coursework 1

Enterprise Application Development

Manuel Benavent Lledo – w1749459

Coursework 1

Enterprise Application Development

University of Westminster

Index

[1. Part A: Requirements 2](#_Toc23879299)

[2. Part B: Use Case Diagrams 4](#_Toc23879300)

[3. Part C – Classes 5](#_Toc23879301)

[1) CRC table 5](#_Toc23879302)

[2) Domain model 7](#_Toc23879303)

# Part A: Requirements

These are the requirements needed for building the time management and reporting tool for personal use:

**R1. The software shall allow the user to insert a new contact and edit the existing ones.**

R1.1. The information about the contact will be stored in a data base

R1.2. The contact must have a first and last name and the system must validate it, that is, it can be composed only by alphabetic characters.

R1.3. The contact may have an email address, if so, the software will validate it.

R1.4. The software will optionally allow the user to add a telephone number, it will check that is only composed by numbers (no more validations will be required since each country has a different format).

R1.5. The user will be able to delete contacts.

**R2. The software shall allow the user to create (or edit existing) events.**

R2.1. All events must have a name.

R2.2. All events must have a start and an end date and time.

R2.3. Events may be one-off or recurring. If the event is recurring, the user will have to introduce the number of days, weeks, months or years that it will be recurring for.

R2.4. There will be different types of events: appointments, tasks, lectures and tutorials.

R2.4.1. Tasks can be marked as completed and in that case they will be shown greyed out in the calendar.

R2.4.2. Lectures will have a field to introduce the lecturer.

R2.4.3. Tutorial will have a field to introduce the lecturer and the lab number.

R2.5. These events will be stored in the database.

R2.6. An event may have one or more contacts associated.

R2.7. An event may have a location.

R2.7.1. A location must have a name, address line, postcode (that will be verified), a city or town and a country. And optionally a second address line.

R2.8. The user will be able to delete an event and if it is recurring, he will be asked if he wants to delete one or all the future ones.

**R3. The software shall display the user interface with the following options:**

R3.1. The user interface will allow the user to perform the operations above mentioned.

R3.2. The system will provide the user with different views of the calendar: weekly and schedule.

R3.3. The system will provide an option to show a list of the contacts and an option to edit each of them.

R3.4. The system will provide the user with a time usage report for the next 4 weeks based on the previous weeks’ time usage.

**R4. The system will use a DDBB as a form of storing data in a persistent way.**

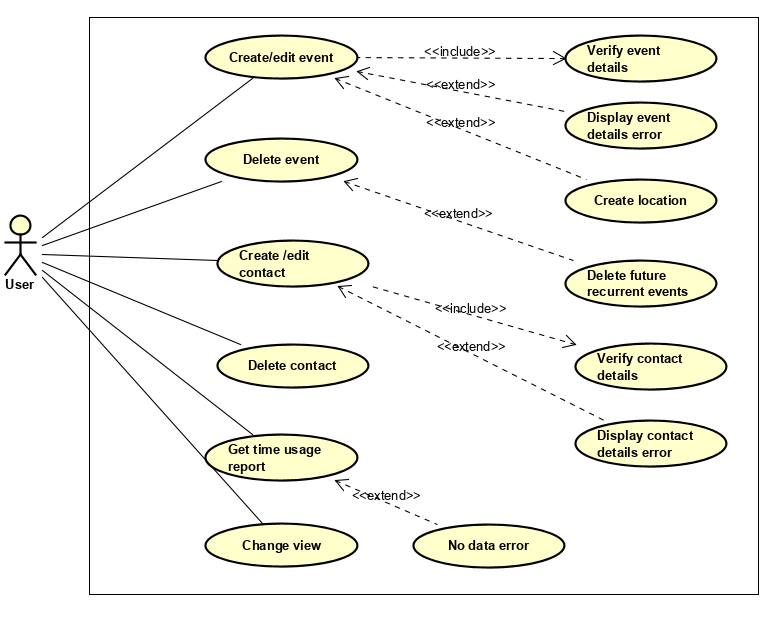
R4.1. The DDBB scheme will be as follows:

Imagen que contiene texto

Descripción generada automáticamente

# Part B: Use Case Diagrams

The designed system has the following use case diagram:



The uses cases shown above have the following descriptions:

Use Case: Create/edit event

Id: UC-001

Description:

The user wants to create or edit an event

Primary Actor:

User: the user of the system

# Part C – Classes

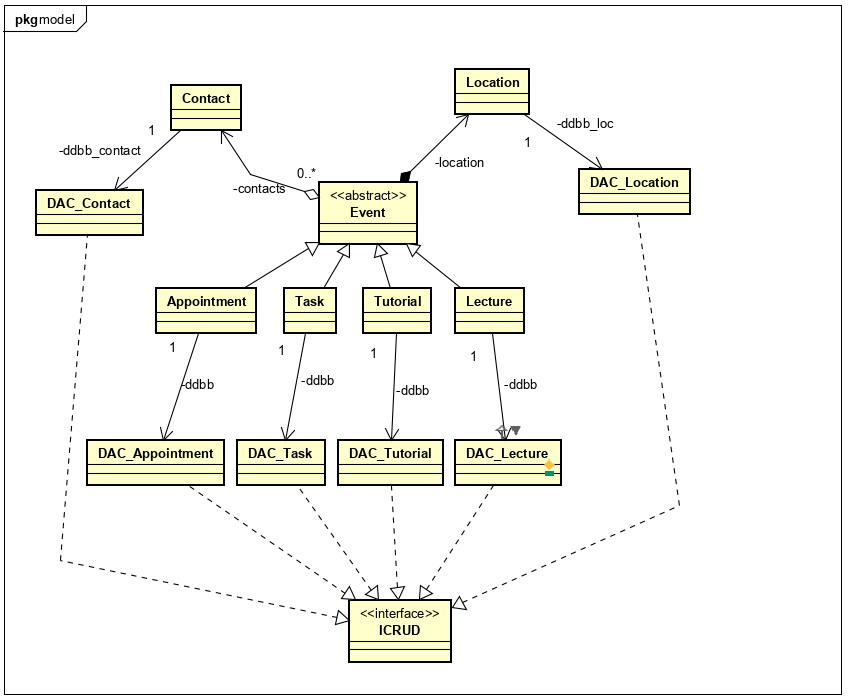
## CRC table

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name | Type | Responsibility | Collaborations |
| Event | Model | Abstract class that has the main features for an event | EventView, EventEditView and its subclasses |
| Task | Model | Type of event | DAC\_Task and super class collaborations |
| Appointment | Model | Type of event | DAC\_Appointment and super class collaborations |
| Lecture | Model | Type of event | DAC\_Lecture and super class collaborations |
| Tutorial | Model | Type of event | DAC\_Tutorial and super class collaborations |
| DAC\_Task | Model | Connects to the DDBB and obtains the desired information | Task |
| DAC\_Appointment | Model | Connects to the DDBB and obtains the information | Appointment |
| DAC\_Lecture | Model | Connects to the DDBB and obtains the information | Lecture |
| DAC\_Tutorial | Model | Connects to the DDBB and obtains the information | Tutorial |
| EventView | View and controller (Form\*) | Display detailed information for an event | Event and its subclasses and EventEditView |
| EventEditView | View and controller (Form\*) | View displayed when creating or editing events | Event and its subclasses |
| Location | Model | Stores the information for a location | DAC\_Location, Event |
| DAC\_Location | Model | Stores the information of a location in the DDBB | Location |
| LocationEditView | View and controller (Form\*) | View displayed when creating a location | Location, EventEditView |
| Contact | Model | Stores the information for contacts | Event, EventEditView, DAC\_Contact, ContactView, ContactEditView and ListContactView |
| DAC\_Contact | Model | Connects to the DDBB and obtains the information | Contact |
| ContactView | View and controller (Form\*) | Displays detailed information for a contact | Contact, ContactEditView |
| ContactEditView | View and controller (Form\*) | View displayed when creating or editing contacts | Contact |
| ListContactView | View and controller (Form\*) | Displays a list with all the contacts so the user can view, edit or delete. | Contact, ContactView |
| WeeklyView | View and controller (Form\*) | Displays a weekly view with all the events belonging to that week and the user is able to perform different operations from this view since it’s the main view | Event, Contact, ListContactView, ScheduleView, ReportView, EventView |
| ScheduleView | View and controller (Form\*) | Alternative view for the main one, it gives us the same options but the events are displayed in a schedule mode | Event, Contact, ListContactView, WeeklyView, ReportView, EventView |
| ReportView | View and controller (Form\*) | Displays the time usage prediction report window | Event |
| WrongNameException | Model | Exception thrown by the model when the introduced name is not correct | Contact |
| WrongEmailException | Model | Exception thrown by the model when the email format is not correct | Contact |
| WrongPhoneException | Model | Exception thrown by the model when the phone number format is not correct | Contact |
| WrongPostCodeException | Model | Exception thrown by the model when the post code format is not correct | Location |
| NoDataException | Model | Exception thrown by the model when we try to calculate the time usage prediction and there is no data |  |

\*Form: In Windows forms, Visual Studio will create 2 different files (.cs and .designer.cs) belonging to the same class but on separated files fulfilling the MVC requirements.

## Domain model

The system will be using the following domain model:



The classes that implement part of the view or the controller are not included in the domain model. However, since some of these classes will be able to throw different types of exceptions there is a domain model for the exceptions:

