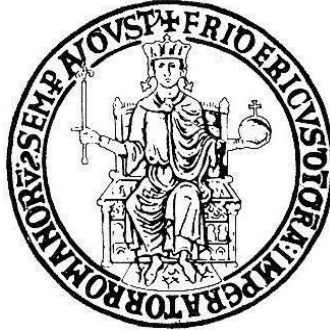


Università degli Studi di Napoli Federico II
Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione



Corso di Laurea in Informatica
Insegnamento di Ingegneria del Software
Anno accademico 2019/2020

Specifica, progettazione, implementazione e
validazione del Sistema Informativo “Consiglia
Viaggi2019”

Autori:

Ivan Capasso N86002587

Docente:

Sergio di Martino

Sommario

Descrizione del progetto	3
Introduzione	3
Documento dei Requisiti Software	4
Modello Funzionale	4
Diagramma dei casi d'uso.....	8
Mockup.....	9
Tabelle di Cockburn.....	16
Modello di dominio:.....	28
Class diagram.....	28
Sequence Diagram	35
Diagramma di stato di analisi: Ricerca Strutture.....	40
Diagramma di attività: Registrazione utente	41
Pianificazione dettagliata dell'attività.....	42
Documento di Design del sistema	43
Analisi dell'architettura.....	43
Diagramma delle classi di design	47
CRC Cards.....	49
Diagrammi di sequenza di design	58
Documento di Testing del sistema	60
Test Plan per System Testing.....	60
Codice xUnit per unit testing di 2 metodi.....	69

Capitolo 1

Descrizione del progetto

Introduzione

Il progetto "Consiglia Viaggi 2019" consiste nella realizzazione di un software il cui scopo è quello di permettere agli utenti che utilizzano l'applicativo di visionare e recensire delle strutture.

La ricerca o visualizzazione delle strutture avviene tramite l'utilizzo di una mappa, oppure può avvenire per nome mediante una apposita barra di ricerca oppure ancora selezionando una categoria di strutture dalla home.

Per rendere più veloce la ricerca, c'è la possibilità di aggiungere dei filtri in base alle stelle o alla distanza della struttura e selezionare il tipo di ordinamento dei risultati. La mappa, invece, implementa dei filtri che permettono di decidere la distanza massima rispetto alla posizione attuale dell'utente.

Una volta selezionata una struttura oltre a visualizzarne le informazioni si potranno visualizzare anche le recensioni inserite dagli utenti, anch'esse filtrabili o ordinabili.

Si possono scrivere nuove recensioni indicando un numero di stelle comprese tra 1 e 5 e una descrizione di almeno 100 caratteri ma per farlo è necessario essere autenticati. Invece, per tutte le altre funzioni precedentemente descritte, non è richiesta la creazione di un account ma si può accedere all'applicazione come "visitatore".

L'utente per registrarsi dovrà inserire: un nome utente ed una password una mail e il nome reale dell'utente.

Capitolo 2

Documento dei Requisiti Software

Modello Funzionale

Iniziamo identificando i requisiti: essi sono composti da un identificativo, un nome ed una descrizione, ritorneranno particolarmente utili nella fase di testing e nelle tabelle di Cockburn.

Requisiti funzionali

Requisiti utente: ottenuti dopo un brainstorming con gli stakeholders

ID	RE01
Nome	Ricerca e visualizzazione di una struttura
Descrizione	<p>L'utente deve avere la possibilità di cercare e visualizzare qualsiasi struttura presente sul software, a prescindere se esso sia autenticato o meno.</p> <p>Per rendere la ricerca più fluida e veloce, il sistema deve permettere di applicare filtri e ordinamenti sulle strutture.</p>

ID	RE02
Nome	Visualizzazione recensioni per una struttura
Descrizione	<p>Una volta aperta una struttura l'utente deve poter visualizzare l'elenco delle recensioni scritte da altri utenti riguardo quella struttura.</p> <p>L'utente può applicare dei filtri e ordinamenti, per esempio in base al numero di stelle, per visionare al meglio le recensioni.</p>

ID	RE03
Nome	Visualizzazione delle strutture su mappa
Descrizione	<p>L'utente deve avere la possibilità di visualizzare qualsiasi struttura presente sull'applicativo tramite una mappa; A prescindere se l'utente sia autenticato.</p> <p>L'utente può effettuare dei filtri sulla mappa in base alla distanza di una località in base alla sua posizione attuale.</p>

ID	RE04
Nome	Pubblicazione di recensioni per una struttura
Descrizione	<p>Una volta aperta una struttura l'utente deve poter pubblicare una recensione attraverso un apposito tasto. Per la scrittura della recensione l'utente deve scegliere una valutazione che va da 1 a 5 stelle, una descrizione di almeno 100 caratteri.</p>

ID	RE05
Nome	Accesso autenticato
Descrizione	<p>L'utente deve avere la possibilità di autenticarsi per la pubblicazione di una recensione.</p> <p>Una volta autenticato sarà autorizzato a recensire qualsiasi struttura presente nell'applicativo.</p> <p>Il sistema dovrà offrire la possibilità di registrarsi o di effettuare un login tramite l'inserimento di credenziali scelte precedentemente.</p> <p>Nel caso in cui l'utente voglia registrarsi perché non in possesso di un account, il sistema richiederà all'utente di inserire i seguenti dati:</p> <ul style="list-style-type: none"> • Nome • Cognome • Nickname • Password • E-mail <p>Nel caso in cui l'utente sia già registrato può autenticarsi fornendo al sistema i propri dati di accesso.</p>

Requisiti non funzionali

Requisiti di sistema: impliciti dai requisiti funzionali.

ID	RENF01
Nome	Visualizzazione della mappa tramite Google Maps
Descrizione	La visualizzazione della mappa deve avvenire tramite le API offerte da Google

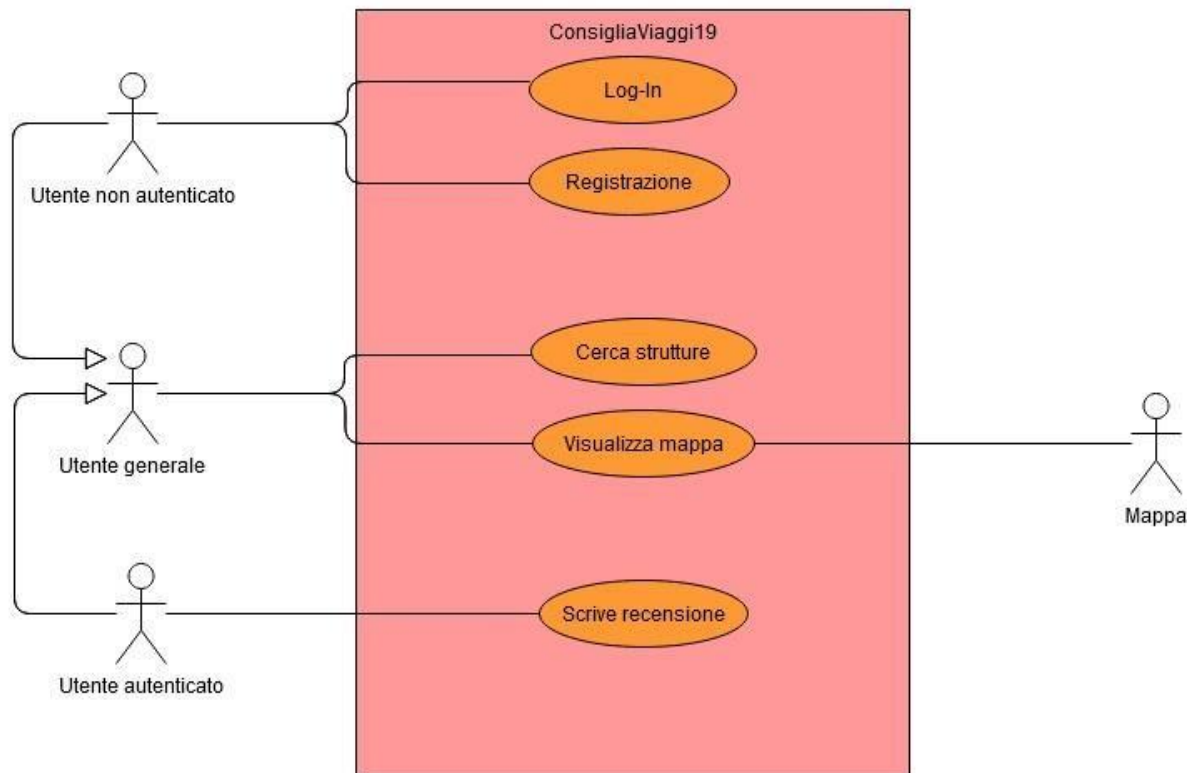
ID	RENF02
Nome	Usabilità mobile
Descrizione	L'applicazione deve funzionare a prescindere dalla grandezza del dispositivo.

ID	RENF03
Nome	Performance ricerca
Descrizione	Una ricerca deve avvenire in meno di 2.5 secondi nel 90% dei casi.

Requisiti di dominio

ID	DOM01
Nome	Rispetto del GDPR
Descrizione	Nella form di registrazione bisogna rispettare il regolamento Generale sulla protezione dei dati GDPR (General Data Protection Regulation).

Diagramma dei casi d'uso



Mockup

Di seguito sono riportati i mockup delle varie schermate dell'applicativo.

Gli errori e le note riguardanti una schermata sono riportati sulla destra della stessa. Per chiarezza, in alcune schermate **NON** sono riportati tutti gli errori possibili ma solo quelli più significativi.

Login utente

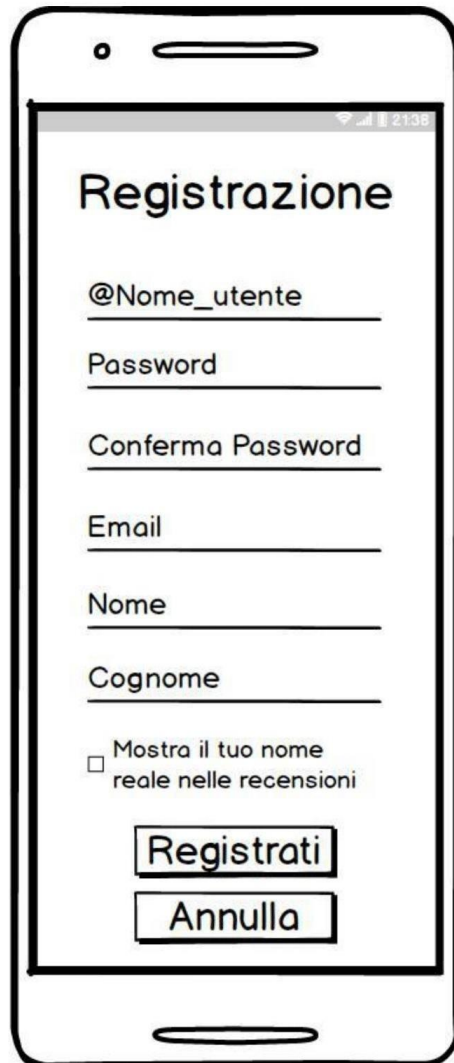
The mockup shows a smartphone screen with the following elements:

- Status bar at the top: signal strength, Wi-Fi, and time 21:38.
- Title: "Consiglia Viaggi 2019" in a large, bold font.
- Section: "Accesso" in a bold font.
- Form fields: "Email" and "Password" with horizontal lines for input.
- Buttons: "Accedi" (highlighted with a black border), "Crea Account", and "Accedi come visitatore".

To the right of the phone is an error dialog box:

- Title: "Attenzione"
- Message: "La password o l'username inseriti non sono validi, riprovare"
- Button: "Ok"

Registrazione



A mobile app registration form displayed on a smartphone screen. The form includes fields for email, password, confirm password, name, and surname. It also has a checkbox for showing the real name in reviews and two buttons at the bottom: 'Registrati' and 'Annulla'.

Registrazione

@Nome_utente

Password

Conferma Password

Email


Nome

Cognome

☐ Mostra il tuo nome reale nelle recensioni

Registrati

Annulla



An error dialog box with a title 'Errore' and a message 'Le password inserite non corrispondono'. It has an 'Ok' button at the bottom.

Errore

Le password inserite non corrispondono

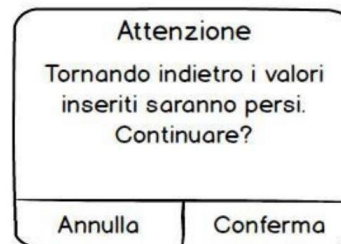
Ok



A success dialog box with a message 'Registrazione avvenuta con successo'. It has an 'Ok' button at the bottom.

Registrazione avvenuta con successo

Ok



A warning dialog box with a title 'Attenzione' and a message 'Tornando indietro i valori inseriti saranno persi. Continuare?'. It has two buttons at the bottom: 'Annulla' and 'Conferma'.

Attenzione

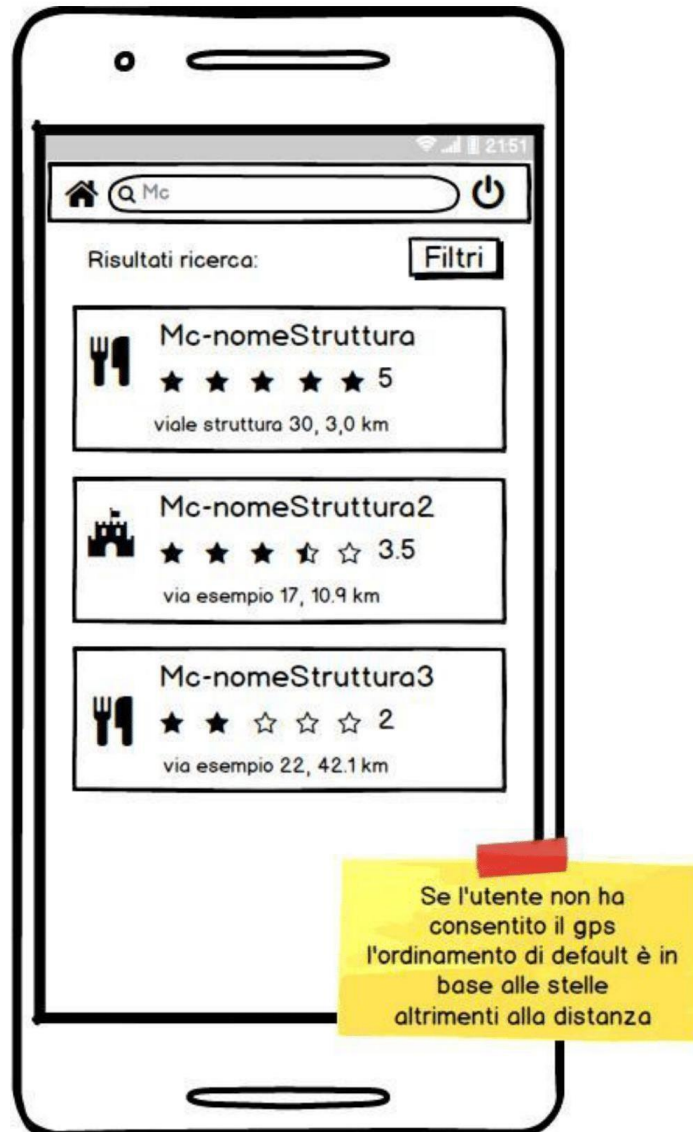
Tornando indietro i valori inseriti saranno persi. Continuare?

Annulla Conferma

Mappa



Ricerca struttura



Filtri ricerca struttura

CV19

Ordinamenti

- ☒ Ordina per numero di stelle crescente
- ☐ Ordina per numero di stelle decrescente
- ☐ Ordina in base alla distanza

Filtri

Numero minimo di stelle

- 1
- 2
- 3
- 4
- 5

Distanza massima

- Nessuna distanza
- 1 chilometro
- 3 chilometro
- 5 chilometri
- 10 chilometri

Conferma

Se l'utente non ha attivato il gps le opzioni per distanza sono disabilitate

Struttura



Se l'utente non ha
effettuato l'accesso o ha
già scritto una recensione
il tasto è disabilitato

Scrittura recensione

The image shows a mobile app interface for writing a review. The app screen has a status bar at the top with 'CV19', a home icon, and a power icon. The main content area is titled 'NomeStruttura' and features five empty star icons for rating. Below the stars is a text input field with the placeholder 'Scrivi una descrizione' and a character count '(-100)'. At the bottom of the screen are two buttons: 'Annulla' (highlighted with a red box) and 'Pubblica'. A yellow callout box points to the 'Pubblica' button with the text: 'Pubblica è disabilitato finché l'utente non seleziona il numero di stelle e non scrive almeno 100 caratteri'. To the right of the app screen is a confirmation dialog box titled 'Attenzione' with the text: 'Tornando indietro i valori inseriti saranno persi. Continuare?'. The dialog has two buttons: 'Annulla' and 'Conferma'.

CV19

NomeStruttura

☆☆☆☆☆

Scrivi una descrizione

(-100)

Annulla Pubblica

Pubblica è disabilitato finché l'utente non seleziona il numero di stelle e non scrive almeno 100 caratteri

Attenzione

Tornando indietro i valori inseriti saranno persi. Continuare?

Annulla Conferma

Tabelle di Cockburn

Le seguenti tabelle di Cockburn sono relative ai casi d'uso definiti nello UCD.

Registrazione utente

Use Case #1	Si registra		
Goal in context	L'utente vuole creare un account		
Precondition	None		
Success end condition	Creazione account		
Failed end condition	Account non creato, chiude la schermata di registrazione		
Primary actor	Utente non autenticato		
Trigger	L'utente preme il tasto "Crea un account" nella schermata "Schermata Iniziale"		
DESCRIPTION	Step	Utente non autenticato	Sistema
	1	Preme il pulsante "Crea un account" nella "Schermata iniziale"	
	2		Crea il nuovo account e mostra pop-up di successo
	3	Preme il tasto "Ok"	
	4		Mostra la schermata "Schermata principale"
EXTENSION #1	Step	Utente non autenticato	Sistema
L'utente lascia vuoto uno o più campi	2.1		Evidenzia tutti i campi e mostra il messaggio "Questo campo non può essere lasciato vuoto" e infine torna allo step 1 dello scenario principale

<i>EXTENSION #2</i>	Step	Utente non autenticato	Sistema
Il nickname scelto dall'utente è inferiore i 4 caratteri	2.2		Evidenzia il campo nome utente con il messaggio "Il nome utente deve essere lungo almeno 4 caratteri" e torna allo step 1 dello scenario principale
<i>EXTENSION #3</i>	Step	Utente non autenticato	Sistema
Le due password inserite dall'utente non coincidono	2.3		Mostra pop-up di avvertenza
	3.3	Preme il tasto "Ok"	
	4.3		Torna allo step 1 dello scenario principale
<i>EXTENSION #4</i>	Step	Utente non autenticato	Sistema
L'utente inserisce una mail non valida	2.4		Evidenzia il campo email con messaggio "Il valore inserito non corrisponde ad una Email" e torna allo step 1 dello scenario principale
<i>EXTENSION #5</i>	Step	Utente non autenticato	Sistema
L'utente inserisce un nickname già in uso	2.5		Evidenzia il campo nome utente con messaggio "Nome utente già in uso" e torna allo step 1 dello scenario principale

<i>EXTENSION #6</i>	Step	Utente non autenticato	Sistema
L'utente inserisce una email già in uso	2.6		Evidenzia il campo e-mail con messaggio "Email già in uso" e torna allo step 1 dello scenario principale
<i>EXTENSION #7</i>	Step	Utente non autenticato	Sistema
L'utente annulla dopo aver riempito uno o più campi	2.7	Preme il tasto "Annulla"	
	3.7		Mostra pop-up di avvertenza
	4.7	Preme il tasto "Conferma"	
	5.7		Torna alla schermata "Schermata iniziale" e termina lo use case
<i>EXTENSION #8</i>	Step	Utente non autenticato	Sistema
L'utente inserisce una password minore di 8 caratteri	2.8		Evidenzia il campo password con messaggio "La password deve essere lunga almeno 8 Caratteri" e torna allo step 1 dello scenario principale

Login utente:

Use Case #2	Effettua log-in		
Goal in context	L'utente vuole effettuare l'accesso		
Precondition	L'utente non è autenticato		
Success end condition	Riesce ad autenticarsi		
Failed end condition	Non riesce ad autenticarsi e cambia schermata		
Primary actor	Utente non autenticato		
Trigger	Viene aperto l'applicativo		
<i>DESCRIPTION</i>	Step	Utente non autenticato	Sistema
	1	Inserisce Nome Utente e Password e preme il tasto "Accedi"	
	2		Effettua il log-in e visualizza la schermata "Schermata principale"
<i>SUBVARIATION #1</i>	Step	Utente non autenticato	Sistema
L'utente accede in un secondo momento	1a	Preme il menu in alto a destra ed "Accedi"	
	2a		Torna allo step 1 dello scenario principale
<i>EXTENSION #1</i>	Step	Utente non autenticato	Sistema
L'utente lascia uno o entrambi i campi vuoti	2.1		Evidenzia i campi vuoti e torna allo step 1 dello scenario principale
<i>EXTENSION #2</i>	Step	Utente non autenticato	Sistema
L'utente inserisce dati non validi per i log-in	2.2		Mostra pop-up di avvertenza
	3.2	Preme il tasto "Ok"	
	4.2		Torna allo step 1 dello scenario principale

Visualizza struttura:

Use Case #3	Visualizza struttura		
Goal in context	L'utente è interessato a visualizzare le informazioni relative ad una struttura		
Precondition	None		
Success end condition	Trova la struttura cercata e la apre		
Failed end condition	Non apre la struttura o cambia schermata		
Primary actor	Utente		
Trigger	L'utente preme la barra di ricerca		
<i>DESCRIPTION</i>	Step	Utente	Sistema
	1	Digita il nome della struttura e preme invio	
	2		Mostra la schermata "Cerca struttura" con una lista di strutture che contengono quel nome
	3	Preme una delle strutture mostrate	
	4		Mostra la schermata "Visualizza struttura" relativa a quella struttura
<i>SUBVARIATION #1</i>	Step	Utente	Sistema
La ricerca della struttura avviene tramite mappa	1a	Effettua gli step da 1 a 4 dello use case "Visualizza mappa"	
	2a		Torna allo step 3 dello scenario principale

<i>SUBVARIATION #2</i>	Step	Utente	Sistema
La ricerca avviene per categoria invece che per nome	1b	Preme una delle tre categorie nella schermata "Schermata principale"	
	2b		Filtra la ricerca in base al tasto premuto e torna allo step 3 dello scenario principale
<i>EXTENSION #1</i>	Step	Utente	Sistema
L'utente aggiunge filtri alla ricerca	3.1	Preme il tasto "Filtri"	
	4.1		Mostra la schermata "Filtri ricerca"
	5.1	Seleziona le sue preferenze e preme il tasto "Conferma"	
	6.1		Filtra la ricerca in base ai filtri ed ordinamenti e torna allo step 3 dello scenario principale
<i>EXTENSION #2</i>	Step	Utente	Sistema
L'utente torna alla homepage	3.2	Preme il tasto "Home"	
	4.2		Mostra la schermata "Schermata principale" e termina lo use case

<i>EXTENSION #3</i>	Step	Utente	Sistema
L'utente effettua una nuova ricerca	3.3	Preme la barra di ricerca, digita un nuovo nome e preme invio	
	4.3		Torna allo step 2 dello scenario principale
<i>EXTENSION #4</i>	Step	Utente	Sistema
L'utente preme il tasto "Accedi"	3.4	Preme il tasto "Accedi"	
	4.4		Mostra la schermata "Schermata iniziale" e termina lo use case

Visualizza mappa:

Use Case #4	Visualizza mappa			
Goal in context	L'utente è interessato a visualizzare le strutture nelle sue vicinanze sulla mappa			
Precondition	None			
Success end condition	La mappa viene visualizzata			
Failed end condition	None			
Primary actor	Utente			
Trigger	L'utente preme la barra di ricerca			
DESCRIPTION	Step	Utente	Sistema	Mappa
	1	Preme il tasto "Apri mappa" nella schermata "Schermata principale"		
	2		Richiede la mappa al provider	
	3			Fornisce la mappa al sistema
	4		Carica la posizione dell'utente sulla mappa	
	5		Carica tutte le strutture sulla mappa	

	6		Mostra la schermata "Schermata mappa"	
<i>SUBVARIATION #1</i>	Step	Utente	Sistema	Mappa
L'utente apre la mappa dalla struttura	1a	Svolge gli step 1-4 dello use case "Visualizza struttura"		
	2a	Preme il tasto "Apri su mappa"		
	3a		Carica la posizione dell'utente e la singola struttura	
	4a		Torna allo step 6 dello scenario principale	
<i>EXTENSION #1</i>	Step	Utente	Sistema	Mappa
L'utente ha il GPS disabilitato e lo abilita	4.1		Mostra pop-up di avvertenza	
	5.1	Preme il tasto "Conferma" ed abilita il GPS		

	6.1		Torna allo step 4 dello scenario principale	
<i>EXTENSION #2</i>	Step	Utente	Sistema	Mappa
L'utente non abilita il GPS	4.2		Mostra pop-up di avvertenza	
	5.2	Preme il tasto "Annulla"		
	6.2		Torna allo step 5 dello scenario principale	

Scrittura recensione:

Use Case #5	Scrivi recensione		
Goal in context	L'utente è interessato a pubblicare una sua recensione relativa ad una determinata struttura		
Precondition	None		
Success end condition	L'utente scrive e pubblica la recensione		
Failed end condition	Annulla la scrittura della recensione o cambia schermata		
Primary actor	Utente Autenticato		
Trigger	Preme "Scrivi recensione"		
DESCRIPTION	Step	Utente Autenticato	Sistema
	1	Svolge gli step 1-4 dello use case "Visualizza struttura"	
	2	Preme il tasto "Scrivi una recensione"	
	3		Mostra la schermata "Scrivi recensione"
	4	Riempie il campo descrizione di almeno 100 caratteri e da un voto da 1 a 5 stelle	
	5		il tasto "Pubblica" viene attivato
	6	Preme il tasto "Pubblica"	
	7		Mostra pop-up di successo
	8	Preme il tasto "Ok"	
	9		Mostra la schermata "Visualizza struttura"

<i>EXTENSION #1</i>	Step	Utente Autenticato	Sistema
L'utente annulla dopo aver scritto qualcosa o aver scelto una valutazione	4.1	Preme il tasto "Home" o "Indietro"	
	5.1		Mostra pop-up di avvertenza
	6.1	Preme il tasto "Conferma"	
	7.1		Mostra la schermata "Visualizza struttura" e termina lo use case

Modello di dominio:

Class diagram

Vengono ora elencati i Class Diagram relativi alle funzionalità che l'applicazione mobile deve offrire. L'individuazione degli oggetti partecipanti è guidata dall'euristica Three-ObjectType, gli oggetti vengono dunque classificati in:

- ▯ *Entity* – Modellano l'informazione persistente.
- ▯ *Boundary* – Modellano le interazioni tra gli attori e il sistema.
- ▯ *Control* – Modellano la logica necessaria a svolgere lo use case.

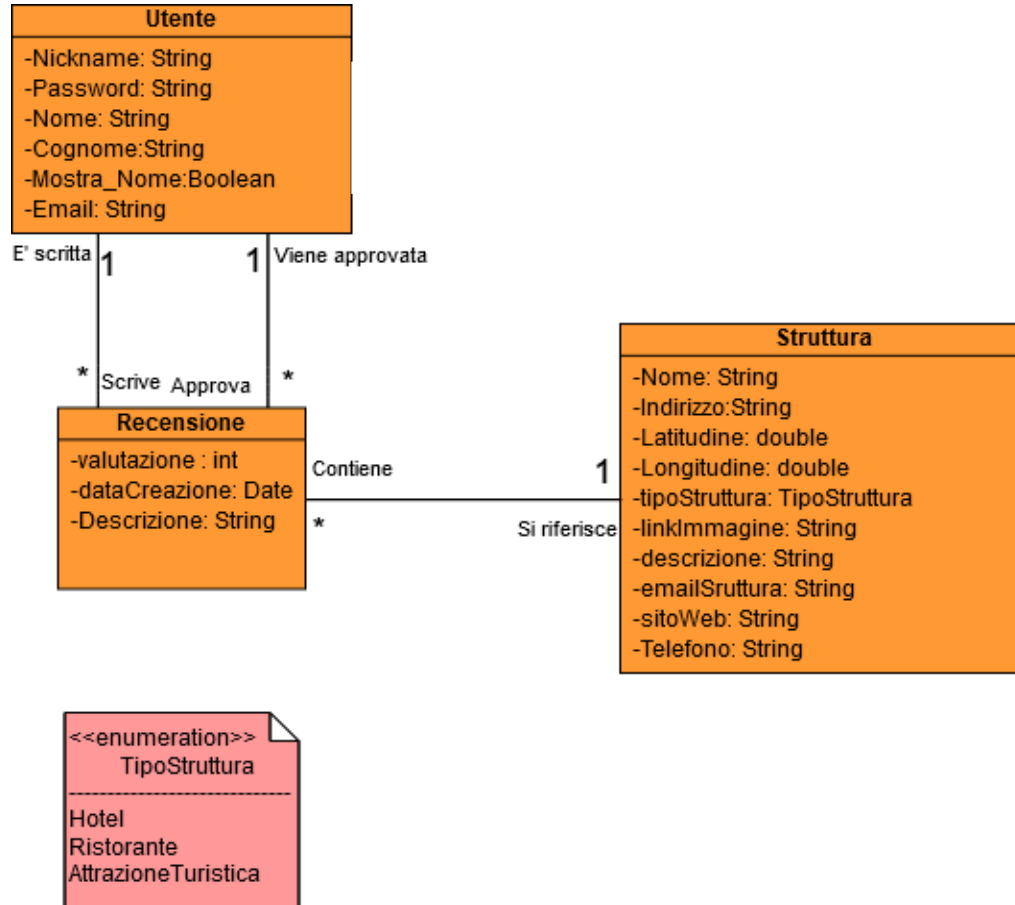
Al fine di aumentare la leggibilità dei Class Diagram entity, boundary e control presentano colori differenti:

- ▯ Le classi *entity* sono colorate in **verde**
- ▯ Le classi *boundary* sono colorate in **viola**
- ▯ Le classi *control* sono colorate in **giallo**

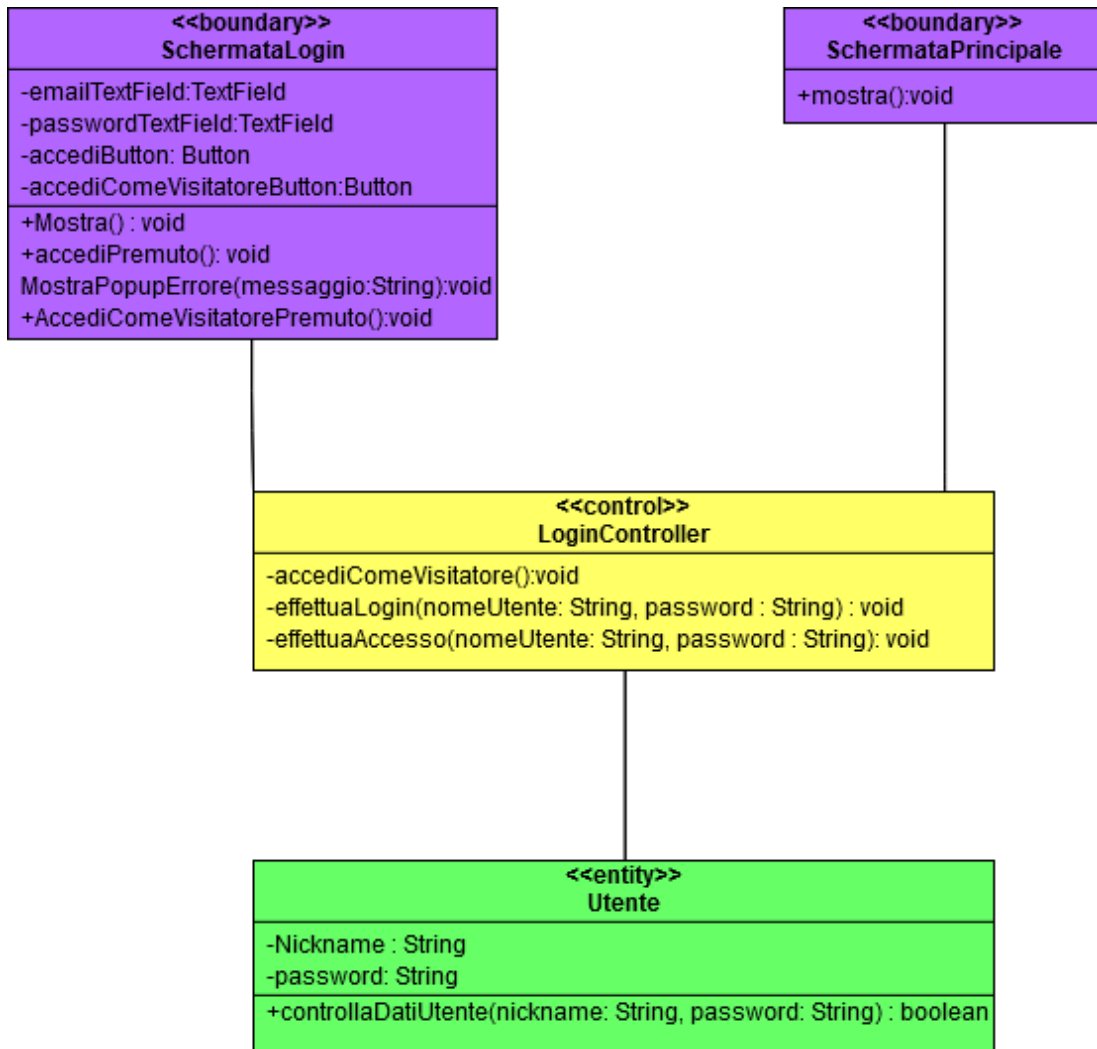
Inoltre, ove possibile, le classi sono raggruppate a seconda del loro tipo.

Unica eccezione all'euristica è il Class diagram del database.

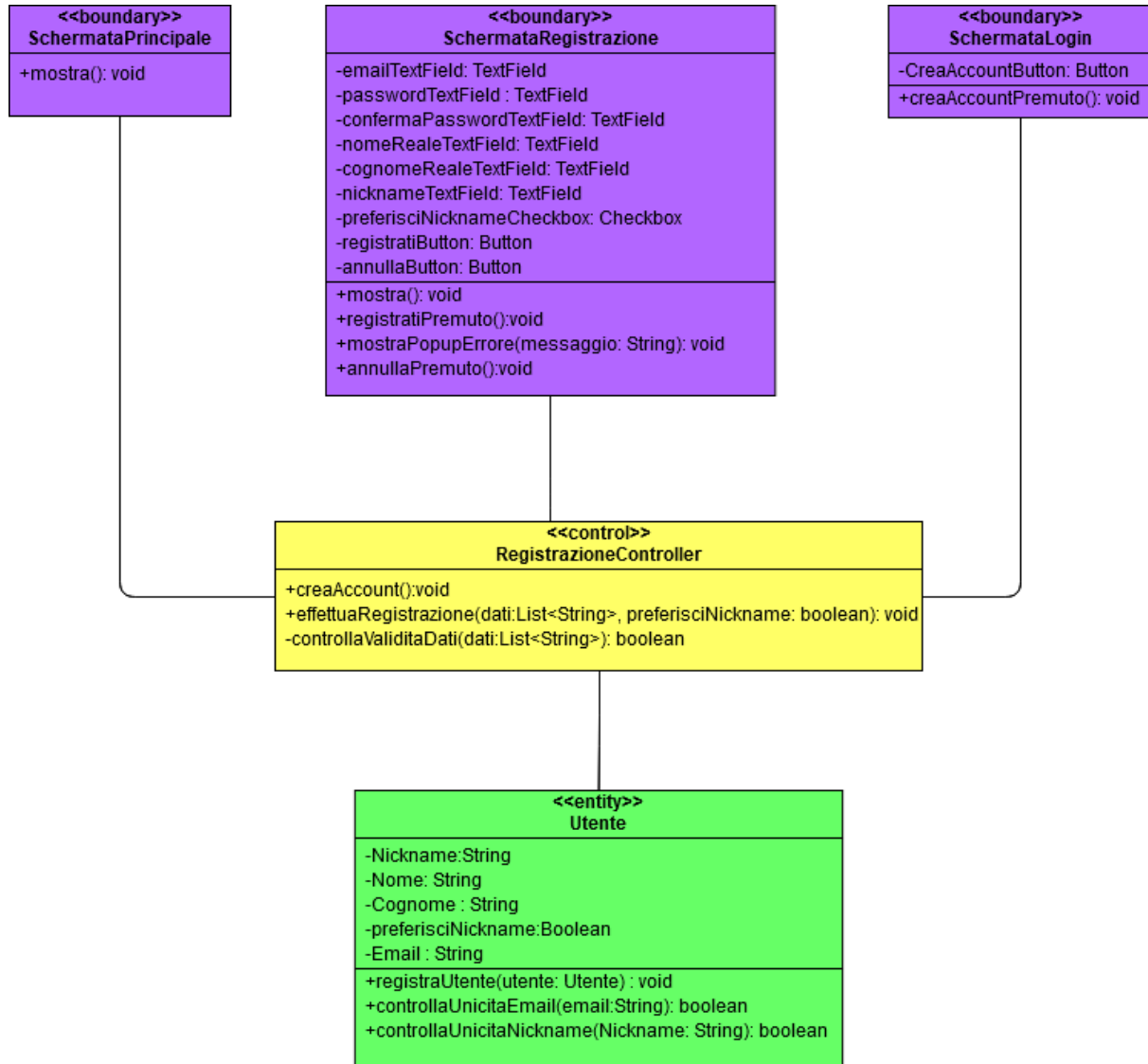
Database



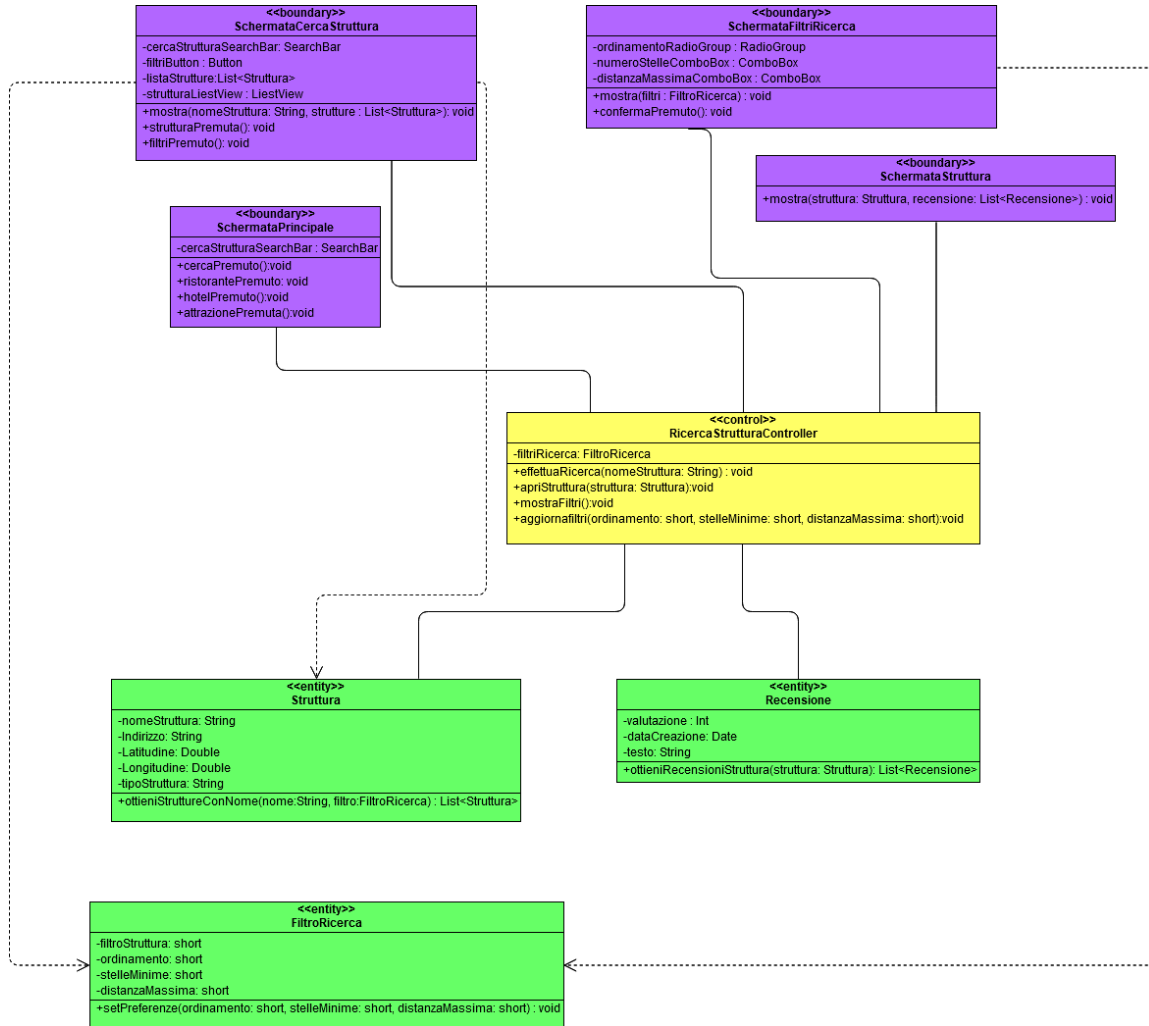
Login utente



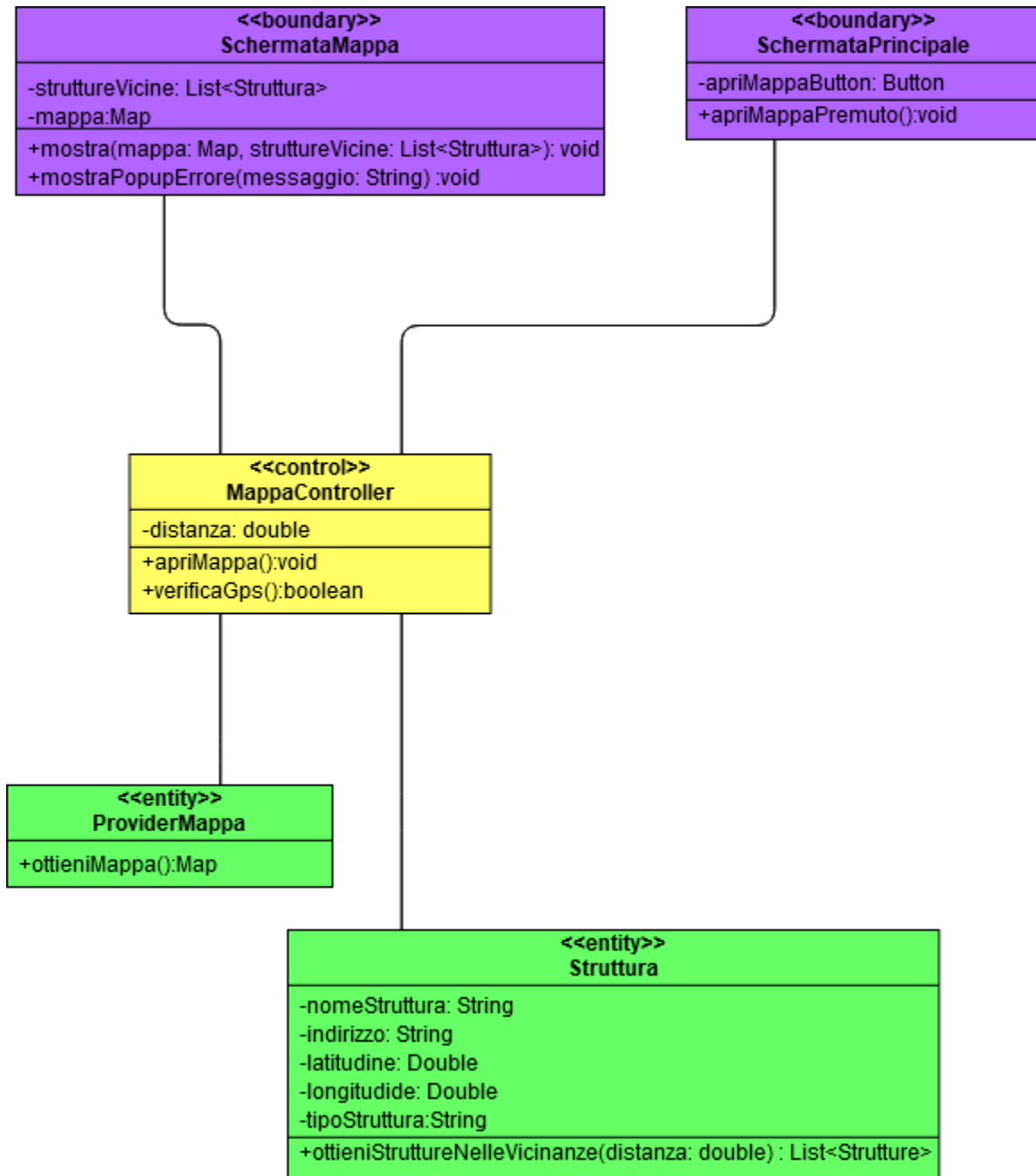
Registrazione



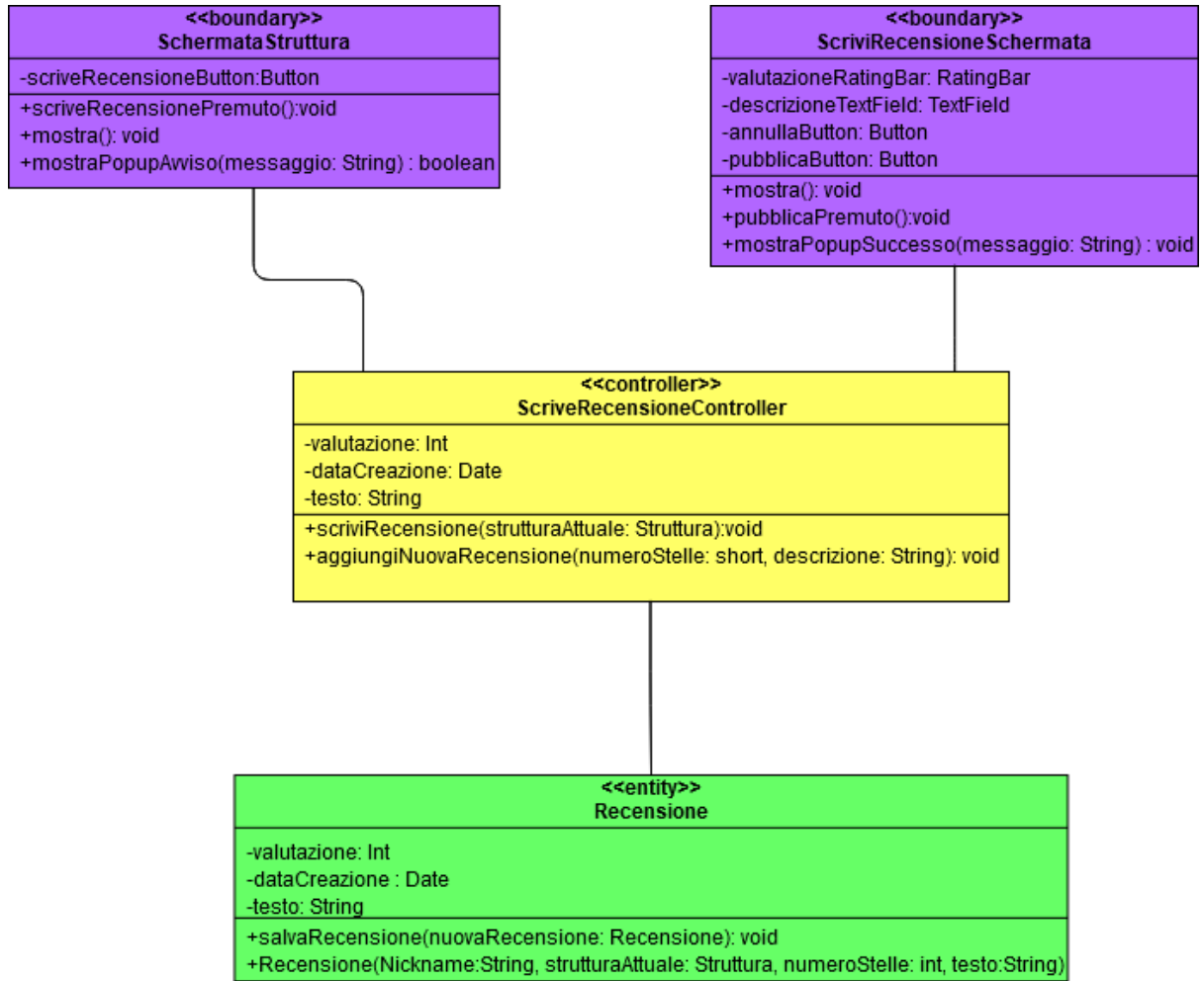
Ricerca Struttura



Visualizza Mappa



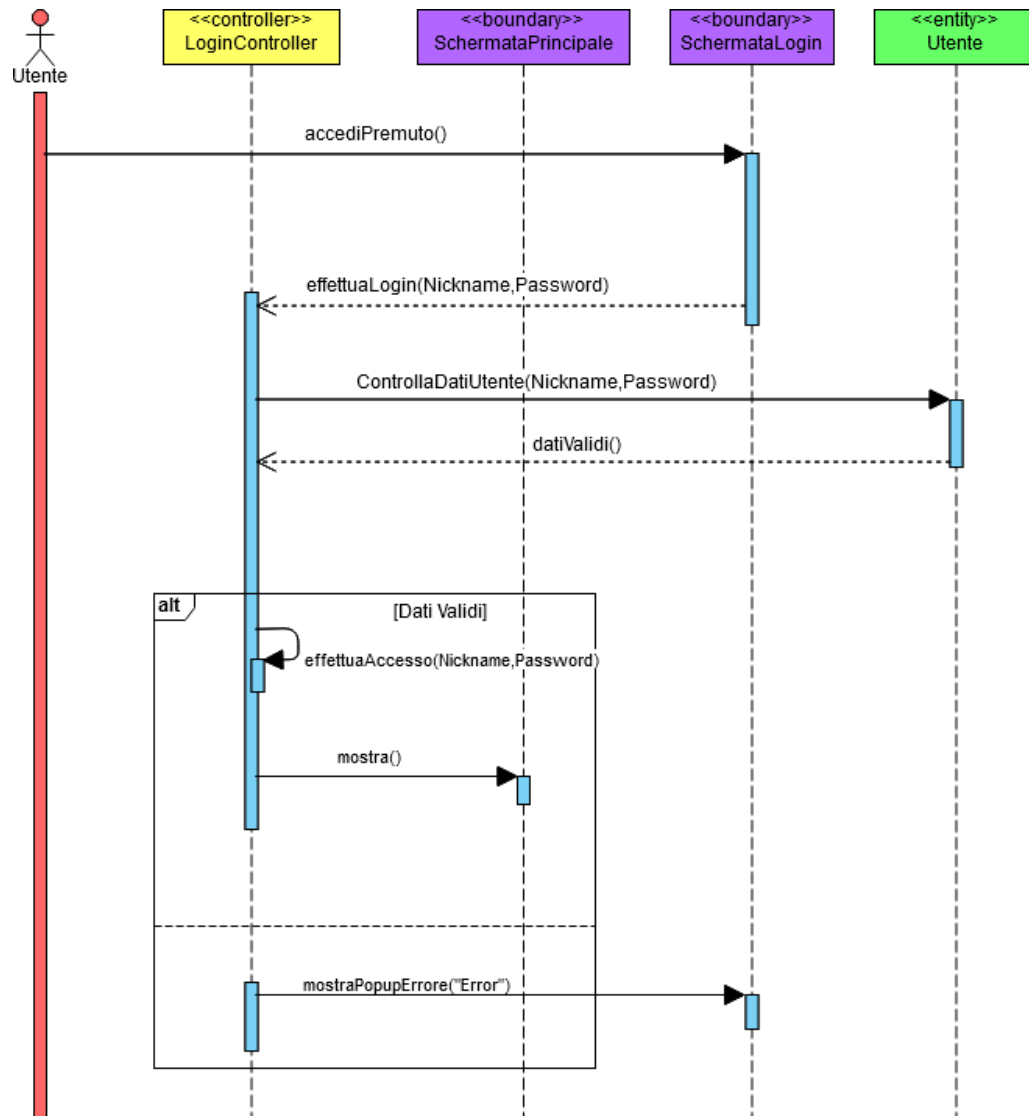
Scrivi Recensione



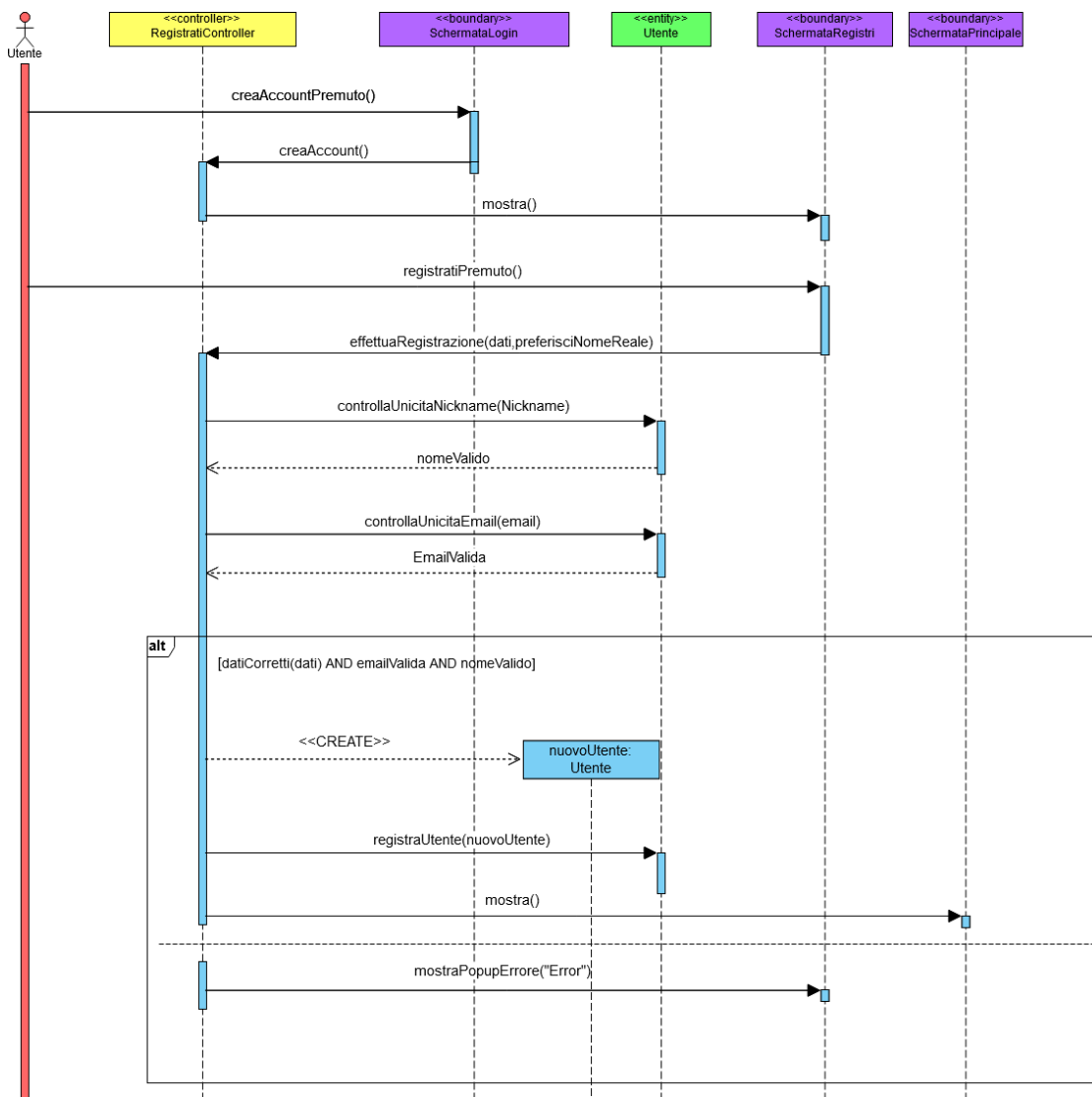
Sequence Diagram

Vengono ora elencati i Sequence Diagram relativi alle funzionalità che l'applicazione deve offrire.

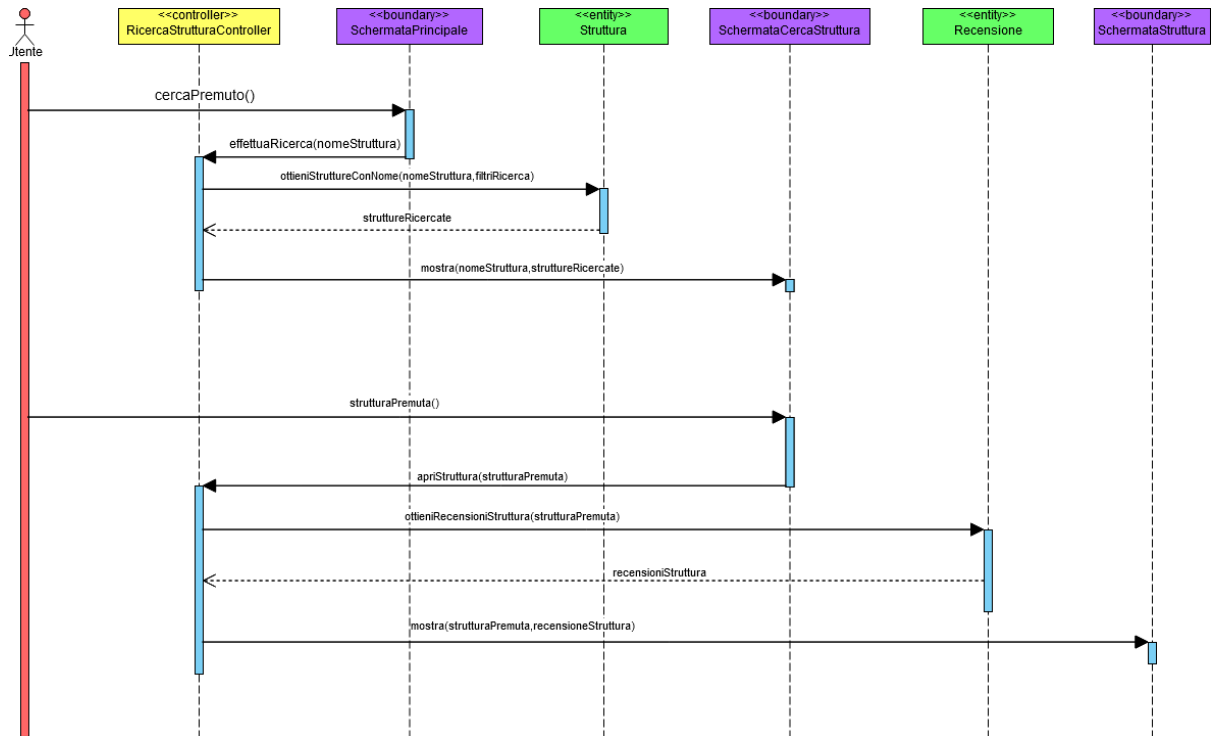
Login Utente



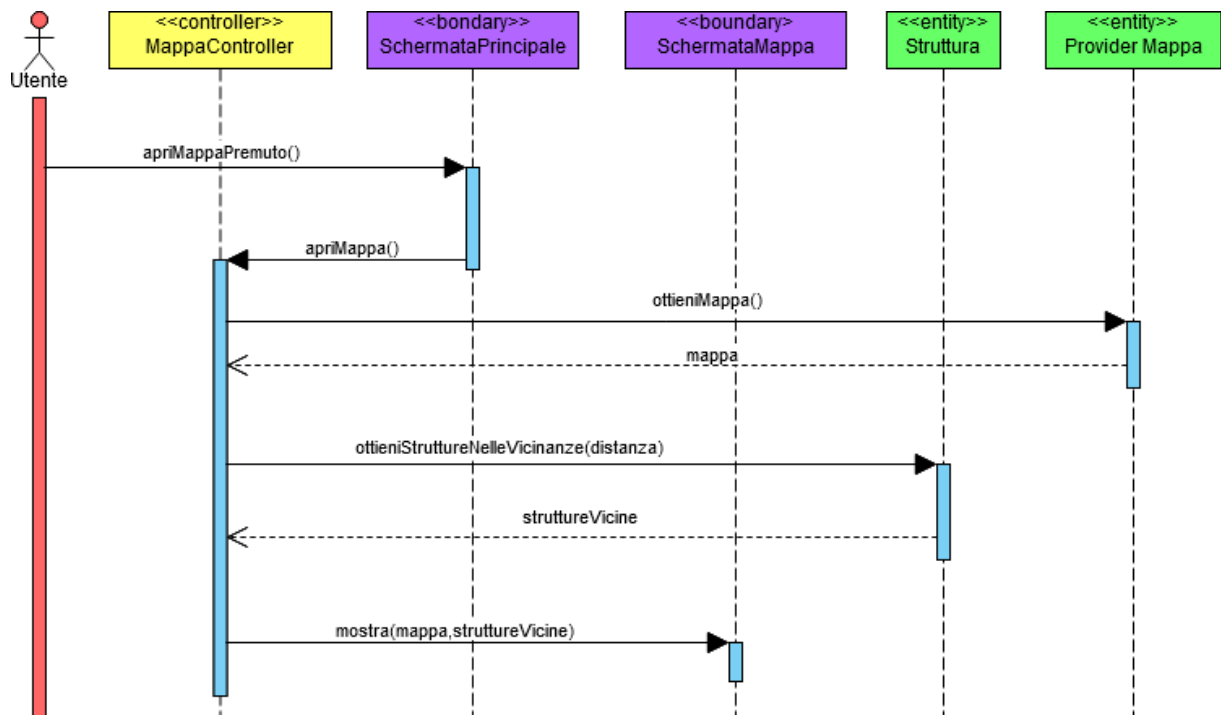
Registrazione



Ricerca Struttura



Visualizza Mappa



Scrivi Recensione

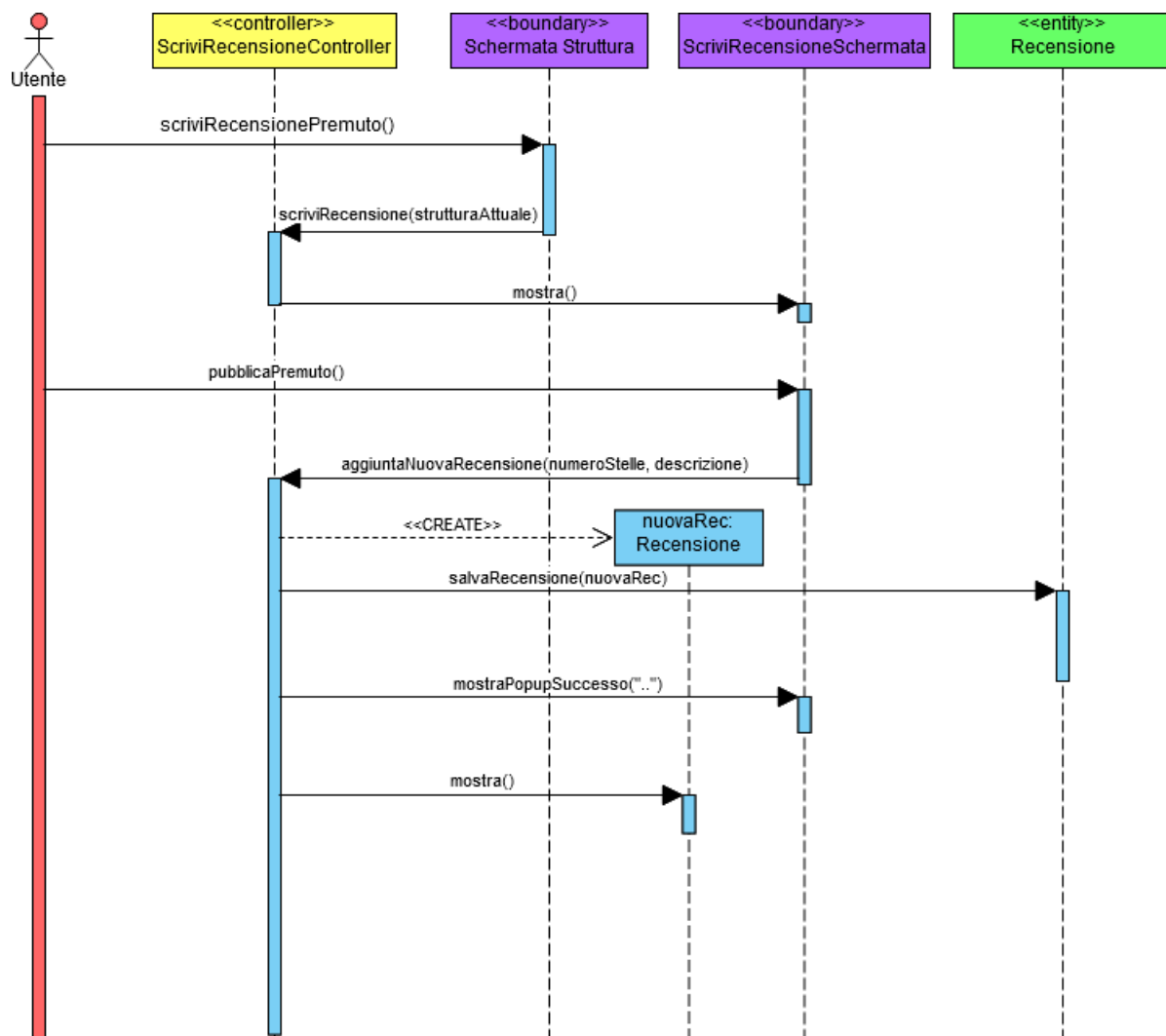


Diagramma di stato di analisi: Ricerca Strutture

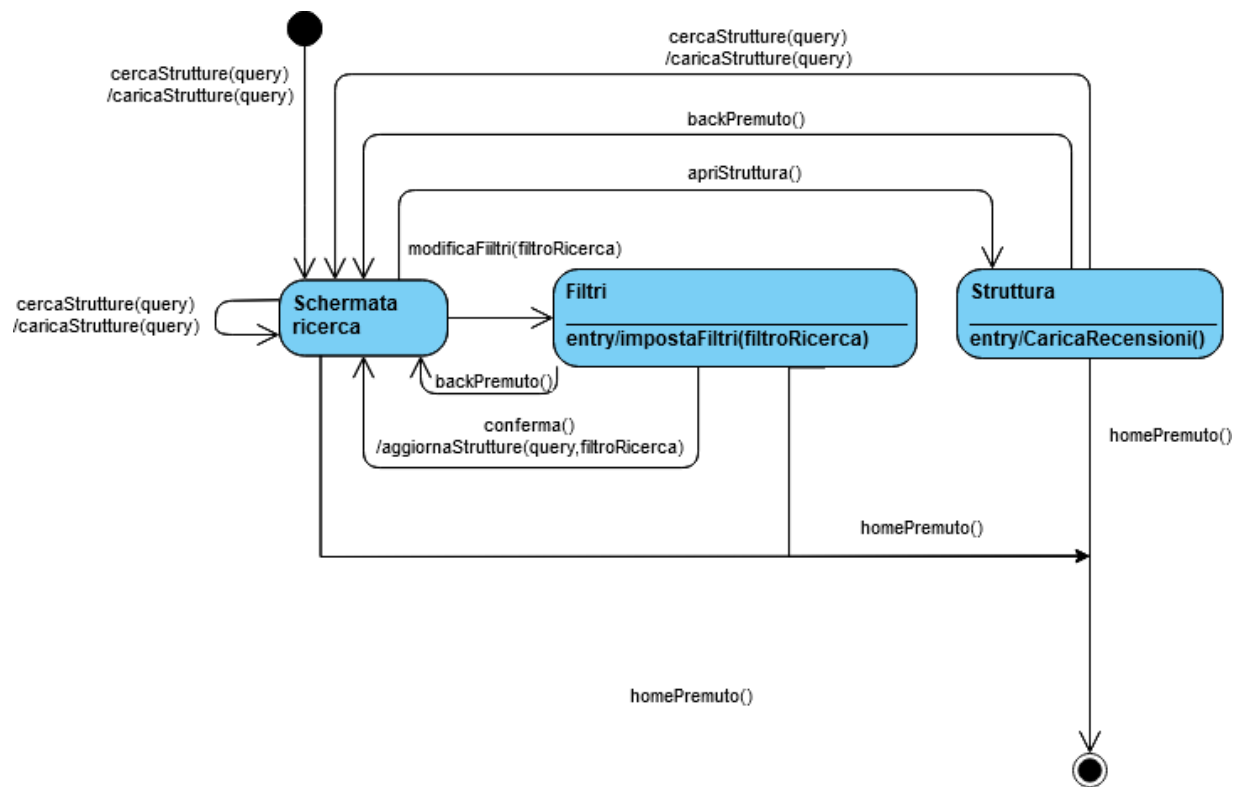
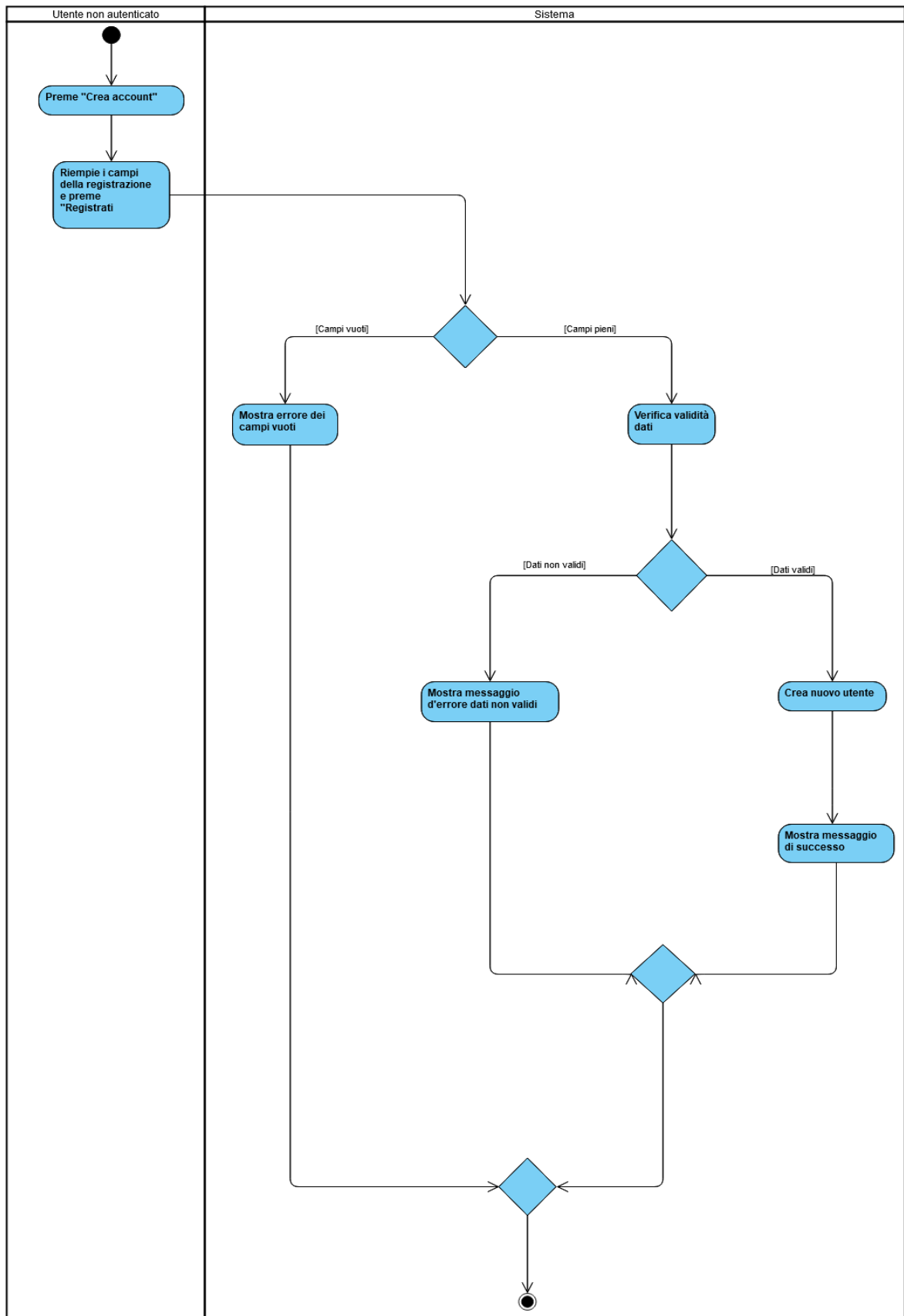


Diagramma di attività: Registrazione utente



Pianificazione dettagliata dell'attività

Viene ora rappresentata una **proposta** di impegno risorse e pianificazione dettagliata delle attività con l'ausilio del seguente diagramma di Gantt:



Come osservabile dal diagramma è stato adottato il modello a cascata essendo questo altamente comprensibile, visibile (distacco quasi netto tra una fase e l'altra), supportabile (non necessità di tool di supporto complessi).

I rischi del modello a cascata sono noti: Dal momento che l'output di ogni fase è di vitale importanza per la fase successiva è necessario minimizzare gli errori al fine di evitare ingenti ritardi dovuti ad un riciclo di tutte le fasi.

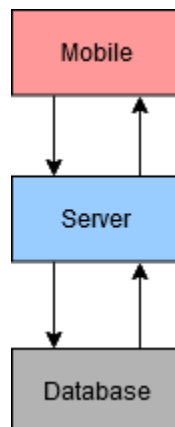
Capitolo 3

Documento di Design del sistema

Analisi dell'architettura

Architettura esterna:

L'architettura esterna del sistema è suddivisa in tre differenti layers, ognuno dei quali può accedere esclusivamente al layer immediatamente sottostante, rendendo dunque l'architettura chiusa.



Il database e l'app potranno essere interpellati esclusivamente da un server. Quest'ultimo ha lo scopo di far da tramite tra l'applicativo mobile e il Database.

Client e server sono stati scritti in Kotlin. Un linguaggio di programmazione che fa della semplicità il suo asso forte: tipi opzionali, interoperabilità con java garantita al 100%, sintassi chiara e immediata e tante altre features che si ritengono hanno agevolato di molto il lavoro, nonché ridotto il tempo di sviluppo.

L'architettura three-type e le relative sotto architetture client/server che sono state utilizzate, permettono di poter eventualmente aggiungere nuovi client

e metodi di memorizzazione in futuro, aspetto considerato fondamentale nell'implementazione dell'applicativo, poiché la volontà è di rendere il servizio quanto più scalare possibile, considerando l'aggiunta di un nuovo client (ES: Desktop) o di un nuovo metodo di memorizzazione dei dati (ES: DB Oracle).

Le misure atte a far ciò sono descritte nei successivi due paragrafi.

Architettura server:

Per il server è stato utilizzato il framework Spring Boot 2.0. Questo framework è pensato per consentire di sviluppare servizi web, permettendo di lavorare su tutti i livelli di sviluppo: dalla sicurezza al web vero e proprio.

Era chiaro già dal modello funzionale che si doveva fare una distinzione tra un'utente visitatore (che non può scrivere recensioni) e utente registrato (che può scriverle).

Per la gestione dell'autenticazione e per rispettare questo requisito è stato utilizzato anche Spring Security che - oltre a gestire l'autenticazione - gestisce anche le autorizzazioni relative ad un singolo utente.

La scelta di questo framework comporta anche un'ottima scalabilità al nostro sistema distribuito: con l'aggiunta di nuove feature sarà immediato dare maggiori permessi agli utenti registrati, o addirittura creare nuovi ruoli.

Come già detto nel paragrafo precedente, il sistema garantisce una scalabilità tale da rimanere stabile alla modifica del modello di persistenza dei dati. Per far ciò è stato adottato come design pattern il DAO (Data Access Object) e il Factory Method.

Il DAO, mediante l'utilizzo di interfacce, permette di modificare come vengono memorizzati i dati, in modo da poter cambiare il database senza andare ad impattare sul resto del codice.

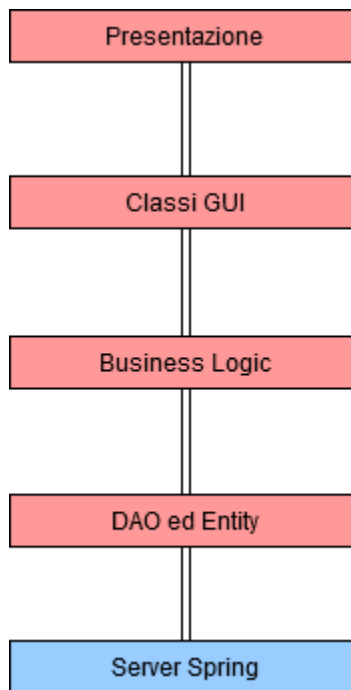
Una problematica di utilizzare una classe DAO come versione concreta è che, alla sua modifica, tutte le classi che dipendono da essa devono essere ricomilate; così come, cambiando il database esso non diventa riciclabile.

Per venire in contro a questo problema si trasforma la classe DAO in un'interfaccia e le sue realizzazioni concrete verranno create mediante la classe Factory. Così facendo, è possibile creare nuove versioni concrete senza impattare il codice preesistente, modificando unicamente la Factory.

Architettura client:

L'applicativo mobile è stato scritto per Android, che per l'interfaccia grafica utilizza XML, fortunatamente grazie ad i vari IDE (Android Studio) è sempre più semplice approcciarsi al design dell'app, rendendo raro lo "sporcarsi le mani" con XML, questo si è tradotto in tempi di sviluppo minori ed efficacia del nostro sistema.

Il client ha un'architettura multilivello illustrata di seguito:



Nel dettaglio:

Livello 0: Server Spring. Al livello più basso risiedono le chiamate al server per ottenere i dati.

Livello 1: DAO ed Entity. Questo livello è formato da interfacce, classi concrete ed una factory. **Sono stati utilizzati gli stessi pattern del server in modo da garantire gli stessi vantaggi anche ai client.**

Livello 2: Business logic: Si occupa di realizzare le funzionalità richieste.

Livello 3: GUI. Formato da classi che controllano l'interfaccia grafica, si occupano di acquisire dati e inizializzare l'interfaccia grafica

Livello 4: Presentazione. Formato da XML che descrive le pagine a livello visivo.

Servizi cloud utilizzati:

AWS è la piattaforma cloud più completa al momento disponibile, di fondamentale importanza è stato per la gestione -in termini di risorse- del server; questo ci ha permesso di concentrarci sui requisiti del cliente e nell'offrigli le risorse a lui più adeguate, in particolare si è utilizzato:

- Elastic Beanstalk: è un servizio web che gestisce automaticamente implementazione, offre un provisioning di capacità e auto scaling al monitoraggio della salute dell'applicazione. Su questo servizio è stato caricato il server Spring dove in pochi minuti era già pronto per l'uso senza dover effettuare alcuna "mano d'opera" sull'infrastruttura o delle risorse.
- Relational Database Service (RDS): è un servizio che fornisce una capacità ridimensionabile a un costo conveniente, ovvero le prestazioni offerte sono proporzionali al numero di richieste (Ottimo per la scalabilità!). Inoltre, automatizza al tempo stesso le attività di amministrazione del database più dispendiose (diminuita la complessità del sistema). Il motore su cui è stato fatto girare il DB è MySQL.

Diagramma delle classi di design

Segue il class diagram di design Per renderlo più chiaro:

- Le associazioni tra classi troppo distanti tra di loro, in particolare quelle più “tecniche” sono state omesse.
- I metodi privati e i metodi get e set sono stati omessi, salvo casi eccezionali.

CRC Cards

In questo paragrafo sono riportate le CRC cards (Class Responsibility Collaborators) utilizzate per dettagliare la collaborazione tra le classi e le loro responsabilità.

Nella sezione “Superclassi” verranno contenute sia interfacce che superclassi, rispettando l’ereditarietà in Kotlin che permette di estendere, allo stesso tempo, una singola classe ma più interfacce.

La superclasse sarà preceduta dalla parola “extends”, le interfacce dalla parola “implements”.

L’ordine in cui sono rappresentate le classi rispetta la sintassi del Kotlin, quindi: prima la superclasse (se presente) e poi le interfacce (sempre se presenti).

Nome classe	FilterRecFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata dei filtri delle recensioni	HomeActivity
	StructureController

Nome classe	FilterSearchFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata dei filtri della ricerca	HomeActivity
	SearchController
	FilterSearch

Nome classe	HomeFragment
-------------	--------------

Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata principale	HomeActivity
	HomeController

Nome classe	MapFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata della mappa	HomeActivity
	MapController
	Structure

Nome classe	SearchFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata di ricerca	StructureAdapter
	SearchController
	Structure

Nome classe	WriteRecFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la schermata di scrittura recensioni	HomeActivity
	StructureController
	Structure

Nome classe	StructureFragment
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators

Nome classe	HomeActivity
Superclasse	extends AppCompatActivity
Sottoclassi	
Responsabilità	Collaborators
Rappresenta l'activity principale dell'applicativo Android	AuthFactory
	AuthProvider
	HomeController
	MapController
	HomeFragment
	WriteRecFragment

Nome classe	MainActivity
Superclasse	extends Activity
Sottoclassi	
Responsabilità	Collaborators
Rappresenta l'activity di Login	LoginController

Nome classe	RegistrazioneActivity
Superclasse	extends Fragment
Sottoclassi	
Responsabilità	Collaborators
Rappresenta l'activity di registrazione	LoginController

Nome classe	RecensioneAdapter
Superclasse	extends ArrayAdapter<Review>
Sottoclassi	

Responsabilità	Collaborators
Rappresenta la singola cella di Recensione una recensione	

Nome classe	StructAdapter
Superclasse	extends ArrayAdapter<Structure>
Sottoclassi	
Responsabilità	Collaborators
Rappresenta la singola cella di una struttura durante la ricerca	Structure

Nome classe	HomeController
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Gestisce la actionBar di Android	HomeActivity
	MainActivity
	AuthProvider
	HomeFragment
	MapFragment
	SearchFragment

Nome classe	LoginController
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Gestisce la funzionalità relativa allo use case "Si registra"	HomeActivity
	MainActivity
Gestisce la funzionalità relativa allo use case "effettua log-in"	RegistrazioneActivity
	AuthFactory

	AuthProvider
	DAOFactory
	UtenteDAO
	User
	ClientHttp

Nome classe	MapController
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Gestisce la funzionalità relativa allo use case “Visualizza mappa”	HomeActivity
	DAOFactory
	StrutturaDAO
	Structure
	MapFragment
	StrutturaFragment

Nome classe	RicercaController
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Gestisce la funzionalità relativa allo use case “Visualizza Struttura”	DAOFactory
	StrutturaDAO
	Structure
	FiltroRicerca
	FiltriRicercaFragment
	SearchFragment
	StrutturaFragment

Nome classe	StructureController
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Gestisce la funzionalità relativa allo use case "Scrivi recensione"	AuthFactory
	DAOFactory
Gestisce i filtri per la visualizzazione delle recensioni	Review
	Structure
	User
	MapFragment
	WriteRecFragment
	FilterRecFragment
	StructureFragment

Nome classe	DAOFactory
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Fabbrica versioni concrete delle interfacce DAO	UserDAO
	UserDAOServerSpring
	ReviewDAO
	ReviewDAOServerSpring
	StructureDAO
	StructureDAOServerSpring
	NoPersistenceTypeException

Nome classe	ReviewDAOServerSpring
Superclasse	implements ReviewDAO
Sottoclassi	
Responsabilità	Collaborators
Implementa l'interfaccia RecensioneDAO per un server spring	Review
	Structure
	User
	ClientHttp
	JsonUtil

Nome classe	StructureDAOServerSpring
Superclasse	implements StructureDAO
Sottoclassi	
Responsabilità	Collaborators
Implementa l'interfaccia StrutturaDAO per un server spring	Struttura
	ClientHttp
	JsonUtil

Nome classe	UserDAOServerSpring
Superclasse	implements UserDao
Sottoclassi	
Responsabilità	Collaborators
Implementa l'interfaccia UtenteDAO per un server spring	User
	ClientHttp
	JsonUtil

Nome classe	ClientHttp
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Permette di effettuare richieste HTTP	AuthFactory
	AuthProvider

Nome classe	JsonUtil
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Converte JSON in entità	Review
	Structure
	User

Nome classe	NoValidAuthenticationException
Superclasse	extends RuntimeException
Sottoclassi	
Responsabilità	Collaborators
Rappresenta un'eccezione che viene lanciata quando non viene trovato il metodo di autenticazione	
Non valido	

Nome classe	NoPersistanceTypeException
Superclasse	extends RuntimeException
Sottoclassi	
Responsabilità	Collaborators
Rappresenta un'eccezione che viene lanciata quando non viene trovato il metodo di persistenza nelle di configurazione	

Nome classe	Review
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Rappresenta una recensione	User
	Structure

Nome classe	Structure
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Rappresenta una struttura	

Nome classe	User
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Rappresenta un utente	

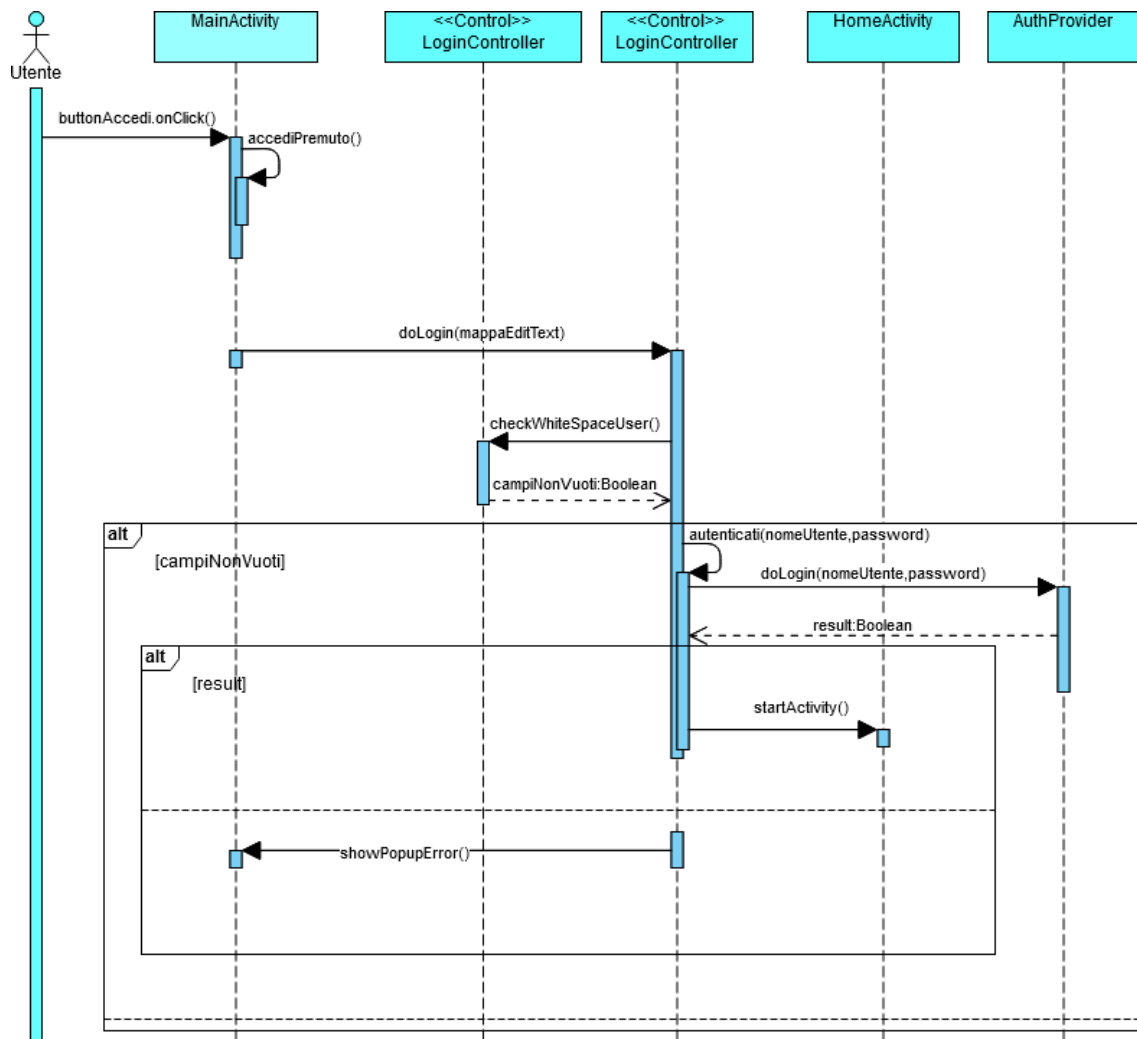
Nome classe	AuthFactory
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Fabbrica versioni concrete di GestoreAutenticazione	AuthProvider
	JwtAutenticazione
	NoValidAuthenticationException

Nome classe	JwtAutenticazione
Superclasse	implements AuthProvider
Sottoclassi	
Responsabilità	Collaborators
Permette l'autenticazione tramite token JWT	JwtToken
	ClientHttp

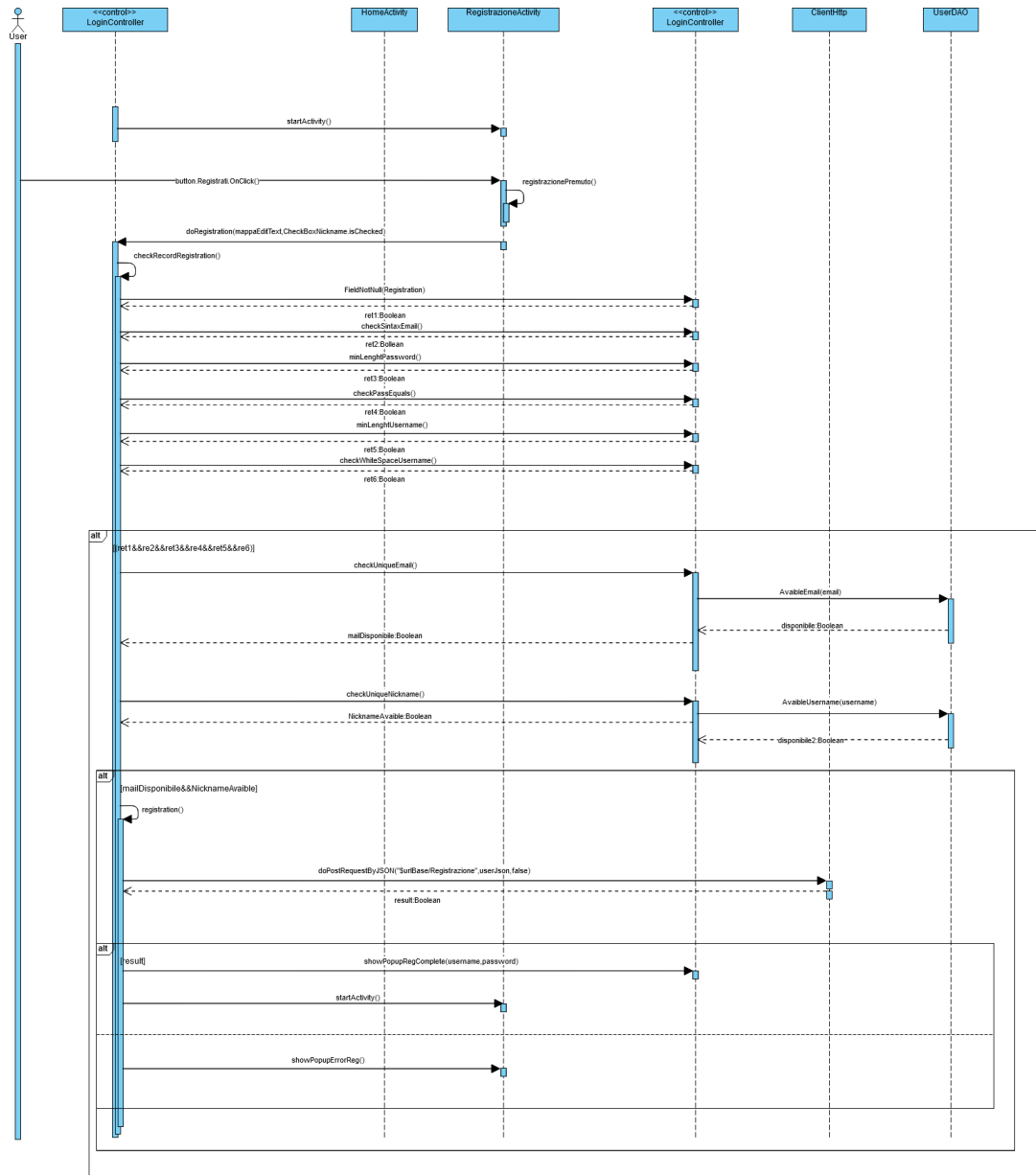
Nome classe	JwtToken
Superclasse	
Sottoclassi	
Responsabilità	Collaborators
Rappresenta un token JWT	

Diagrammi di sequenza di design

Login utente



Registrazione



Capitolo 4

Documento di Testing del sistema

Test Plan per System Testing

Il test plan che segue è stato scritto per organizzare la fase di testing del sistema al fine di verificare e dimostrare che tutti i requisiti esplicitati precedentemente siano stati correttamente implementati e siano funzionanti.

Test registrazione

Test ID	1	
Nome test	Test registrazione	
Descrizione test	L'obiettivo di questo test è verificare la funzionalità di registrazione	
Input	Risultato desiderato	Risultato ottenuto
Riempie i campi con dati validi e preme il tasto "Registrati"	Registrazione effettuata, visualizza il messaggio di successo "Registrazione avvenuta con successo"	Superato
Preme il tasto "Ok" al messaggio di successo	Visualizza la schermata "Schermata Principale"	Superato
Preme il tasto "Registrati" lasciando tutti i campi vuoti	Registrazione fallita, evidenzia tutti i campi con il messaggio. Il campo non può essere vuoto	Superato

Preme il tasto "Registrati" lasciando un qualsiasi singolo campo vuoto	Registrazione fallita, evidenzia il campo vuoto con il messaggio "Il campo non può essere vuoto"	Superato
Inserisce "Noctino52 " nel campo nome Utente, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo Nome Utente con il messaggio "Nome utente già in uso"	Superato
Inserisce "a" nel campo nome Utente, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo Nome Utente con il messaggio "Il nome utente deve essere lungo almeno 4 caratteri"	Superato
Inserisce "a" nel campo email, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo email con il messaggio "Il valore inserito non corrisponde ad una email"	Superato
Inserisce "NoctinoMisterioso@it.it" nel campo email, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo email con il messaggio "Email già in uso"	Superato
Inserisce "1234" nel campo password, "1234" nel campo conferma password, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo password con il messaggio "La password deve essere lunga almeno 8 caratteri"	Superato
Inserisce "12345678" nel campo password, "1234" nel campo conferma password, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, evidenzia il campo conferma password con il messaggio "La password deve essere lunga almeno 8 caratteri"	Superato

Inserisce "12345678" nel campo password, "12345678910" nel campo conferma password, riempie gli altri con dati validi e preme il tasto "Registrati"	Registrazione fallita, mostra il messaggio d'errore "Le password non Corrispondono"	Superato
Preme il tasto "Annulla" con tutti i campi riempiti	Mostra il messaggio d'errore Tornando indietro i valori inseriti saranno persi. Continuare?"	Superato
Preme il tasto "Annulla" con un qualsiasi singolo campo riempito	Mostra il messaggio d'errore Tornando indietro i valori inseriti saranno persi. Continuare?"	Superato
Preme il tasto "Annulla" senza inserire niente	Visualizza la schermata "Schermata Iniziale"	Superato

Dati validi:

Nome utente = "Noctino52" -

Password = "12345678"

Conferma password = "12345678"

Email = "NoctinoMisterioso@it.it"

Nome = "Ivan"

Cognome = "Capasso"

Test login utente

Test ID	2	
Nome test	Test login utente	
Descrizione test	L'obiettivo di questo test e veri care la funzionalità di login per gli utenti	
Input	Risultato desiderato	Risultato ottenuto
Inserisce "Noctino52" nel campo Nome Utente, "ConsigliaViaggi" nel campo password e preme il tasto	Login effettuato, visualizza la schermata "SchermataPrincipale"	Superato

"Login"		
Inserisce "Ivan" nel campo Nome Utente, "ConsigliaViaggi" nel campo password e preme il tasto "Login"	Login fallito, visualizza messaggio di errore "Nome Utente o password errati. Controlla i valori inseriti e Riprova"	Superato
Inserisce "Noctino52" nel campo Nome Utente, "ingsw" nel campo password e preme il tasto "Login"	Login fallito, visualizza messaggio di errore "Nome Utente o password errati. Controlla i valori inseriti e Riprova"	Superato
Inserisce "aaa" nel campo Nome Utente, "bbb" nel campo password e preme il tasto "Login"	Login fallito, visualizza messaggio di errore "Nome Utente o password errati. Controlla i valori inseriti e Riprova"	Superato
Preme il tasto "Login" senza inserire niente	Login fallito, evidenzia entrambi i campi con il messaggio "Il campo non può essere vuoto"	Superato
Inserisce "Noctino52" nel campo Nome Utente e preme il tasto "Login"	Login fallito, evidenzia il campo password con il messaggio "Il campo non può essere vuoto"	Superato

Test filtra strutture

Test ID	3	
Nome test	Test filtra strutture	
Descrizione test	L'obiettivo di questo test è verificare che l'utente riesca a filtrare le strutture	
Input	Risultato desiderato	Risultato ottenuto
Ha il GPS abilitato e cerca "s" sulla barra di ricerca	Viene mostrato l'elenco delle strutture ordinate per distanza	Superato
Ha il GPS disabilitato e cerca "s" sulla barra di ricerca	Viene mostrato l'elenco delle strutture ordinate per numero crescente di stelle	Superato
Apri i filtri di ricerca con il GPS abilitato	Viene mostrato l'elenco dei filtri con tutti i filtri abilitati	Superato

Apri i filtri di ricerca con il GPS disabilitato	Viene mostrato l'elenco dei filtri con i filtri "Distanza massima per le Strutture" e "ordina per distanza" disabilitati	Superato
Seleziona l'ordinamento "Ordina per stelle Crescenti", nessun filtro e preme il tasto "Conferma"	Viene mostrato l'elenco delle strutture ordinate per numero crescente di stelle	Superato
Seleziona l'ordinamento "Ordina per stelle Decrescenti", nessun filtro e preme il tasto "Conferma"	Viene mostrato l'elenco delle strutture ordinate per numero decrescente di stelle	Superato
Note	Il database deve contenere le strutture elencate in basso.	

Test visualizza mappa

Test ID	4	
Nome test	Test visualizza mappa	
Descrizione test	L'obiettivo di questo test e verificare che l'utente riesca a visualizzare la mappa	
Input	Risultato desiderato	Risultato ottenuto
Preme il tasto "Apri Mappa" dalla schermata principale con GPS abilitato	Viene aperta la mappa, sono visibili tutte le strutture e la posizione dell'utente	Superato
Preme il tasto "Apri Mappa" dalla schermata principale con GPS disabilitato	Viene mostrato il messaggio di avvertenza "Il GPS sembra essere disabilitato, alcune funzionalità saranno limitate. Vuoi attivarlo?"	Superato
Preme il tasto "Conferma" sul pop-up del GPS disabilitato	Viene reindirizzato alle impostazioni del dispositivo per abilitarlo	Superato
Preme il tasto "Annulla" sul pop-up del GPS disabilitato	Viene aperta la mappa, sono visibili tutte le strutture ma non la posizione dell'utente	Superato
Preme il tasto "Visualizza su mappa" dalla schermata di una struttura con GPS abilitato	Viene aperta la mappa con solo la struttura e la posizione dell'utente	Superato
Preme il tasto "Visualizza su mappa" dalla schermata di una struttura con GPS disabilitato	Viene mostrato il messaggio di avvertenza "Il GPS sembra essere disabilitato, alcune funzionalità saranno limitate. Vuoi attivarlo?"	Superato
Preme il tasto "Visualizza su mappa" e "Annulla" sul pop-up del GPS disabilitato	Viene aperta la mappa con solo la struttura e senza la posizione dell'utente	Superato

Test scrittura recensione

Test ID	5	
Test name	Test scrittura recensione	
Test description	L'obiettivo di questo test e verificare la funzionalità di scrittura recensioni	
Input	Risultato desiderato	Risultato ottenuto
Scrive una recensione maggiore di 100 caratteri e preme una delle 5 stelle	Il tasto "Pubblica" è abilitato	Superato
Scrive una recensione maggiore di 100 caratteri, preme una delle 5 stelle e preme il tasto "Pubblica"	Viene mostrato il messaggio di successo "Pubblicazione Avvenuta con successo!"	Superato
Preme una delle 5 stelle e preme il tasto "Home"	Viene mostrato il messaggio di avvertenza "Tornando alla home perderai i dati inseriti. Continuare?"	Superato
Scrive "test" come recensione e preme il tasto "Home"	Viene mostrato il messaggio di avvertenza "Tornando alla home perderai i dati inseriti. Continuare?"	Superato
Preme una delle 5 stelle e preme il tasto "Annulla"	Viene mostrato il messaggio di avvertenza "Tornando indietro i valori inseriti saranno persi. Continuare?"	Superato
Scrive "test" come recensione e preme il tasto "Annulla"	Viene mostrato il messaggio di avvertenza "Tornando indietro i valori inseriti saranno persi. Continuare?"	Superato
Preme una delle 5 stelle e non scrive nulla	Il tasto "Pubblica" è disabilitato	Superato

Preme una delle 5 stelle e scrive una descrizione minore di 100 caratteri	Il tasto "Pubblica" è disabilitato	Superato
Scrivo una recensione maggiore di 100 caratteri ma non preme nessuna delle 5 stelle	Il tasto "Pubblica" è disabilitato	Superato
Note	L'utente è autenticato e non ha già recensito la struttura	

Codice xUnit per unit testing di 2 metodi

I test di unita che seguono sono stati scritti ed eseguiti utilizzando il framework JUnit.

Unit testing del metodo ottieniStella

Il metodo ottieniStella si occupa di valutare se una stella debba essere rappresentata come una stella vuota, mezza o piena.

Il metodo ha due parametri: il primo rappresenta la stella da esaminare, il secondo la valutazione media della struttura. Restituisce:

Una stella piena se la differenza tra la stella e la valutazione è minore di 0.3;

Una stella a meta se la differenza tra la stella e la valutazione è maggiore di 0.3 ma minore di 0.75;

Una stella vuota altrimenti.

Una stella per essere valida deve avere un valore compreso tra 1 a 5 nel dominio degli interi, mentre la valutazione di una struttura nell'intervallo [0; 5] nel dominio dei float.

```
1.     fun ottieniStella(numeroStella: Int, valutazione: Float): Int {
2.         if(numeroStella < 1 || numeroStella > 5 || valutazione < 0 ||
   valutazione > 5) {
3.             throw IllegalArgumentException()
4.         }
5.
6.         return when {
7.             (numeroStella-valutazione < 0.3) ->
   R.drawable.ic_star_black_24dp
8.             (numeroStella-valutazione <= 0.75) ->
   R.drawable.ic_star_half_black_24dp
9.             else -> R.drawable.ic_star_border_black_24dp
10.        }
11.    }
```

Questo metodo è stato testato sia tramite strategia Black Box che White Box, il primo per verificare il rispetto dei requisiti, il secondo per verificare che non ci fossero comportamenti anomali in alcune parti del metodi.

Per testare il metodo in black box sono state identificate le seguenti classi di equivalenza:

Parametro numeroStella:

{ CE1: [Int:MIN VALUE; 0] (Non valido)

{ CE2: [1; 5] (Valido)

{ CE3: [6; Int:MAX VALUE] (Non valido)

Parametro valutazione

{ CE4: [Float:MIN VALUE; 0] (Non valido)

{ CE5: [0; 5] (Valido)

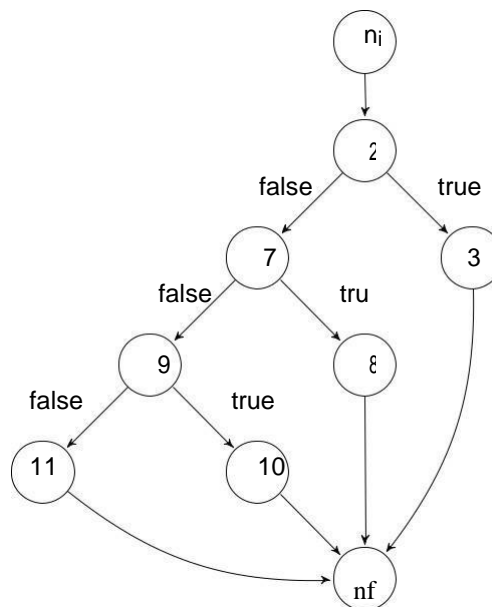
{ CE6: (5; Float:MAX VALUE] (Non valido)

Seguendo la strategia Weak Equivalence Class Testing sono stati scritti 5 test, uno per ogni classe non valida ed uno massimizzando quelle valide:

1. ottieniStellaConNumeroStellaMinoreDiUno()
2. ottieniStellaConNumeroStellaMaggioreDiCinque()
3. ottieniStellaConValutazioneMinoreDiUno()
4. ottieniStellaConValutazioneMaggioreDiCinque()
5. ottieniStellaConParametriValidi()

```
1. class StructureFragmentTest {
2.     private val stellaVuota = R.drawable.ic_star_border_black_24dp
3.     private val stellaMedia = R.drawable.ic_star_half_black_24dp
4.     private val stellaPiena = R.drawable.ic_star_black_24dp
5.
6.     private lateinit var strutturaFragment: StructureFragment
7.
8.     @Before
9.     fun istanziaFragment() {
10.         val strutturaStub = Struttura("", 2.0, 3.0)
11.         strutturaFragment = StructureFragment(structuraStub)
12.     }
13.
14.     @Test (expected = IllegalArgumentException::class)
15.     fun ottieniStellaConNumeroStellaMinoreDiUno() {
16.         strutturaFragment.ottieniStella(-1, 2f)
17.     }
18.
19.     @Test (expected = IllegalArgumentException::class)
20.     fun ottieniStellaConNumeroStellaMaggioreDiCinque() {
21.         strutturaFragment.ottieniStella(42, 3.5f)
22.     }
23.
24.     @Test (expected = IllegalArgumentException::class)
25.     fun ottieniStellaConValutazioneMinoreDiUno() {
26.         strutturaFragment.ottieniStella(2, -5f)
27.     }
28.
29.     @Test (expected = IllegalArgumentException::class)
30.     fun ottieniStellaConValutazioneMaggioreDiCinque() {
31.         strutturaFragment.ottieniStella(4, 33f)
32.     }
33.
34.     @Test
35.     fun ottieniStellaConParametriValidi() {
36.         val stella = strutturaFragment.ottieniStella(3, 4.5f)
37.         assertEquals(stellaPiena, stella)
38.     }
}
```

Questo invece è il grafo che rappresenta il flusso di controllo del whiteboxing:



Da questo GFC sono stati rilevati i seguenti test per ottenere una Branch Coverage:

```
1. // Test White box
2. @Test (expected = IllegalArgumentException::class)
3. fun whiteboxPath_2_3() {
4.     strutturaFragment.ottieniStella(-6, 2f)
5. }
6.
7. @Test
8. fun whiteboxPath_2_7_8() {
9.     val stella = strutturaFragment.ottieniStella(1, 4.5f)
10.    assertEquals(stellaPiena, stella)
11. }
12.
13. @Test
14. fun whiteboxPath_2_9_10() {
15.     val stella = strutturaFragment.ottieniStella(3, 2.5f)
16.     assertEquals(stellaMedia, stella)
17. }
18.
19. @Test
20. fun whiteboxPath_2_9_11() {
21.     val stella = strutturaFragment.ottieniStella(5, 4f)
22.     assertEquals(stellaVuota, stella)
23. }
24. }
```


Unit testing del metodo verificaNomeUtenteDisponibile

Il metodo verificaNomeUtenteDisponibile è un metodo del server utilizzato nella creazione di un nuovo utente che si occupa di valutare se un nome utente è disponibile oppure già utilizzato da un altro utente: il suo parametro è il nome utente da verificare.

Il valore di ritorno è True se il nome utente è libero ed utilizzabile; False se il nome utente è già utilizzato.

Un nome utente può contenere qualsiasi carattere ad esclusione degli spazi; inoltre, se la sua lunghezza non è compresa tra 4 e 16 caratteri, ritorna false. Questo per coerenza con la definizione del dominio del nickname presente nel DB.

```
1. override fun verificaNomeUtenteDisponibile(nomeUtente: String):  
   Boolean{  
2.     if(nomeUtente.length < 4 || nomeUtente.length > 16 ||  
       nomeUtente.contains(" ")) {  
3.       return false4.  
     }  
5.  
6.     val querySql = "SELECT COUNT(id) as occupato FROM utente WHERE  
       nomeUtente = ?"  
7.     val preparedStatement = MySQLConnection.preparaQuery(querySql)  
8.     preparedStatement.setString(1, nomeUtente)  
9.     val resultSet = preparedStatement.executeQuery()  
10.  
11.     resultSet.next()  
12.     return !resultSet.getBoolean("occupato")  
13. }
```

Per testare il metodo in black box sono state identificate le seguenti classi di equivalenza per il parametro nomeUtente:

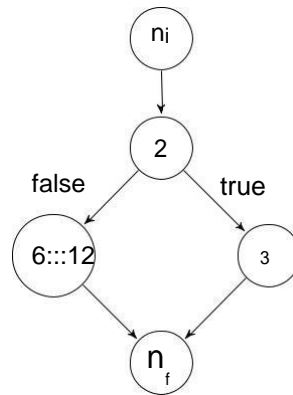
- CE1: nomeUtente.length 2 [Int:MIN VALUE; 4) (Non disponibile)
- CE2: nomeUtente.length 2 (16; Int:MAX VALUE] (Non disponibile)
- CE3: nomeUtente.length 2 [4; 16] AND nomeUtente.lowercased (Disponibile)
- CE3: nomeUtente.length 2 [4; 16] AND nomeUtente.uppercased (Disponibile)
- CE4: nomeUtente.contains(" ") (Non disponibile)
- CE5: @nomeUtente2 Database (Non disponibile)
- CE6: @nomeUtente2 Database (Disponibile)

Seguendo la strategia Weak Equivalence Class Testing sono stati scritti 6 test, uno per ogni classe “non disponibile” ed uno massimizzando quelle “disponibile”:

1. nomeUtenteNonDisponibileConLunghezzaMinoreDiQuattro()
2. nomeUtenteNonDisponibileConLunghezzaMaggioreDiSedici()
3. nomeUtenteNonDisponibileConSpazi()

4. nomeUtenteNonDisponibileInMinuscoloNelDatabase()
5. nomeUtenteNonDisponibileInMaiuscoloNelDatabase()
6. nomeUtenteDisponibileNelDatabase()

Questo invece è il grafo che rappresenta il flusso di controllo del whiteboxing:



Da questo GFC è stata ottenuta una Node Coverage e Branch Coverage attraverso i metodi black box:

1. nomeUtenteNonDisponibileConSpazi()
2. nomeUtenteNonDisponibileInMinuscoloNelDatabase()

```
1. class UtenteDAOMySQLTest {
2.     private lateinit var utenteDAO: UtenteDAOMySQL
3.
4.     @Before
5.     fun istanziaUtente() {
6.         utenteDAO = UtenteDAOMySQL()
7.     }
8.
9.     @Test
10.    fun nomeUtenteNonDisponibileConLunghezzaMinoreDiQuattro() {
11.        assertFalse(utenteDAO.verificaNomeUtenteDisponibile("abc"))
12.    }
13.
14.    @Test
15.    fun nomeUtenteNonDisponibileConLunghezzaMaggioreDiSedici() {
16.        assertFalse(utenteDAO.verificaNomeUtenteDisponibile("0123456789abcdefgh"
17.        ))
18.    }
19.
20.    @Test
21.    fun nomeUtenteNonDisponibileConSpazi() {
22.        assertFalse(utenteDAO.verificaNomeUtenteDisponibile("ciao
23.        mondo"))
24.    }
25.
26.    @Test
27.    fun nomeUtenteNonDisponibileInMinuscoloNelDatabase() {
28.        assertFalse(utenteDAO.verificaNomeUtenteDisponibile("Noctino52"))
29.    }
30.
31.    @Test
32.    fun nomeUtenteNonDisponibileInMaiuscoloNelDatabase() {
33.        assertFalse(utenteDAO.verificaNomeUtenteDisponibile("NOCTINO52"))
34.    }
35.
36.    @Test
37.    fun nomeUtenteDisponibileNelDatabase() {
38.        assertTrue(utenteDAO.verificaNomeUtenteDisponibile("test4321"))
39.    }
40. }
```