

Sistema di Bike Sharing

1 Descrizione del problema

1.1 Analisi e specifica dei requisiti

Il sistema di bike sharing è previsto solo per utenti in possesso di un **abbonamento**.

Tre tipologie di abbonamento:

- Giornaliero(Ha inizio nel momento in cui si preleva la prima bici)
- Settimanale(Ha inizio nel momento in cui si preleva la prima bici)
- Annuale(Ha inizio da subito)

L'unico metodo di pagamento è la **carta di credito**.

Il sistema è in grado di produrre alcuni dati statistici di base sugli abbonamenti scelti dagli utenti.

Tre tipologie di bicicletta:

- Normale(30 minuti gratuito, poi si aggiunge un supplemento in base all'utilizzo. Per gli **studenti** è sempre gratuito.)
- Elettrica(tariffa a consumo)
- Elettrica con seggiolino(tariffa a consumo)

Le biciclette sono posizionate su delle rastrelliere e agganciate alla propria tipologia di **morsa**.

Ogni **rastrelliera** ha un **totem** con cui l'utente può interagire per il ritiro e la restituzione delle bici.

Ulteriori dettagli sull'utilizzo delle biciclette:

- Tempo di utilizzo massimo per noleggio: **2 ore**
- 150 euro di penale, sommati al consumo, per la non restituzione della bicicletta entro i limiti orari del noleggio specificati poc'anzi.

Fase di registrazione abbonamento: L'**utente** inserisce i propri dati anagrafici, una carta di credito, sceglie un tipo di abbonamento e una *password*. Il sistema restituisce un *codice univoco* valido per la durata dell'abbonamento.

Fase di ritiro della bicicletta: viene inserito il *codice univoco* ottenuto durante la registrazione dell'abbonamento e la propria *password* scelta sul totem della rastrelliera. Il sistema risponderà con il *numero di posteggio della bici* che verrà sbloccata.

Fase di restituzione della bicicletta: la bici viene restituita in un aggancio libero di una rastrelliera e di corretta tipologia per la bici in questione. Verifica della corretta consegna tramite l'inserimento del codice utente sul totem, il sistema risponderà con un messaggio di conferma. *Inoltre il sistema calcolerà l'ammontare in denaro dovuto dall'utente per l'utilizzo.*

Controllo sblocco morsa: dalla fase di ritiro si evince il fatto che vi sarà una componente che svolgerà il ruolo di verificare che le operazioni vengano fatte in modo corretto così da garantire lo sblocco della morsa per il prelievo delle biciclette.

Controllo restituzione: un'altra componente dovrà occuparsi di controllare che la restituzione della bicicletta avvenga in modo corretto. Dunque verifica che almeno un posto sia effettivamente libero e che il codice utente corrisponda allo stesso usato per il prelievo.

Controllo accesso su rastrelliera: non solo gli utenti del servizio devono poter accedere ai totem delle rastrelliere ma anche **membri del personale** per spostare eventualmente delle biciclette da una rastrelliera piena a un'altra con poche bici disponibili. Dunque il totem deve essere in grado di capire che tipo di utente sta effettuando l'accesso.

Interfaccia grafica:

- Tre bottoni *registrazione*, *accesso rastrelliera* e *restituzione* nella finestra di avvio.
- Registrazione porta a una finestra dedicata dove poter acquistare un abbonamento.
- Accesso rastrelliera porta a una finestra dedicata in cui inserendo i propri dati e il numero della rastrelliera, si accede ad essa.

Una volta sulla finestra relativa alla rastrelliera scelta vi sono due casi:

- Se si tratta di un utente normale, potrà scegliere quale bici prelevare.
- Se si tratta di un membro del personale, potrà effettuare lo spostamento di una bicicletta da quella rastrelliera a un'altra e ripetere l'azione in caso se ne vogliano spostare altre.

Se si clicca su restituzione:

Vi sarà una casella di testo dove specificare la rastrelliera in cui si sta restituendo la bici con annesso bottone per confermare e una casella di testo dove inserire il proprio codice utente per verificare la corretta consegna con un bottone per la verifica.

Ulteriori dettagli sui controlli e alcuni esempi di schermata sono riportati nel paragrafo dedicato.

Scelte effettuate sulle tariffe biciclette/abbonamenti:

Per quanto riguarda le tariffe delle biciclette è stato deciso:

bicicletta normale: 0.50 € ogni mezz'ora(escludendo la mezz'ora gratuita).

Bicicletta elettrica e con seggiolino: 1.50 € ogni mezz'ora(con un parametro che definisce se c'è o meno il seggiolino).

Non sono state applicate modifiche per quanto riguarda i prezzi degli abbonamenti, ovvero 4.50 per il giornaliero, 9 per il settimanale e 36 per l'annuale.

Salvataggio dati persistenti:

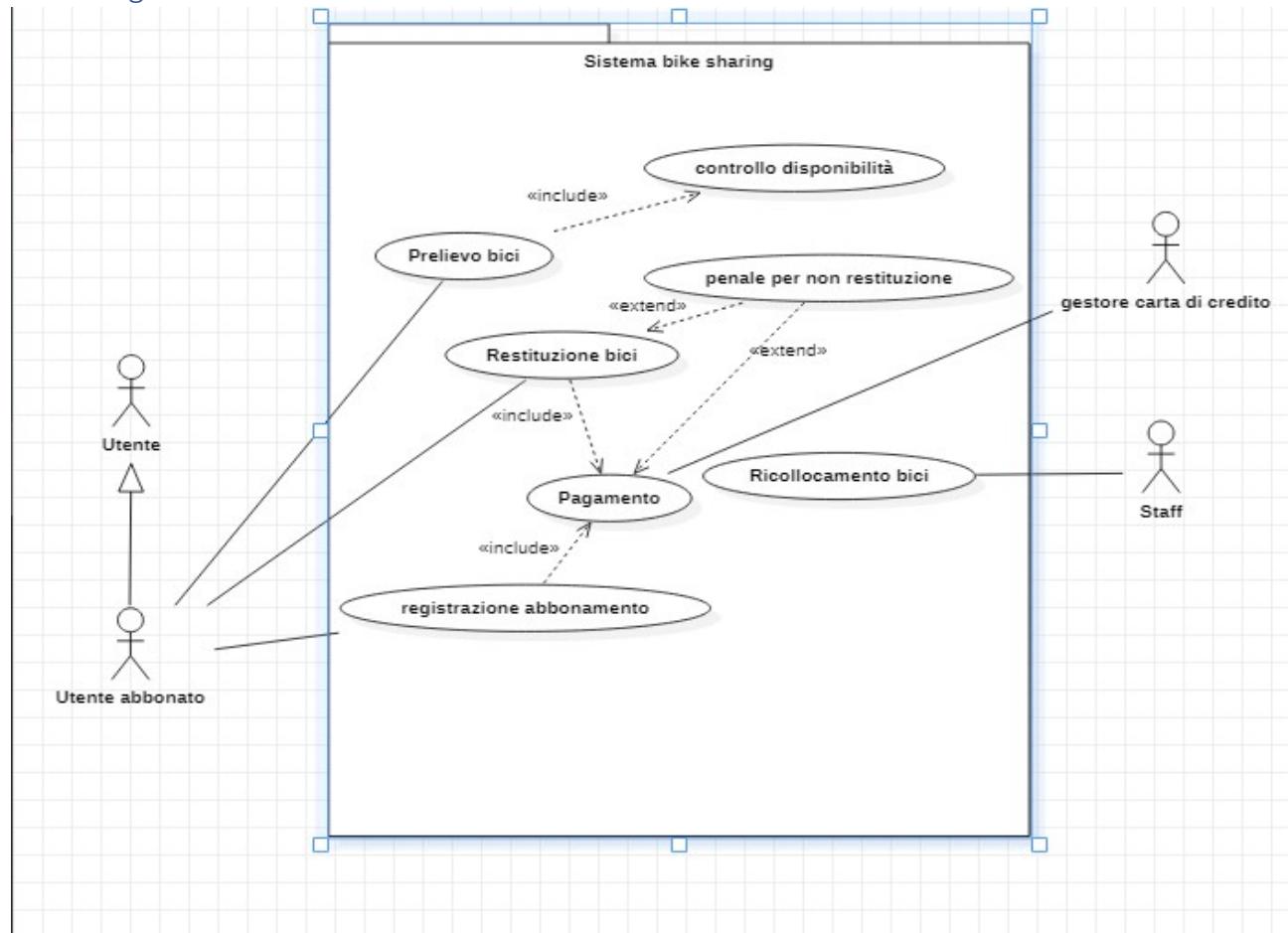
È stato scelto MySQL come DBMS ed è stato deciso che i dati da salvare in modo persistente sono quelli che riguardano gli utenti, le rastrelliere e le biciclette(che siano in possesso di un utente o posteggiate).

È possibile trovare sia il file .jar connettore necessario per collegarsi al database, sia i file dump necessari per riprodurre il database costruito. All'interno vi sono dei dati pre-inseriti che verranno specificati nel dettaglio nei paragrafi dedicati insieme a tutta la struttura della base di dati.

In fondo alla relazione sarà possibile visionare ulteriori dettagli per quanto riguarda la compilazione e esecuzione dell'applicazione.

2 Progettazione del Sistema

2.1 Diagramma dei casi d'uso



Prelievo bici: un utente in possesso di un abbonamento valido inserisce il proprio codice utente e la password per ottenere lo sblocco di una morsa e utilizzare una bicicletta.

Controllo disponibilità: il sistema controlla se vi sono biciclette disponibili e se l'abbonamento dell'utente è valido.

Restituzione bici: un utente che in precedenza aveva prelevato una bici da una rastrelliera, la restituisce presso la stessa o un'altra rastrelliera.

Pagamento: Ogni volta che viene restituita una bici o registrato un nuovo abbonamento, l'utente è tenuto a effettuare un pagamento.

Registrazione abbonamento: un utente che è intenzionato a utilizzare il servizio di bike sharing è obbligato a sottoscrivere un abbonamento fornendo i propri dati anagrafici e una password.

Penale per non restituzione: un utente che non abbia restituito la bici prelevata entro il tempo massimo di noleggio che è di 2 ore, è tenuto a pagare una penale oltre ai costi dell'utilizzo.

Ricollocamento bici: un membro del personale che accede a una rastrelliera ha i privilegi che lo autorizzano a ricollocare una bici da una rastrelliera all'altra dove è necessario.

2.2 Descrizione degli scenari

Nome	Registrazione abbonamento
Scopo	Acquisto di un abbonamento da parte di un utente per usare il servizio
Attore/i	Utente standard
Pre-condizioni	Non essere in possesso di un abbonamento
Trigger	Voler utilizzare il servizio

Descrizione sequenza eventi	<ol style="list-style-type: none"> 1. l'utente clicca il bottone per la registrazione. 2. Il sistema accoglie la richiesta e chiede i dati necessari all'utente. 3. L'utente inserisce i dati anagrafici, sceglie una password, il tipo di abbonamento, i dati della carta di credito. 4. L'utente clicca il bottone di conferma. 5. Il sistema in base al tipo di abbonamento scelto scala il denaro dalla carta di credito dell'utente. 6. Il sistema restituisce il codice utente univoco
Alternativa/e	<p>2a. Il sistema rifiuta la richiesta poiché alcuni dati inseriti non sono corretti/validi.</p> <p>3. Il sistema chiude la comunicazione.</p>
Post-condizioni	L'utente è in possesso di un abbonamento valido

Nome	Prelievo bici
Scopo	Prelievo di una bici da una rastrelliera da parte di un utente
Attore/i	Utente abbonato
Pre-condizioni	Utente in possesso di abbonamento;bici disponibile alla rastrelliera.
Trigger	Utente che prova ad accedere alla rastrelliera.
Descrizione sequenza eventi	<ol style="list-style-type: none"> 1. L'utente clicca il bottone per accedere alla rastrelliera. 2. Il sistema richiede codice utente, password e numero rastrelliera. 3. L'utente inserisce i dati richiesti. 4. L'utente clicca il bottone di conferma. 5. Il sistema mostra le bici disponibili sulla rastrelliera. 6. L'utente sceglie il tipo di bici da noleggiare cliccando il bottone. 7. Il sistema sblocca la morsa consentendo il prelievo della bici.
Alternativa/e	<p>5a. Il sistema rifiuta la richiesta perché la password è errata.</p> <p>5b. Il sistema rifiuta la richiesta perché l'abbonamento associato al codice utente è scaduto e quindi non risulta più valido.</p> <p>6. Il sistema chiude la comunicazione.</p>
Post-condizioni	Vi è una bici in meno nella rastrelliera.

Nome	Restituzione bici
Scopo	Restituzione di una bici presso una rastrelliera da parte di un utente.
Attore/i	Utente abbonato
Pre-condizioni	Essere in possesso della bicicletta prelevata
Trigger	Voler restituire la bici
Descrizione sequenza eventi	<ol style="list-style-type: none"> 1. l'utente clicca il bottone per la restituzione bici. 2. Il sistema chiede numero rastrelliera e codice utente. 3. l'utente inserisce i dati richiesti. 4. l'utente clicca il bottone di conferma. 5. Il sistema restituisce una conferma della corretta restituzione. 6. L'utente paga l'importo dovuto. 7. Il sistema chiude la comunicazione.
Alternativa/e	<p>2a. Il sistema rifiuta la richiesta perché l'utente non è in possesso di una bici.</p> <p>4a. Il sistema restituisce un messaggio di errore perché il numero della rastrelliera non è valido.</p> <p>4b. Il sistema restituisce un messaggio di errore perché la rastrelliera non</p>

	ha posti disponibili per depositare la bicicletta.
Post-condizioni	Vi è una bici in più nella rastrelliera.

Nome	Ricollocamento bici
Scopo	Spostamento di una bici da una rastrelliera a un'altra
Attore/i	Staff
Pre-condizioni	Essere un membro dello staff
Trigger	Voler spostare una bici
Descrizione sequenza eventi	<p>1. l'utente clicca il bottone per l'accesso alla rastrelliera.</p> <p>2. Il sistema richiede codice utente, password e numero rastrelliera.</p> <p>3. l'utente inserisce le proprie credenziali.</p> <p>4. l'utente clicca il bottone di conferma.</p> <p>5. il sistema mostra i dati relativi alla rastrelliera.</p> <p>6. l'utente decide quale bici intende spostare.</p> <p>7. l'utente clicca il bottone per spostare la bici.</p> <p>8. Il sistema chiede su quale rastrelliera spostare la bici.</p> <p>9. l'utente inserisce il numero di rastrelliera sulla quale spostare.</p> <p>10. il sistema conferma l'avvenuto ricollocamento.</p>
Alternativa/e	<p>5a. Il sistema restituisce un messaggio di errore per credenziali non valide.</p> <p>10a. Il sistema non convalida il ricollocamento perché la rastrelliera sulla quale si vuole spostare la bici è già piena.</p>
Post-condizioni	Ho una bici in meno su una rastrelliera e una bici in più in un'altra rastrelliera.

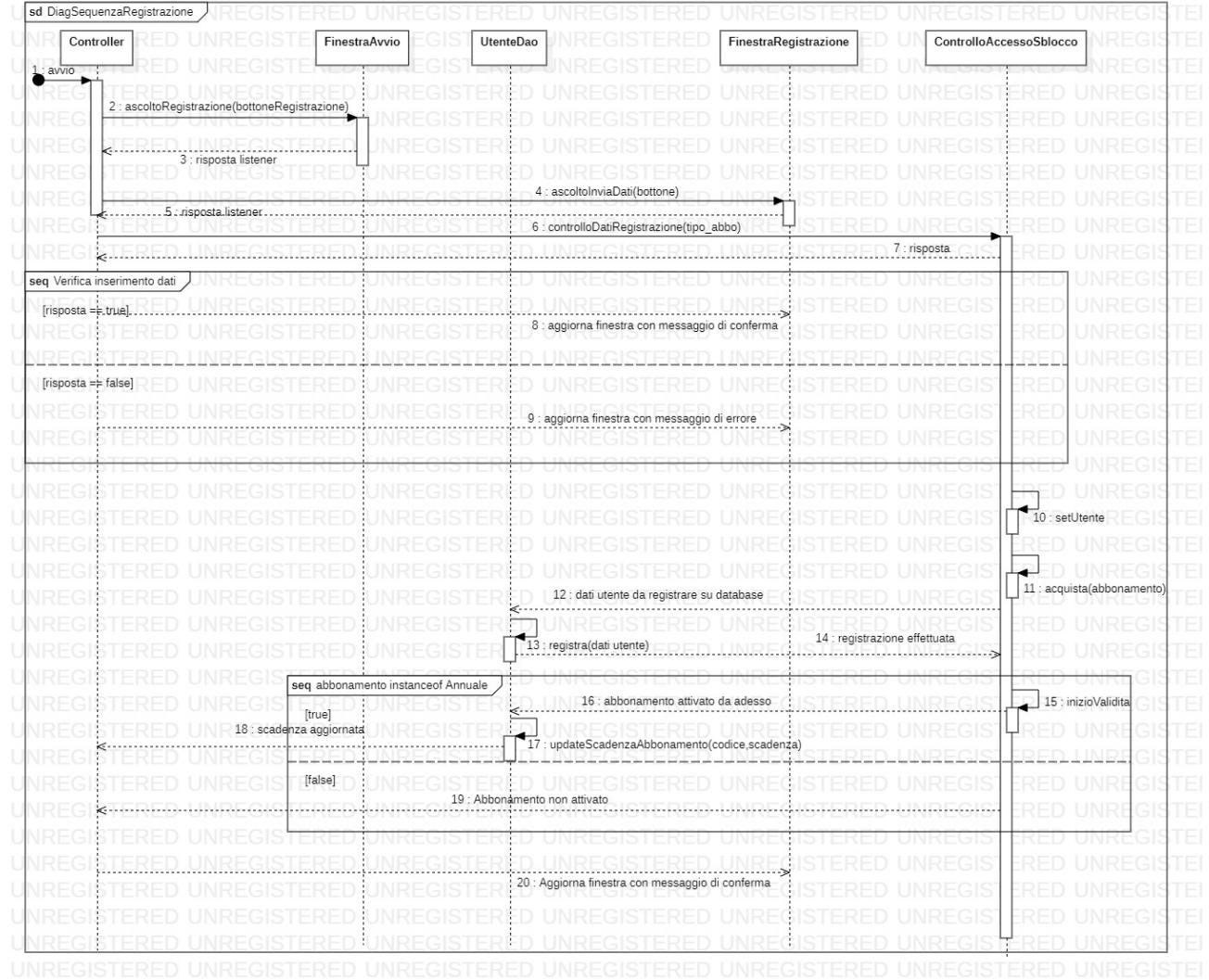
È stato deciso di riportare un unico diagramma delle classi (quello di programma), in esposizione più avanti nella relazione al relativo paragrafo. Qui di seguito sono illustrati i diagrammi di sequenza dei casi d'uso: Registrazione, Prelievo bici, Restituzione e ricollocamento bici.

Dunque all'interno di questi diagrammi di sequenza e nei diagrammi di attività poco più avanti sono già presenti le classi che utilizzano i design pattern scelti per la progettazione, i quali verranno giustificati e esposti nel paragrafo dedicato.

Se le immagini di alcuni diagrammi risultano poco leggibili per via della loro risoluzione abbassata a causa del pdf, sotto ognuno di essi c'è scritto il nome del file immagine originale esportato da starUML, disponibile all'interno della cartella "immagini diagrammi".

Inoltre, sempre nella cartella appena citata, sarà possibile trovare il file .mdj nativo con la lista dei diagrammi UML, nel caso nemmeno le immagini fossero sufficientemente leggibili.

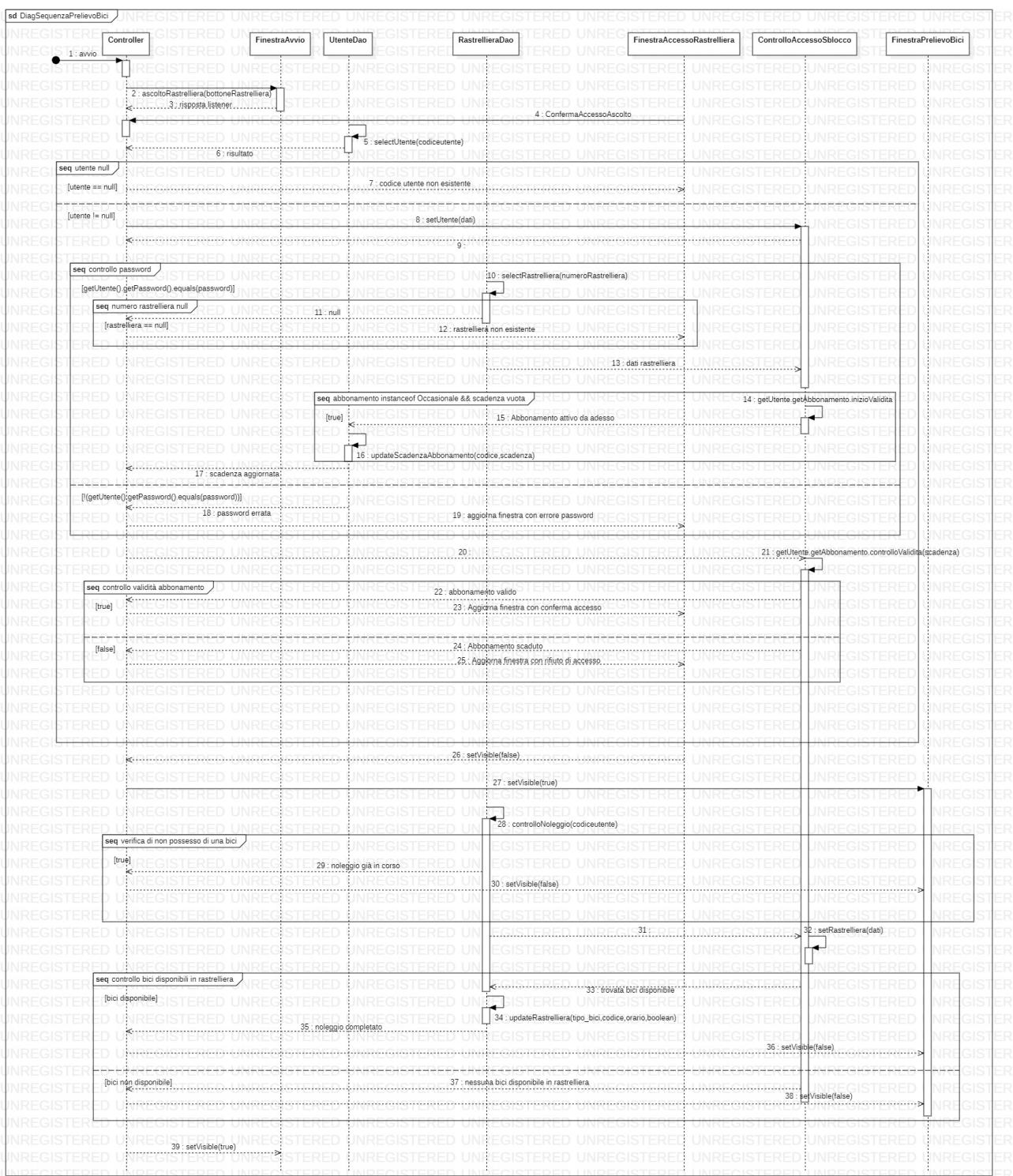
2.3 Diagrammi di sequenza



[DiagSequenzaRegistrazione.jpg]

Il primo diagramma di sequenza esposto rappresenta il caso d'uso relativo alla **registrazione di un abbonamento**.

Cliccando sul bottone “registrazione” viene eseguito il metodo **ascensoRegistrazione()** che prende come argomento una classe ActionListener e il Controller grazie a questa componente, intercetta il click del bottone e apre la finestra dedicata alla registrazione. L’utente provvederà all’inserimento dei campi richiesti nelle caselle di testo e una volta fatto, cliccherà il bottone per confermare l’invio dei dati che farà intervenire un altro ActionListener sul Controller. A questo punto, viene effettuato un controllo sul corretto inserimento del tipo di abbonamento scelto e in caso non lo sia, viene restituito un messaggio di errore e l’utente potrà modificare l’inserimento e riprovare. Questo avviene tramite il metodo **controlloDatiRegistrazione()** della classe ControlloAccessoSblocco. Una volta passato questo controllo, viene settato un oggetto di tipo utente con i dati inseriti in precedenza e viene effettuato l’acquisto dell’abbonamento scelto scalando il denaro dal residuo della carta di credito(ogni utente appena generato ha 1000 € di default). L’acquisto viene effettuato tramite l’oggetto utente settato con il metodo **acquista()** nella classe ControlloAccessoSblocco. Fatto ciò l’utente viene registrato sul database e in caso l’abbonamento scelto sia di tipo Annuale, viene subito attivato, altrimenti no. L’attivazione di un abbonamento avviene tramite il metodo **inizioValidita()** dentro l’oggetto utente nella classe ControlloAccessoSblocco. Il tutto si conclude con un messaggio di conferma della registrazione, tornando poi al menù principale di avvio.



[DiagSequenzaPrelevobici.jpg]

Questo diagramma di sequenza illustra il caso d'uso relativo al **prelevobici**.

Cliccando sul bottone “accesso rastrelliera” viene eseguito il metodo **ascoltoRastrelliera()** che prende come argomento una classe ActionListener e il **Controller** grazie a questa componente, intercetta il click del bottone e apre la finestra dedicata all’accesso presso una rastrelliera.

L’utente provvede a inserire i dati richiesti per accedere. Viene eseguito il metodo **selectUtente()** dalla classe UtenteDao che non fa altro che prelevare i dati dal database in base al codice utente

inserito. Qui viene effettuato un controllo poiché se il risultato è null, viene restituito che il codice utente inserito non esiste sul sistema.

Altrimenti, si prosegue con i controlli e viene verificato che la password inserita sia corretta, altrimenti viene restituito un messaggio di password errata. Fatto questo viene controllato se l'abbonamento utilizzato sia di tipo occasionale e se non abbia ancora la data di scadenza. In questo caso si tratta del primo utilizzo quindi viene attivato con il metodo ***inizioValidita()*** dell'oggetto utente all'interno della classe ControlloAccessoSblocco.

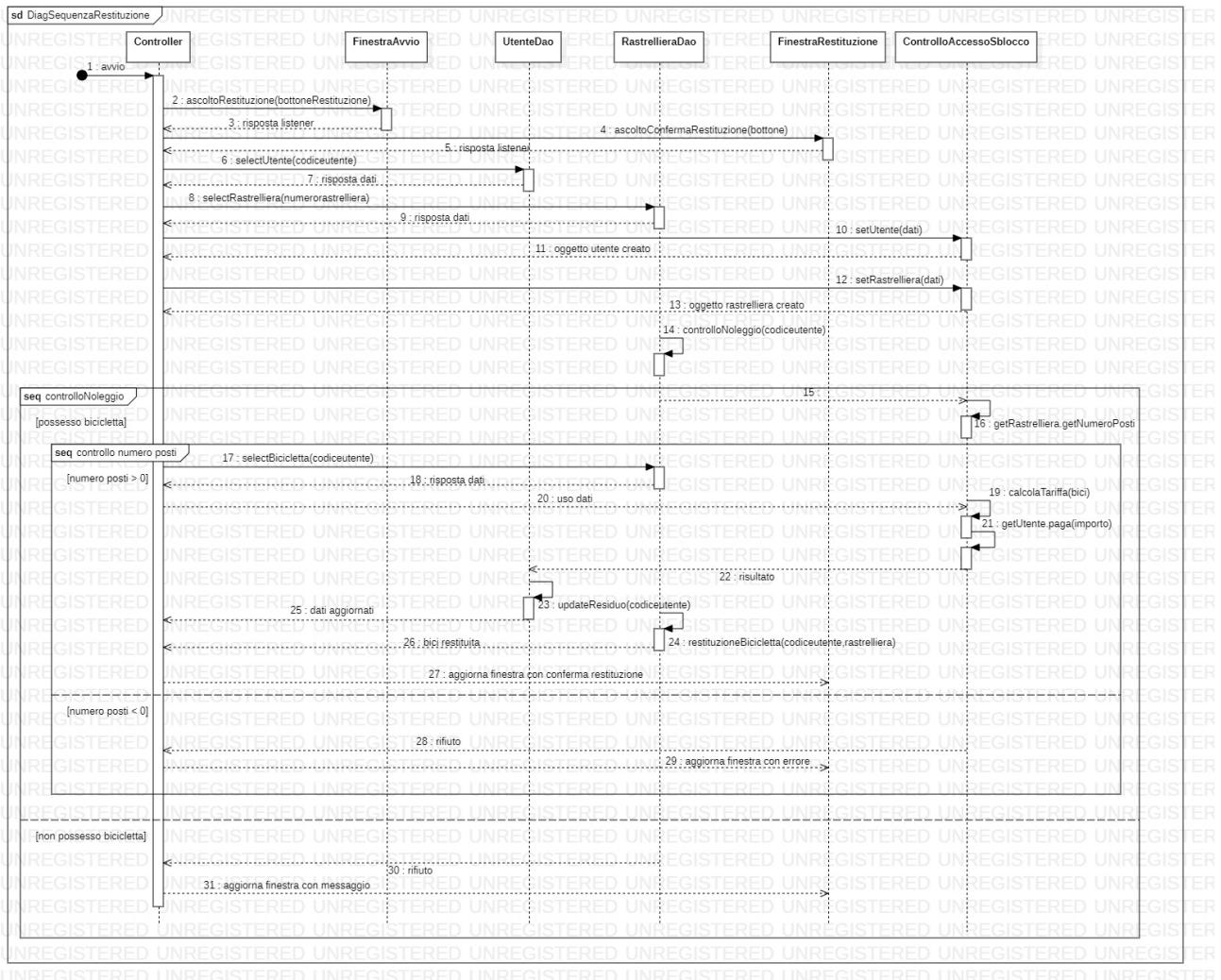
Passato anche questo controllo viene verificato che il numero della rastrelliera inserito sia di un oggetto rastrelliera esistente, altrimenti viene restituito un messaggio di errore. Questo viene controllato tramite il metodo ***selectRastrelliera()*** della classe RastrellieraDao.

Infine viene anche controllato se l'abbonamento dell'utente è ancora valido o se risulta scaduto. Questo avviene utilizzando il metodo ***controlloValidita*** nella classe ControlloAccessoSblocco. Se sono stati superati tutti questi controlli, allora il sistema chiude la finestra di accesso e apre quella dedicata al noleggio delle bici puntata sui dati della rastrelliera scelta.

L'utente potrà scegliere tra i tre tipi di bicicletta e cliccando uno dei tre button farà scattare un controllo che verificherà prima, che l'utente in questione non stia già noleggiando una bicicletta e poi, che in quella rastrelliera ci sia una bici disponibile di quel tipo.

Nel caso in cui non ci fossero biciclette disponibili, l'utente potrà accedere a una rastrelliera diversa e riprovare.

Una volta prelevata la bici, verranno aggiornati i valori sul database relativi a quella bicicletta che ora non è più disponibile ma è in possesso di un utente marchiata anche con l'orario in cui è stata presa in noleggio. Quest'ultima azione viene effettuata tramite il metodo ***updateRastrelliera()*** della classe RastrellieraDao.



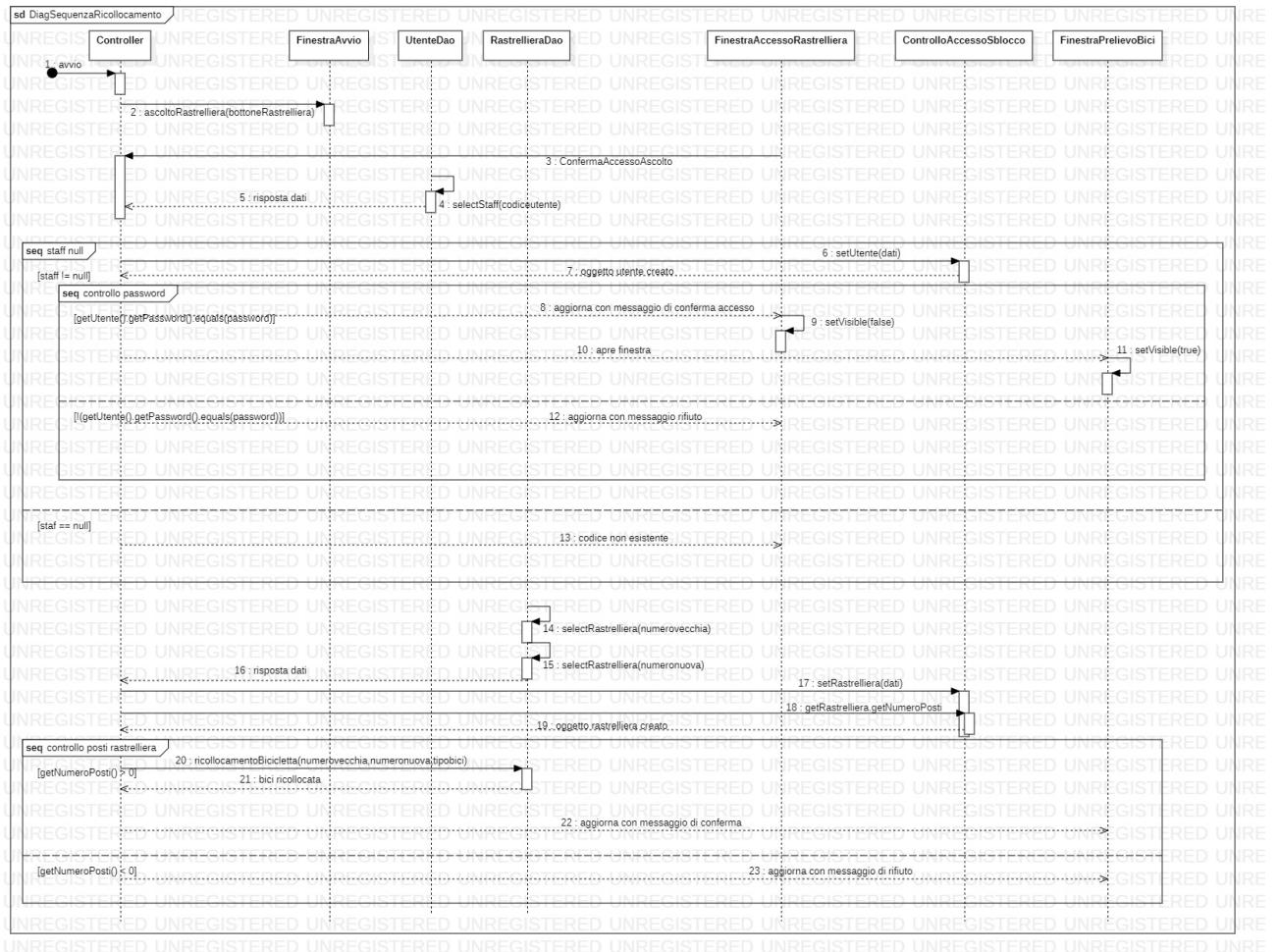
[DiagSequenzaRestituzione.jpg]

Il diagramma di sequenza sopra illustrato mostra il caso d'uso della **restituzione** di una bici. Cliccando sul bottone Restituzione viene eseguito il metodo **ascoltoRestituzione()** che prende come argomento una classe ActionListener e il **Controller** grazie a questa componente, intercetta il click del bottone e apre la finestra dedicata alla restituzione di una bicicletta precedentemente presa in noleggio. Dopo aver recuperato i dati inseriti nelle caselle di testo(codice utente e numero di rastrelliera), le classi **UtenteDao** e **RastrellieraDao** recuperano i dati dal database rispettivamente con **selectUtente()** e **selectRastrelliera()** così da effettuare successivamente i controlli necessari.

A questo punto, dopo aver prelevato le informazioni dalla base di dati, vengono eseguiti **setUtente()** e **setRastrelliera()** dalla classe **ControlloAccessoSblocco** creando così i due oggetti che serviranno in seguito.

Viene inizialmente fatto un controllo sull'effettivo possesso di una bicicletta da restituire da parte dell'utente tramite **controllo Noleggio()** e in caso di valore di ritorno a true, si prosegue altrimenti viene restituito un messaggio di rifiuto della restituzione. Dunque se true, viene effettuato un ulteriore controllo, questa volta sulla rastrelliera, tramite **getRastrelliera().getNumeroPosti()** così da ricavare il numero di posti disponibili e verificare che ce ne sia almeno uno dove depositare la bicicletta. In caso negativo, verrà restituito un messaggio di rifiuto. In caso affermativo invece, vengono prelevate le informazioni specifiche sulla bicicletta con **selectBicicletta()** e poi una volta ottenuto un oggetto di tipo Bicicletta, viene passato al metodo **calcolaTariffa()** per ottenere

l'importo da pagare da passare al metodo **getUtente().paga()** per effettuare il pagamento dovuto per il noleggio. Infine verrà restituito un messaggio di conferma della restituzione.



[DiagSequenzaRicollocaimento.jpg]

Questo diagramma di sequenza rappresenta il caso d'uso del **ricollocamento di bici** da parte di un membro del personale.

Cliccando sul bottone “accesso rastrelliera” viene eseguito il metodo **ascoltoRastrelliera()** che prende come argomento una classe ActionListener e il **Controller** grazie a questa componente, intercetta il click del bottone e apre la finestra dedicata all’accesso presso una rastrelliera.

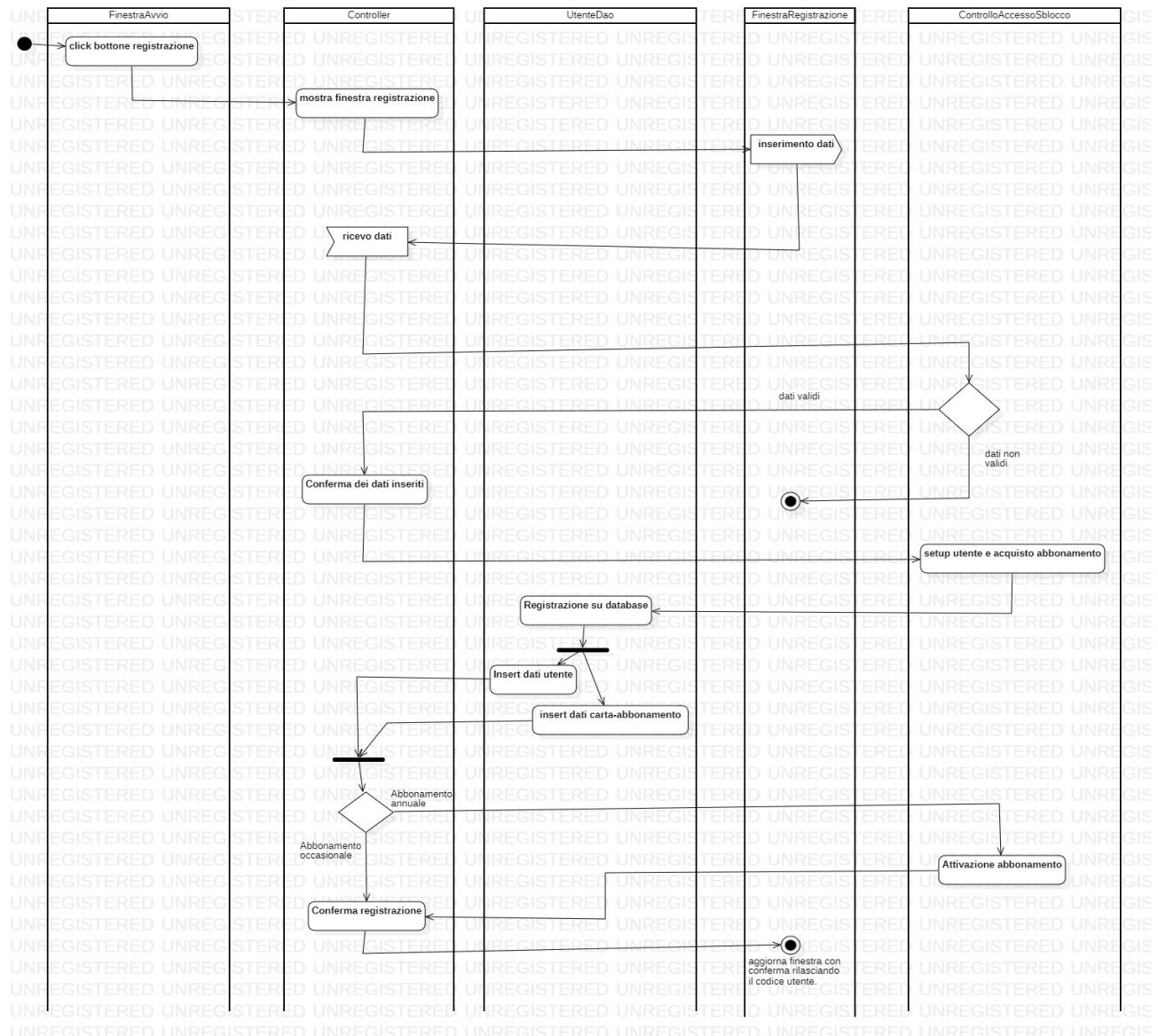
Il membro del personale dunque provvede a inserire le proprie credenziali “speciali” per accedere. Viene eseguito il metodo **selectStaff()** dalla classe *UtenteDao* che non fa altro che prelevare i dati dal database in base al codice utente inserito. Qui viene effettuato un controllo sul risultato dell’esecuzione poiché se il risultato è null, viene restituito un messaggio che comunica la non esistenza del codice utente inserito.

In caso contrario, si prosegue con i controlli e viene verificato che la password inserita sia corretta, altrimenti viene restituito un messaggio di password errata.

A questo punto, vengono recuperati i dati relativi alla rastrelliera alla quale l’utente staff ha fatto accesso tramite l’esecuzione di **selectRastrelliera()**. Attraverso una casella di testo aggiuntiva, è ora possibile inserire il numero della rastrelliera sulla quale depositare la bicicletta che si vuole spostare, dunque verrà eseguito nuovamente **selectRastrelliera()** questa volta sulla rastrelliera appena richiesta. I dati recuperati sulla nuova rastrelliera vengono usati per creare un oggetto di tipo *Rastrelliera* con **setRastrelliera()** e verrà eseguito **getRastrelliera().getNumeroPosti()** dalla classe **ControlloAccessoSblocco** per effettuare un controllo sul numero di posti disponibili nella

rastrelliera target. In caso negativo, viene restituito un messaggio in cui si comunica che la rastrelliera risulta piena e l'utente potrà sceglierne un'altra re-inserendo un nuovo numero oppure uscire dal programma.

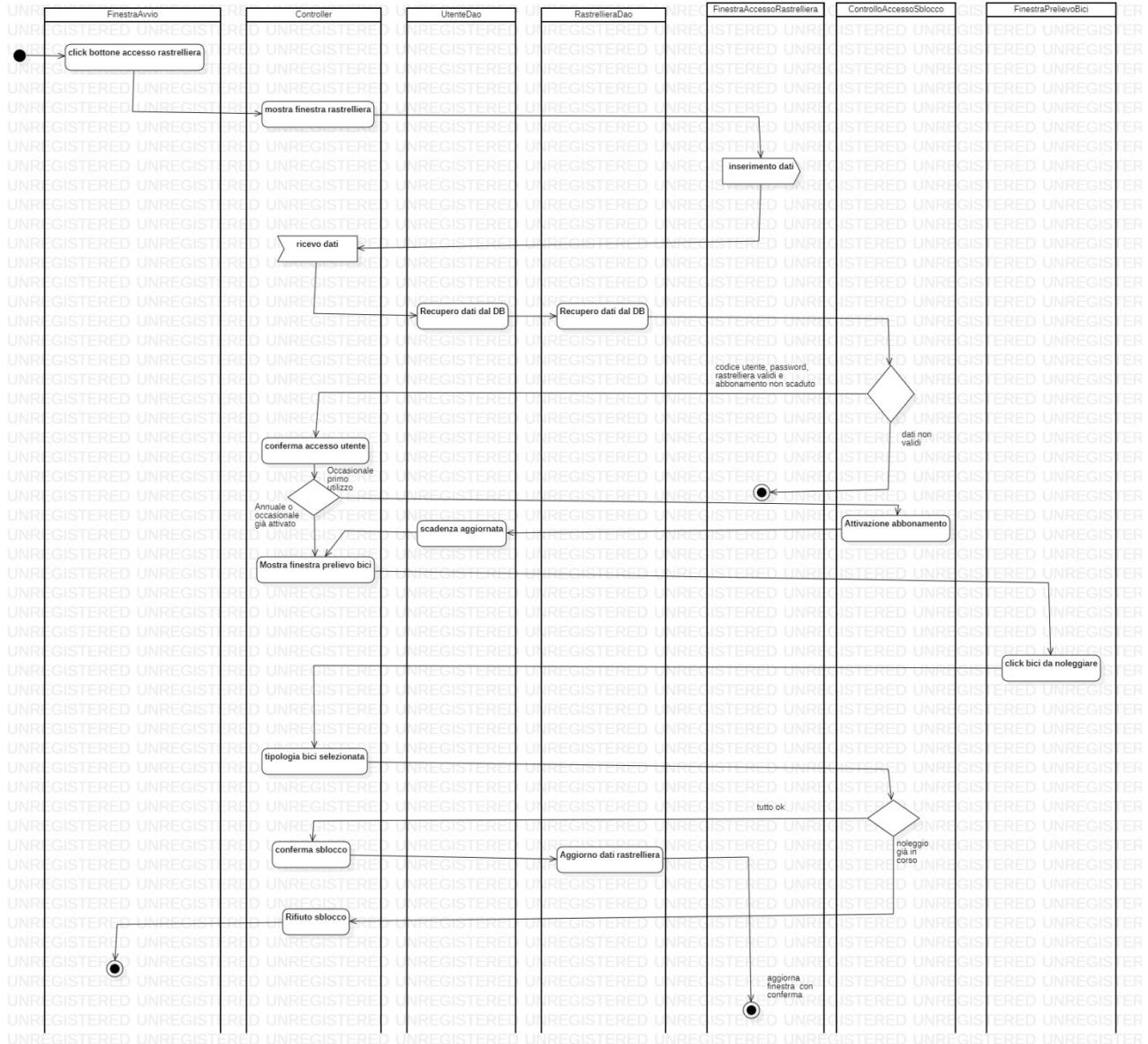
2.4 Diagrammi delle attività



[DiagActRegistrazione.jpg]

Diagramma di attività che illustra i passi che vengono seguiti durante la registrazione di un utente sul sistema.

Viene scelta l'attività di registrazione. L'utente provvede all'inserimento di tutti i dati richiesti e se validi, viene costruito il profilo utente e viene acquistato l'abbonamento scelto in fase di inserimento. Tutti questi dati vengono registrati su una base di dati e in caso di abbonamento annuale, viene attivato immediatamente. Infine viene confermata la registrazione mostrando il codice utente univoco generato.

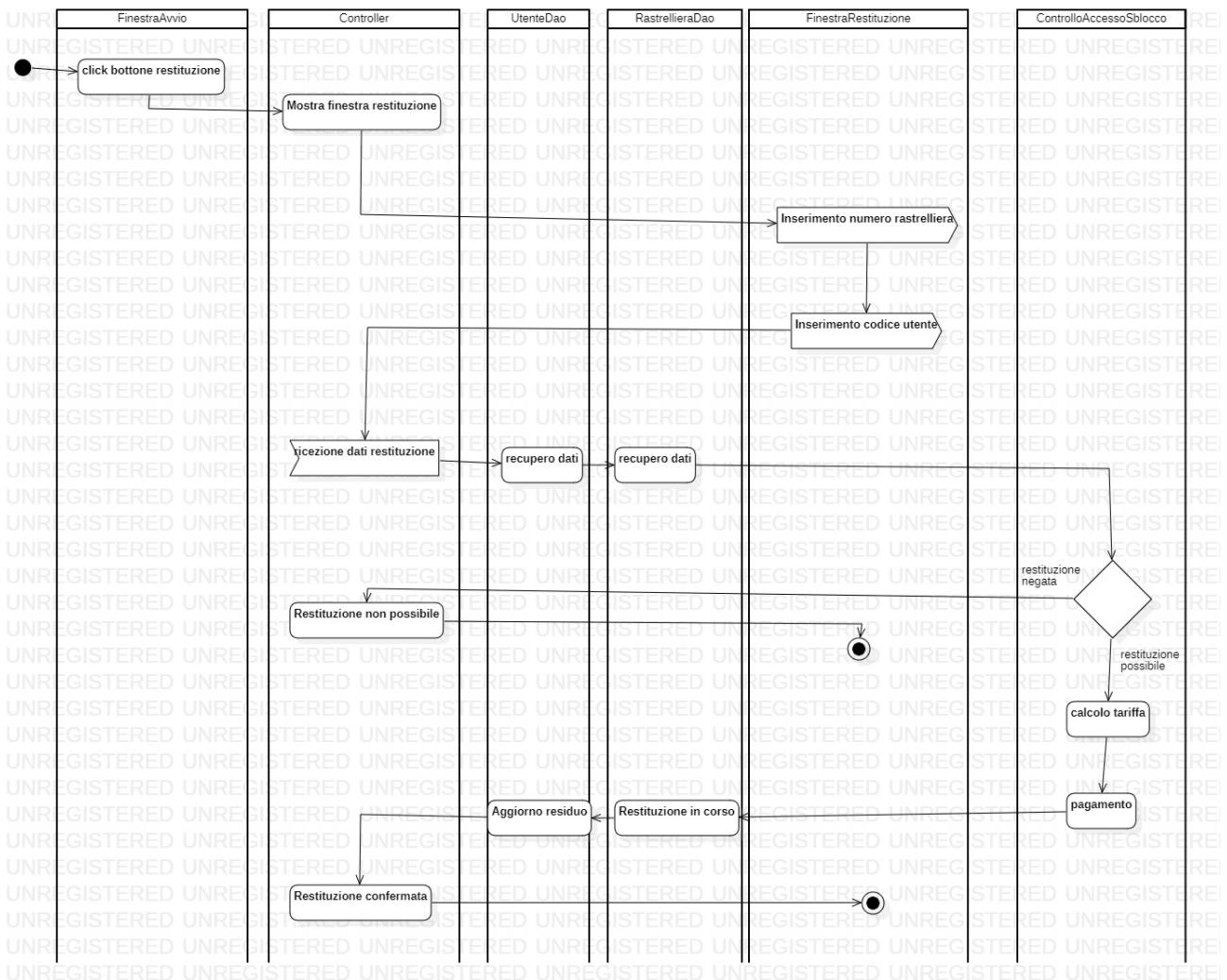


[DiagActPrelievoBici.jpg]

Diagramma di attività che illustra i passi che vengono seguiti durante il prelievo di una bici da parte di un utente sul sistema.

Viene effettuato l'accesso a una rastrelliera tramite l'inserimento del codice utente, password e numero della rastrelliera. Viene controllato se i dati inseriti risultano validi e se l'abbonamento associato al codice utente sia ancora valido oppure no. In caso negativo non viene effettuato l'accesso.

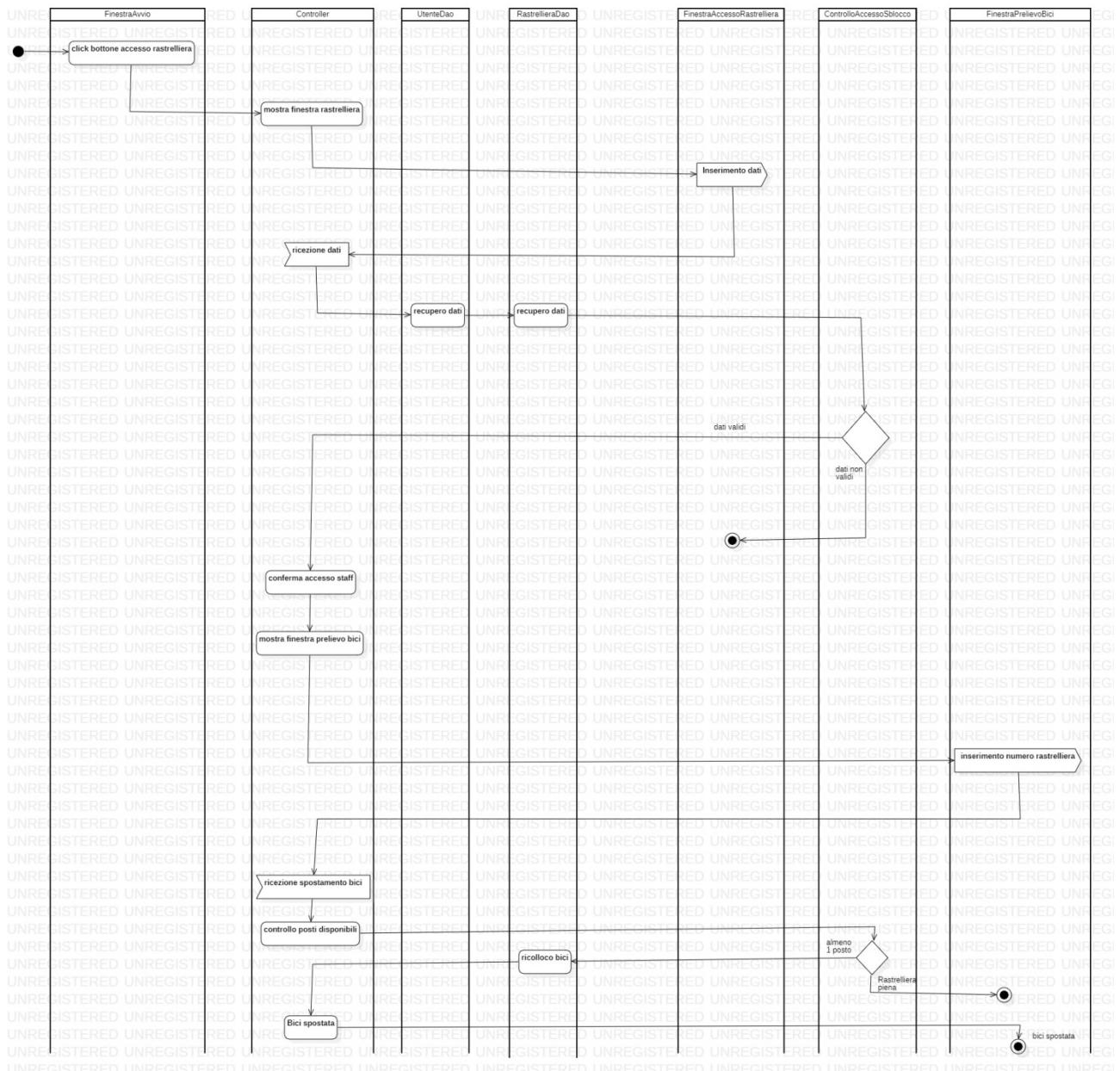
Passato questo controllo, nel caso di un abbonamento occasionale, viene verificato se sia il primo accesso in assoluto e dunque viene attivato aggiornandone la data di scadenza. L'utente può dunque decidere che tipo di bicicletta noleggiare sulla rastrelliera e una volta scelta, la morsa si sbloccherà rilasciandola. Nel caso in cui l'utente avesse fatto accesso ma fosse già in possesso di una bicicletta in noleggio, un ulteriore noleggio viene rifiutato.



[DiagActRestituzione.jpg]

Diagramma di attività che illustra i passi che vengono seguiti durante la restituzione di una bici da parte di un utente seguendo lo scenario del caso d'uso relativo.

Viene scelta l'attività di restituzione. Una volta inseriti codice utente e numero di rastrelliera dove restituire la bicicletta, viene controllato se la restituzione è possibile oppure no. In caso affermativo, viene calcolata la tariffa e effettuato il pagamento dovuto, confermando infine la restituzione avvenuta con successo.

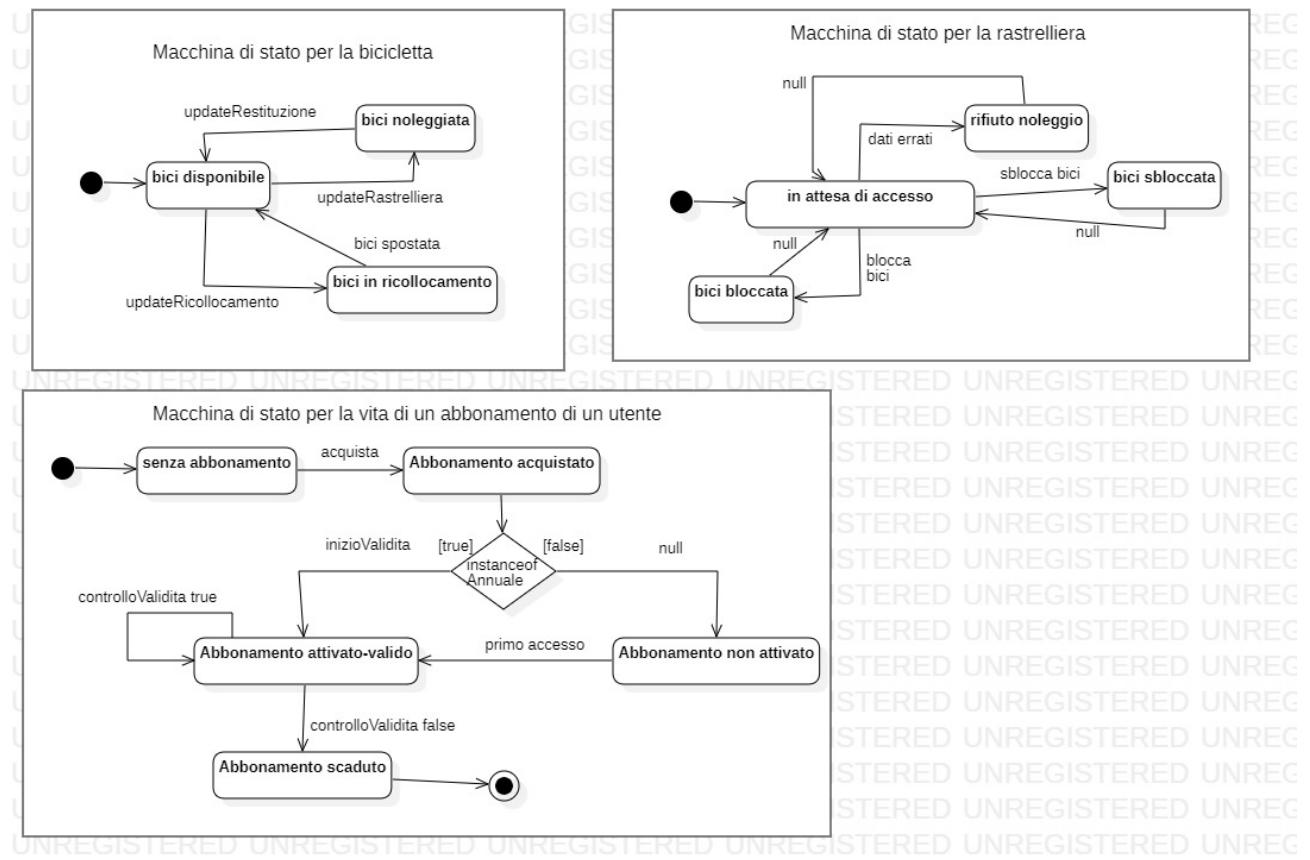


[DiagActRicollocamento.jpg]

Diagramma di attività che illustra i passi che vengono seguiti durante il ricollocamento di una bici da parte di un membro del personale seguendo lo scenario del caso d'uso relativo.

Viene effettuato l'accesso a una rastrelliera da parte di un membro del personale con credenziali speciali. Una volta inserite viene controllata la loro correttezza e in caso affermativo, viene confermato l'accesso speciale come membro dello staff. A questo punto è possibile selezionare il tipo di bici da spostare e successivamente inserire il numero di rastrelliera sulla quale spostarla. Viene dunque fatto un controllo sui posti disponibili della rastrelliera target e se risulta almeno 1 posto libero, viene confermato il ricollocamento della bicicletta, altrimenti viene rifiutato.

2.5 Macchine di stato



[Macchine-di-stato.jpg]

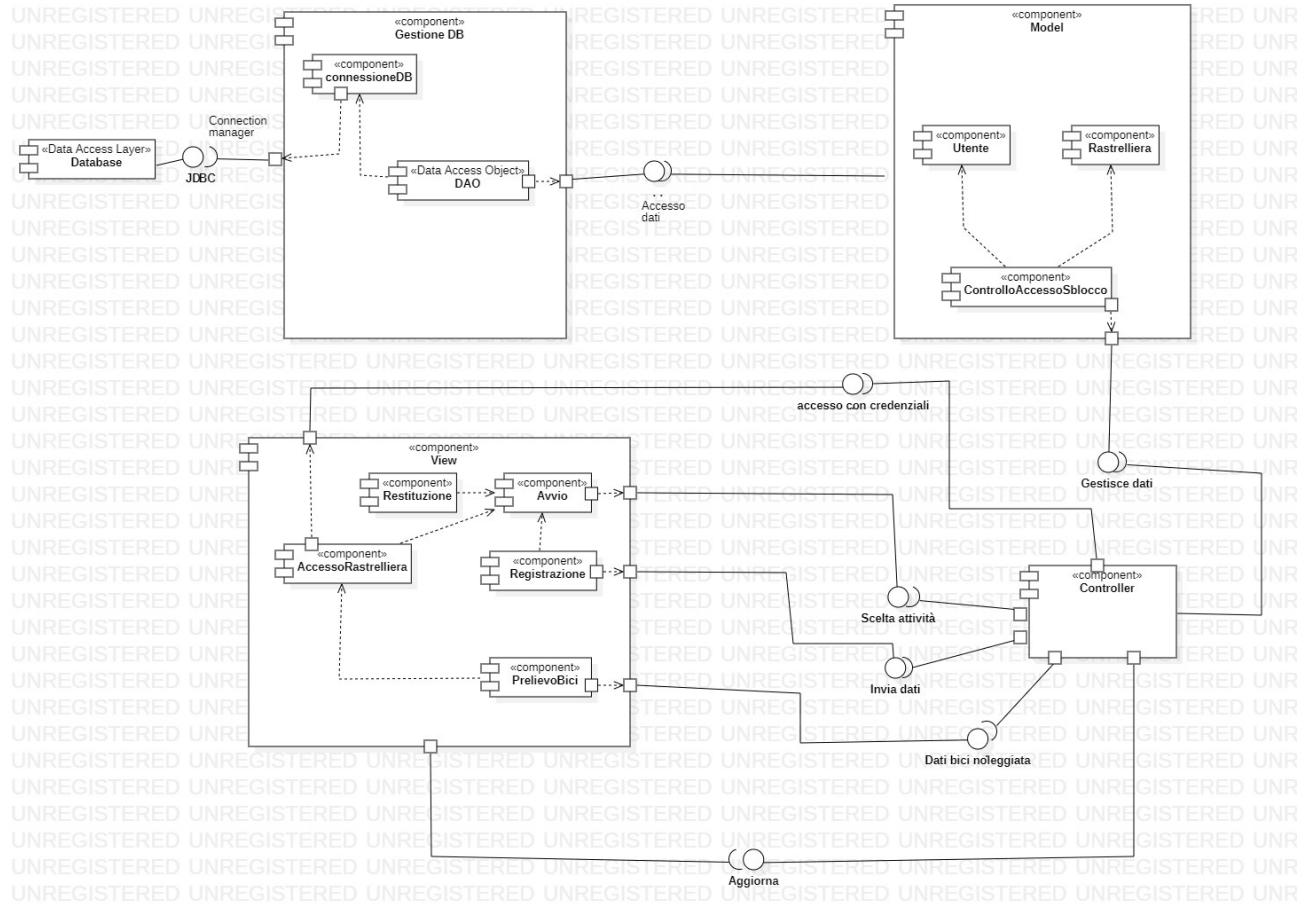
Qui sono illustrate alcune macchine di stato che ripercorrono alcuni degli scenari possibili all'interno del sistema.

La **bicicletta** nel suo stato iniziale risulta disponibile presso una rastrelliera e può essere noleggiata o ricollocata. Mentre la bicicletta è nello stato di noleggiata o è in corso di ricollocamento, nessun utente può richiederla per il noleggio.

La **rastrelliera** all'inizio è in attesa di un accesso per noleggiare o restituire una bicicletta e in base ai dati inseriti può sbloccare o bloccare la morsa della bici o rifiutare la richiesta.

L'**abbonamento** infine può non esistere o essere stato acquistato. Se acquistato viene attivato subito se risulta essere di tipo annuale altrimenti viene attivato al primo utilizzo se di tipo occasionale(giornaliero o settimanale). Una volta in stato di validità, ad ogni accesso viene fatto un controllo per verificare lo sia ancora e in caso contrario diviene scaduto e non più utilizzabile per accedere al servizio.

2.6 Diagramma dei componenti

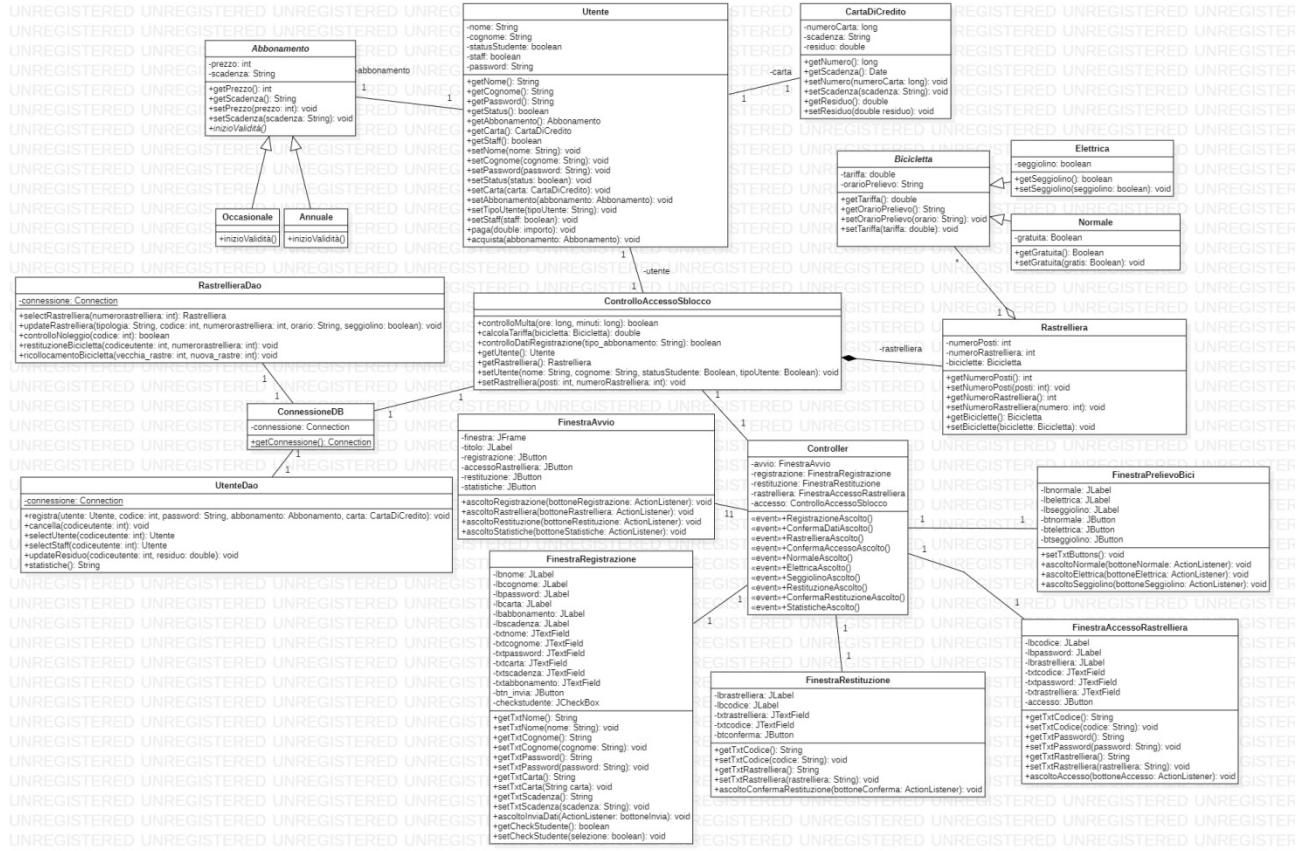


[DiagComponenti.jpg]

Qui viene presentato il diagramma dei componenti del sistema. È possibile vedere come la macro componente **Model** sia costituita al suo interno da tre componenti le quali sono **ControlloAccessoSblocco** che fa uso delle altre due che sono **Utente** e **rastrelliera**. **Model** interagisce con altre due macro componenti che sono **GestioneDB** e **Controller**. La prima per poter accedere ai dati salvati in modo persistente quando richiesti, la seconda per inviare i risultati delle operazioni. Il **Controller** dunque prenderà quei risultati e li restituirà alla componente **View** che deve essere aggiornata in funzione dello scenario in cui si trova il sistema.

3 Implementazione del sistema

3.1 Diagramma delle classi (modello di programma)



[*DiagClassiProgramma.jpg*]

In seguito verranno mostrate alcune parti zoomate su questo diagramma, ma per una visione completa e dettagliata è consigliato aprire il file .jpg citato.

3.1.1 Discussione dei Design Pattern utilizzati

Design pattern utilizzati: MVC, Singleton, DAO.

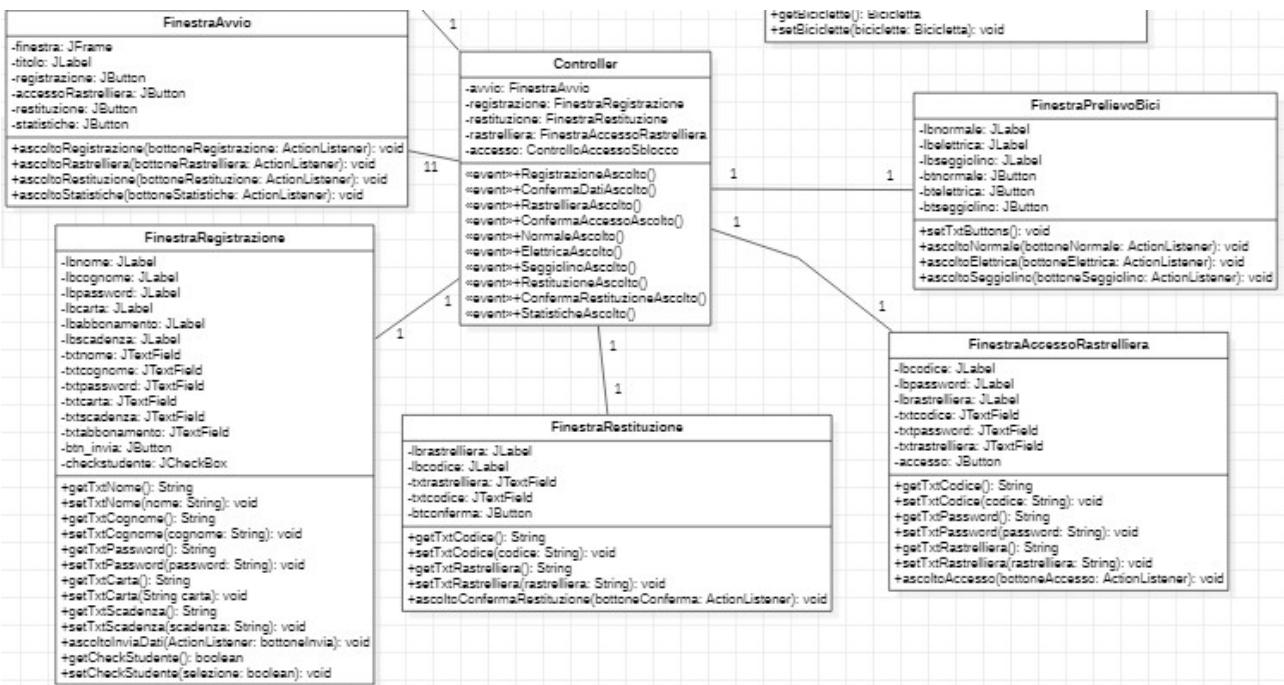
MVC(Model View Controller): le cassi coinvolte in questo design pattern sono molte e divise nelle tre macro categoria Model, View e Controller.

Le classe **ControlloAccessoSblocco** è la classe **Model** che racchiude tutte le altre model, quindi è in grado di gestire i dati degli oggetti di tipo Utente e Rastrelliera e si interfaccia direttamente con la classe Controller.

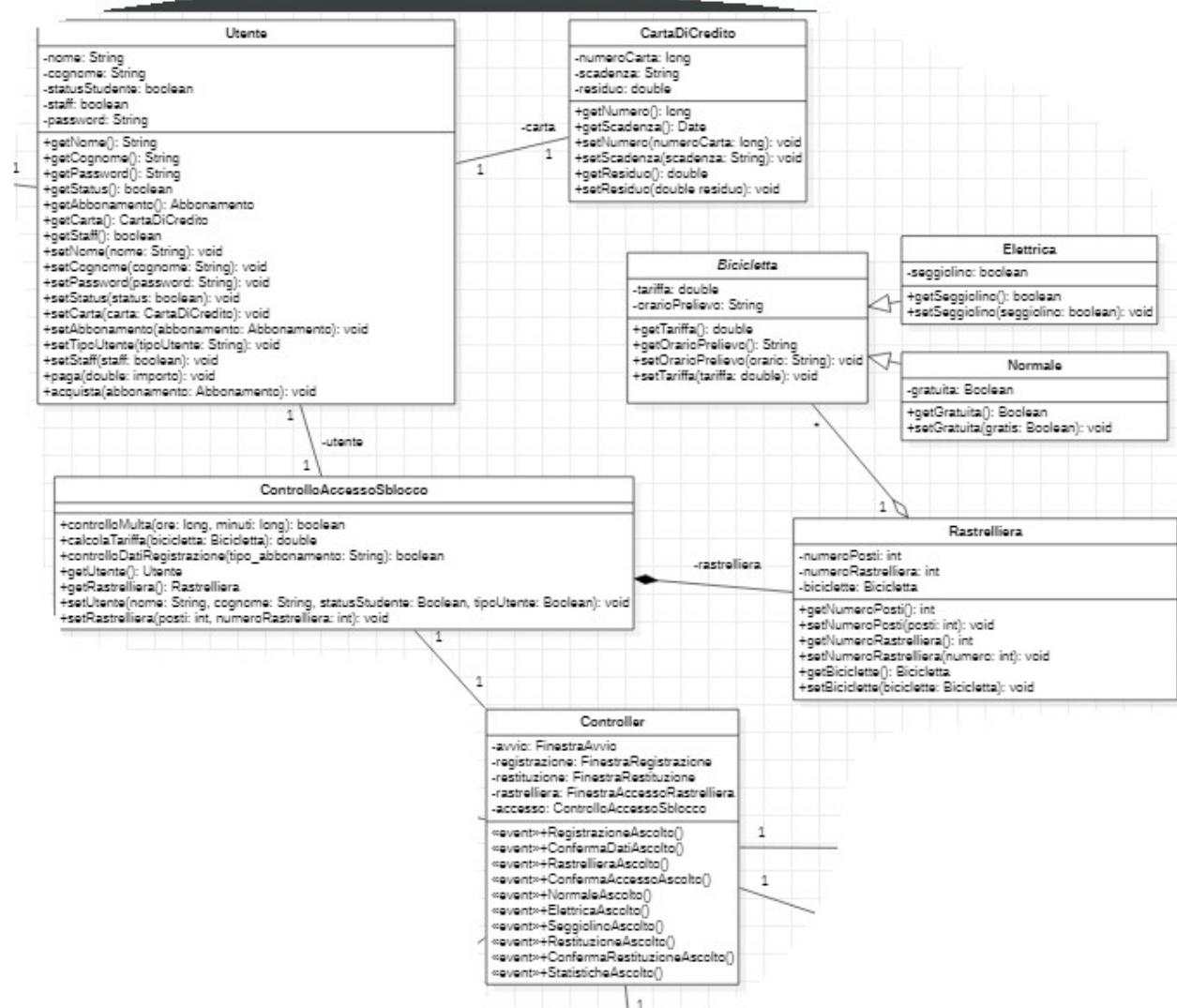
Per quanto riguarda le classi **View**, abbiamo tutte quelle classi che formano l'interfaccia grafica del sistema e anch'esse parlano con il *Controller* direttamente.

Il **Controller** intercetta le varie richieste e di volta in volta aggiorna il Model o la View relativa tramite componenti interne al Controller.

L'uso di questo pattern favorisce la manutenzione del codice nel lungo periodo in quanto è possibile sviluppare codice in ciascuna parte in modo indipendente dall'altra, rendendo il processo più veloce e meno costoso.



[La classe Controller e le classi View.]



[La classe Controller e alcune delle classi Model(per renderlo più visualizzabile).]

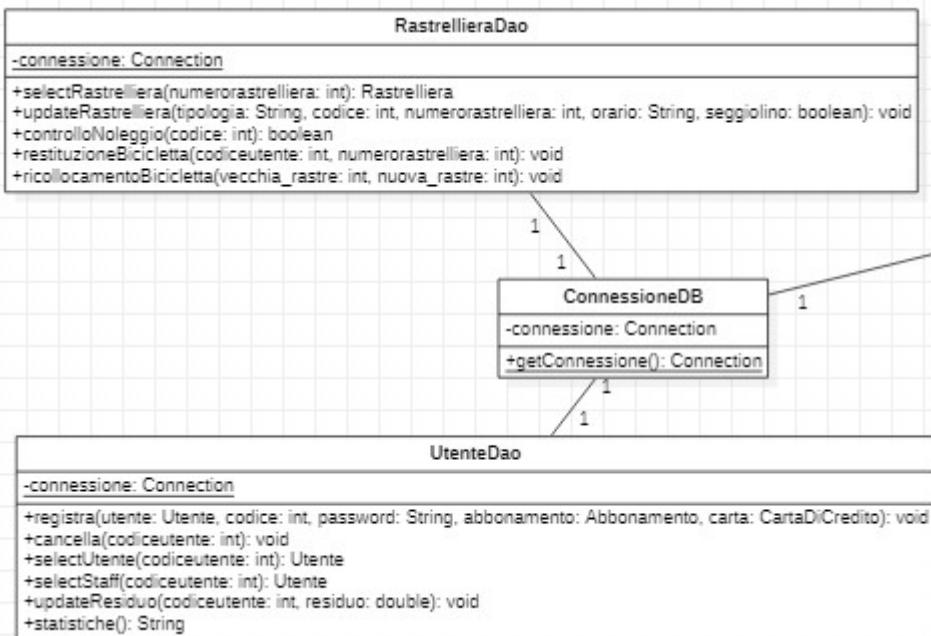
Singleton: la classe coinvolta in questo caso è **ConnessioneDB** e il pattern è stato usato per effettuare la connessione al database in un'unica classe dotata di istanza unica.

In questo modo non è stato necessario istanziare ogni volta la connessione quando serviva, ma viene semplicemente istanziata la prima volta e successivamente viene richiamata tutte le volte che risulta necessaria per delle operazioni.



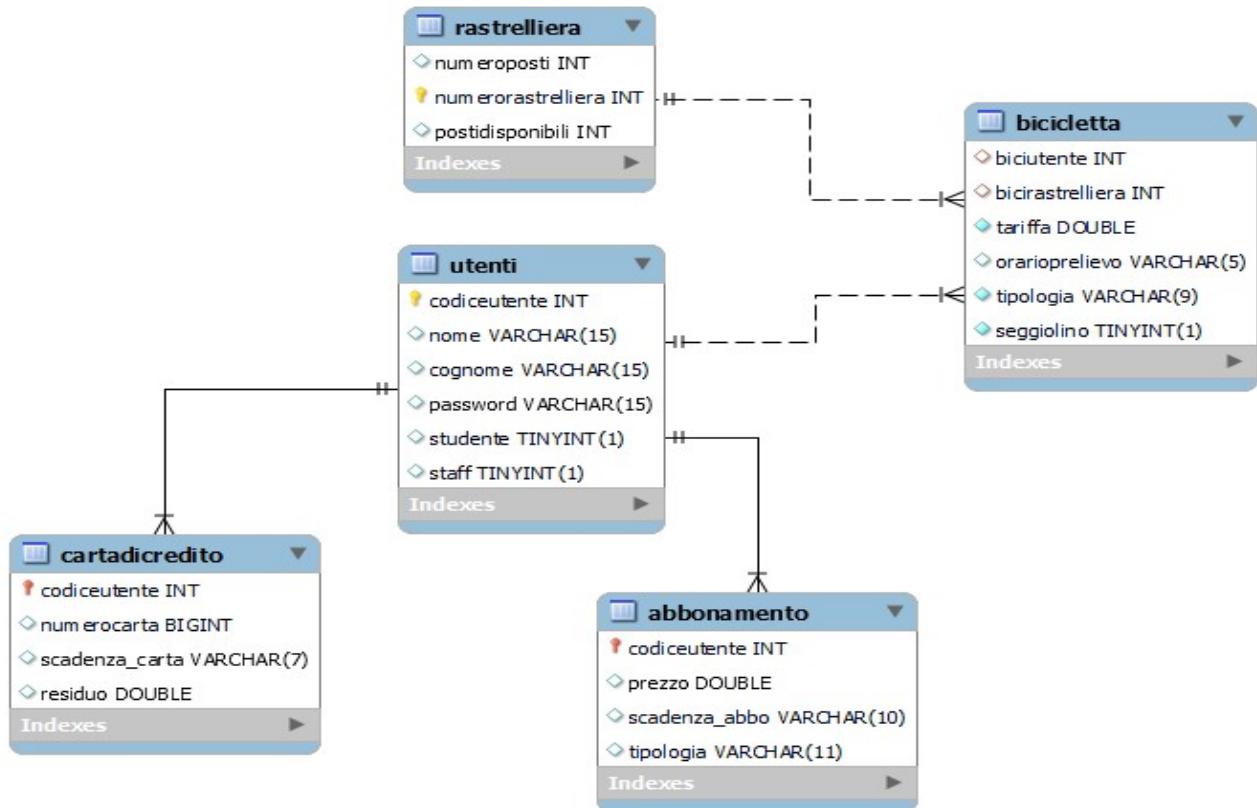
[La classe Singleton ConnessioneDB]

DAO(Data Object Access): l'uso di questo pattern ha coinvolto le classi **UtenteDao** e **RastrellieraDao** aventi come attributo il *Singleton* citato poco fa per la connessione al database. Grazie al *DAO* è possibile tenere tutte le interazioni con il database in classi distinte e indipendenti dal resto del sistema. Dunque tutte le eventuali query per recuperare dati essenziali durante l'esecuzione sono incapsulate in metodi specifici di queste due classi. In questo modo non è necessario scrivere query SQL all'interno di altre classi come il *Controller*, ma è sufficiente richiamare il metodo della classe *DAO* relativa. Si ha quindi una separazione rigida tra le componenti dell'applicazione e il mantenimento risulta essere a basso costo e molto più rapido.



[Le classi DAO insieme alla classe Singleton per la connessione al database.]

3.2 Gestione dei dati persistenti



[Database.jpg]

Il DBMS utilizzato è stato **MySQL**.

Dall'immagine è possibile visionare la struttura del database costruito a supporto del sistema per il salvataggio dei dati persistenti.

Vi sono 5 tabelle: *utenti*, *cartadicredito*, *abbonamento*, *rastrelliera* e *bicicletta*.

La tabella **utenti** è quella su cui sono presenti i dati principali di un utente come ad esempio il codice utente univoco, la password e se sia studente o membro dello staff. Questa tabella ha tre relazioni verso le tabelle **cartadicredito**, **abbonamento** e **bicicletta**.

Le prime due contengono ulteriori dati personali dell'utente quali sono per esempio il residuo sulla carta o la tipologia di abbonamento e la sua scadenza. La relazione con la tabella bicicletta invece è necessaria nel momento in cui è in corso un noleggio così da tenere traccia tramite il codice utente quale bicicletta sia in utilizzo e non disponibile.

Infine la tabella **rastrelliera** ha anch'essa una relazione con la tabella *bicicletta* e serve a tenere traccia di quali biciclette sono in quale rastrelliera e dunque sapere quanti posti sono occupati o disponibili.

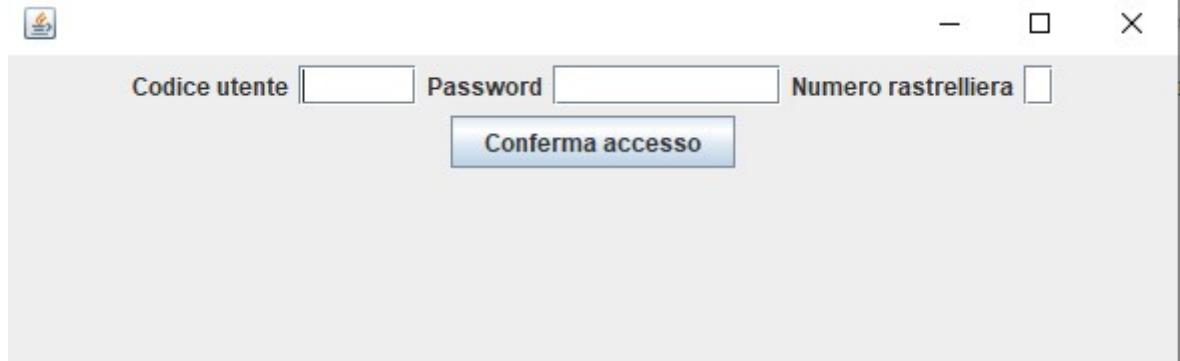
3.3 Descrizione dell'Interfaccia Grafica



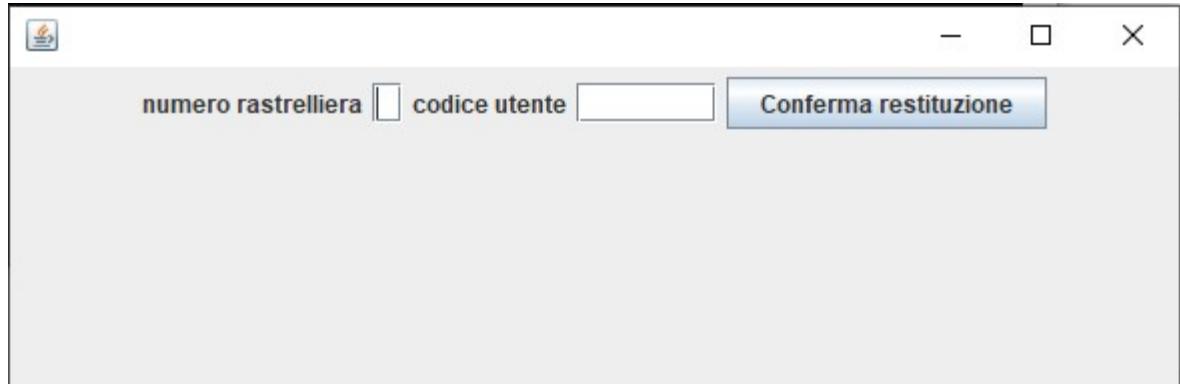
Finestra alla quale si viene riportati appena si lancia l'applicazione.



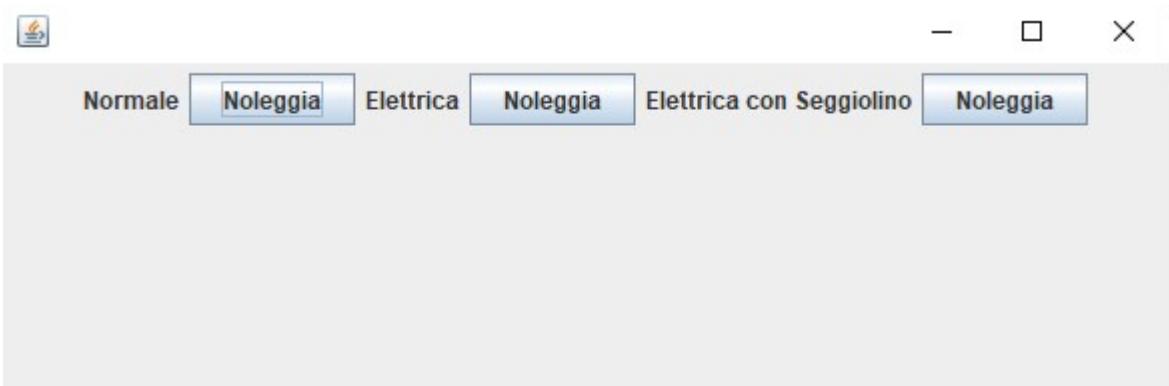
Finestra per effettuare la registrazione di un abbonamento.



Finestra che consente l'accesso a una rastrelliera.



Finestra che consente la restituzione di una bici presso una rastrelliera.

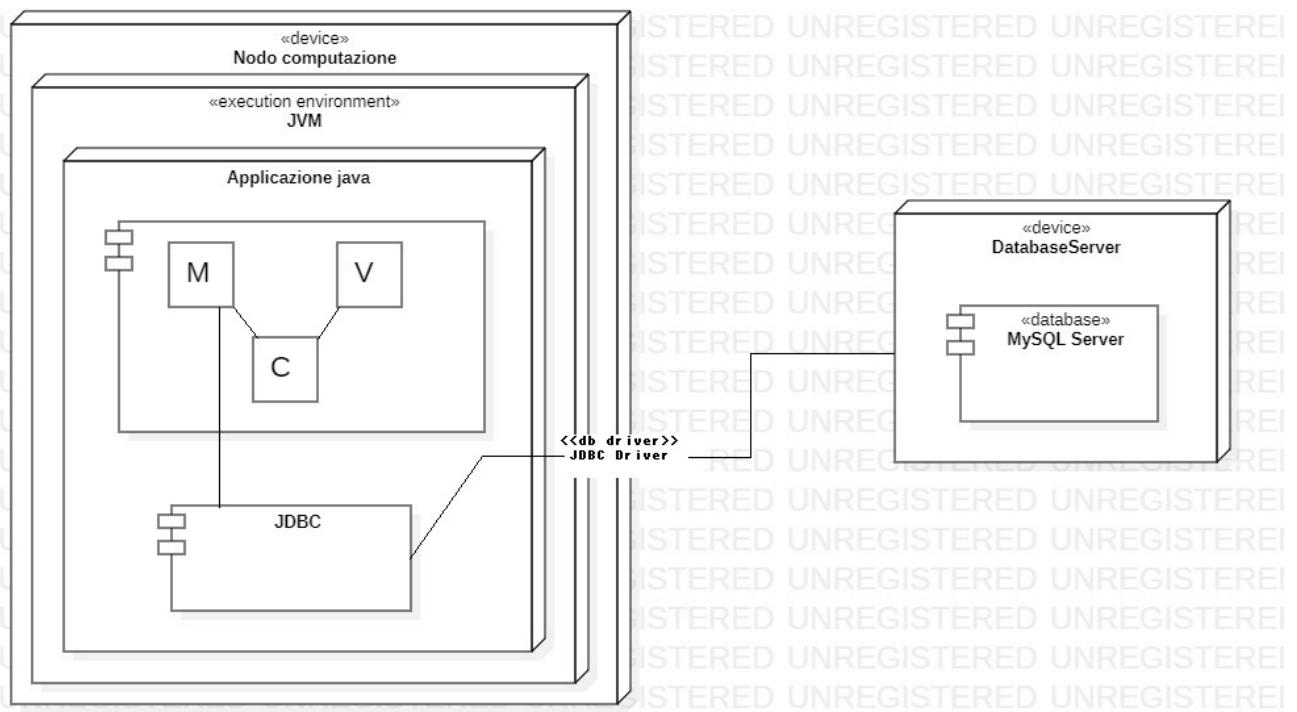


Finestra che consente di scegliere quale tipo di bicicletta noleggiare.

Alcuni controlli sull'input:

1. È stato effettuato un controllo durante la fase di registrazione per verificare che il tipo di abbonamento sia stato inserito correttamente tra Giornaliero, Settimanale e Annuale.
2. Un altro controllo è stato effettuato in fase di noleggio per verificare che l'utente non stia già in possesso di una bicicletta noleggiata.
3. In fase di restituzione viene controllato che l'utente abbia una bicicletta altrimenti non è possibile restituire niente.

3.4 Diagramma di deployment

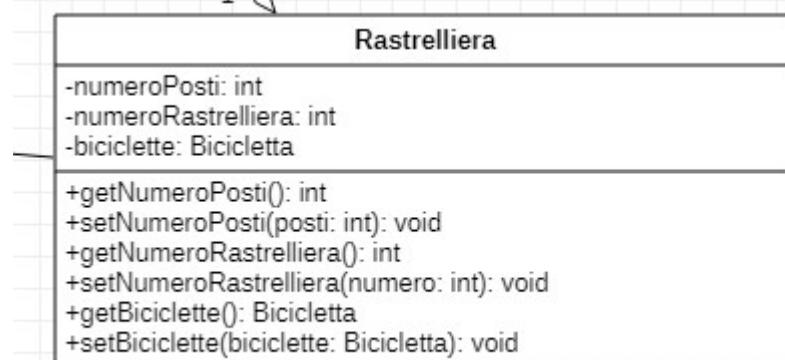


[DiagDeployment.jpg]

Nel diagramma di deployment qui esposto si può osservare come l'applicazione abbia come unica interazione esterna quella con il database tramite l'uso di **JDBC**.

3.5 Specifica e verifica dei vincoli

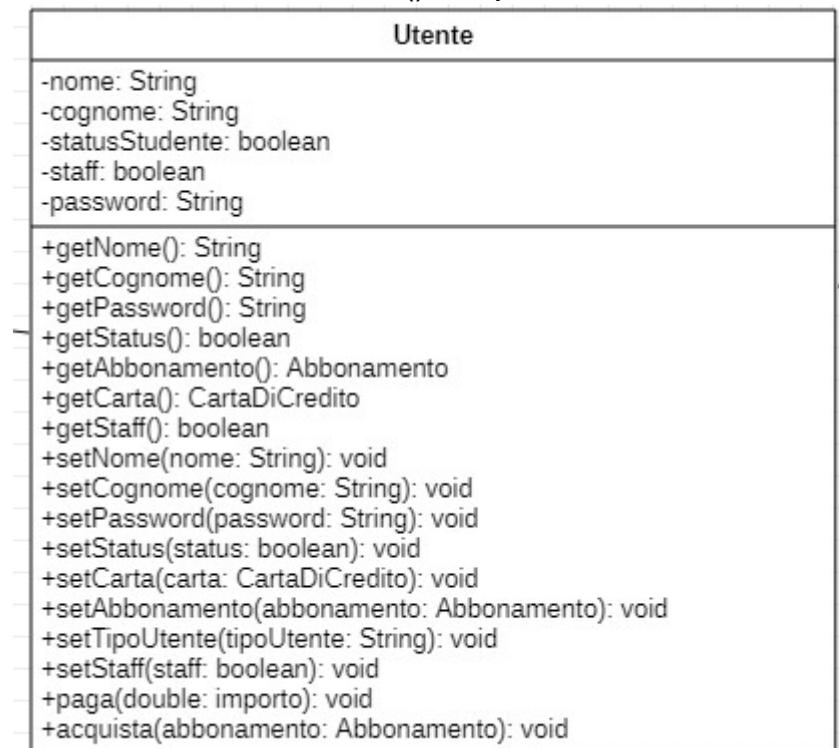
Qui di seguito è possibile dei vincoli OCL scritti per le classi del sistema.



Una rastrelliera ha al massimo 5 posti per le biciclette.

{ context Rastrelliera

 inv: self.biciclette->size() <= 5 }



Un utente ha una sola carta di credito.

{context Utente

 inv: self.allInstances()->isUnique(carta) }

Un utente ha un solo abbonamento.

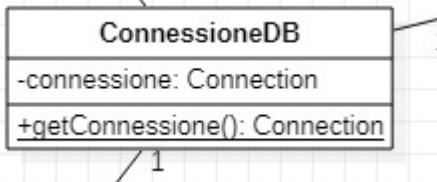
{context Utente

 inv: self.allInstances()->isUnique(abbonamento) }

La password di un utente deve essere lunga almeno 7 caratteri.

{context Utente

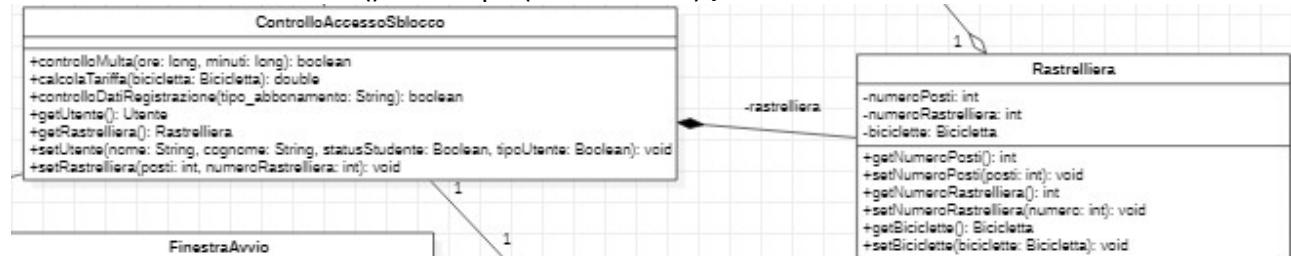
 inv: self.password.length >= 7 }



La connessione in ConessioneDB esiste in un'unica istanza.

{context ConessioneDB

inv: self.allInstances()->isUnique(conessione) }



Esistono solo 3 rastrelliere, quindi il numeroRastrelliera può essere al massimo 3.

{context ControlloAccessoSblocco

inv: self.rastrelliera.numeroRastrelliera <= 3}

3.6 Descrizione del testing

I test effettuati sono presenti all'interno delle seguenti classi: **TestUtente.java**,

TestAbbonamento.java e **TestControlloAccessoSblocco.java**.

Nella classe *TestUtente* è stato testato il funzionamento dei metodi *paga()* e *acquista()* così da verificare che il residuo della carta di credito venisse modificato correttamente.

Nella classe *TestAbbonamento* è stato testato che i metodi *controlloValidità()* e *inizioValidità()* funzionassero correttamente e che un abbonamento di tipologia qualsiasi sia attivato nel momento corretto e sia valido se la scadenza non è ancora sopraggiunta.

Nella classe *TestControlloAccessoSblocco* sono stati testati i metodi *controlloMult()* e *controlloDatiRegistrazione()*.

3.7 Note per l'installazione e l'utilizzo

Il progetto è stato realizzato interamente utilizzando il linguaggio **Java**.

La fase di sviluppo è stata quasi interamente svolta su piattaforma **Linux**, dunque è consigliabile ma ho riportato anche gli eventuali comandi che dovrebbero funzionare anche su piattaforma **Windows**.

All'interno della cartella *libs* è possibile trovare il file .jar necessario per collegarsi al database con il DBMS usato per questo progetto(**MySQL**).

Dentro la cartella *dumpDB* si possono trovare i dump per ricostruire lo schema usato. Dunque va importato dentro il DBMS per poter essere utilizzato.

```

String url = "jdbc:mysql://localhost:3306/bikesharing?user=user&password=password";
try {

```

Come è evidenziato in questa immagine, le credenziali usate per accedere al database sono user e password. Nel caso in cui sul proprio DBMS in locale si utilizzassero altre credenziali, è necessario modificare questa linea di codice con le proprie credenziali nel file sorgente **ConessioneDB.java**.

Inoltre all'interno dello schema è già presente un utente di tipo staff con i seguenti dati d'accesso: codice utente 117 e password "staff".

Quindi per provare questa specifica funzionalità, bisogna accedere con questi dati pre-inseriti.

Per quanto riguarda la compilazione dei sorgenti dare il seguente comando da terminale:

```
javac -cp libs/junit-4.13.2.jar:. *.java (Linux)  
javac -cp libs/junit-4.13.2.jar:. *.java (Windows)
```

Per eseguirlo(una volta avviato il database) dare il seguente comando:

```
java -cp libs/mysql-connector-java-8.0.24.jar:. SistemaBikeSharing (Linux)  
java -cp libs/mysql-connector-java-8.0.24.jar;. SistemaBikeSharing (Windows)
```

Compilazione e lancio dei test:

(Linux)

```
javac -cp libs/junit-4.13.2.jar:. Test*.java (se si vuole compilare solo i file di test)  
java -cp libs/junit-4.13.2.jar:libs/hamcrest-core-1.3.jar:. org.junit.runner.JUnitCore TestUtente  
TestAbbonamento TestControlloAccessoSblocco
```

(Windows)

Su Windows a causa di problemi di compatibilità sulla linea di comando è stato utilizzato Eclipse nella versione reperibile al seguente link: http://fmse.di.unimi.it/sw/eclipsePSS_win.zip
Una volta lanciato con il file *eclipse_java7.bat*, è necessario importare le classi di test e le classi coinvolte per poter essere lanciati.