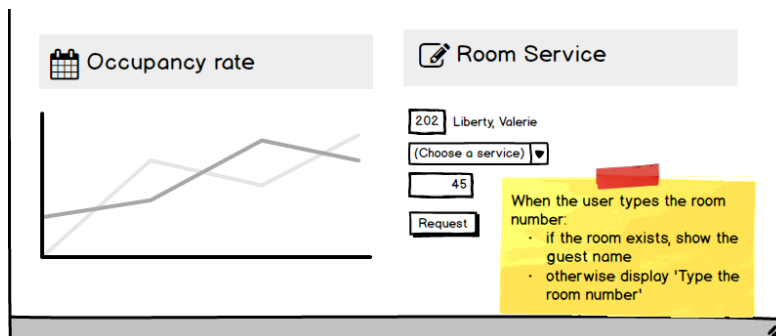


Figure 37. Mockup of a section of the Homepage Screen, with Room Service

- To present the Guest Name when the clerk inputs the Room Number, use the OnChange Event of the Input Widget.

2. Charge for the Room Service on the Check-Out

Change the Check-Out confirmation page to include both the Room Price and the Room Services required during the stay. The following mockup displays how the Popup should work.

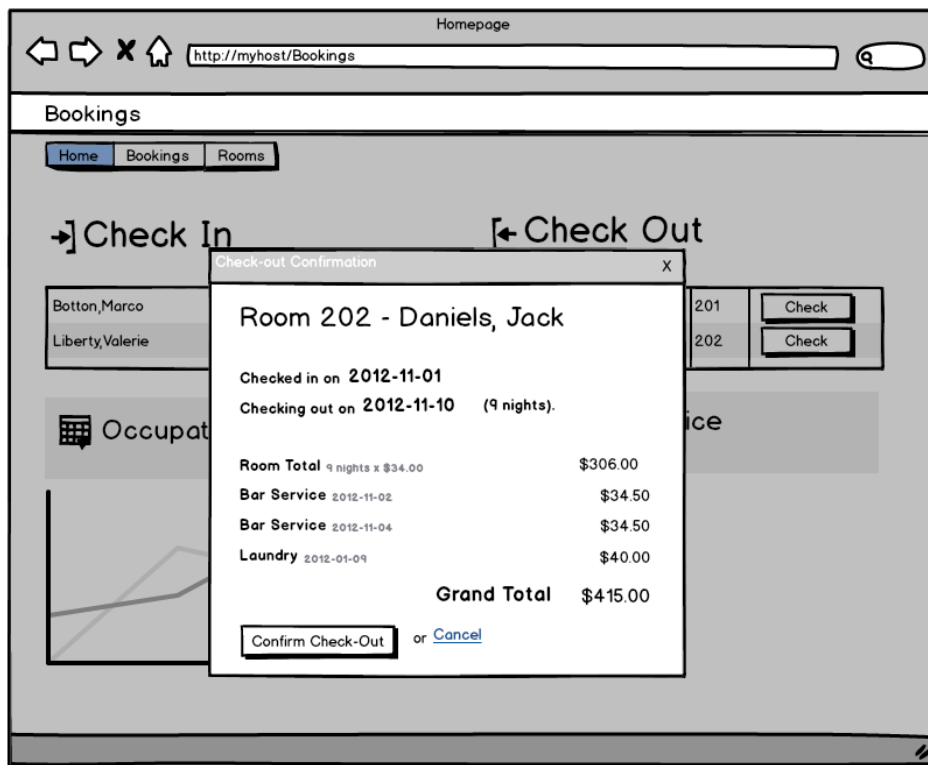


Figure 38. Checkout including Room Service

List of Figures

| | |
|---|----|
| Figure 1. Create a New Application | 3 |
| Figure 2. Select the Web App option | 3 |
| Figure 3. Select the Web App template | 4 |
| Figure 4. Give a name and logo to the application..... | 4 |
| Figure 5. Creating a BookingsCore Blank module | 5 |
| Figure 6. Room, Booking and Status Entities | 5 |
| Figure 7. Unset the BookingsCore as the Home Module | 7 |
| Figure 8. Create a new Module | 7 |
| Figure 9. Choosing the Module Type to be Responsive..... | 7 |
| Figure 10. Mockup of the Rooms Screen..... | 8 |
| Figure 11. Mockup of the RoomDetail Screen | 8 |
| Figure 12. Bookings Data Model, with the relations between Entities | 10 |
| Figure 13. Mockup of the Bookings Screen | 11 |
| Figure 14. Mockup of the BookingDetail Screen | 12 |
| Figure 15. BookingDetail Screen, after adding the search functionality | 14 |
| Figure 16. Part of the Data Model with Amenity and RoomAmenity Entities | 15 |
| Figure 17. Mockup of the RoomAmenities Screen..... | 16 |
| Figure 18. Mockup of the RoomDetail Screen with the Amenities Table..... | 16 |
| Figure 19. Mockup of the BookingDetail Screen with the Check In, Check Out and Cancel Bookings functionality | 17 |
| Figure 20. Mockup of the Booking_Checkout Screen | 18 |
| Figure 21. Mockup of the Homepage Screen | 19 |
| Figure 22. Mockup of the validations in the BookingDetail Screen..... | 20 |
| Figure 23. Example of the usage of the Web Block for the Guest Name | 24 |
| Figure 24. Example of the usage of the Web Block for the Guest Name | 27 |
| Figure 25. Tablet Mode Preview | 27 |
| Figure 26. Widget Tree for the Homepage Screen | 28 |
| Figure 27. Icon Widget in the elements tree | 29 |
| Figure 28. Homepage after applying the CardWhite style..... | 29 |
| Figure 29. Final look and feel of the guest lists in the Homepage Screen..... | 30 |
| Figure 30. Final look and feel of the guest lists in the Homepage Screen..... | 31 |
| Figure 31. Booking Checkout Popup Screen. | 32 |
| Figure 32. Bookings Screen with List_Navigation. | 33 |
| Figure 33. Homepage with Feedback_Message..... | 33 |
| Figure 34. Homepage with Occupancy Rate Chart..... | 34 |
| Figure 35. Occupancy Rate Chart. | 35 |
| Figure 36. Fragment of the data model with RoomService and RoomServiceType Entities | 36 |
| Figure 37. Mockup of a section of the Homepage Screen, with Room Service . | 36 |
| Figure 38. Checkout including Room Service | 37 |