



Computer Engineering – Università del Salento Software  
Engineering Course

Software developed for 2019-2020 academic year

# **PROJECT DOCUMENTATION**

# **MyAirBnb.com**

Professore: Luca Mainetti

Capoccia Gianmarco  
Vedruccio Andrea

## INDICE

INTRODUZIONE .....	3
TECNOLOGIE UTILIZZATE.....	4
ANALISI DEI REQUISITI .....	5
CASI D'USO DETTAGLIATI .....	6
DIAGRAMMA DEI CASI D'USO .....	11
PROGR. CONCETTUALE E LOGICA DELLA BASE DI DATI .....	12
PROGETTAZIONE UML DELL'ARCHITETTURA DEL SOFTWARE .....	13
DIAGRAMMA DELLE DIPENDENZE .....	13
DIAGRAMMA DI SEQUENZA.....	14
DESIGN PATTERNS .....	15
TEST COVERAGE AND MCCABE CYCLOMATIC COMPLEXITY.....	17
SOFTWARE TESTING .....	18
SPRINT BACKLOG .....	18
BURNDOWN CHART .....	19

## **INTRODUZIONE**

“MyAirBnb.com” è un software progettato per la gestione di un sistema di prenotazione di beni e servizi, come : auto, alloggi ed escursioni.

Il sistema proposto, composto da una piattaforma web e mobile, presenta uno stile sobrio e lineare che trova nelle sue interfacce semplici e intuitive, il suo maggiore punto di forza, permettendo un apprendimento rapido e un utilizzo piacevole.

Questa sua caratteristica, rende il sistema fruibile a un ampio bacino di utenti, i quali potranno usufruire del sistema a partire dalla modalità ‘guest’, senza dunque una preventiva registrazione.

L’utente guest durante la sua esperienza , potrà godere di numerosi annunci e navigare fra vantaggiose offerte, scrutando tra quelle più vicine alle proprie esigenze, favorito da un form di ricerca.

Una volta individuato il bene o servizio desiderato, l’utente potrà procedere con facilità alla sua prenotazione, non prima però di essersi registrato.

La registrazione consentirà all’ utente di usufruire di numerose funzionalità aggiuntive, che completeranno la sua esperienza.

Tra queste, l’utente registrato , potrà :

- commentare un bene o servizio , indicandone un indice di gradimento (che va da una a cinque stelle)
- rispondere a commenti relativi alle proprie proposte
- effettuare una o più prenotazioni
- affittare i propri beni e servizi
- controllare e/o modificare le proprie proposte
- controllare le proprie prenotazioni
- restare aggiornato riguardo le avvenute prenotazioni e/o inserimenti di nuove proposte , attraverso un efficace sistema di notifica

Ogni proposta sarà preventivamente verificata da un amministratore, il quale potrà decidere quali proposte saranno idonee alla pubblicazione e quali no .

All’amministratore , verrà infine data la possibilità di aggiungere nuove tipologie di servizi prenotabili, in modo da rendere il sistema dinamico e sempre all’avanguardia.

## **TECNOLOGIE UTILIZZATE**

Per la programmazione del software proposto sono stati utilizzati i seguenti frameworks e strumenti di lavoro :

- **AngularJS** : per la programmazione dell'interfaccia Web;
- **ionic** : per la programmazione dell'interfaccia Mobile;
- **Visual Studio** : Ambiente di sviluppo per la programmazione del front-end;
- **Spring Tool Suite** : Framework per la programmazione del back-end;
- **Php my admin** : per la costruzione del database;
- **Postman** : per testare le richieste http inviate dal back-end;
- **JUnit framework** : Per i test compiuti;
- **UML** : come linguaggio di modellazione per l'analisi dei requisiti ;

## **ANALISI DEI REQUISITI**

*In questa fase analizziamo le sequenze di passi che descrivono le interazioni tra utente e sistema (scenario), partendo dalla determinazione dei ruoli di utenti che interagiscono con il sistema (attori) e continuando con la focalizzazione dei goal (scopi finali degli utenti-attori). Ciò ci permetterà l'approfondimento dei requisiti funzionali del sistema attraverso la tecnica dei casi d'uso.*

### **Attori:**

- Amministratore
- Utente proponente
- Utente acquirente
- Guest

### **Goal**

- **Amministratore**
  - Creazione nuova tipologia di servizi
  - Gestione delle proposte
- **Utente proponente**
  - Proporre beni e servizi di sua proprietà
  - Gestione delle proprie proposte ( modifica , cancellazione )
  - Rispondere ai commenti relativi alle proprie proposte
  - Verificare le proprie notifiche
- **Utente acquirente**
  - Prenotazione beni e servizi
  - Visualizzazione proprie prenotazioni
  - Lasciare feedback sottoforma di commenti e/o indici di gradimento
  - Verificare le proprie notifiche
- **Guest**
  - Navigazione nella parte pubblica
  - Registrazione al sistema

## **CASI D'USO DETTAGLIATI**

Analizziamo nel dettaglio i vari passi che compongono i casi d'uso precedentemente trovati, sottolineando eventuali pre-condizioni (condizioni necessarie per l'esecuzione di un caso d'uso), post-condizioni (cosa accadrà alla fine del caso d'uso) e estensioni (scenari alternativi a quello principale).

(LATO MOBILE)

### **1. GUEST NAVIGA NELLA PARTE PUBBLICA**

PRE Essere un utente guest

1. Utente preme il bottone "Accedi come Guest"

2. Sistema mostra Homepage

POST Utente Guest è abilitato alla navigazione tra i beni e servizi del sistema

(LATO MOBILE)

### **2. GUEST SI REGISTRA AL SISTEMA**

PRE Non essere ancora registrato al sistema

1. Utente guest preme il bottone "Registration"
2. Sistema predispone form di registrazione
3. Utente compila form e preme "Register"
4. Sistema mostra messaggio di avvenuta registrazione e mostra interfaccia login

ESTENSIONE 3.a ) Un campo del form non viene compilato

- 3.1. Sistema mostra un messaggio di errore e ripropone form di registrazione
- 3.2. Compila campi mancanti e preme "Register"
- 3.3. Sistema mostra messaggio di avvenuta registrazione e mostra interfaccia login

POST Utente è registrato al sistema

(LATO MOBILE)

### 3. UTENTE SI CONNETTE AL SISTEMA

PRE Essere un utente registrato al sistema

1. Utente preme il bottone “Login”
2. Sistema predispone form di Login
3. Utente compila form e preme “Sign In”
4. Sistema mostra Homepage

POST Utente accede al sistema

(LATO WEB)

### 4. AMMINISTRATORE GENERA NUOVA TIPOLOGIA DI SERVIZI

PRE Essere un amministratore registrato al sistema

1. Amministratore si connette al sistema
2. Sistema mostra Homepage
3. Amministratore preme “ Crea Servizio”
4. Sistema predispone form per la creazione di una nuova tipologia di servizio
5. Amministratore compila form e preme “Conferma”
6. Sistema mostra messaggio di avvenuta creazione

POST Crea una nuova tipologia di servizi

(LATO WEB)

### 5. AMMINISTRATORE GESTISCE PROPOSTE

PRE Essere un amministratore registrato al sistema

1. Amministratore si connette al sistema
2. Sistema mostra le nuove proposte ricevute
3. Amministratore preme il tasto “Evadi” alla destra della proposta ricevuta
4. Sistema evade proposta e invia notifica all’utente acquirente che ha effettuato la proposta

ESTENSIONE 3.a ) Amministratore decide di non evadere proposta

- 3.1 Amministratore preme il tasto “Rifiuta” alla destra della proposta ricevuta
- 3.2 Sistema elimina proposta e invia notifica all’utente acquirente che ha effettuato la proposta

POST Proposta cambia il suo stato in “evasa” e risulta disponibile a tutti gli utenti

(LATO WEB)

## **6. UTENTE PROPONENTE PROPONE BENI E/O SERVIZI**

PRE Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente preme “Registra bene/servizio”
4. Sistema predispone form per la proposta di un alloggio
5. Utente compila tutti i campi e preme “Conferma”
6. Sistema mostra messaggio di conferma

ESTENSIONE 5.a) Utente proponente intende proporre un’auto

- 5.1. Utente preme “Auto”
- 5.2 Sistema predispone form per la proposta di una nuova auto
- 5.3. Utente compila tutti i campi e preme “Conferma”
- 5.4. Sistema mostra messaggio di conferma

POST Nuova proposta di Alloggio

(LATO WEB)

## **7. UTENTE PROPONENTE GESTISCE LE PROPRIE PROPOSTE**

PRE Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente preme “Le mie proposte”
4. Sistema mostra lista con tutte le proposte effettuate
5. Utente preme “Modifica” nella parte inferiore della card della proposta che intende modificare
6. Sistema predispone form con i campi che è possibile modificare
7. Utente modifica i campi desiderati e preme “Conferma”
8. Sistema mostra messaggio di avvenuta modifica

POST La proposta fatta è stata modificata



(LATO MOBILE)

## **8. UTENTE PROPONENTE RISPONDE AI COMMENTI SOTTO UNA PROPOSTA**

PRE Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente preme il bottone “Le mie proposte” (contenuto nello slide menu)
4. Sistema mostra le proposte effettuate
5. Utente seleziona Proposta di cui intende visualizzare i commenti
6. Sistema mostra scheda dettagliata della proposta con gli eventuali commenti ricevuti
7. Utente inserisce risposta al commento e preme il tasto rispondi , sotto il commento a cui intende rispondere
8. Sistema pubblica risposta al commento

POST Utente proponente ha risposto a un commento

(LATO MOBILE)

## **9. UTENTE ACQUIRENTE PRENOTA UN BENE/SERVIZIO**

PRE Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente compila form di ricerca di un alloggio
4. Sistema mostra risultati relativi alla ricerca
5. Utente seleziona card del risultato a cui è interessato
6. Sistema mostra scheda dettagliata dell'alloggio selezionato
7. Utente preme “Prenota ora”
8. Sistema mostra scheda di riepilogo
9. Utente preme “Prenota”
10. Sistema mostra messaggio di avvenuta prenotazione

ESTENSIONE 3.a) Utente intende prenotare un'escursione

- 3.1. Utente sceglie Escursione, tramite una combo box
- 3.2. Sistema mostra form di ricerca delle escursioni
- 3.3. Utente compila form di ricerca
- 3.4. Sistema mostra risultati relativi alla ricerca

- 3.5. Utente seleziona card del risultato a cui è interessato
- 3.6. Sistema mostra scheda dettagliata dell'escursione selezionata
- 3.7. Utente acquirente preme "Prenota ora"
- 3.8. Sistema mostra scheda di riepilogo
- 3.9. Utente preme "Prenota"
- 3.10. Sistema mostra messaggio di avvenuta prenotazione e invia notifica all'utente proponente

**POST** Utente effettua prenotazione di un bene o di un servizio

(LATO MOBILE)

## **10. UTENTE ACQUIRENTE COMMENTA UN BENE/SERVIZIO**

**PRE** Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente seleziona proposta a cui è interessato
4. Sistema mostra dettagli della proposta selezionata
5. Utente inserisce un commento nell'apposito spazio , esprime un indice di gradimento con una valutazione da 1 a 5 stelle e preme "invia commento"
6. Sistema inserisce commento

**ESTENSIONE** 6.a) Utente scorda di inserire indice di gradimento

- 6.1. Sistema invita utente a inserire indice di gradimento

**POST** Utente esprime un commento e un indice di gradimento in merito a una proposta

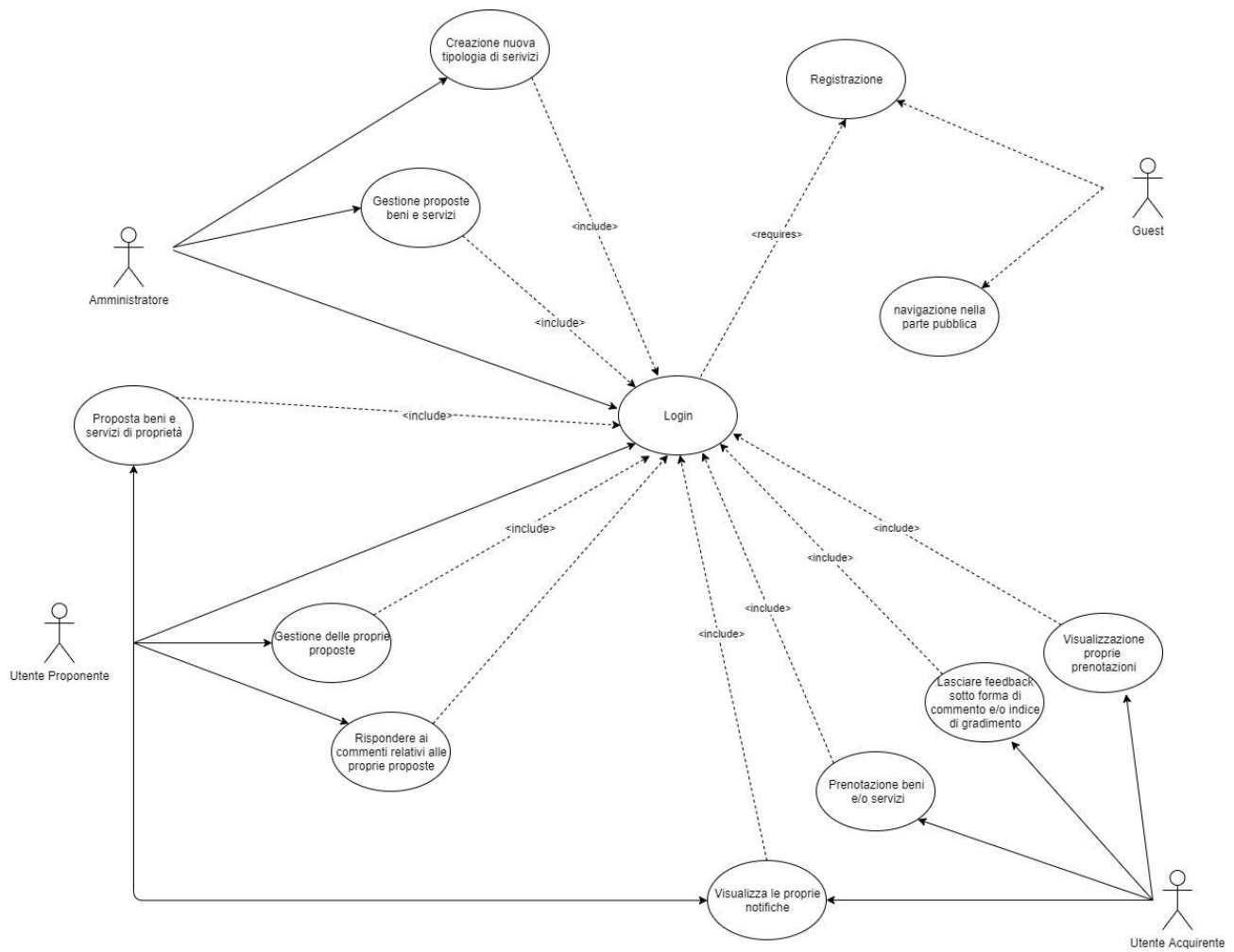
## **11. UTENTE (GENERIC) CONTROLLA LE PROPRIE NOTIFICHE**

**PRE** Essere un utente registrato al sistema

1. Utente si connette al sistema
2. Sistema mostra Homepage
3. Utente preme sull'icona delle notifiche (campanellina posta in alto a sinistra)
4. Sistema mostra notifiche ricevute

**POST** Utente Visualizza le proprie notifiche

## DIAGRAMMA DEI CASI D'USO



## MODELLO ER



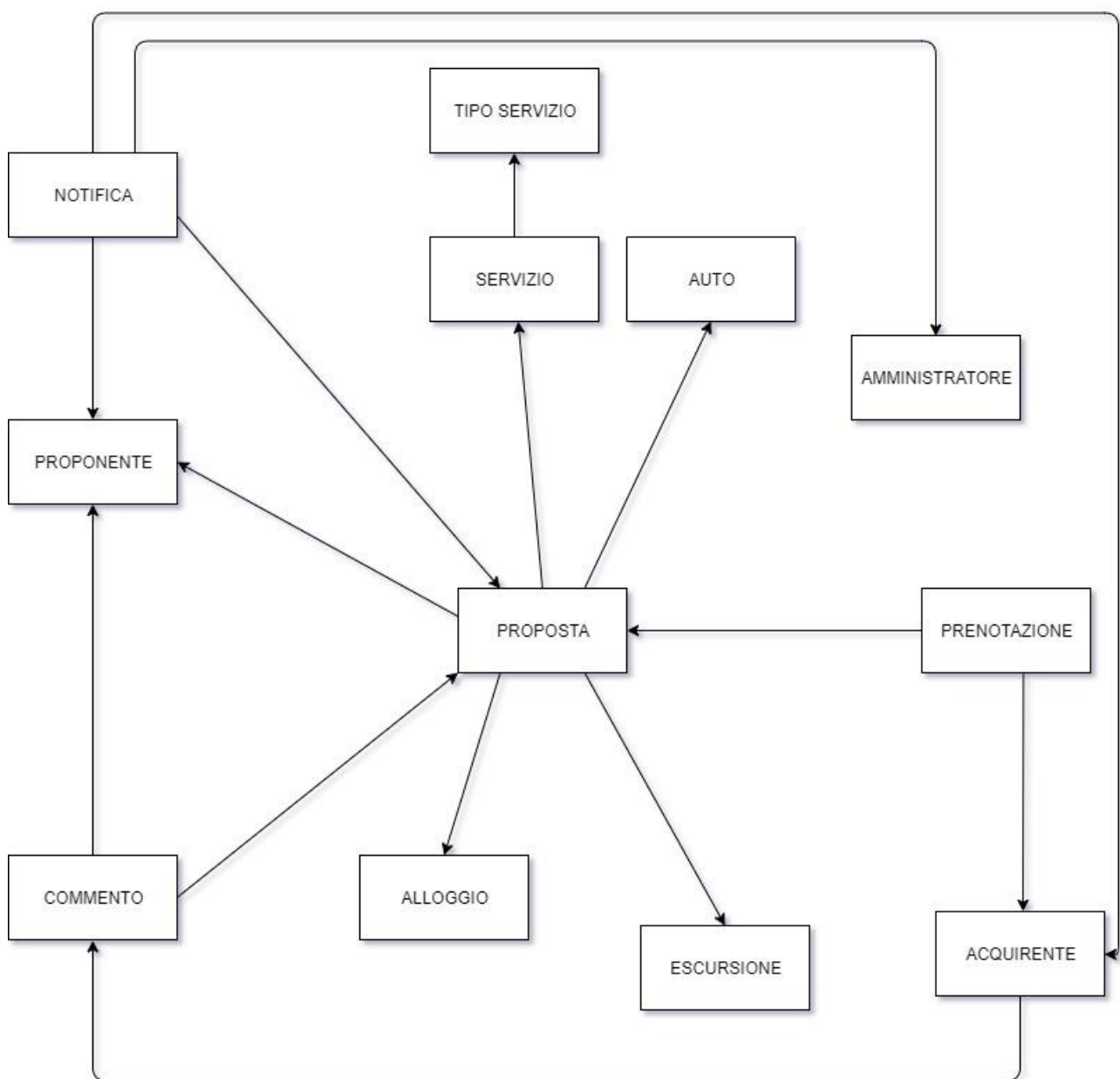
## PROGETTAZIONE UML DELL'ARCHITETTURA DEL SOFTWARE

Utilizziamo il linguaggio visuale UML (Unified Modeling Language) per creare una notazione semi formale del nostro software.

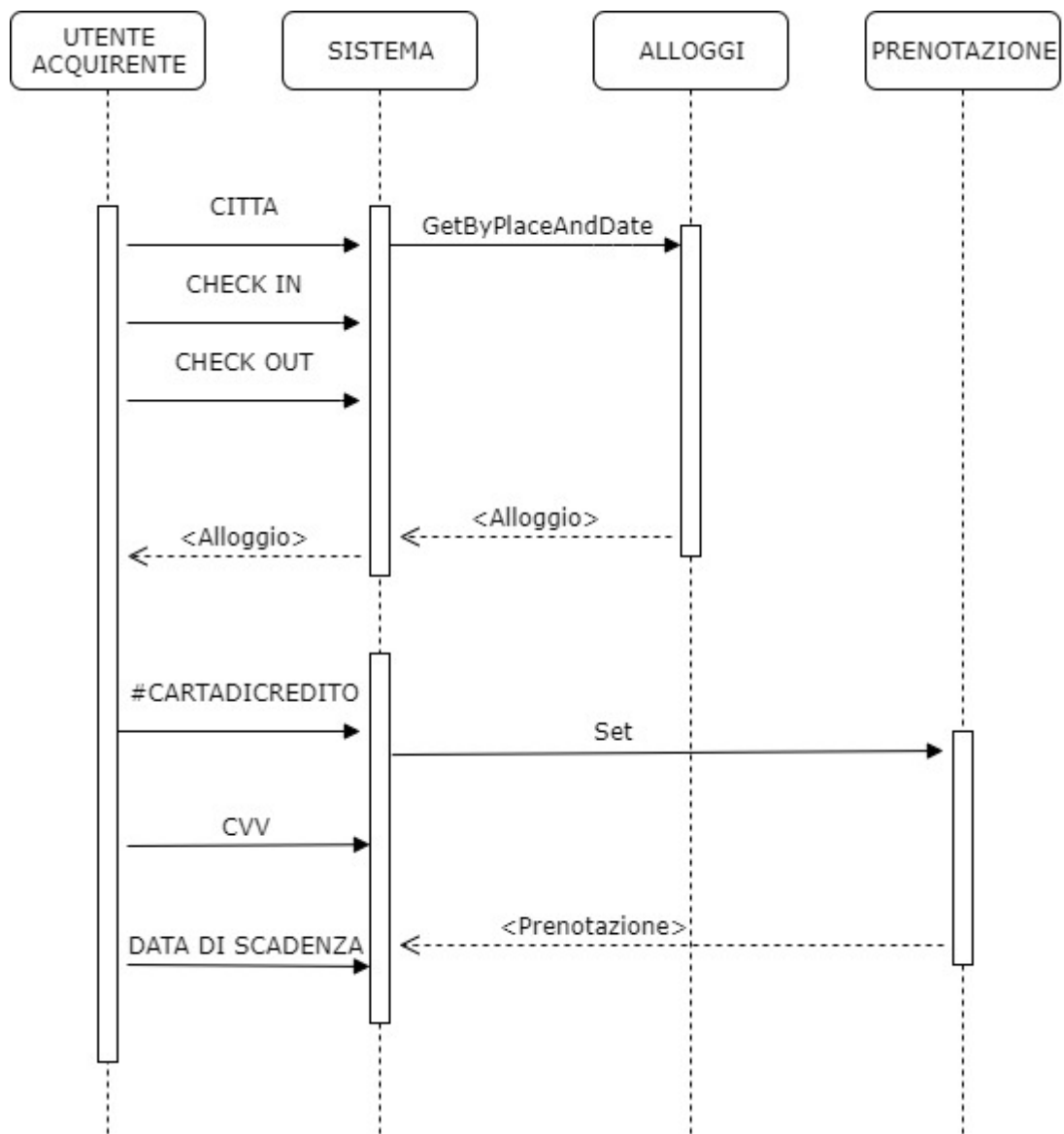
L'analisi del progetto attraverso questi diagrammi strutturali consente di visualizzare, specificare, costruire e documentare gli aspetti statici di un sistema.

Di seguito è illustrato il diagramma delle classi (o dipendenze).

### DIAGRAMMA DELLE DIPENDENZE



## DIAGRAMMA DI SEQUENZA : UTENTE PRENOTA ALLOGGIO



## **DESIGN PATTERNS**

I problemi incontrati nello sviluppare il software sono stati spesso ricorrenti e prevedibili. Le soluzioni a questi problemi sono state date dai design pattern, schemi utilizzabili nel progetto di un sistema che hanno permesso quindi di non inventare da capo soluzioni ai problemi già risolti, ma di utilizzare delle strutture di provata

I Design Patterns utilizzati nel nostro software sono i seguenti.

### **MODEL-VIEW-CONTROLLER**

Lo schema che abbiamo identificato è esattamente quello proposto dal pattern MVC. In particolare:

Model: contiene i metodi di accesso ai dati.

View: si occupa di visualizzare i dati all'utente e gestisce l'interazione fra quest'ultimo e l'infrastruttura sottostante.

Controller: riceve i comandi dell'utente attraverso il View e reagisce eseguendo delle operazioni che possono interessare il Model e che portano generalmente ad un cambiamento di stato del View.

Questo design pattern è stato utilizzato sia lato back-end che front-end

### **OBSERVER**

Il pattern Observer permette di definire una dipendenza uno a molti fra oggetti, in modo tale che se un oggetto cambia il suo stato interno, ciascuno degli oggetti dipendenti da esso viene notificato e aggiornato automaticamente. E' utilizzato dal front-end in ogni chiamata al back-end che ritorna un oggetto <Observable> .

### **DECORATOR**

Il design pattern **Decorator** fornisce un'alternativa flessibile all'ereditarietà per estendere la funzionalità degli oggetti. Tale pattern consente di arricchire dinamicamente, a **run-time**, un oggetto con nuove funzionalità. Questo pattern è stato utilizzato nello sviluppo del lato front-end dell'applicazione attraverso le annotazioni "@Component" e "@ViewChild".

Queste annotazioni inserite nelle classi di typescript implementano il template della pagina scritto in html e il form della pagina. Attraverso questa tecnica è possibile aggiungere nuove proprietà alla classe typescript modificando dinamicamente i modelli dell'HTML.

## INVERSION OF CONTROL

“L’Inversion of Control è un principio architetturale, basato sul concetto di invertire il controllo del flusso di sistema (Control Flow) rispetto alla programmazione tradizionale. Questo principio è molto utilizzato nei framework e ne rappresenta una delle caratteristiche basilari che li distingue dalle API.

In questo progetto è usato in ogni classe .ts lato front-end in modo da gestire la forte dipendenza tra la classe e oggetti esterni come il Router, ActivatedRoute e i Services

## STRATEGY

Si tratta di un pattern comportamentale basato su oggetti e viene utilizzato per definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Nel nostro progetto, è usato nelle classi “Repository” Java. Queste classi sono interfacce pubbliche che supportano diverse implementazioni dello stesso metodo.

Una prima implementazione implica l’annotazione del metodo con “@Query” e la sua attuazione (Es. `NotificaRepository`) tramite una query SQL esplicita, mentre una seconda implementazione implica che il metodo sia annotato da “Spring Jpa” e da implementare attraverso le funzioni di Spring.
































## FRONT CONTROLLER

Il front controller design pattern è usato per costruire un generico template per la view delle pagine e un’unica pagina di avvio attraverso le classi “index.html”, “styles.scss” and “approuting.ts” dove quest’ultima gestisce la visualizzazione delle nuove pagine, in base alle scelte effettuate dall’utente.

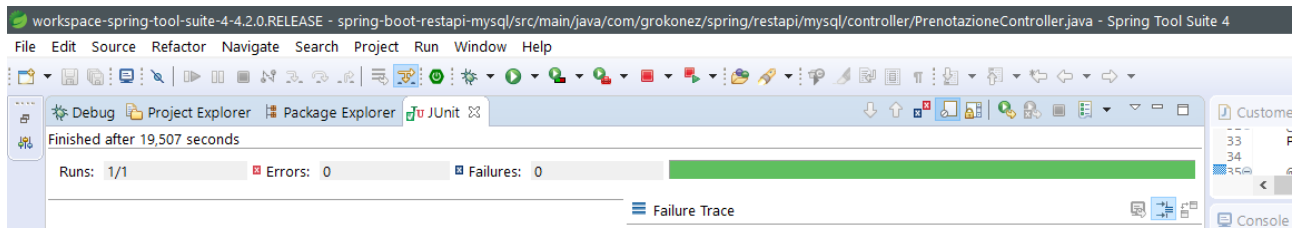


## TEST COVERAGE AND MCCABE CYCLOMATIC COMPLEXITY

(Plugin utilizzato ECL EMMA)

spring-boot-restapi-mysql (21-gen-2020 16.26.11)	
Element	Coverage
▼ spring-boot-restapi-mysql	 46,4 %
▼ src/test/java	100,0 %
> com.grokonez.spring.restapi.mysql	 100,0 %
▼ src/main/java	 35,9 %
▼ com.grokonez.spring.restapi.mysql	50,0 %
> SpringBootRestApiMySQLApplication.java	 50,0 %
▼ com.grokonez.spring.restapi.mysql.model1	 33,3 %
> Customer.java	 16,7 %
> Escursione.java	 50,0 %
> Servizi.java	 33,3 %
▼ com.grokonez.spring.restapi.mysql.controller	 25,4 %
> PrenotazioneController.java	 24,2 %
> CommentoController.java	 24,2 %
> CommentoRispostaController.java	 3,3 %
> Appartamento1Controller2.java	 56,7 %
> AppartamentoController.java	 56,7 %
> TiposervizioController.java	 60,0 %
> NotificaController.java	 5,3 %
> UserController.java	 7,1 %
> AutoController.java	 6,2 %
> EscursioneController.java	 7,5 %
> CustomerController.java	 28,3 %
> ServiziController.java	 8,3 %
▼ com.grokonez.spring.restapi.mysql.model	 25,6 %
> Notifica.java	 12,5 %
> Tiposervizio.java	 12,5 %
> Commento_risposta.java	 25,0 %
> Commento.java	 25,6 %
> User.java	 7,7 %
> Prenotazione.java	 16,7 %
> Appartamento2.java	 55,0 %
> Appartamento.java	 8,3 %
> Auto.java	 7,1 %

## SOFTWARE TESTING



## SPRINT BACKLOG

	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
	25/11 - 1/12	2/12 - 8/12	9/12 - 16/12	17/12 - 24/12	27/12 - 4/01	4/01 - 11/01	12/01 - 19/01	20/01 - 27/01
Analisi dei requisiti, casi d'uso, documentazione	43	4	2	//	//	//	2	24
Implemetazione Database	24	8	7	//	//	//	//	//
Sviluppo Back end	8	48	36	5	//	8	//	1
Sviluppo Front end	//	6	25	48	74	68	54	33
Testing	//	3	5	21	3	//	3	12
Revisione codice	2	2	2	2	3	2	8	3
Consultazione Online	1	5	1	6	1	1	1	//

## **BURNDOWN CHART**

