RELAZIONE PROGETTO

Programmazione e modellazione a oggetti

REALIZZATO DA:

Pugnali Luca

a.a. 2019/2020

SPECIFICA DEL SOFTWARE

Il progetto consiste in un software grafico per la gestione delle prenotazioni dei vari servizi di un camping (3 bungalow, piscina e campi da gioco). In particolare il programma ha tre diverse schermate che permettono di:
1) Scegliere un servizio ed effettuare una nuova prenotazione
2) Scegliere un servizio per vedere l'elenco delle prenotazioni di quest'ultimo e decidere quale eliminare
3) Scegliere un giorno del calendario (che evidenzia i giorni con delle prenotazioni) per poter avere una panoramica di tutte le prenotazioni di tutti i servizi relativi alla data selezionata dall'utente
(anche le prime due schermate hanno un calendario, per avere un'idea delle prenotazioni del servizio selezionato, con il quale però non si può interagire)
Ogni prenotazione comprende le seguenti informazioni:
- Anno mese e giorno di inizio prenotazione
- Anno mese e giorno di fine prenotazione
- Nome del cliente che desidera prenotare

Il software è inoltre in grado di memorizzare in file interni al progetto le liste delle prenotazioni dei vari servizi del campeggio.

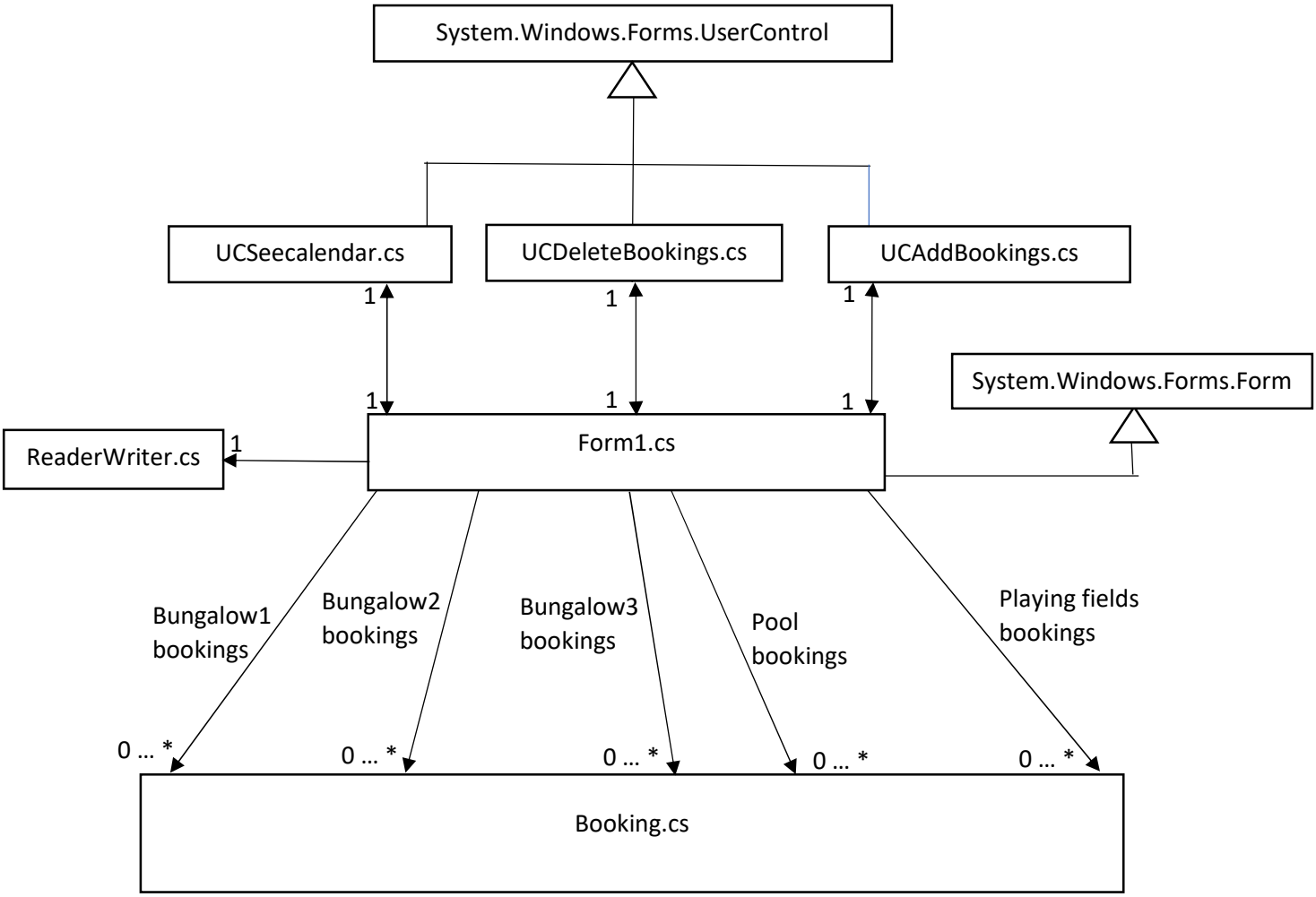
STUDIO DEL PROBLEMA

I punti critici del software sono legati alle principali funzionalità del programma:

- Aggiungere una nuova prenotazione: l'utente deve aprire l'apposita schermata per poi scegliere il servizio di cui si vuole effettuare la nuova prenotazione, compilando in seguito tutti i campi necessari alla creazione della nuova prenotazione, cioè: anno mese e giorno delle date di inizio e fine prenotazione, nome del cliente e scelta degli eventuali servizi aggiuntivi. In questa fase il programma deve analizzare i dati inseriti dall'utente e mostrare un messaggio di errore nel caso in cui: l'utente non ha compilato tutti i campi obbligatori (data di inizio fine e nome), una delle due date sia inesistente (es. 31/9/20 o 30/2/20), la data di inizio prenotazione è dopo quella di fine oppure nel caso in cui nell'intervallo di date scelte ci siano uno o più giorni già occupati (NB quest'ultimo requisito viene valutato solo nel caso in cui si sta prenotando un bungalow in quanto si assume che la piscina e i campi da gioco possano ospitare più clienti contemporaneamente).
- Eliminare una nuova prenotazione: l'utente deve aprire l'apposita schermata per poi scegliere il servizio del quale cancellare una prenotazione. A questo punto vengono elencate tutte le prenotazioni del servizio selezionato e l'utente deve solo fare click su quella da eliminare e confermare la sua scelta.
- Salvataggio e caricamento delle prenotazioni: le liste vengono memorizzate in più file interni al progetto (un file per ogni servizio del camping) dai quali si caricano i dati all'apertura del software e nei quali si scrivono tutti i dati ogni volta che il programma viene chiuso.

SCELTE ARCHITETTURALI E DESCRIZIONE TECNICA DELL'ARCHITETTURA

Il programma ha una classe principale, Form1.cs, nella quale sono state inserite le altre tre schermate utili al funzionamento del software, realizzate ognuna con un apposito `UserControl`. Infine il progetto contiene una classe `Booking.cs` che viene istanziata tramite costruttore ogni volta che si aggiunge una nuova prenotazione e una classe `ReaderWriter.cs` che, una volta istanziata, rappresenta l'oggetto a cui fare riferimento per leggere e scrivere i dati nei file di salvataggio.



Form1.cs

La principali funzioni di questa classe sono: quella di gestire e coordinare tutte le altre e contenere le strutture dati utilizzate per memorizzare le prenotazioni.

DESIGN:

La Form ha dei bottoni nel suo lato sinistro che permettono di muoversi tra le varie schermate, le quali sono state posizionate nel rettagolo blu dell'immagine una sovrapposta all'altra. Quando l'utente preme un bottone si mette in evidenza la schermata relativa al bottone e si nascondono le altre.

ATTRIBUTI:

- 5 liste tipizzate con la classe Booking, ognuna delle quali rappresenta la lista di prenotazioni di un servizio, come i rispettivi nomi fanno capire: `bungalowsBookings1, bungalowsBookings2, bungalowsBookings3, poolsbookings, fieldsbookings.` Come struttura dati è stata scelta la lista generica per la sua dinamicità e per la sua facilità di gestione degli elementi. Tutte e 5 le liste vengono tenute private ma è possibile accedervi `dalle` altre classi tramite le omonime proprietà pubbliche di lettura e scrittura.
- Una variabile booleana `isCollapsed` utile al funzionamento del menù a tendina grazie alla quale si tiene traccia dello stato del panel grazie al quale è stato implementato.
- Un oggetto `ReaderWriter` (altra classe implementata) grazie al quale si effettuano letture e scritture dei dati negli appositi file di salvataggio.

METODI:

Principalmente questa classe ha metodi per gestire gli eventi degli elementi grafici (click dei bottoni) i quali hanno il semplice ed unico compito di mostrare la schermata corrispondente al bottone.

- `ColorCalendar` è un metodo utilizzato da tutte le altre schermate per mettere in grassetto nel calendario le date già occupate. I parametri sono una lista di prenotazioni e il calendario interessato. Dati questi due parametri si agisce nel seguente modo: si scorre la lista di prenotazioni e ogni giorno di ogni prenotazione si aggiunge a una nuova list creata dentro questo metodo. Una volta inseriti tutti i giorni nella lista, quest'ultima viene trasformata in un array con elementi dello stesso tipo il quale verrà poi associato alla proprietà `BoldedDates` del calendario, grazie alla quale le date risulteranno in grassetto. L'ultima operazione eseguita in questo metodo è lo spostamento del calendario nella prima data utile (se c'è) in grassetto.
- `Form1:` il costruttore di questa classe svolge diverse istruzioni fondamentali in quanto, essendo questa Form il contenitore di tutti gli oggetti del programma, può essere considerato come l'inizio dell'esecuzione del programma. Innanzitutto si assegna all'apposito campo delle altre classi una copia dell'istanza della Form1 in modo che, dalle altre classi, si possa comunque accedere al metodo `ColorCalendar` e alle varie liste memorizzate appunto in questa classe principale. Dopo di che si richiama il metodo dell'oggetto `ReaderWriter` per caricare i dati delle prenotazioni e si inseriscono nelle liste dei vari servizi. Infine si richiamano i metodi `SetUc` per rendere pronte da subito le varie schermate e si aggiunge un nuovo `EventHandler` all'evento `ApplicationExit` che permette di sovrascrivere i dati nei file di salvataggio ogni volta che l'applicazione viene chiusa.

ReaderWriter.cs

Per questa classe si è deciso di utilizzare il design pattern di tipo creazionale SINGLETON per far in modo che ci sia sempre e comunque un solo oggetto istanziato durante l'esecuzione. Questa scelta è giustificata dal fatto che se ci fossero più oggetti di questo tipo si rischierebbe, ad esempio nella fase di scrittura nei file, di effettuare più salvataggi sovrascrivendo magari liste di prenotazioni che effettivamente andrebbero memorizzate

ATTRIBUTI:

Gli attributi rappresentano appunto i vari dati di ogni prenotazione e vengono assegnati nella creazione dell'oggetto tramite costruttore.

- Due variabili di tipo `DateTime` per memorizzare le due date di inizio e fine prenotazione.

- Una stringa per memorizzare il nome del cliente che desidera prenotare
- Un oggetto di tipo `XmlSerializer` per serializzare e deserializzare le strutture da salvare/caricare.
- Una stringa assegnata dal costruttore che contiene il percorso della cartella contenente i file di salvataggio. Precisamente alla stringa viene assegnato il seguente valore:

```
//part of the path
path = Path.Combine(Environment.CurrentDirectory, @"Data\");
```

  `Enviornment.CurrentDirectory` fa riferimento alla cartella interna al progetto con percorso bin -> Debug. Facendo riferimento alla cartella del progetto in questo modo il riferimento al file rimane valido a prescindere dalla macchina in cui il software viene eseguito.
- Con questo modo per far riferimento ai singoli file basta concatenare a questa stringa il nome dell'xml.
- Un oggetto statico che istanzia la classe stessa utilizzato durante l'esecuzione come riferimento all'unico oggetto creato. Il modificatore `static` è molto importante per l'implementazione del singleton in quanto, non potendo istanziare nuovi oggetti liberamente tramite il costruttore (private), la creazione dell'oggetto avviene tramite il metodo a sua volta statico `GetIstance` il quale controlla ad ogni invocazione che il principio del design pattern venga rispettato.

METODI:

- `GetInstance` come già spiegato sopra è il metodo che ha come compito quello di far rispettare il design pattern. In questo metodo molto semplice si controlla che non sia stato ancora creato un oggetto di questo tipo, valutando la variabile `instance` della classe stessa: se la variabile è `null` significa che ancora bisogna istanziare un oggetto di quel tipo, altrimenti non si crea il nuovo oggetto e si restituisce quello già esistente.
- `SaveData` viene richiamato durante la chiusura dell'applicazione per salvare i dati nei file xml. In questo metodo si ripetono le stesse operazioni per ogni tipo di servizio presente nel camping:
  1) si assegna alla variabile di tipo File una nuova istanza, assegnandole tramite costruttore il percorso del file relativo al servizio cui si vuole fare riferimento (ottenuto concatenando la stringa `path` al nome del file).
  2) Si invoca il metodo `Serialize` dell'`XmlSerializer` che riceve come parametri l'istanza del file contenente la lista del servizio e la lista stessa. Una volta eseguita questa istruzione il serializzatore avrà serializzato la lista e avrà memorizzato nei file xml tutte le prenotazioni
- `GetData` viene invece invocato, come già detto, durante la creazione della classe principale e si utilizza per leggere i dati dai file xml di salvataggio. Questo metodo non ha parametri e restituisce una lista tipizzata in liste di prenotazioni (List<List<Booking>>) contenente le liste di tutti i servizi. Ciò che fa il metodo è semplicemente chiamare `GetListFromFile` (ultimo metodo della classe di seguito spiegato) su tutti i file xml contenenti le liste.
- `GetListFromFile` è un metodo che restituisce una lista leggendo e deserializzando le prenotazioni da un file xml (passato tramite il suo nome come parametro di ingresso della funzione) contenuto nella cartella di salvataggio interna al progetto invocando il metodo `Deserialize` sui file di tutti i servizi.

UcAddBooking.cs

È la schermata che permette di aggiungere una nuova prenotazione a un servizio.

DESIGN:

Per scegliere il servizio basta fare click in una delle immagini nella parte alta della schermata. Per selezionare le due date di inizio e fine prenotazione sono state disposte 6 `combobox`: due per gli anni, due per i mesi e altrettante per i giorni. La scelta di questo controllo per acquisire i dati è dovuta al fatto che in questo modo l'utente deve per forza inserire uno dei valori che gli vengono messi a disposizione e questo rimane comodo poi nella validazione dei dati inseriti. Sotto ancora troviamo una `textbox` per inserire il nome del cliente e due `checkbox` da spuntare per scegliere un servizio aggiuntivo tra piscina e campi da gioco. La schermata di prenotazione cambia in base al servizio per cui la si sta facendo; in particolare se si sta prenotando per un bungalow verranno visualizzati tutti gli elementi sopra elencati mentre, se si sta prenotando per piscina o campo, verranno rimosse dalla schermata le `combobox` per impostare la data di inizio prenotazione e le `checkbox` per scegliere eventuali servizi aggiuntivi. Nell'eventualità in cui si spunti uno dei due servizi aggiuntivi (quindi se si sta prenotando un bungalow) verrà creata una prenotazione indipendente e parallela a quella del bungalow ma della stessa durata; in questo modo se il cliente dovesse chiedere di togliere solo i servizi aggiuntivi basterà rimuovere quella prenotazione, lasciando invariata quella del bungalow. Infine questa schermata ha un calendario che segna in grassetto i giorni già occupati rispetto al servizio selezionato; se durante la prenotazione l'utente dovesse provare a prenotare in giorni già occupati o inesistenti, il programma provvederà a stampare un messaggio d'errore, invitandolo a controllare il calendario. Il calendario serve dunque per far capire a chi usa il programma per quale motivo la loro prenotazione non viene inoltrata.

ATTRIBUTI:

Oltre ai controlli grafici sopra descritti la classe ha i seguenti attributi:

- Una variabile `form` utilizzata come riferimento alla classe principale di tipo Form1. L'assegnazione dell'istanza viene effettuata nel costruttore della Form1.cs attraverso la seguente istruzione: `this.ucAddBookings1.Form = this.`

- `actualService` è una lista tipizzata in Booking alla quale si fa riferimento per aggiungere una nuova prenotazione. A questa struttura viene assegnata la giusta lista di prenotazioni (contenute nella Form1.cs) ogni volta che l'utente sceglie un servizio nella parte alta della schermata. Basandosi su questa considerazione basta far riferimento ad `actualService` quando si vuole aggiungere un servizio e la prenotazione verrà automaticamente messa nella giusta lista.

METODI:

- Per rendere i pulsanti nella parte alta funzionante è stato assegnato a ognuno una funzione all'evento click nella quale si aggiorna l'`actualService`, nel calendario si colorano i giorni occupati del servizio selezionato e infine si mostrano/nascondono i controlli grafici utili alla prenotazione del servizio selezionato (se è stato selezionato un bungalow si mostreranno i checkBox per la scelta dei servizi aggiuntivi mentre se è stata selezionata la piscina questi dovranno essere nascosti.
- `Hidecomboboxlabel` e `showcomboboxlabel` servono a mostrare o a nascondere i giusti controlli grafici in base a quale servizio è stato selezionato, come spiegato in precedenza.
- `btnAddBook_Click` contiene il codice da eseguire quando l'utente vuole provare a inoltrare una prenotazione. Il suo codice è molto semplice, quest'ultimo infatti non fa altro che verificare quale sia il servizio attuale e in base a questo richiamare uno dei due metodi `ValidateBungalow` oppure `Validatepoolorfields`
- `Validate bungalow` si occupa di inoltrare la prenotazione dopo aver svolto tutti i controlli del caso. La prima verifica fatta è quella per assicurarsi che la prenotazione abbia almeno le due date di inizio e fine ed il nome; questa verifica si effettua tramite il metodo `AreCompiled`, descritto meglio in seguito. Una volta verificato che tutti i campi sono stati compilati, dentro una struttura `try catch` si inseriscono le due date scelte dall'utente in due variabili di tipo `DateTime` e si verifica che quella di inizio sia prima di quella di fine; se la data inserita dall'utente dovesse essere inesistente alla creazione della variabile `DateTime` verrà sollevata un'eccezione che viene gestita settando a false la variabile `booleana isValid` sulla quale si effettueranno poi dei controlli per capire se qualcosa non ha superato una parte della validazione. Successivamente si invoca la funzione `ItsFree` (metodo per sapere se quel giorno ha già altre prenotazioni) su tutti i giorni compresi nell'intervallo tra le date di inizio e fine prenotazione; se si dovesse trovare anche solo un giorno occupato si setterebbe a false la variabile di controllo `isValid`. Infine si fa una verifica sulla variabile `isValid`, se l'esito è positivo, la prenotazione del bungalow e di eventuali servizi aggiuntivi verrà inoltrata, altrimenti si emette un messaggio d'errore per comunicare all'utente che è impossibile prenotare e lo si invita a ricontrollare i dati inseriti ed il calendario per risolvere capire il problema.
- `ValidatePoolOrFields` è molto simile al metodo appena descritto, con la differenza che bisogna effettuare meno controlli in quanto per prenotare la piscina o i campi basta selezionare una data e non due. Questo metodo controlla che l'utente abbia inserito almeno una data e un nome (invocando il metodo `AreCompiled`), per poi assicurarsi che il giorno selezionato sia libero tramite il metodo `ItsFree` descritto di seguito.
- `ItsFree` riceve una data in entrata e questa viene confrontata con tutte le date evidenziate nel calendario, se quella data è già evidenziata (e quindi occupata) si restituisce `false`, altrimenti `true`.
- `AreCompiled` inizialmente controlla che le `combobox` per inserire la data di inizio prenotazione e la `textbox` per il nome siano state compilate dall'utente; successivamente, solo se si sta prenotando un bungalow, controlla che anche le `combobox` per la data di fine prenotazione siano compilate.
- `SetUc` è l'ultimo metodo della classe e viene chiamato ogni volta che questa UC viene aperta e svolge le seguenti azioni: assegna ad `actualService` la lista del primo bungalow e evidenzia le date occupate nel calendario.

UcDeleteBooking.cs

È la schermata che permette di eliminare una prenotazione esistente relativa a un servizio.

DESIGN:

Per scegliere il servizio basta fare click in una delle immagini nella parte alta della schermata. Una volta scelto il servizio nella parte centrale della schermata verranno elencate tutte le prenotazioni relative a quel servizio; per eliminarne una basterà farci click e confermare la propria volontà di eliminarla. Per stampare le prenotazioni si utilizza una lista dinamica di etichette che vengono aggiunte e rimosse dal `flowLayoutPanel` in modo da averle sempre disposte in modo ordinato dall'alto verso il basso; la proprietà `AutoScroll` di quest'ultimo viene impostata su `true` in modo da poter scorrere tra le prenotazioni nel caso in cui non ci stessero tutte in altezza o larghezza.

ATTRIBUTI:

Oltre ai controlli grafici sopra descritti la classe ha i seguenti attributi:

- Una variabile `form` utilizzata come riferimento alla classe principale di tipo Form1. L'assegnazione dell'istanza viene effettuata nel costruttore della Form1.cs attraverso la seguente istruzione: `this.ucDeleteBookings1.Form = this.`
- `listLblBook` è la lista dinamica nella quale vengono inserite le etichette aggiunte al `flowLayoutPanel`, ognuna delle quali contiene le informazioni di una prenotazione (date e nome).
- `actualService` è una lista tipizzata in Booking alla quale si fa riferimento per aggiungere una nuova prenotazione. La sua funzione è uguale a quella all'omonima lista nella classe sopra descritta.

METODI:

- Per rendere i bottoni nella parte alta funzionante è stato assegnato a ognuno una funzione all'evento click nella quale si aggiorna l'`actualService`, nel calendario si colorano i giorni occupati del servizio selezionato e infine si mostra la lista di prenotazioni relativa al servizio.

- `LastItem` è un metodo privato utile nella gestione della lista dinamica di etichette in quanto, data una lista di etichette questa funzione restituisce l'ultimo elemento della lista.
- `BuildBookString` è un metodo privato che permette di ottenere una stringa con tutti i dati relativi ad una prenotazione partendo dalla prenotazione stessa che viene passata come input della funzione. Questo metodo rende più snello il codice nella parte in cui bisogna elencare le prenotazioni tramite le etichette, in quanto il testo dell'etichetta non sarà altro che il valore restituito dalla funzione.
- `SetProperties` è una funzione alla quale che ha il seguente scopo: si passa come input un'etichetta appena creata per modificarne le proprietà di default e renderle funzionali e gradevoli esteticamente. Di questo metodo troviamo due versioni:
  1) La prima riceve come input un'etichetta ed un oggetto di tipo Booking. Dell'etichetta passata come parametro si modificano dimensione, visibilità, testo (usando la funzione `BuildBookString`), colore e font; infine si associa all'evento click dell'etichetta un metodo specifico che permetterà, a tempo d'esecuzione, di cancellare la prenotazione e l'etichetta stessa quando si fa click su quest'ultima.
  2) La seconda versione riceve invece solo un'etichetta come parametro l'unica differenza con la prima è che alla proprietà `Text` dell'etichetta non si associano i dati di una prenotazione ma si scrive "Non ci sono prenotazioni". Questo metodo viene infatti invocato quando non ci sono prenotazioni e l'unica etichetta da aggiungere alla lista dovrà avere questo testo.

  In conclusione se la lista da stampare ha delle prenotazioni si crea una nuova etichetta per ogni prenotazione e si modificano tutte con la prima versione del metodo mentre se la lisat da stampare è vuota si utilizza una sola etichetta modificata con la seconda versione.

- ClearFlowPanAndLblArray è un metodo molto semplice che chiama il metodo clear (ne elimina tutti gli elementi) sulla lista delle etichette listLblBook e sul pannello flwPanelBookingsList contenente tutte le etichette.
- `SetUc` è l'ultimo metodo della classe e viene chiamato ogni volta che questa UC viene aperta e svolge le seguenti azioni: assegna ad `actualService` la lista del primo bungalow, ne stampa a schermo le prenotazioni e evidenzia le date occupate nel calendario della schermata.

Booking.cs

Questa classe è molto semplice, verrà istanziata ad ogni nuova prenotazione, quindi ogni oggetto booking rappresenta una prenotazione e verrà aggiunto o tolto alle liste presenti nella classe principale del programma (`Form1.cs`).

ATTRIBUTI:

Gli attributi rappresentano appunto i vari dati di ogni prenotazione e vengono assegnati nella creazione dell'oggetto tramite costruttore.

- Due variabili di tipo `DateTime` per memorizzare le due date di inizio e fine prenotazione.
- Una stringa per memorizzare il nome del cliente che desidera prenotare

UcSeeCalendar.cs

È la schermata che permette di eliminare una prenotazione esistente relativa a un servizio.

DESIGN:

Questa è la schermata mostrata all'apertura del software. Come già detto questa schermata non serve a cancellare o aggiungere nuove prenotazioni ma serve solo a consultare quelle esistenti in base alla data selezionata nel calendario. Per disporle è stato utilizzato lo stesso controllo della schermata precedente ossia un `flowLayoutPanel`.

ATTRIBUTI:

oltre a quelli grafici descritti sopra la classe ha i seguenti attributi:

- Una variabile `form` utilizzata come riferimento alla classe principale di tipo Form1. L'assegnazione dell'istanza viene effettuata nel costruttore della Form1.cs attraverso la seguente istruzione: `this.ucSeeCalendar1.Form = this.`
- `listLblBook` è la lista dinamica nella quale vengono inserite le etichette aggiunte al `flowLayoutPanel`, ognuna delle quali conterrà solo il nome del cliente.
- dates è infine una lista dinamica in cui vengono memorizzate le date relative a tutte le prenotazioni di tutti i servizi per poi evidenziarle nel calendario.

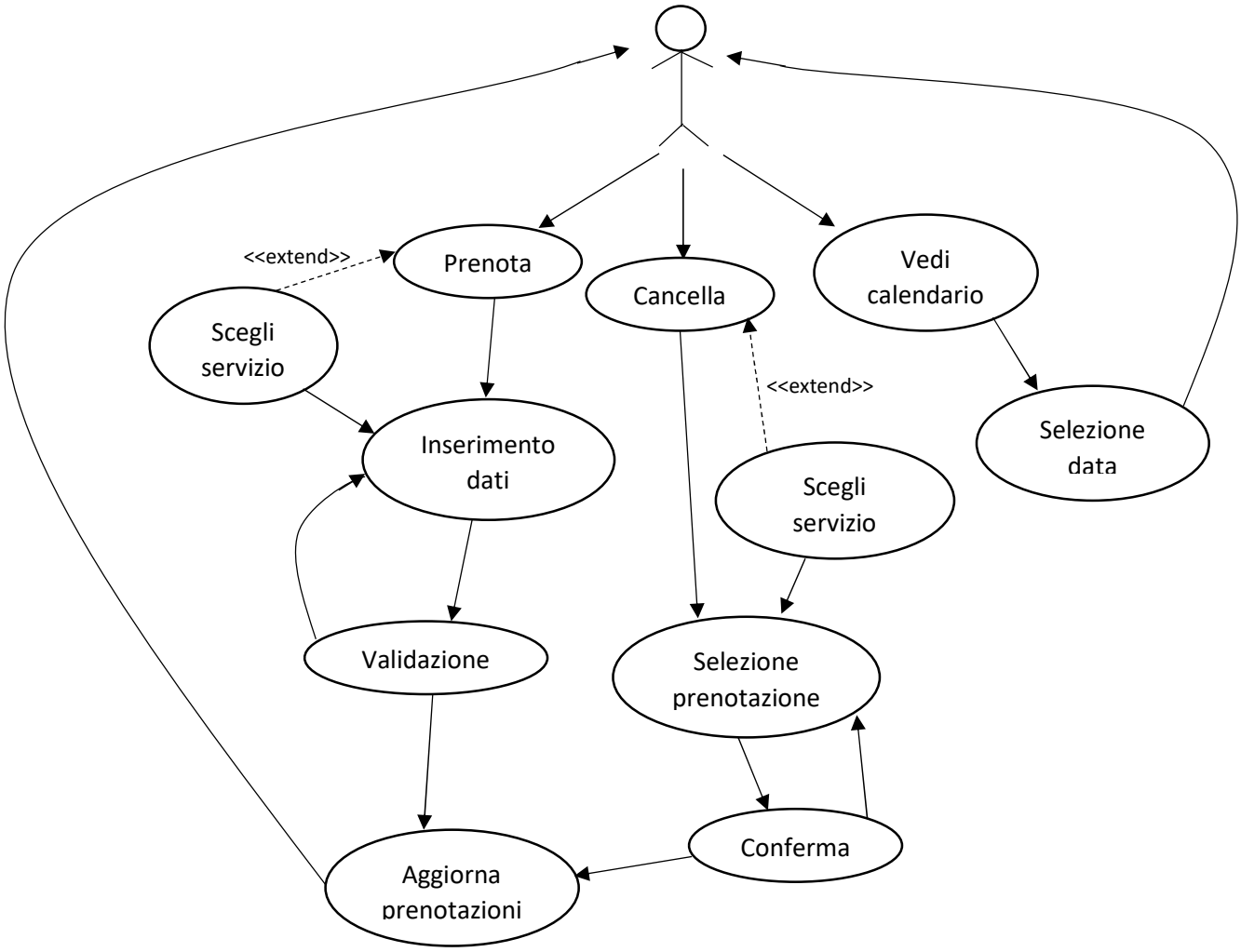
METODI:

- `ClearFlowPanAndLblArray` è un metodo molto semplice che chiama il metodo clear (ne elimina tutti gli elementi) sulla lista delle etichette `listLblBook` e sul pannello `flwPanelBookingsList` contenente tutte le etichette.
- `SetProperties` è la medesima funzione della schermata sopra descritta ma l'unica differenza sta nel fatto che nella prima versione, ossia quella che accetta come parametri un oggetto prenotazione e un'etichetta, non associa alla proprietà Text di quest'ultima solo "`Nome :` "+`book.BookerName` dove book è il primo parametro sopra citato della funzione. A causa di questa differenza il metodo non è stato reso condiviso tra le varie classi (come `ColorCalendar` della classe `Form1.cs`) ma è stato reso interno a ogni classe che necessitava la sua variante.
- `LastItem` è un metodo privato utile nella gestione della lista dinamica di etichette in quanto, data una lista di etichette questa funzione restituisce l'ultimo elemento della lista.

- Per evidenziare le date con qualche prenotazione, non si utilizza il metodo condiviso `ColorCalendar` della classe `Form1.cs`, perché questo riceve una lista di prenotazioni e mette in grassetto le relative date, differentemente in questa schermata, il calendario deve evidenziare le date occupate di tutti i servizi. In particolare in questo metodo si svolgono le seguenti istruzioni: si inseriscono nella lista privata `dates` i giorni occupati di tutti i servizi (tramite il metodo `AddBoldDates` spiegato di seguito), si trasforma la lista in array per renderla compatibile con la proprietà `BoldedDates` e la si assegna a quest'ultima evidenziando i giorni di tutti i servizi.
- `AddBoldDates` è un metodo utile a `ColorCalendar` e svolge un ruolo molto semplice: riceve una lista come parametro, per poi inserire tutti i giorni con prenotazioni nella lista interna alla classe chiamata `dates`. Invocando cinque volte questa funzione (una volta per ogni servizio) si avranno tutte le date utili in un'unica struttura dati.
- Il penultimo metodo di questa classe è `SearchAndPrint` che accetta come parametri una lista di oggetti Booking e un oggetto `DateTime`. Nel corpo della funzione si controlla se nella data specificata c'è qualche prenotazione della lista, in questo caso stampa i nomi delle persone che hanno prenotato in quel giorno, altrimenti stampa "Non ci sono prenotazioni per questo servizio" o "Non ci sono prenotazioni nella data selezionata" in base alla causa della mancanza di prenotazioni.

CASI D'USO

Di seguito viene rappresentato il diagramma che comprende tutti i casi d'uso del programma:



Descrizione dei casi d'uso:

Use case: Prenota	ID: UC1	Attore: Utente
Precondizione: /		
Corso degli eventi: l'utente clicca nel menù a sinistra il bottone per vedere la schermata per prenotare		
Post-condizione: UC2 o UC3		
Alternativa: UC3 se si vuole cambiare il servizio attualmente selezionato		

Use case: Scegli servizio	ID: UC2	Attore: Utente
Precondizione: UC1		
Corso degli eventi: L'utente seleziona un servizio diverso da quello attuale o da quello selezionato di default (se ha appena aperto il programma) in modo che il programma disponga la schermata nel modo migliore.		
Post-condizione: UC3		
Alternativa: /		

Use case: Inserimento dati	ID: UC3	Attore: Utente
Precondizione: UC1 se è solo stata aperta la schermata, UC2 se si si modifica il servizio un altro servizio UC3		
Corso degli eventi: L'utente inserisce la data di inizio e fine prenotazione e il nome del cliente		
Post-condizione: UC4		
Alternativa:		

Use case: Validazione	ID: UC4	Attore: /
Precondizione: UC3		
Corso degli eventi: Quando l'utente preme su prenota, si effettuano le seguenti verifiche: l'utente deve aver inserito i dati necessari, la data di inizio sia minore di quella di fine prenotazione, tutte le date comprese tra queste ultime due siano non occupate e esistenti.		
Post-condizione: Se tutte le condizioni sono rispettate si può procedere a UC5		
Alternativa: Se una di queste condizioni non viene rispettata si ritorna alla precondizione UC3 per far modificare all'utente i dati inseriti.		

Use case: Aggiorna prenotazioni	ID: UC5	Attore: /
Precondizione: UC4 o UC9		
Corso degli eventi: Si aggiornano le liste delle prenotazioni riportando le modifiche richieste dall'utente (si rimuove un elemento se la precondizione è UC9 o se ne aggiunge una se la precondizione è UC4		
Post-condizione: Le liste sono aggiornate e l'utente può effettuare altre operazioni.		
Alternativa: /		

Use case: Cancella	ID: UC6	Attore: Utente
Precondizione: /		
Corso degli eventi: l'utente clicca nel menù a sinistra il bottone per vedere la schermata per eliminare una prenotazione.		
Post-condizione: UC7 o UC8		
Alternativa: /		

Use case: Scegli servizio	ID: UC7	Attore: Utente
Precondizione: UC6		
Corso degli eventi: se non va bene il servizio selezionato se ne sceglie un altro per visualizzarne la lista		
Post-condizione: UC8		
Alternativa: /		

Use case: Selezione prenotazione	ID: UC8	Attore: Utente
Precondizione: UC6 o UC7		
Corso degli eventi: tra tutte quelle elencate l'utente clicca sulla prenotazione da eliminare		
Post-condizione: UC9		
Alternativa: /		

Use case: Conferma	ID: UC9	Attore: Utente
Precondizione: UC8		
Corso degli eventi: si apre una finestra di dialogo con l'utente per confermare la sua volontà di eliminare la prenotazione.		
Post-condizione: Se l'utente conferma si procede a UC5.		
Alternativa: altrimenti si ritorna a UC8 per selezionare un'altra prenotazione.		

Use case: Vedi calendario	ID: UC10	Attore: Utente
Precondizione: /		
Corso degli eventi: l'utente clicca nel menù a sinistra il bottone per vedere la schermata per consultare il calendario		
Post-condizione: UC11		
Alternativa: /		

Use case: Selezione data	ID: UC11	Attore: Utente
Precondizione: UC10		
Corso degli eventi: l'utente clicca nel menù a sinistra il bottone per vedere la schermata per consultare il calendario		
Post-condizione: La lista di prenotazioni relativa alla data selezionata è consultabile a schermo		
Alternativa: /		