RELAZIONE FEDERICO FIORIO 922735

1 Analisi del problema

1.1 Analisi e specifica dei requisiti

Il sistema presentato è un software per il noleggio delle biciclette.
E' suddiviso in 3 parti indipendenti tra loro le quali riescono ad utilizzare gli stessi dati persistenti tramite il db di postgres.
Le tre parti si suddividono in:
Totem: E' il software che viene installato su ogni totem per il noleggio e la riconsegna di biciclette, esso permette di:

- Noleggiare una bicicletta (normale – elettrica – elettrica con seggiolino) a seconda della disponibilità della rastrelliera del totem.
  Il noleggio deve avvenire dopo almeno 5 minuti dall'ultima riconsegna.
- Riconsegnare una bicicletta sia da parte di un utente abbonato sia da parte di un utente di root che svolge un'operazione di riallocazione di biciclette. Questa operazione viene fatta tramite 2 parti principali del sistema, la parte utilizzata solo dagli impiegati/addetti comunali in cui è possibile verificare la disponibilità di biclette nei vari totem del sistema e dalla parte “sul campo” in cui un addetto preleva le bici da un totem e le rialloca in un altro a sua scelta in base alla disponibilità di biciclette che ritiene opportuna.
  Pertanto sul Totem possono riconsegnare queste due diverse categorie di utenti, la differenza principale è che un utente abbonato può noleggiare e quindi consegnare una solo bici (quella noleggiata) mentre l'utente di root può sganciare e agganciare bici a sua scelta dai vari totem per svolgere il compito di riallocazione.
- Permette di segnalare una bicicletta con eventuali danni, questo solo se la bicicletta è in noleggio all'utente per evitare false segnalazioni.
- Permette di ricontrollare l'avvenuta consegna della bicicletta tramite un pulsante.

Per permettere queste funzionalità in modo che non ci sia contrasto tra utente abbonato e di root è previsto un login separato tra i due utenti che porta ad avere interfacce diverse con funzionalità diverse.
Il pagamento di un noleggio avviene alla riconsegna in modo automatico, se un utente abbonato non ha abbastanza fondi per permettere il pagamento, non sarà permessa la consegna e il noleggio continuerà fino a riconsegna avvenuta(e quindi pagamento avvenuto).

La seconda parte indipendete è il “Sito” dove vengono creati gli abbonamenti da parte di utenti nuovi o già abbonati.
La creazione di un nuovo abbonamento consiste nel compilare dei form e inviare i dati inseriti, vengono fatti dei controlli per verificare che i campi non siano vuoti (e ad esempio la mail debba contenere @ per simulare appunto una vera mail).
In caso di studente è richiesta una compilazione extra di forms che verrano poi controllati per verificare la veridicità dei dati inseriti.
E' presente un rinnovo di abbonamento per evitare la ricompilazione di forms da parte di un utente che ha già utilizzato il sistema; in tal caso l'abbonamento precedente deve essere scaduto e tramite nome utente e password è possibile attivare un nuovo abbonamento.

Gli abbonamenti permettono la creazione di un codice univoco utilizzato sul totem insieme alla password scelta dall'utente per accedere ai servizi offerti dal Totem.
Tramite il sito viene fatto l'addebito della penale di 150 euro all'utente che non restituirà la bicicletta nelle 24h successivo all'inizio del noleggio.
Viene effettuato in questo elemento perchè essendo il sito dovrà essere sempre attivo.

Il terzo e ultimo software indipendete è l'applicativo per gli impiegati/addetti comunali.
Questo elemento ha diverse funzionalità:

- Aggiunge una bicicletta al sistema; l'aggiunta di bicicletta viene registrata sul db ma l'effettivo inserimento sui totem deve avvenire tramite la funzionalità di riconsegna dei totem. Solo l'utente di root che ha inserito la nuova bici nel db potrà aggiungerla ad un nuovo totem.
- Elimina Biciclette, solo quelle staccate dai totem e non in noleggio.
- Aggiunge una Rastrelliera
- Elimina una Rastrelliera con nessuna bici attaccata
- Visualizzazione di dati statistici
- Visualizzazione di disponibilità di biciclette nei vari totem presenti sul sistema per effettuare la riallocazione.

2 Progettazione del Sistema

2.1 Diagramma dei casi d'uso

zip->usecase

Un utente non registrato può abbonarsi, questo include il pagamento con carta di credito e il compilare i forms per l'iscrizione.
Un utente non abbonato deve pagare con carta di credito ma non è necessario che compili di nuovo i forms in quanto già registrato.
Gli abbonamenti possibili sono giornaliero, settimanale e annuale.

Solamente l'utente abbonato che può anche essere uno studente può noleggiare una bici (normale, elettrica) questo include la riconsegna ed un eventuale segnalazione dei danni.

Un addetto comunale può aggiungere – togliere bici e rastrelliere ma anche verificare dati statistici e riallocare le biciclette.

2.2 Descrizione degli scenari

Nome	**Abbonamento**
Scopo	Creare un abbonamento
Attore/i	Utente non registrato
Pre-condizioni	Possedere abbastanza soldi sulla carta di credito
Trigger	Tramite applicazione sito
Descrizione sequenza eventi	**Azioni attore** 1.Seleziona il tipo di abbonamento

	3.inserisce carta di credito e prosegue **Risposte del sistema** 2.il sistema chiede di riempire un form dedicato per l'inserimento dei dati dell'utente(compresa la password ed eventuali dettagli studente). 2b.richiede carta di credito per il pagamento 4. valida il pagamento e rilascia codice univoco da utilizzare per il noleggio
Alternativa/e	Azioni attore 1. = 3. = Risposte del sistema 2. = 4. non valida il pagamento e non valida neanche la registrazione e non rilascia codice.
Post-condizioni	restituito un codice univoco e password che saranno validi per tutta la durata del suo abbonamento e sarà registrato dall'applicazione

Nome	**Abbonamento (dopo il primo)**
Scopo	Rinnovare un abbonamento
Attore/i	Utente non abbonato
Pre-condizioni	Possedere abbastanza soldi sulla carta di credito
Trigger	Tramite applicazione sito
Descrizione sequenza eventi	**Azioni attore** 1.Seleziona il tipo di abbonamento e inserisce nome utente e password **Risposte del sistema** 2.il pagamento va a buon fine e viene rilasciato il codice univoco
Alternativa/e	**Azioni attore** 1. = **Risposte del sistema** 2. non valida il pagamento e non rilascia codice.
Post-condizioni	restituito un codice univoco e password che saranno validi per tutta la durata del suo abbonamento

Nome	**Noleggio**
Scopo	Un'utente abbonato vuole noleggiare una bici
Attore/i	Utente Abbonato
Pre-condizioni	l'utente deve possedere un abbonamento funzionante e deve trovare una rastrelliera con delle bici disponibili
Trigger	Tramite applicazione Totem
Descrizione	**Azioni attore**

sequenza eventi	1.inserisce codice utente e password 3.decide che bicicletta scegliere tramite le opzioni del totem 5.restituisce la bicicleta con un tempo di noleggio non superiore a 2 ore **Risposte del sistema** 2a.verifica credenziali 2b.visualizza le scelte possibili di biciclette 4.il sistema indicherà il numero di posteggio della bicicletta da prelevare sbloccandola.
Alternativa/e	**Azioni attore** 1. = **Risposte del sistema** 2a. non viene riconosciuto e costringe a rifare il procedimento **<u>Variante utente inadempiente</u>** uguali fino al 4 **Azioni attore** 5. l'utente non restituisce la bicicletta superando le 24 ore **Risposte del sistema** 6. viene addebitata una penale di 150 euro, oltre ai supplementi, sulla **<u>Variante bici danneggiata.</u>** fino al 4 uguale **Azioni attore** 5.l'utente si accorge che la bici prelevata è danneggiata 6.tramite opzione del totem inserisce le informazioni riguardo al danneggiamento. **Risposte del sistema** 7. il sistema avvertirà gli utenti successivi dei danni presenti alla bicicletta (non permettere il noleggio potrebbe compromettere il sistema da parte di qualche malintenzionato)
Post-condizioni	l'utente avrà a sua disposizione una bicicletta

Nome	**Gestione bici-rastrelliere**
Scopo	Aggiungere-togliere bici-rastrelliere
Attore/i	Addetto Comunale
Pre-condizioni	
Trigger	Tramite applicazione impiegati
Descrizione sequenza eventi	**Azioni attore** 1 l'attore sceglie di aggiungere una bici 3 l'attore sceglie il modello della bici da aggiugere **Risposte del sistema** 2.chiede di inserire il modello tra quelli disponibili

	4. viene generato un codice univoco per la bici aggiunta
Alternativa/e	Analoghe, cambia soggetto e qualche attributo.
Post-condizioni	Inserimento o eliminazione di nuove bici e/o rastrelliere

Nome	**Riallocazione bici**
Scopo	Portare una bici da un totem pieno ad uno vuoto
Attore/i	Addetto Comunale
Pre-condizioni	
Trigger	Tramite applicazione impiegati e totem
Descrizione sequenza eventi	Azioni attore 1. l'attore verifica la disponibilità di biciclette nella zona 3.l'addetto comunale si reca nelle postazioni più opportuna con maggior numero di biciclette 4.inserisce dati per farsi identificare dal totem tramite root 6 inserisce i posti da sbloccare 8 porta le biciclette alla postazione più carente e le aggancia Risposte del sistema 2.evidenzia le disponibilità di biciclette dei totem in zona 5a.il totem identifica l'operatore 5b.chiede quali posti sbloccare 7sblocca i posti corrispondenti
Alternativa/e	
Post-condizioni	Una o più bici verrano tolte da una rastrelliera e aggiunte ad un'altra

2.3 Diagramma delle classi (modello di programma)

Il diagramma delle classi è quello di programma
zip → DiagClassiProgramma

2.4 Diagrammi di sequenza

zip → DiagrammaSequenza

2.5 Diagrammi delle attività

zip → DiagrammaAttività

2.6 Macchine di stato

I 3 applicativi si comportano come oggetti statefull perciò rispondono in modo diverso a seconda del loro stato, perciò ho fatto 3 fsm generali di questi 3 applicativi.

zip → Fsm

2.7 Diagramma dei componenti

zip → DiagrammaComponenti

3 Implementazione del sistema

3.1 Diagramma delle classi (modello di programma)

zip → DiagClassiProgramma

3.1.1 Discussione dei Design Pattern utilizzati

MVC:
utilizzato tramite javaFx mi ha permesso di avere un model(totem oppure abbonamento oppure impiegatoSw) che fornisca i metodi e i dati da utilizzare, un controller (diversi in base alla GUI) esso mi permette sulla base dell'input dell'utente di utilizzare le funzioni del model e la view cioè i file .fxml

OBSERVER:
Utilizzato all'interno della GUI di javafx, quando un bottone viene schiacciato ad esempio nella visulizzazione dei modelli di bici disponibili nel totem, questo bottone permette di aggiornare il testo sull'interfaccia e quindi permettere una visione sincrona degli eventi all'utente.

DAO
Utilizzato per l'accesso al database da parte dei vari model, in questo modo le azioni a basso livello vengono divise da altre azioni di alto livello in modo da mantenere alta coesione e basso accoppiamento.
Per questo pattern sono stati introdotte molte classi DAO all'interno del diagramma delle classi che altrimenti sarebbero state unite ai vari model

CONTROLLER
Riassunto anche nel pattern MVC, il ruolo dei controller è quello di rappresentare la mente di una schermata di GUI. La scelta rientra nel Facade Controller.

POLIMORFISMO
Questo pattern è stato utilizzato per le gerarchie di classi di abbonamento e di bici permettendo maggiore modularità.

INDIRECTION
I vari models ovvero Totem, impiegatosw e Abbonamento possono rientrare in questo gruppo perchè permettono di svolgere il ruolo di intermediario, ad esempio Totem permette di svolgere il ruolo di intermediario tra rastrelliera e pagamento, tenendo separate le due classi e mantenendo alta coesione e basso accoppiamento

SINGLETON
utilizzato in Totem, impiegatosw, Abbonamento e Rastrelliera questi elementi vengono utilizzati in 3 parti del sw indipendenti (Totem e rastrelliera per il Totem), pertanto è necessario che si possa istanziare una sola ed unica istanza di questi elementi.
Esempio: ho due totem in due postazioni diverse della città, un totem non può svolgere il ruolo di entrambi perciò è singleton in modo che ogni totem svolga il proprio ruolo ovvero che venga istanziato correttamente una sola volta.
Il swImpiegato è analogo solo che viene utilizzato su macchine diverse utilizzate solo da impiegati.
Il sito ovvero Abbonamento dev'essere unico, gli utenti non verrano indirizzati su siti diversi con stesse funzionalità ma utilizzerano solamente un unico sito.

3.2 Gestione dei dati persistenti

I dati persistenti vengono mantenuti tramite due database, uno (ProgettoUML) contiene tutti i dati relativi al funzionamento dell'applicazione, l'altro (BancaSimulazione) è usato per mantenere i dati relativi ai saldi delle carti di credito degli utenti, serve appunto a simulare l'interazione con un ente esterno banca.

BancaSimulazione è appunto formato da una sola tabella per questo scopo, perchè mantenere i dati dei saldi relativi alle carte di credito sul db dell'applicazione non sarebbe stato corretto.

L'altro database (ProgettoUML) contiene 8 tabelle:

- Abbonamento per gli abbonamenti attivi da parte degli utenti, per attivi intendo anche quegli abbonamenti occasionali che dovranno essere attivati al primo noleggio.
- Biciclette che contiene le varie biciclette presenti nel db riconosciute univocamente da un id
- Impiegati contiene le informazioni relativi agli user impiegati (di root)
- ImpiegatiRitiroBici è una tabella in cui si segnano le bici che vengono sbloccate e ribloccate in altri totem dagli impiegati o utenti di root, viene utilizzato per avere maggiore controllo sui propri utenti nel caso qualche credenziale di root andasse rubata o se qualche impiegato volesse fare il furbo questa tabella segna ogni spostamento di biciclette degli utenti di root
- Noleggio: questa tabella viene utilizzata per segnare i noleggi avvenuti con i vari orari e rastrelliere in cui sono avvenuti

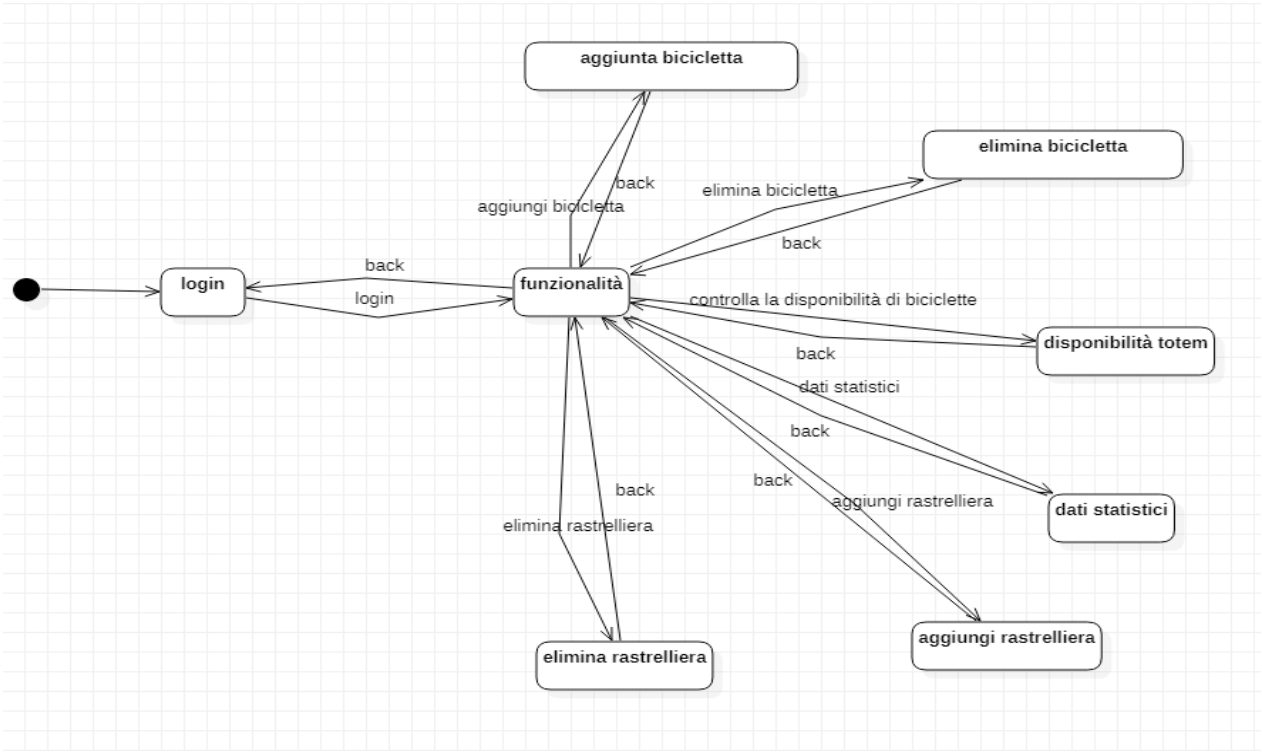
- Totem contiene le informazioni relative ai totem ovvero rastrelliere, cioè dove sono posizionati e il loro id
- Università è usata per inserire i dati degli utenti che si identificano alla registrazione come studenti, questi dati potranno essere verificati da qualche admin e in caso di dati falsi potranno essere applicate delle sanzioni
- Utenti contiene le informazioni relative agli utenti principalmente le credenziali di login e il codice relativo all'abbonamento

3.3 Descrizione dell'Interfaccia Grafica

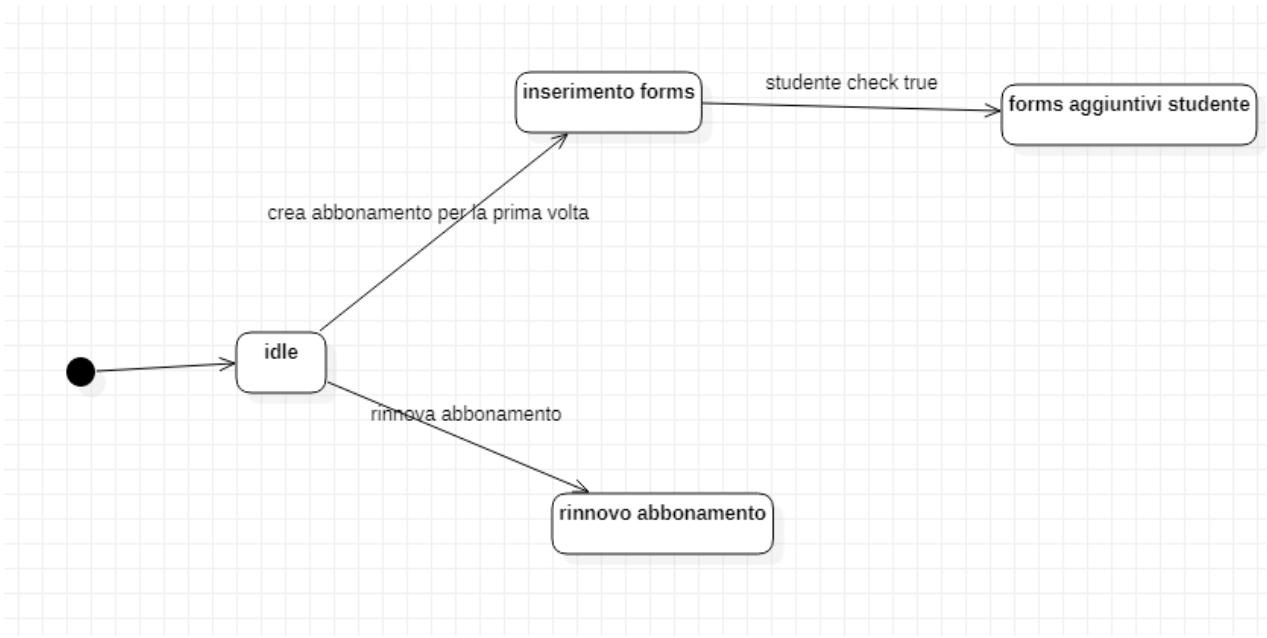
I controlli effettuati riguardano ogni interfaccia.
Nelle interfacce che richiedono un input da parte dell'utente è stato fatto il controllo che questo non sia vuoto e per le schermate che richiedono riscontro col database è stato effettuato il controllo dei dati inseriti dall'utente.
Tutte le query fatte al database sono prepared in modo che non possa essere effettuata nessuna SQL injection.
Per i dati in cui mi volevo assicurare che fossero inseriti sensatamente è permesso un'inserimento solamente tramite choice box o formati simili in cui l'utente non ha altra scelta se non quella di selezionare le opzioni date.
Esempio: nella schermata per i dati aggiuntivi dello studente la mail viene controllata in modo che ci sia una @, oppure per inserimento di bici i campi che voglio che venghino rispettati vengono inseriti tramite choicebox.

Nota: le navigation map vogliono rappresentare in modo semplice la navigazione all'interno della GUI le notazioni delle fsm potrebbero non essere rispettate

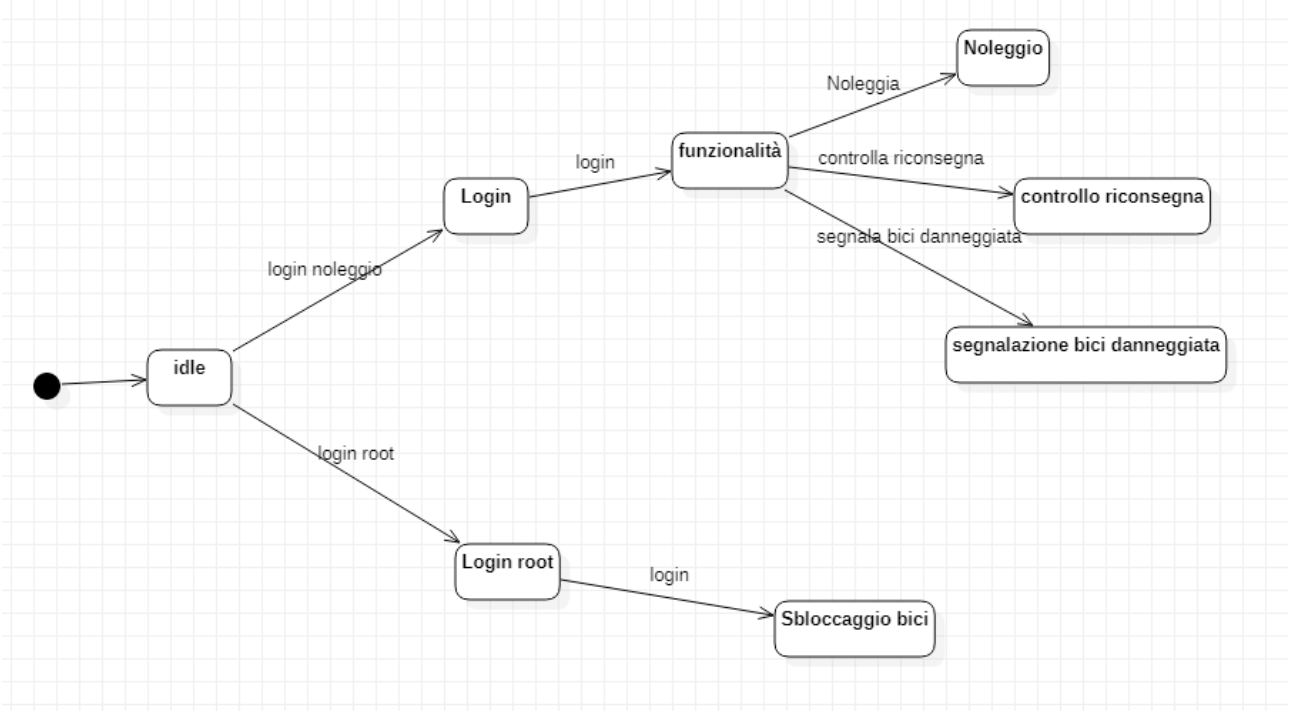
Navigation map GUI swImpiegati (APPLICATIVOimpiegati)



Navigation map Abbonamento (SITOprogetto)

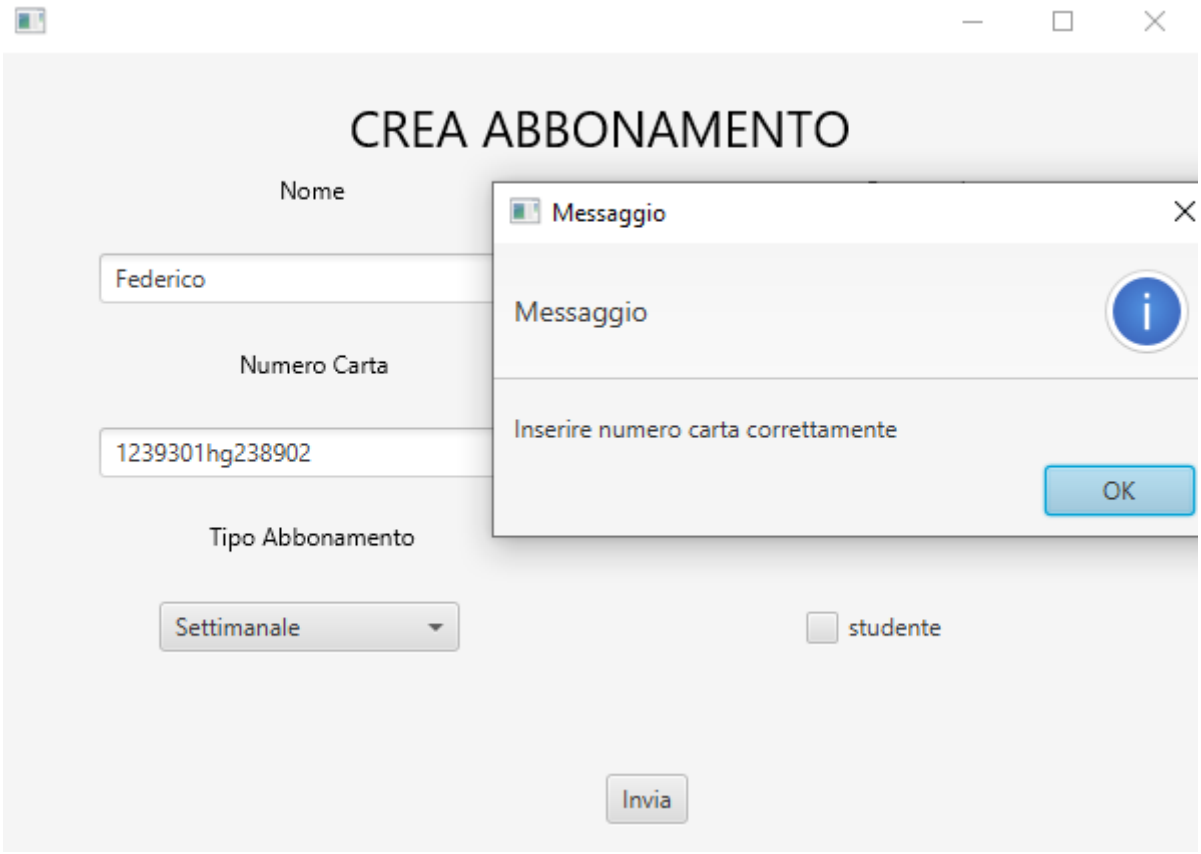


Navigation map TOTEM

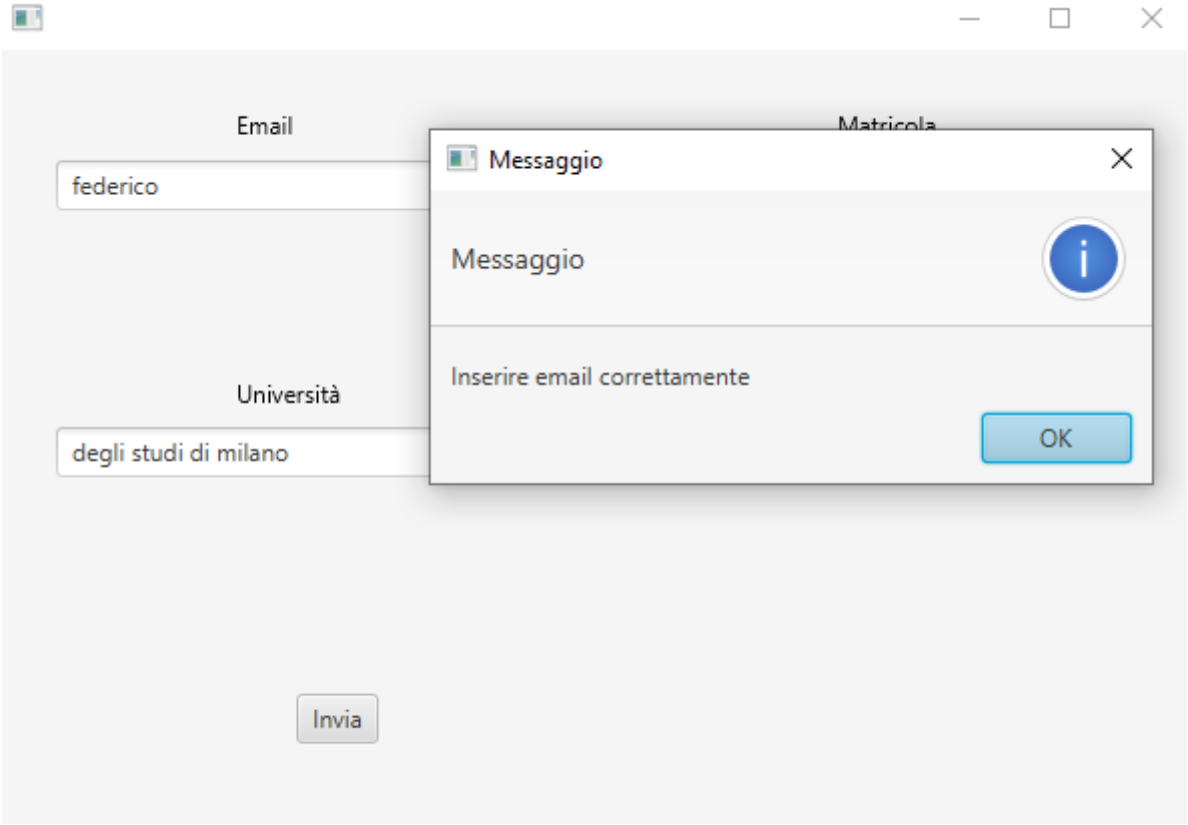


Esempio di utilizzo di choiceBox:

Esempio controllo su numero di carta:



Esempio controllo sulla mail:



3.4 Diagramma di deployment

zip → DeploymentDiagram

3.5 Specifica e verifica dei vincoli

OCL e JML con riferimento al diagramma delle classi di programma

TOTEM----------

context Totem
inv: ID >= 0
inv: idUtente >= 0

context TotemInterface::getPostoBici(String modello): int
post: if !(self.Totem.RastrellieraInterface.Rastrelliera.Bici -> exists(Bici b |b.getModello = modello))
then result = -1

context TotemInterface:: getBiciArray(): Bici[]
post: result = self.Totem.RastrellieraInterface.Rastrelliera.Bici //collezione

context TotemInterface:: getIdUtenteLoggato(): String
post: result = self.Totem.ID

context TotemInterface:: getModelloBici(int id): String
post: if ! self.Totem.RastrellieraInterface.Rastrelliera.Bici -> contains(Bici b |b.getCodiceUnivoco = id)
then result = null

context TotemInterface::calcolaImportoNoleggio(long minutiTrascorsi, String modello, boolean studente) : double
post: result >= 0

RASTRELLIERA:----------

//@ requires r.length == 8;   <-jml
context Rastrelliera::Rastrelliera(Bici[] r)
pre: r.length = 8

//@ ensures \result >=0 && \result <= 8; <-jml
context RastrellieraInterface::disponibilitàBiciclette(): int
post: result >=0 && result <= 8

context RastrellieraInterface::getPostiLiberi(): ArrayList<Integer>
post: result-> forAll(Integer i | i >=0 && i<=7) // i posti vanno da 0 a 7

//@ ensures \result.length == 8; <-jml
context RastrellieraInterface::disponibilitàBiciArray(): Bici[]
post:result.length = 8

//@ensures \result== -1 ==> !(\exists int j; j >= 0 && j<r.length; r[j].getModello() == modelloBici) ; <- jml
context RastrellieraInterface::getPosto(String modelloBici): int
post: if !(self.Rastrelliera.Bici-> exists(Bici b| b.getModello = modelloBici)) then return -1

//@ensures posto >=0 && posto< 8   <-jml
context RastrellieraInterface::getBici(int posto): Bici
pre: posto >=0 && posto <8

//@ensures posto >=0 && posto< 8   <-jml
context RastrellieraInterface::sblocca(int posto)
pre: posto >=0 && posto <8

BICI----------

context BiciNormale

inv: modello = 'normale'

context BiciElettrica
inv: modello = 'elettrica'

context BiciElettricaSeggiolino
inv: modello = 'elettrica con seggiolino'

context BiciElettrica::sprint()
post: self.batteria =batteria@pre - 1

context BiciElettrica::ricarica()
post: self.batteria = 100

ABBONAMENTO ----------

context AbbonamentoGiornaliero
inv: SADO_DEFAULT = 100

context AbbonamentoSettimanale
inv: SADO_DEFAULT = 100

context AbbonamentoAnnuale
inv: SADO_DEFAULT = 100

//analogo per AbbonamentoSettimanale e AbbonamentoAnnuale per l'annuale deve scadere dopo 365days
context AbbonamentoGiornaliero::creaAbbonamentoFirstTime(String nome, String password, String numeroCarta, Date dataValidità,boolean studente, String tipo): String
pre: dataValidità > current_date + 30 days //dataValidità indica la validità della carta di credito dello user

3.6 Descrizione del testing

I test drivers sono 3 uno per ogni elemento principale del software.
Il criterio utilizzato per il testing è stato quello di copertura delle istruzioni, si è cercato di coprire la maggior parte del codice possibile evitando il testing dei controller che generano javaFx, questo perchè in essi non sono presenti operazioni “intelligenti” se non quelle di controllo di input, la vera struttura del programma è data dalle altre classi, che sono state testate cercando di non generare operazioni tramite il database che potrebbero provocare incosistenza dei dati, per questo alcuni metodi e operazioni non vengono testati o perchè troppo dinamici (esempio la struttura della rastrelliera) o perchè provocherebbero incosistenze nel database(esempio inizio noleggio e riconsegna del totem).

3.7 Note per l'installazione e l'utilizzo

Ambiente di sviluppo e compilazione: eclipse
Version: 2020-12 (4.18.0)
Build id: 20201210-1552

Credenziali per accesso al database : utente: postrgres password: postgres
account di test:
Root: codice accesso: 1 password: prova (da tabella impiegati)
Utente Prova: codice accesso totem: 4714186072 password:1 (da tabella utenti)
Nel login per il noleggio del totem va inserito il codice univoco rilasciato dopo l'abbonamento NON il nome dell'utente. Per il rinnovo dell'abbonamento (solo se il precedente scaduto) va inserito il nome dell'utente e la password

Il codice relativo al progetto è nella cartella (CODICE APPLICATIVO).
il codice è formato da 3 progetti eclipse che rappresentano i 3 sw indipendenti.

Sono presenti i dump dei databases con i rispettivi nomi usati nel progetto.

Per utilizzare un totem piuttosto che un altro basta cambiare l'id di esso(attributo) nel codice(nella classe Totem), esso dovrà matchare con gli id dei totem presenti nel db.
Si ricorda che la rastrelliera viene inizializzata a partire dal login nel Totem, se è necessario usar più totem può essere fatto facendo partire più programmi tramite il main, basta che il secondo o terzo totem ecc.. sia avviato dopo che è stato cambiato l'id dopo il login del totem precedente.
Altrimenti con dei file eseguibili non si avrebbe questo problema(quello che mi aspetto nella realtà), questo perchè l'id non verrebbe cambiato in corso di esecuzione del programma.
Per la prova sono stati inseriti solamente due totem id 1 e 2.