Università degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione

Clup - Customer Line Up

Progetto d'esame del corso di *Ingegneria del Software*
Anno Accedemico 2020 - 2021

Autori
Elvis Nasic
Tommaso Zanelli

30 gennaio 2021

Indice

Capitolo 1

Abstract

L'emergenza *coronavirus* ha messo a dura prova la società su molti livelli, a causa dell'imposizione di blocchi da parte di molti paesi che consentono alle persone di uscire dalle loro case solo per esigenze essenziali e l'applicazione di regole severe anche quando le persone sono giustificate a uscire (come il limite del numero di accessi agli edifici e il mantenimento di una distanza di almeno un metro tra le persone).
In particolare, fare la spesa può diventare una sfida in presenza di regole così severe.
In effetti, i supermercati devono migliorare l'accesso ai loro negozi per evitare di avere folle all'interno, il che in genere si traduce in lunghe code che si formano all'esterno, che sono a loro volta una fonte di pericoli.
In questi tempi difficili, le persone si rivolgono alla tecnologia, e in particolare alle applicazioni software,per affrontare le sfide create dalle restrizioni imposte.
L'obiettivo di questo progetto è quello di sviluppare un'applicazione facile da usare che, da un lato, consenta ai responsabili dei negozi di regolare l'afflussodi persone nell'edificio e, dall'altro, risparmi il dover star in coda per ore ed ore.

Capitolo 2

Introduzione

La seguente documentazione ha lo scopo di accompagnare lo sviluppatore dalla fase di *Analisi e Progettazione del Software*, alla *fase di sviluppo.*
In particolar modo, verranno presentati:

- Il diagramma dei casi d'uso;
- Il diagramma delle classi a livello di dominio;
- Il diagramma delle classi a livello di progettazione;
- Il diagramma SSD;
- Il diagramma di sequenza di progettazione;
- Il diagramma dell'architettura software;
- Il diagramma degli stati;
- Il diagramma delle attività.
- Pattern Architetturali e Design Principles
- La scelta dei linguaggi di sviluppo
- Analisi del codice

Capitolo 3

Analisi

3.1 Attori primari

Gli attori primari individuati, ovvero gli attori che integrasicono direttamente con il sistema, sono i seguenti:

- Customer
- Manager

In particolar modo il *customer* rappresenta l'attore al quale il sistema è diretto, ovvero è colui che prenota un posto in un determinato negozio in una determinata fascia oraria.
Il *manager* invece, è il responsabile del negozio, ovvero la persona che visualizza le prenotazioni, può modificarle e può modificare il numero massimo di prenotazioni effettuabili.

3.2 Casi d'uso

I casi d'uso che vengono individuati sono relativi alle funzionalità che un attore può compiere per raggiungere un obiettivo.

> *Definizione: Per* **casi d'uso** *si intende la descrizione di un insieme di interazioni tra un utente ed un sistema che consentono all'utente di raggiungere un obiettivo o di svolgere un compito.*

Nella figura riportata in seguito, si illustra il diagramma dei casi d'uso, comprensivo di attori primari e di obiettivi raggiungibili da ognuno di essi.

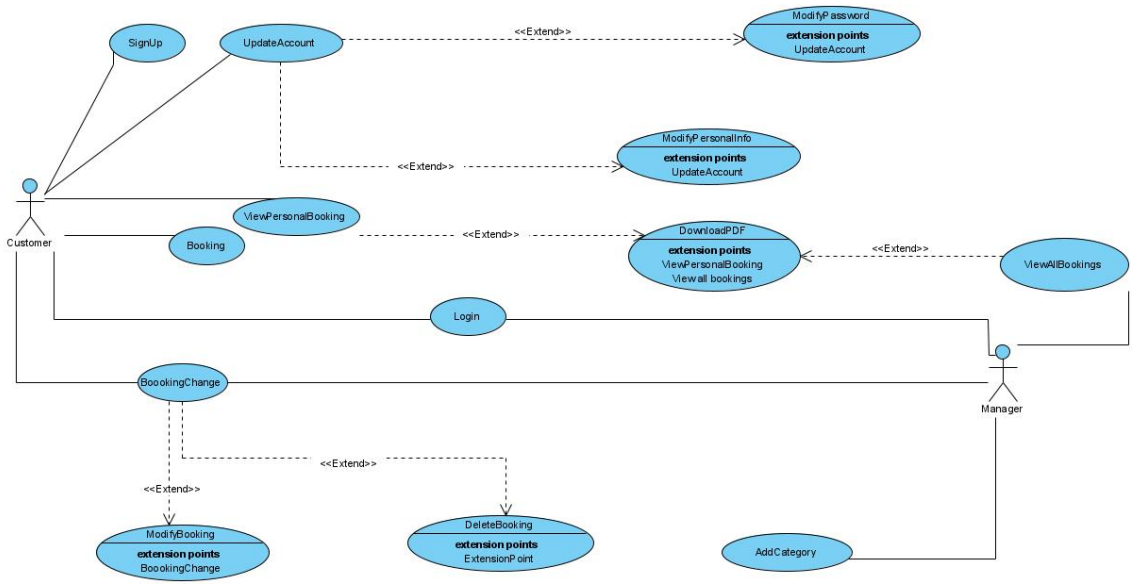

Figura 3.1: Diagramma dei casi d'uso

Di seguito sono riportati i casi d'uso in formato non dettagliato, ovvero gli use case vengono presentati in maniera sintetica.

3.2.1 Casi d'uso in formato non dettagliato

Di seguito vengono riportati i casi d'uso in formato non dettagliato, ovvero con una breve descrizione per il raggiungimento dell'obiettivo.

- **UC001 - Login**
  Il cliente effettua il login all'interno del sistema, andando ad inserire le credenziali inserite durante la fase di registrazione, potendo così proseguire la navigazione all'interno dell'applicazione con i permessi a lui attribuiti.
- **UC002 - SignUp**
  Il cliente provvede alla registrazione, andando ad inserire nella sezione dedicata i propri dati personali.

- **UC003 - ModifyPassword**
  Il cliente provvede ad inserire, nel caso in cui si sia dimenticato la propria chiave d'accesso, lo username e l'indirizzo e-mail con il quale ha effettuato la registrazione, e successivamente provvede ad inserire una nuova password.

- **UC004 - ModifyPersonalInfo**
  Il cliente effettua il login all'interno del sistema ed accedendo nella sezione relativa, provvede ad aggiornare i propri dati nel caso in cui ci siano state delle modifiche.

- **UC005 - Booking**
  L'attore, una volta loggato, ha la possibilità di prenotarsi presso un negozio in un determinato slot orario. Egli, una volta effettuato l'accesso alla pagina relativa, seleziona il negozio, l'orario ed eventuali categorie di prodotti interessate. Effettuaua la registrazione, l'utente riceve un'e-mail di conferma.

- **UC006 - ModifyBooking**
  L'attore, una volta loggato, può modificare una prenotazione se e solo se questa risulta ancora attiva, ovvero se la data non è passata rispetto al giorno seguente. L'attore, dopo aver effettuato l'accesso nella sezione corrispondente, può modificare data, orario ed eventualmente le categorie di interesse. Dopo aver confermato la modifica, l'attore riceve un'e-mail di conferma.

- **UC007 - DeleteBooking**
  L'attore, una volta loggato ed entrato nella corrispondente sezione, può decidere di eliminare una prenotazione. Dopo aver confermato tale volontà, riceve un'e-mail di riepilogo.

- **UC008 - ViewPersonalBookings**
  L'attore, dopo aver effettuato il login ed essere entrato nella sezione corrente, può visualizzare le prenotazioni effettuate attraverso il sistema.
  La visualizzazione può essere personalizzata attraverso determinati filtri.

- **UC009 - ModifyStoreInfo**
  L'attore manager, dopo aver effettuato il login nel sistema ed essere entrato nella corrispondente sezione, può modificare i dati del negozio. Una volta confermati i nuovi dati, riceverà un'email di conferma.

- **UC010 - AddCategory**
  L'attore, dopo aver effettuato il login ed essere entrato nella sezione corrente, può aggiungere una categoria al proprio negozio, selezionandola tra quelle disponibili.

- **UC011 - ViewAllBookings**
  L'attore, dopo aver effettuato il login ed essere entrato nella sezione corrente, può visualizzare tutte le prenotazioni effettuate presso il proprio negozio.
  La visualizzazione può essere personalizzata attraverso l'uso di determinati filtri.

Capitolo 4

Progettazione

Durante la fase di progettazione, vengono proodtti i diagrammi necessari a progettare il funzionamento del sistema.

4.1 Diagramma delle classi

Di seguito vengono riportati i diagrammi relativi al *diagramma delle classi a livello di dominio* ed al *diagramma delle classi a livello di progettazione*.

4.1.1 Diagramma delle classi a livello di dominio

Nel diagramma di dominio, vengono rappresentati solamente i concetti da implementare, senza considerare le classi software, andando così a rappresentare i concetti e le relazioni interessate.
Di deguito si roporta un'immagine relativa al diagramma delle classi a livello di dominio.

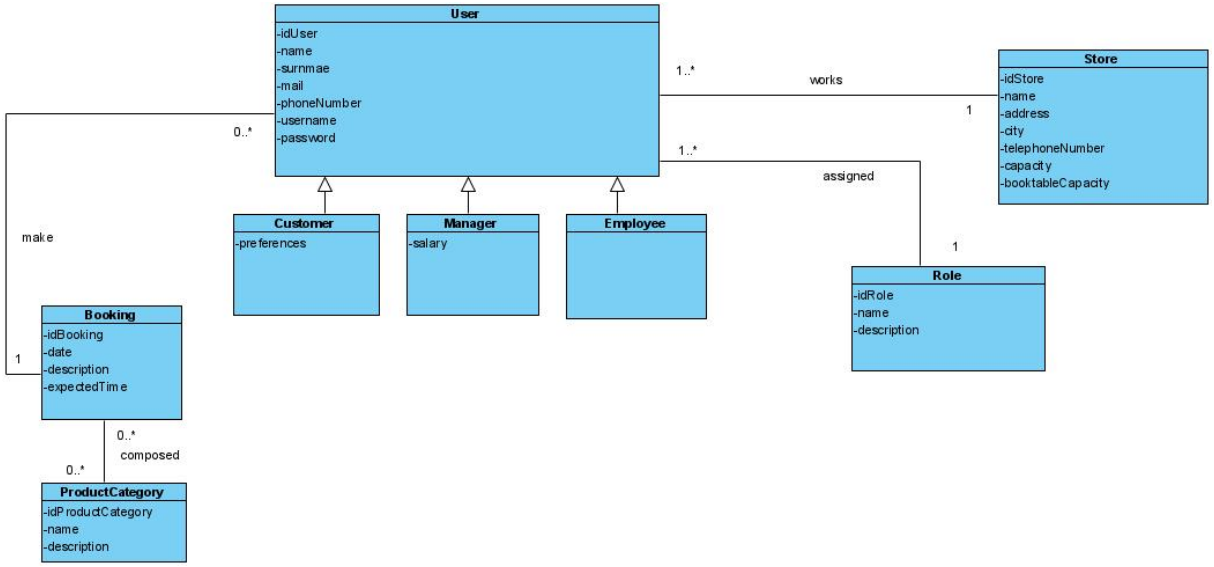

Figura 4.1: Domain Model

Come si deduce da un'analisi della figura precedente, nel modello di dominio sono rappresentati solamente i concetti da implementare durante la fase di sviluppo, oltre alle relazioni che sono presenti tra essi.
In particolare vengono evidenziate le seguenti classi da implementare a livello di dominio:

- **User** (*con Customer - Manager - Employee che rappresentanno le classi figlie*)
- **Booking**
- **ProductCategory**
- **Store**
- **Role**

4.1.2 Diagramma delle classi a livello di progettazione

Nel diagramma delle classi a livello di progettazione, vengono rappresentate, oltre ai concetti e le relazioni, le classi software, ed eventuali classi di supporto(es: classi per implementare i DAO).
Di seguito si riporta la figura rappresentante il diagramma delle classi a livello di progettazione. .

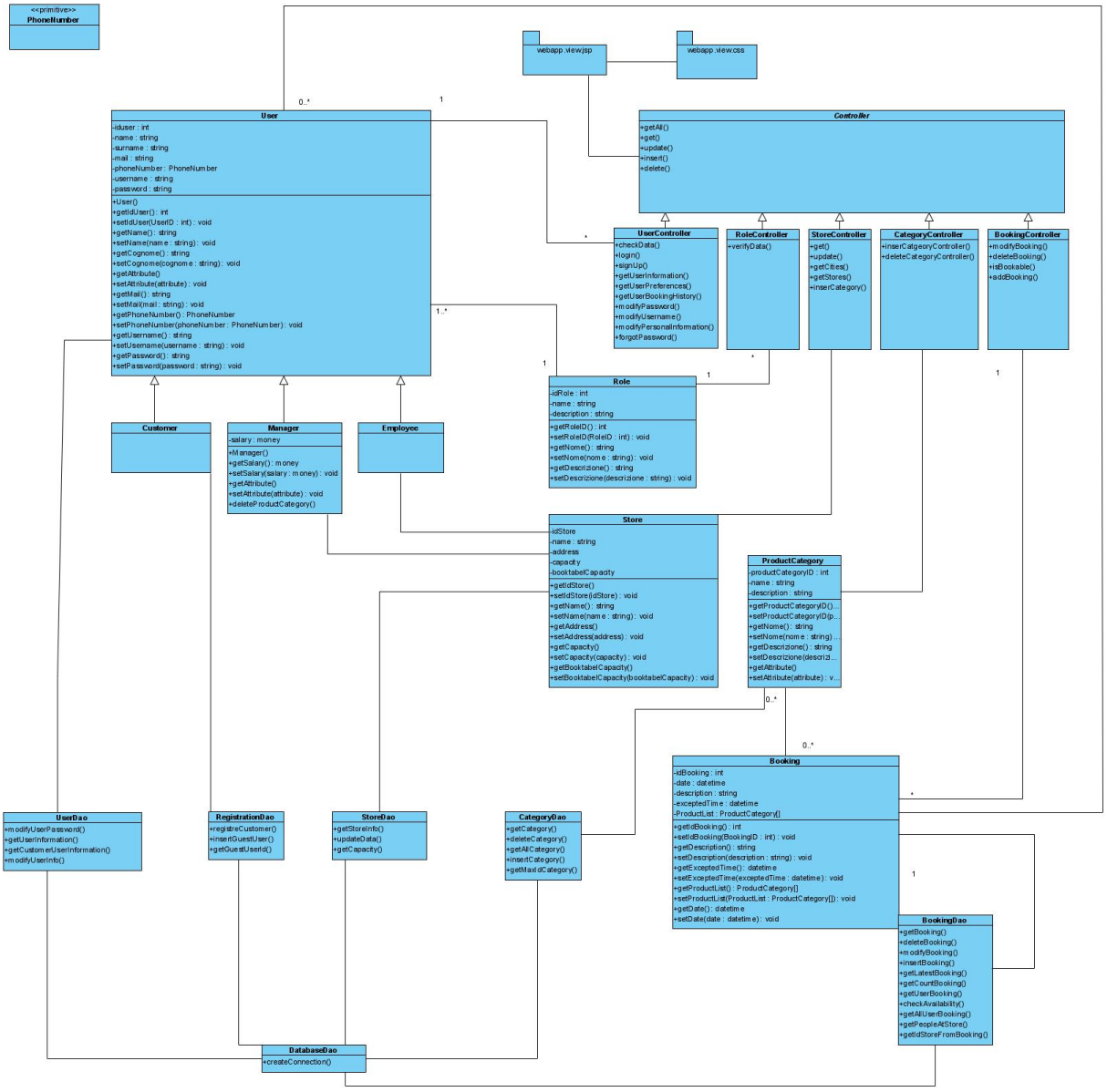

Figura 4.2: Class Diagram

A differenza del diagramma di progettazione a livello di dominio, in questo caso vengono introdotte le classi software da implementare ed eventualmente le classi di supporto (es: DAO). Oltre alla presenza delle classi di dominio, sono state introdotte classi Controller, ovvero classi che si occupano di gestire le azione che vengono svolte dall'utente, appoggiandosi su classi di suporto come i DAO.
Le classi di supporto introdotte sono:

- LoginDao
- RegistrationDao
- BookingDao
- CategoryInBookingDao
- RegistrationDao
- StoreDao
- UserDao

Le classi descritte in precedenza hanno lo scopo di accedere alle informazioni memorizzate all'interno della base dati, andando a supportare le azioni svolte dai controller.
La descrizione del pattern DAO (*Data Access Object*) è riportata nel *capitolo 5*.

4.2 SSD

Un diagramma di sequenza del sistema (SSD) è un diagramma di sequenza che mostra, per un particolare scenario di un caso d'uso, gli eventi generati da attori esterni, il loro ordine e possibili eventi inter-sistema.

Di seguito si riporta il diagramma SSD relativo alla fase di prenotazione.

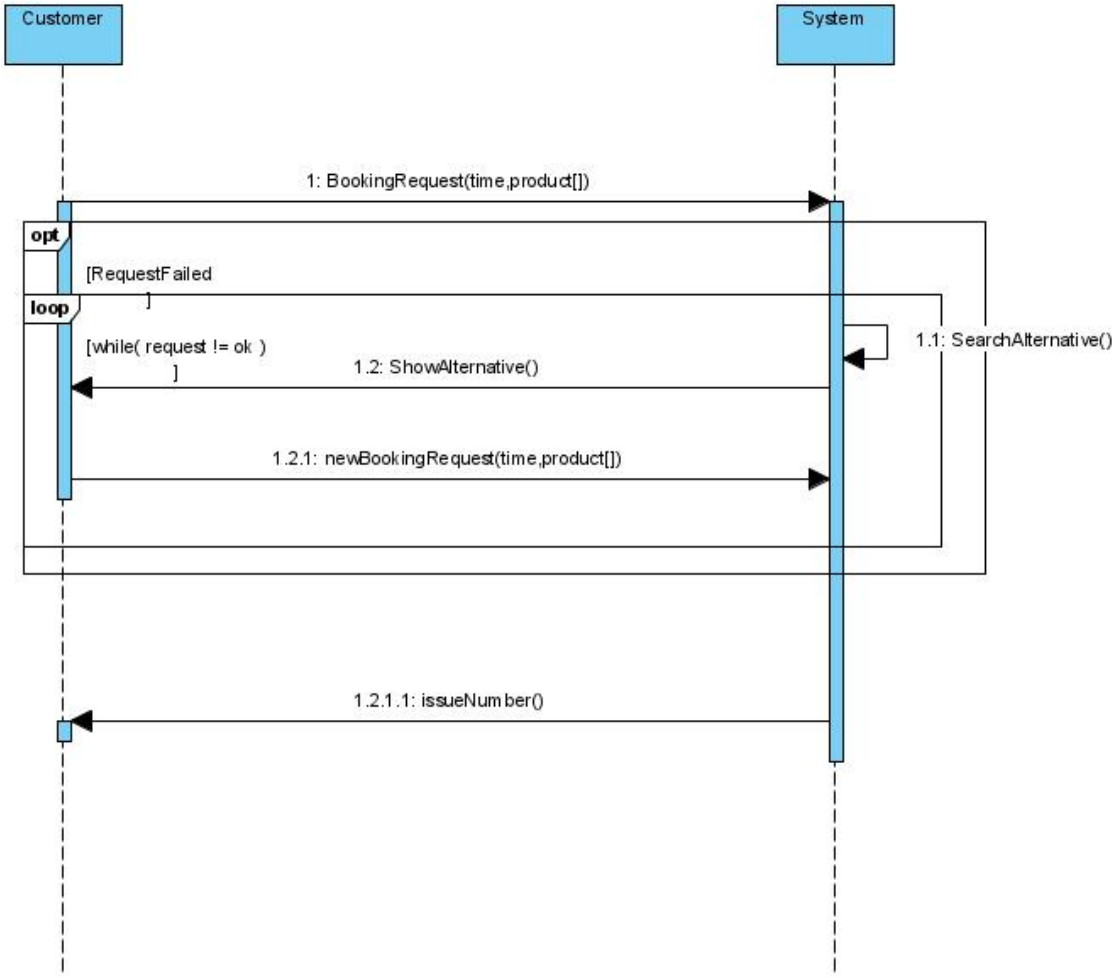

Figura 4.3: Diagramma SSD prenotazione

Analizzando la figura precedente, si può dedurre facilmente che il cliente che desidera effetture una prenotazione, effettua una richiesta al sistema, passando come parametri la data-ora e una lista di prodotti desiderati.
Ricevuta la richiesta, il sistema la elabora, andando a verificare se è possibile effettuare una prenotzione in quel determinato slot orario. Nel caso in cui questo non fosse possibile, si mostrano all'utente delle alternative in termini di orario; nel caso in cui, invece, lo slot è disponibile, si mostra all'utente un messaggio di conferma.

4.3 Diagramma di sequenza di progettazione

Il diagramma di sequenza di progettazione, mostra la sequenza di operazioni che si verificano per il raggiungimento di un obiettivo. In questo caso si descrive la sequenza affinchè l'attore utente possa effetture una prenotazione.

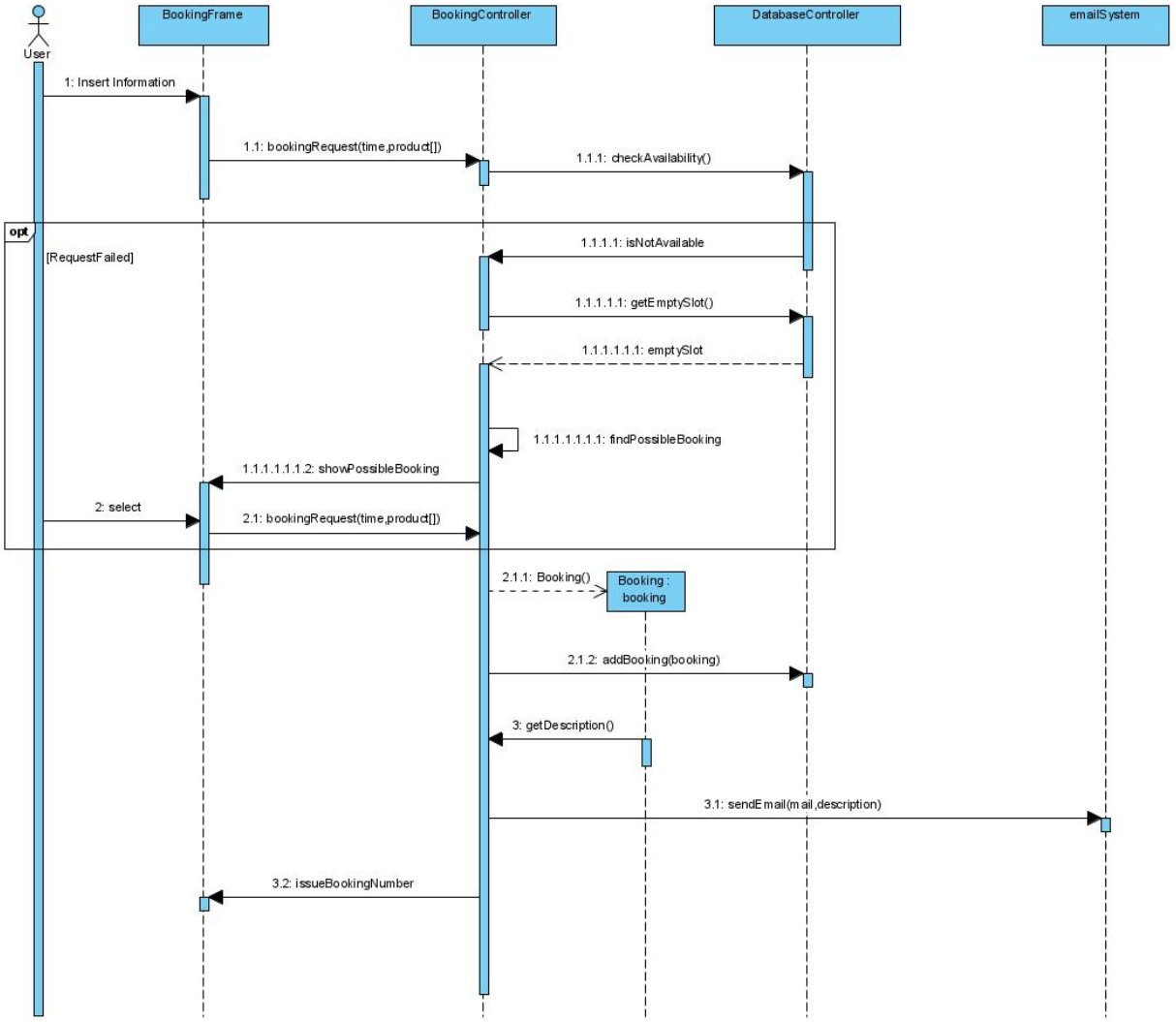

Figura 4.4: Diagramma di sequenza di progettazione - Prenotazione

4.4 Diagramma dell'architettura software

Il diagramma dell'archietttura software è utile affinchè si riesca a comprendere in dettaglio l'implementazione architetturale del sistema, andando ad evidenziare i package presenti.
In particolar modo, implementando un'architettura MVC, vi saranno tre pacakge distinti: *Model - View - Controller*.
E' presente, inoltre, un quarto package, pacakge DA=, contenente le classi di supporto implementate.
Di seguito si riporta il diagramma.

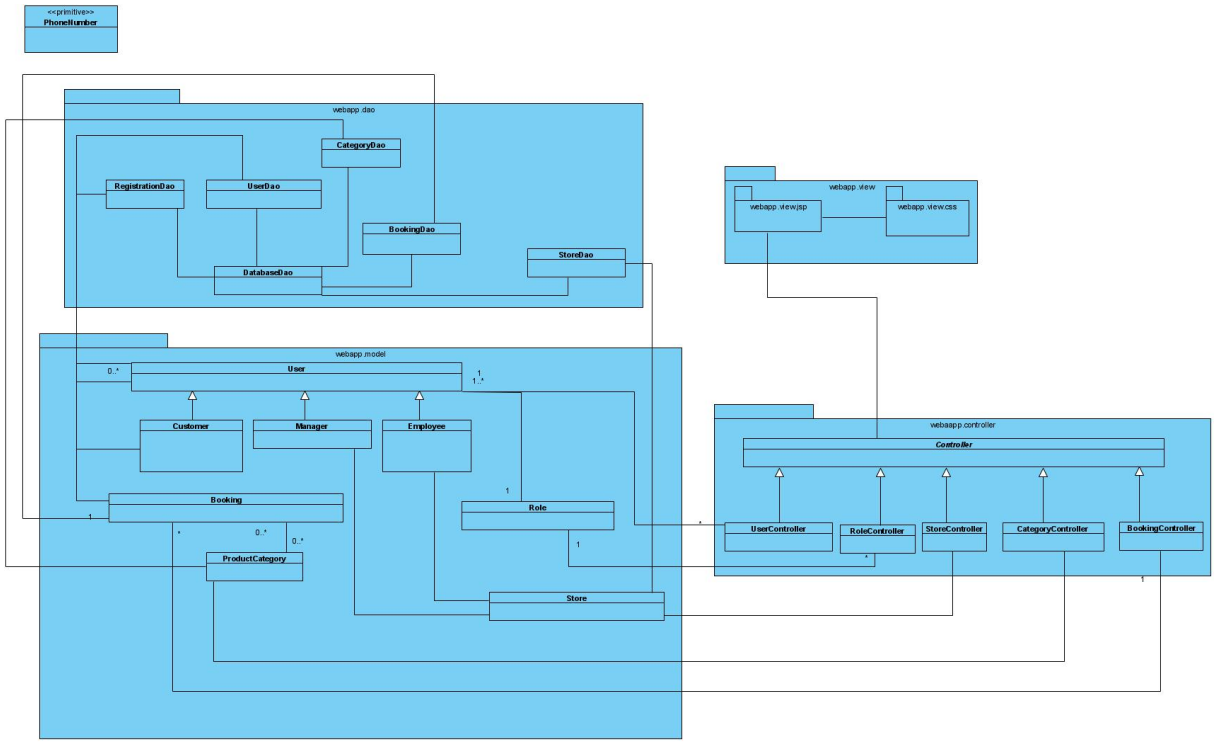

Figura 4.5: Diagramma dell'architettura software

4.5 Diagramma degli stati

I diagrammi degli stati, conosciuti anche come *State Machine Diagrams*, mostrano come gli oggetti di una classe reagiscono a determinati eventi, ovvero quale azione compie un oggetto all'accadere di un determinato evento.

Di seguito si riporta il diagramma degli stati relativo alla fase di *login*.

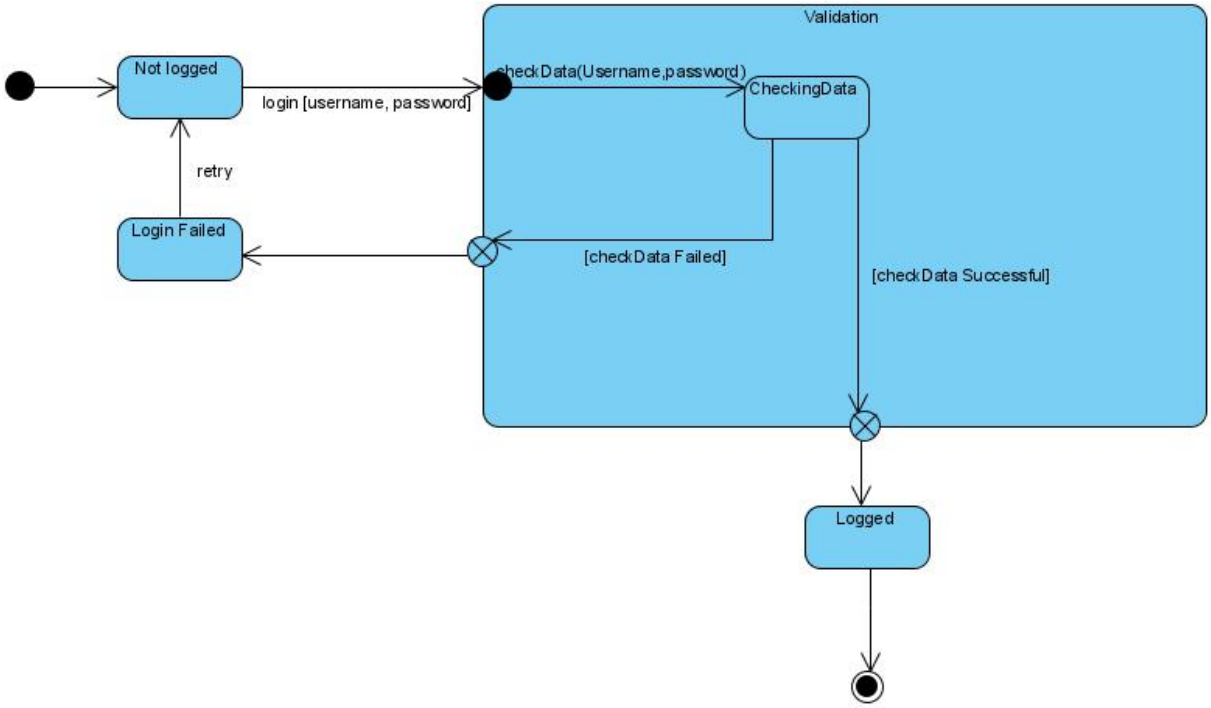

Figura 4.6: Diagramma degli stati - Login State Machine

Come si può dedurre dall'analisi della figura precedente, gli stati che si possono assumere durante la fase di *login* sono i seguenti:

- **NotLogged**: rappresenta lo stato di un utente non acora loggato nel sistema;
- **CheckingData**: rappresenta lo stato assunto dal sistema nel momento in cui controlla i dati;
- **Logged**: rappresenta lo stato che l'utente assume quando le credenziali sono corrette e la fase di *CheckingData* restituisce un esito positivo;
- **LoginFailed**: rappresenta lo stato che assumente l'utente quando le credenziali sono errate e la fase di *CheckingData* restituisce un esito negativo.

4.6 Diagramma delle attività

Il diagramma delle attività riportato in seguito, rappresenta una descrizione di come le attività sono coordinate affinchè possano servire il servizio di login.

> *Definizione: Per* **diagramma di attività** *si intende la descrizione di come le attività sono coordinate tra di loro con l'obiettivo di fornire un determinato servizio.*

Di seguito si riporta il diagramma delle attività relativo alla funzionalità descritta in precedenza.

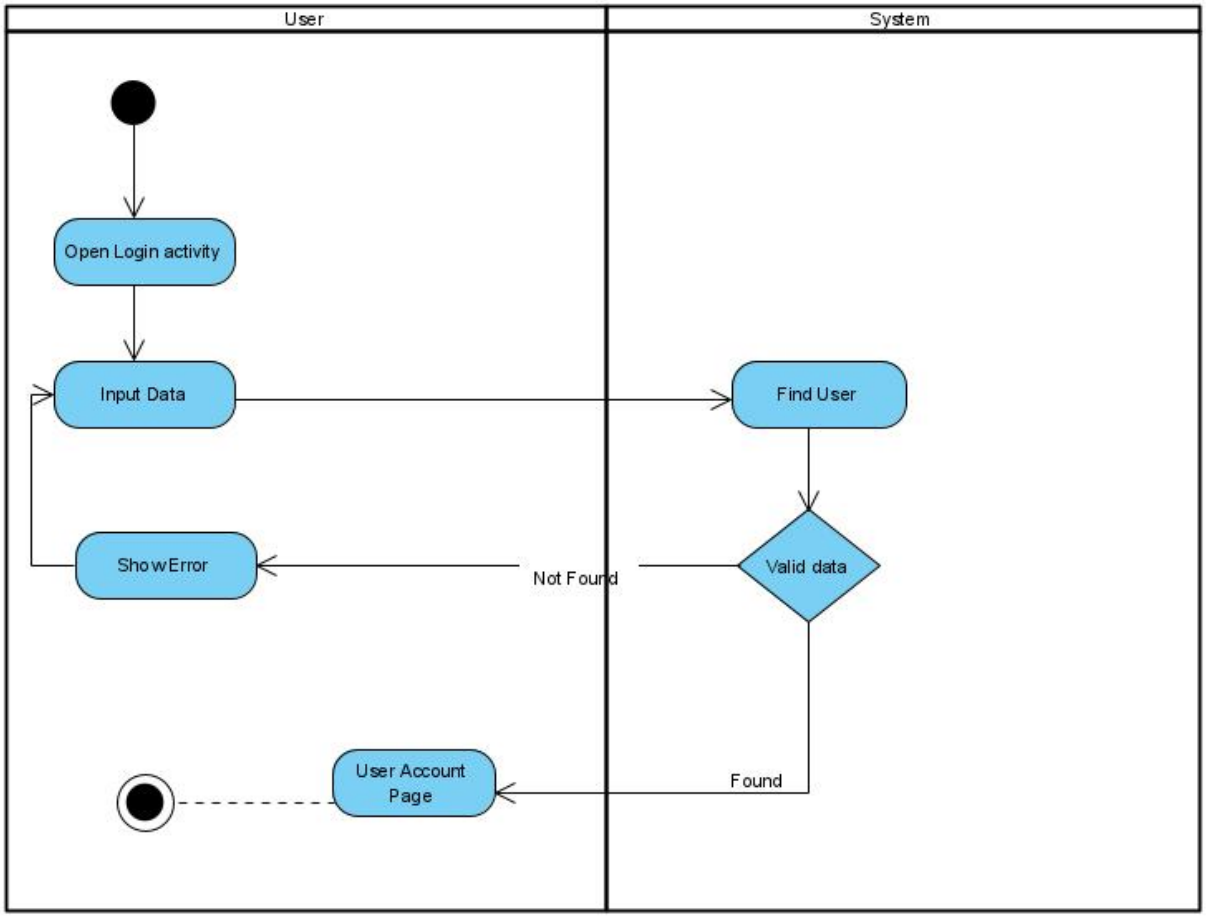

Figura 4.7: Diagramma delle attività - Login Activity

Come si deduce dalla figura, l'utente, dopo aver aperto la pagina relativa al login, inserisce i propri dati, per poi inoltrarli al sistema. Quest'ultimo, dopo averli ricevuti, controlla la loro validità, e nel caso in cui non corrispondessero a quelli memorizzati all'interno della base dati, manderà un errore all'utente; in caso contrario, inoltrerà l'utente alla corrispondente homepage, a seconda del ruolo.

Capitolo 5

Pattern

5.1 Pattern Architetturali

Il sistema *Clup* è stato pensato come un'applicazione web in grado di fornire determinati servizi all'utente finale.
Avendo questa base, il pattern architetturale in grado di sostenere il sistema è rappresentato dal Web Presentation Patterns, in particolare il pattern **MVC**(*Model-View-Controller*).
Il pattern MVC è solitamente utilizzato per lo sviluppo di sistemi in grado di dividere l'interfaccia utente dalla logica del sistema, in modo tale da separare la *business logic* dal modo in cui i dati vengono presentati all'utente finale.
In particolare si ha:

- **Model**: Rappresenta il componente principale del sistema in quanto è indipendente dall'interfaccia utente ma controlla direttamente i dati e la logica del sistema.
- **View**: Corrisponde alla rappresentazione delle informazioni ricevute dal controller in modo da renderle visibili all'utente. Contiene l'interffacia grafica per la visualizzazione dei dati.
- **Controller**: Rappresenta la business logic del sistema, ovvero esegue le istruzioni ricevute in input, in modo da soddisfare le richieste dell'utente.

Di seguito si riporta il funzionamento del pattern descritto in precedenza.

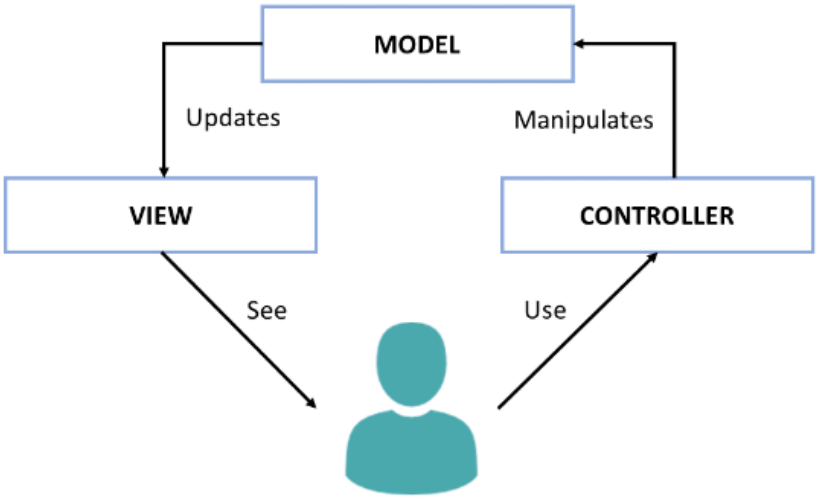

Figura 5.1: Model-View-Controller Pattern

5.2 Design Principles

Durante la fase di analisi del progetto, e sucessivamente durante la fase di progettazione, è stato individuato, e sucessivamente applicato, il Design Principles relativo all'Object Oriented systems(LSP-Liskov Substitution).
In particolar modo, il design priciples Object Oriented systems(LSP - Liskov Substitution), è stato applicato in quanto le classi figlie possno essere sostituite dalla corrispondente classe padre, evitando così la creazione di oggetti poco funzionali, in quanto la loro sotituizione non comporta delle modifiche a livello architetturale, logico ed implementativo.

Capitolo 6

Sviluppo

6.1 Scelta dei linguaggi

La fase di sviluppo necessita, in primis, la scelta dei linguaggi da utilizzare per l'implementazione del sistema.
Lo svilppo si è incentrato prevalentemente sull'uso di Java per la parte back-end, e di HTML, con l'inclusione di alcuni CSS, per l'implementazione della parte front-end.

6.2 Analisi di sviluppo

Durante la fase di sviluppo del progetto, si è reso necessario ricorrere a due strumenti in grado di evidenziarne le criticità, in modo da risolverle migliroando il quality gate complessivo.

6.2.1 SonarCloud

SonarCloud è un tool in grado di evidenziare la presenza di eventuali *Bug*, *Vulnerabilities*, *Security Hotspots* e *Code Smells*.
L'uso dello strumento descritto, ha permesso di portare al termine il progetto con un quality gate ottimale, senza la presenza di criticità che avrebbero potuto ridurre la qualità del progetto.
Di seguito si riporta la schermata di SonarCloud contenente il livello raggiunto dall'applicazione.

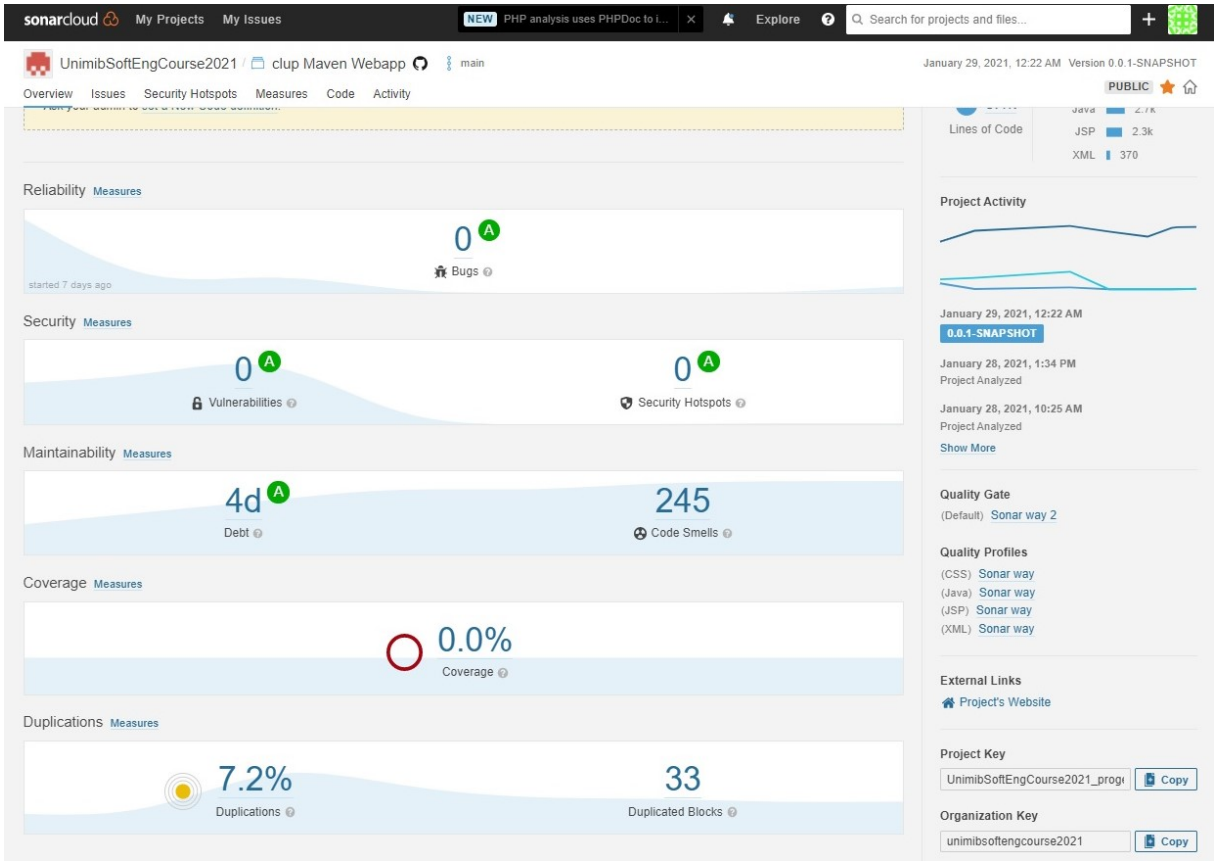

Figura 6.1: Analisi del codice - SonarCloud

6.2.2 Understand

L'uso di Understand ha permesso di evidenziare la presenza di ciriticità relativa agli code smell e la presenza di eventuali antipattern strutturali legati a problemi di dipendenza. Di seguito si riporta la schermata principale del tool, in modo da evidenziare i principali risultati.

Figura 6.2: Understand