Wydział Matematyki i Nauk Informacyjnych

Politechnika Warszawska

Title	Project #1: Hostel Management
Subject	Enterprise Applications in .NET Framework
Year	2016-2017
Version	1.3
Authors	Mario Juez Gil juezm@student.mini.pw.edu.pl Maryan Plakhtiy plakhtiym@student.mini.pw.edu.pl

Revision history

Date	Author	Changes	Revision
19/10/2016	Mario Juez	First version of this document.	1.0
20/10/2016	Mario Juez	- Removed general project introduction covering information about three iterations. - Improved business description with detailed information. - Added and modified user histories to fit use cases definition. - Detailed schedule.	1.1
17/11/2016	Maryan Plakhtiy	- Added Architecture of a project. - Added System requirements - Manual for the application - References	1.2
18/11/2016	Mario Juez	- Added libraries used. - Modified functional requirements. - Modified use case diagram.	1.3 (current)

Table of contents

Table of figures

1 Project #1: Hostel Management

1.1 Business Description

The first project product will be a desktop application for generic hostel management. We understand hostel management as guest check-ins and bedrooms management by a hostel manager.

Guest check-in will be a form that the manager will have to fill up with guest personal data like ID number, name, surname, birth date, and sex. That form will also contain check-in and check-out dates, and room number fields. Depending of bedroom size, the form will allow to the manager register one or more guests into the same check-in.

For bedroom management, the application will have a view with bedroom listing including filtering options by size, price, bathroom type, bed type and availability. Through that view, manager will be able to open the bedroom editor for creating or modifying a bedroom, remove bedrooms, or open the check-in registration view for one specific bedroom.

In order to track operations and avoid usage of non-authenticated users, hostel manager login will be obligatory to use the application.

1.1.1 Technologies

This project is going to use the following technologies:

1. **Windows Presentation Foundation (WPF)**: User Interface.
2. **NoSQL Database MongoDB**: Database.

1.2 Functional Requirements

This part of the document, defines project functional requirements by usage of use case diagrams and user stories.

1.2.1 Use Case Diagram

The following figure shows project use case diagram.

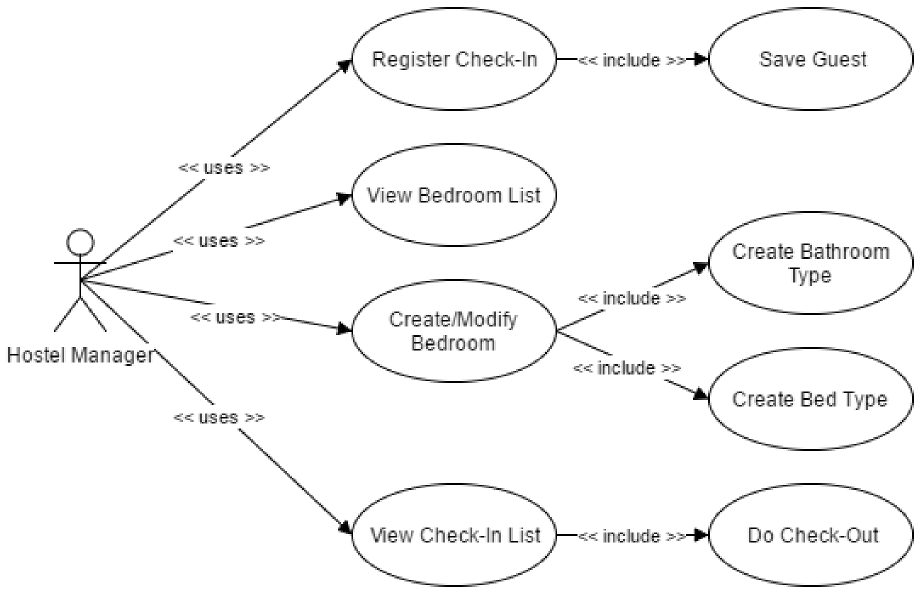


Figure 1: Use Case Diagram.

Actor	Name	Description
Hostel Manager	Register Check-In	The user will enter check-in and check-out date, select the bedroom associated to, and finally the guest information.
	Save Guest	The user will fill the Guest information (ID, Name, Surname, Birth Date and Sex) and then it will be stored when saving the check-in.
	View Bedroom List	The user is able to view the hostel bedroom list, this view provides filtering options to filter the rooms by its features like price, bathroom type… The list will show information about bedroom availability. Inside this view, user will find buttons to create a new room, and modifying or deleting existing ones.
	Create/Modify Bedroom	If bedroom does not exist, user will enter bedroom number, bed number and type, bathroom type, and regular price per night into a blank form. That form will contain bedroom data if the user is modifying it. Bathroom and bed type fields has to have autocomplete feature for those types which already exists.
	Create Bathroom Type	If the introduced bathroom type does not exist in the database, the system has to create it.
	Create Bed Type	If the introduced bed type does not exist in the database, the system has to create it.
	View Check-In List	The user is able to view the hostel current check-in list (without check-out). In this view, the user can remove a check-in (check-out).
	Do Check-Out	The user is able to remove a check-in (check-out operation) It will change the check-in state from active to inactive.

Table 1: Functional requirements.

1.2.2 User Stories

1. Check-In Registration:
   As hostel manager, I need an agile method to register guest check-ins, that allows me to enter guest required information as quick as possible.
2. Guest Storage:
   As hostel manager, I need the system to store all hostel guest data in order to comply with the law.
3. Bedroom Listing:
   As hostel manager, I need an easy method to list all bedrooms of the hostel and see its features and availability, in order to offer my clients the bedroom that best fits their needs.
4. Bedroom creation and modification:
   As hostel owner, I have a hostel chain with different hostels placed in different cities. I need flexible solution which allows my hostel managers to configure each specific hostel bedrooms. Besides, bedroom features like price are subject to change along the time, and managers should have the possibility to modify it.
5. Bathroom types:
   As hostel owner, bedrooms in my hostels has many bathroom configurations, like private bathroom + private shower, shared bathroom, private bathroom + shared shower etc.

6. Bed types:
   As hostel owner, bedrooms in my hostels has many beds configurations, like single bed, double bed, two single beds, bunk bed, etc.
7. Check-In Listing:
   As hostel manager, I need a method to listing current check-ins without check-out.
8. Check-out Operation:
   As hostel manager, when a guest leaves the hostel, I need to remove the check-in from the current check-in list, in other words, I need to do a check-out.

1.3 Non-Functional Requirements

Non-functional requirements are the following:

1. Interface:
   a. Desktop application must work properly on screens with a minimum resolution of 1366x768.
2. Portability:
   a. Desktop application must work properly on Microsoft Windows 10 and above.
3. Security:
   a. User must authenticate into the application with valid credentials.

1.4 Schedule

Date Range	Task
14/10/2016 – 21/10/2016	Initial documentation definition.
21/10/2016 – 27/10/2016	Technologies study and architecture design.
27/10/2016 – 29/10/2016	Data model and business logic development.
29/10/2016 – 02/10/2016	Presentation layer.
02/10/2016 – 02/10/2016	Testing.
02/10/2016 – 04/10/2016	Presentation.
04/10/2016 – 11/11/2016	Final documentation and Delivery.

Table 2: Project schedule.

1.5 Project architecture

1.5.1 Domain layer architecture

Our domain layer is composed by three different layers:

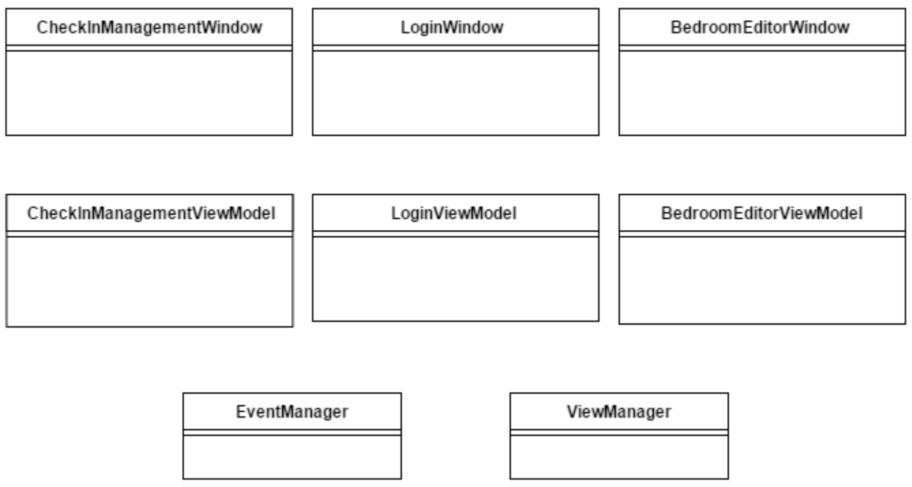
1. Model Layer: Provides POCOs classes. In our solution, we have 6 different models:
   a. CheckIn
   b. Bedroom
   c. BathroomType
   d. BedType
   e. Guest
   f. User

   Each model class corresponds to a MongoDB collection.
2. Data Layer: In this layer CRUD operations with the database (MongoDB) are performed. Is responsible of the data mapping. 6 repositories and one persistence factory are defined here:
   a. CheckInRepository
   b. BedroomRepository
   c. BathroomTypeRepository
   d. BedTypeRepository

   e. GuestRepository
   f. UserRepository
   g. PersistenceFactory
3. Service Layer: Provides interfaces to the UI layer that allows it to perform the operations needed. Our solution has 5 services:
   a. AmenityService: Groups operations with bathroom types and bed types (amenities).
   b. AuthenticationService: Operations to log into the application.
   c. BedroomService: Operations with bedrooms.
   d. CheckInService: Operations with checkins.
   e. GuestService: Operations with guests.

1.5.2 User Interface layer architecture

We have used the MVVM pattern in order to provide a clean separation of concerns between the user interface controls and their logic. There are three essential components in the MVVM pattern: the model, the view, and the view model. Each serves a distinct and separate role. The view is responsible for defining the structure, layout, and appearance of what the user sees on the screen. The model in MVVM is an implementation of the application's domain model that includes a data model along with business and validation logic. The view model acts as an intermediary between the view and the model, and is responsible for handling the view logic and state. Typically, the view model interacts with the model by invoking methods in the model classes. The view model then provides data from the model in a form that the view can easily use.



There are three Views in our applications: LoginView, CheckInManagementView and BedroomEditor. And they have three ViewModels with respective names. Also, we have implemented two helper classes in order to make our application easy to extend and to change. These are EventManager and ViewManager. The former is responsible for all events which take place in Views and it invokes methods which are subscribed on certain event. The latter is responsible for running correct Views and for their refreshing.

1.5.3 Libraries used

As additional libraries we used MongoDB C# Driver which includes BSON Library that provides a serialization infrastructure. That driver has Apache License 2.0.

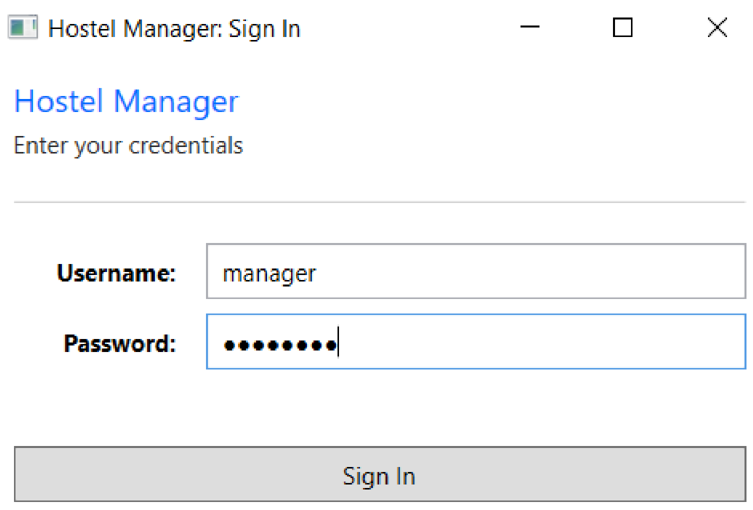
1.6 System requirements

- 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor
- 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit)
- 5 MB available hard disk space (32-bit or 64-bit)
- DirectX 9 graphics device with WDDM 1.0 or higher driver.

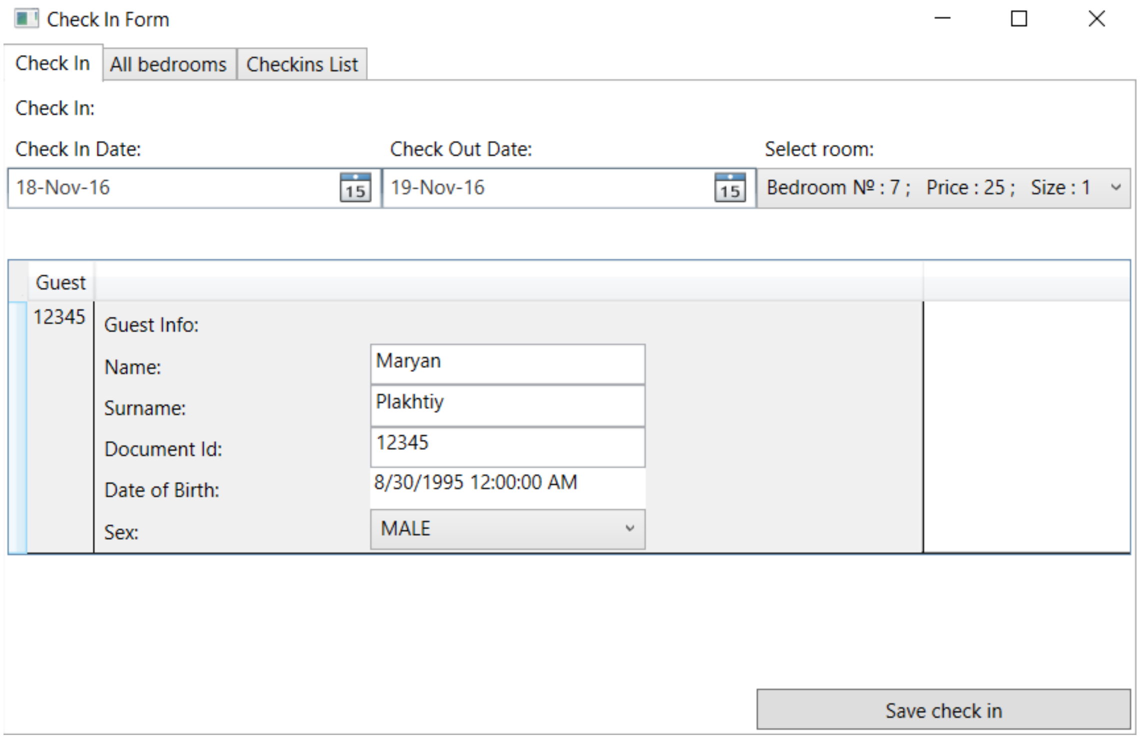
1.7 Manual for the application

The source code is here: https://github.com/mjg0015/.NET-Hostel-Manager

1. Download ISO File from https://github.com/mplakhtiy/.NET-Hostel-Manager/blob/master/HostelManagement.iso.
2. Run the .iso file.
3. Extract all the files to chosen folder.
4. Run the DesctopClient.exe file.
5. Sign in to the application. User name: manager. Password: password.



6. Select room and dates for the check in. Afterwards fill in the guest information fields. All fields are obligatory to fill in.



Check In:

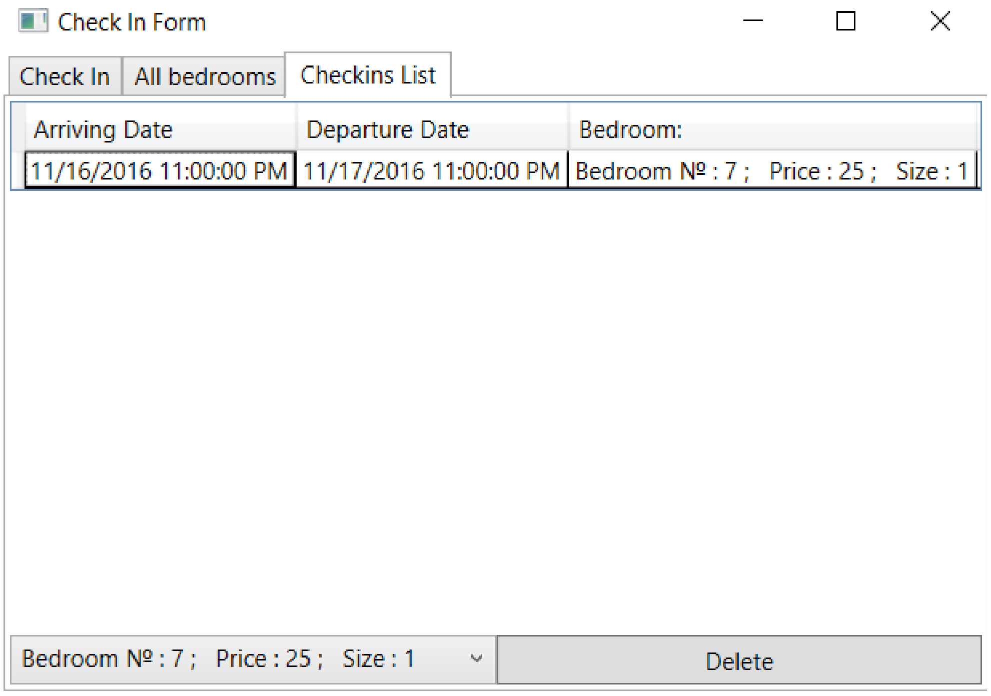
Check In Date:	Check Out Date:	Select room:
18-Nov-16	19-Nov-16	Bedroom № : 7 ; Price : 25 ; Size : 1

Guest
12345

Guest Info:

Name:	Maryan
Surname:	Plakhtiy
Document Id:	12345
Date of Birth:	8/30/1995 12:00:00 AM
Sex:	MALE

7. When all fields are filled press the "Save check in" button. Subsequently you would be able to see that in a tab "Checkins List" your check in has occurred.

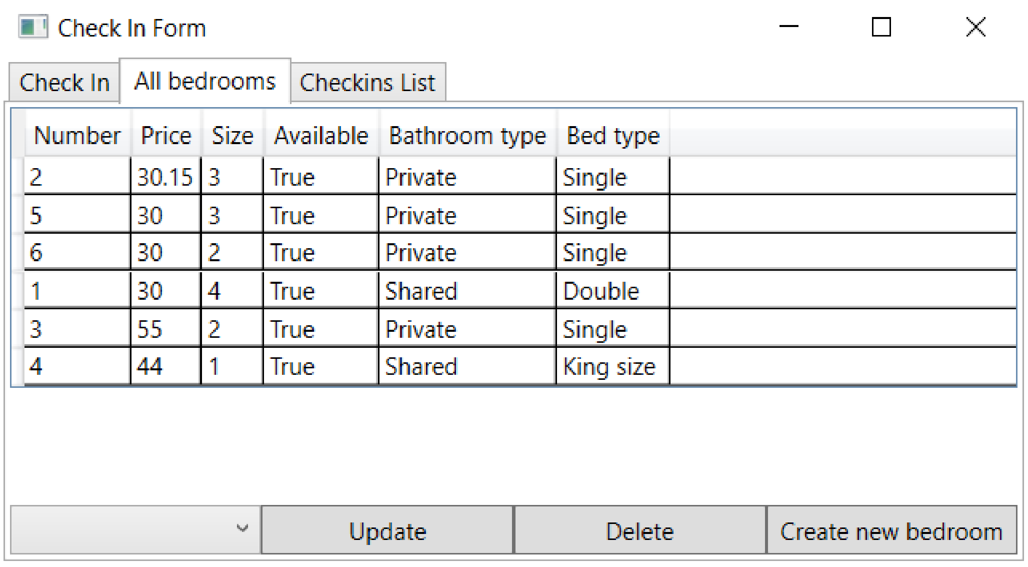


Arriving Date	Departure Date	Bedroom:
11/16/2016 11:00:00 PM	11/17/2016 11:00:00 PM	Bedroom № : 7 ; Price : 25 ; Size : 1

Bedroom № : 7 ; Price : 25 ; Size : 1

8. To delete a certain check in you need firstly to find it in a combo box and then press the "Delete" button.

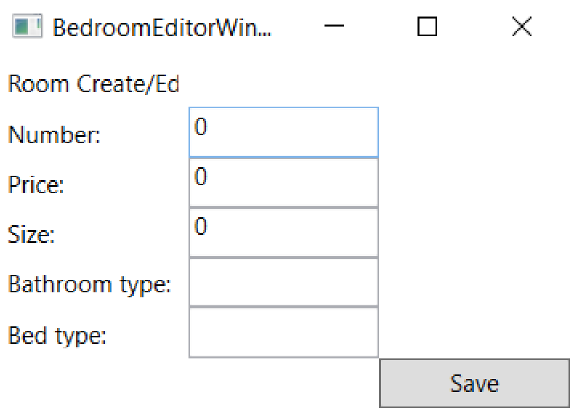
9. In a “All bedrooms” tab you can manage your rooms.



Number	Price	Size	Available	Bathroom type	Bed type
2	30.15	3	True	Private	Single
5	30	3	True	Private	Single
6	30	2	True	Private	Single
1	30	4	True	Shared	Double
3	55	2	True	Private	Single
4	44	1	True	Shared	King size

10. The availability of a room sets automatically.

11. By clicking on “Update” or “Create new bedroom” you will open a Bedroom Editor window.



Room Create/Ed

Number:	0
Price:	0
Size:	0
Bathroom type:	
Bed type:	

12. Remember to fill in all the fields.

1.8 Requirement changes

Change	Requirement affected	Description
1	Save Guest	Initially saving a guest gives hostel manager possibility of autofilling form by the input of the guest ID. Our services provide interfaces to achieve that requirement, but is not implemented as user interface feature.
2	View Check-In List	Added requirement.
3	Do Check-Out	Added requirement.

Table 3: Requirement changes.

1.9 References

- MongoDB Documentation: https://docs.mongodb.com/manual/introduction/
- Apache License 2.0: http://www.apache.org/licenses/LICENSE-2.0
- MongoDB Licensing: https://www.mongodb.com/community/licensing
- Martin Fowler Service layer: http://martinfowler.com/eaaCatalog/serviceLayer.html
- Introduction to WPF (MSDN): https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx
- What is WPF (WPF Tutorial): http://www.wpf-tutorial.com/about-wpf/what-is-wpf/
- The Repository Pattern (MSDN): https://msdn.microsoft.com/en-us/library/ff649690.aspx