

# Bikesharing

## 1. Descrizione del problema

### 1.1. Analisi e specifica dei requisiti

Il servizio sviluppato consiste in un sistema di bikesharing, ispirato al servizio BikeMi presente a Milano, utilizzabile dai cittadini in possesso di un abbonamento.

Sono previsti diversi tipi di abbonamento, suddivisi in base alla durata, in particolare:

Descrizione	Prezzo
Abbonamento Annuale	36,00 €
Abbonamento Settimanale	9,00 €
Abbonamento Giornaliero	4,50 €

Alla fine della registrazione all'utente vengono fornite le credenziali di accesso al servizio sotto forma di codice utente e password.

In fase di registrazione è possibile segnalare lo status di studente, sono previste delle agevolazioni per i clienti di questo tipo.

Nel caso di abbonamento giornaliero o settimanale l'abbonamento non ha inizio nel momento della registrazione ma al primo ritiro di una bicicletta.

Le biciclette messe a disposizione, sono di diversi tipi e prevedono tariffe di noleggio diverse:

- Biciclette normali
- Biciclette elettriche
- Biciclette elettriche con seggiolino

Ad ogni tipo di bicicletta è associato un prezzo di base (PRICE) e una rapporto di incremento (INCREMENT\_RATIO), il totale del noleggio viene calcolato tramite la seguente formula:

$$\text{Prezzo} = \text{PRICE} * (\lfloor \frac{\text{durata}}{30} \rfloor + \text{INCREMENTARIO})$$

In questo modo il prezzo del noleggio aumenta all'aumentare delle mezz'ore di utilizzo consecutive, scelta fatta poiché il servizio è stato pensato per l'utilizzo in noleggi di breve durata.

Bicicletta classica		Bicicletta elettrica	
Tariffa base	Incremento	Tariffa base	Incremento
0,25 €*	1,2	0,5 €	1,4

\* I noleggi di biciclette classiche sotto i 30 minuti sono sempre gratuiti, inoltre per gli studenti è previsto il noleggio completamente gratuito delle biciclette classiche.

Sono previsti alcuni vincoli nell'utilizzo del servizio:

- Il tempo di utilizzo massimo del servizio è di 2 ore consecutive, il superamento di tale limite per 3 volte determina l'annullamento dell'abbonamento.
- Se il tempo di utilizzo supera le 24 ore vengono addebitati 150 € di penale sulla carta del cliente.
- Il tempo minimo fra due noleggi dello stesso cliente è di 5 minuti.

Le biciclette sono posizionate su apposite rastrelliere dotate di morsa, per le biciclette elettriche e quelle elettriche con seggiolino sono previste delle morse con ricarica.

E' possibile ritirare la bicicletta presso una delle rastrelliere e riconsegnarla in qualsiasi altra rastrelliera a condizione la rastrelliera di destinazione abbia morsa libere del tipo corrispondente alla bicicletta utilizzata.

Per ritirare una bicicletta è sufficiente effettuare il login nel totem posizionato di fianco ad ogni rastrelliera, dopo di che selezionare ritira e scegliere il tipo di bicicletta; Il sistema restituirà il numero di bicicletta da ritirare e sbloccherà la morsa, a questo punto il cliente può ritirare la bicicletta per iniziare il noleggio, se il ritiro della bicicletta non avviene entro 3 minuti il sistema chiude la morsa e annulla il noleggio.

Per restituire una bicicletta il cliente effettua il login nel totem della rastrelliera di destinazione e seleziona riconsegna; Il sistema restituisce, se disponibile, il numero di morsa alla quale agganciare la bicicletta, dopo di che il cliente posiziona la bicicletta nella rastrelliera per terminare il noleggio.

Effettuando nuovamente il login in una qualsiasi rastrelliera il cliente ha la possibilità di verificare la presenza di noleggi attualmente attivi.

Il totale del noleggio viene addebitato automaticamente sulla carta inserita nel momento dell'abbonamento, non è necessario che l'utente inserisca i dati della carta ad ogni riconsegna. Per questioni di sicurezza si è scelto di non memorizzare i dati della carta di credito dei clienti ma di prevedere che questo compito sia affidato al servizio di pagamento, come spesso avviene nella realtà, in questo modo sarà comunque possibile effettuare il pagamento del noleggio senza richiedere nuovamente i dati della carta semplicemente riferendo al servizio di pagamento il cliente che deve pagare.

Il servizio prevede anche una sezione di manutenzione all'interno della quale gli utenti manutentori possono occuparsi di:

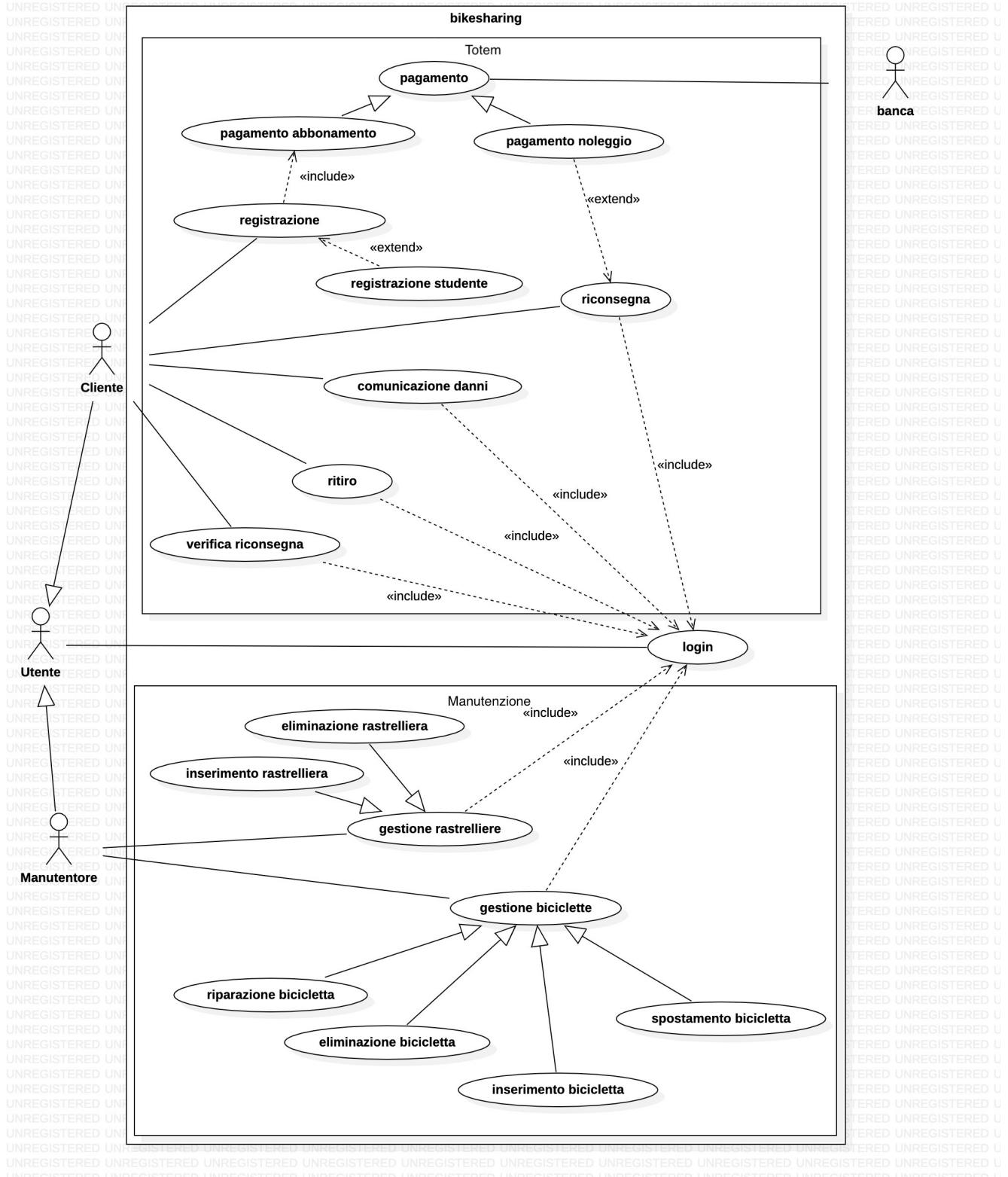
- Creazione e eliminazione rastrelliere
- Creazione, eliminazione, riparazione e spostamento delle biciclette

## 1.2. Glossario

**Bikesharing:** Il termine bikesharing (bici condivisa) si riferisce ad uno strumento di mobilità sostenibile, spesso gestito dalle amministrazioni pubbliche, che consiste appunto nella condivisione fra cittadini di una flotta di biciclette.

## 2. Progettazione del Sistema

### 2.1. Diagramma dei casi d'uso



## 2.2. Descrizione degli scenari

<b>Nome</b>	Registrazione
<b>Scopo</b>	Registrare utente e abbonamento associato
<b>Attori</b>	Cliente
<b>Pre-condizioni</b>	
<b>Trigger</b>	Bottone “registri” nella home page
<b>Descrizione</b>	<ol style="list-style-type: none"> <li>1. Il cliente inserisce i propri dati anagrafici in un apposito form e fa click per confermare</li> <li>2. Il sistema registra l’utente e gli restituisce un codice univoco da utilizzare come codice utente</li> <li>3. Il sistema passa al <u>pagamento dell’abbonamento</u></li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1a. Se i dati del form non sono corretti il sistema restituisce un errore</li> <li>2a. Se la registrazione non va a buon fine il sistema restituisce un errore</li> </ol>
<b>Post-condizioni</b>	Il cliente registrato e possiede i parametri di accesso

<b>Nome</b>	Pagamento abbonamento
<b>Scopo</b>	Effettuare il pagamento dell’abbonamento
<b>Attori</b>	Cliente, Banca
<b>Pre-condizioni</b>	
<b>Trigger</b>	Registrazione del cliente confermata o login di cliente con abbonamento scaduto
<b>Descrizione</b>	<ol style="list-style-type: none"> <li>1. Il cliente seleziona il tipo di abbonamento che desidera sottoscrivere</li> <li>2. Il cliente inserisce i dati della carta di credito e fa click per confermare</li> <li>3. Il servizio della banca conferma i dati inseriti</li> <li>4. Il cliente effettua il pagamento</li> <li>5. Il pagamento ha successo</li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>2a. Se vi sono dati mancanti/errati il sistema restituisce un errore e riavvia il caso d’uso pagamento abbonamento</li> <li>5a. Il pagamento non ha successo il sistema restituisce un errore e riavvia il caso d’uso pagamento abbonamento</li> </ol>
<b>Post-condizioni</b>	Il cliente ha attivato un abbonamento

<b>Nome</b>	Login (modalità TOTEM)
<b>Scopo</b>	Effettuare l'accesso al servizio
<b>Attori</b>	Utente
<b>Pre-condizioni</b>	
<b>Trigger</b>	Bottone "login" nella home page
<b>Descrizione</b>	<p>1. L'utente inserisce il codice univoco e la password all'interno di un apposito form.</p> <p>2. Il sistema effettua il controllo dei dati inseriti e conferma</p> <p>3. Il sistema riconosce l'utente come cliente e lo reindirizza alla home page clienti</p>
<b>Alternative</b>	<p>2a. Se i dati inseriti non sono corretti il sistema restituisce un errore e riavvia il caso d'uso</p> <p>2b. Se il cliente non ha un abbonamento attivo il sistema avvia il caso d'uso <u>pagamento abbonamento</u></p> <p>3a. Se i dati di accesso non corrispondono ad un cliente il sistema restituisce un messaggio di errore</p>
<b>Post-condizioni</b>	Il cliente ha accesso al servizio

<b>Nome</b>	Login (modalità MAINTENANCE)
<b>Scopo</b>	Effettuare l'accesso al servizio
<b>Attori</b>	Utente
<b>Pre-condizioni</b>	
<b>Trigger</b>	Bottone "login" nella home page
<b>Descrizione</b>	<p>1. L'utente inserisce il codice univoco e la password all'interno di un apposito form.</p> <p>2. Il sistema effettua il controllo dei dati inseriti e conferma</p> <p>3. Il sistema riconosce l'utente come tecnico e lo reindirizza alla home page manutenzione</p>
<b>Alternative</b>	<p>2a. Se i dati inseriti non sono corretti il sistema restituisce un errore e riavvia il caso d'uso</p> <p>3a. Se i dati di accesso non corrispondono ad un tecnico il sistema restituisce un messaggio di errore</p>
<b>Post-condizioni</b>	Il tecnico ha accesso al servizio

<b>Nome</b>	Ritiro
<b>Scopo</b>	Il cliente ritira la bicicletta
<b>Attori</b>	Cliente
<b>Pre-condizioni</b>	Il cliente è abbonato
<b>Trigger</b>	Il cliente preme ritira bicicletta
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il cliente seleziona il tipo di bicicletta e fa click per confermare</li> <li>2. Il sistema restituisce il numero della morsa e la sblocca</li> <li>3. Il cliente preleva la bicicletta</li> </ul>
<b>Alternative</b>	<ul style="list-style-type: none"> <li>2a. Se non ci sono biciclette disponibili del tipo selezionato il sistema restituisce un messaggio di errore</li> <li>3a. Se il cliente non ritira la bicicletta entro un tempo prestabilito il sistema annulla il noleggio e richiude la morsa</li> </ul>
<b>Post-condizioni</b>	Il cliente ha avviato un noleggio

<b>Nome</b>	Restituzione Bicicletta
<b>Scopo</b>	Permettere ai clienti di restituire una bicicletta
<b>Attori</b>	Cliente
<b>Pre-condizioni</b>	
<b>Trigger</b>	Bottone restituisci bicicletta
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema verifica la presenza di noleggi attivi</li> <li>2. Il sistema restituisce il numero di morsa alla quale collegare la bicicletta</li> <li>3. Il cliente inserisce la bicicletta nella morsa indicata</li> <li>4. Il sistema effettua il <u>pagamento del noleggio</u></li> </ul>
<b>Alternative</b>	2a. Se non ci sono morse libere il sistema restituisce un errore
<b>Post-condizioni</b>	Il noleggio è terminato

<b>Nome</b>	Pagamento noleggio
<b>Scopo</b>	Effettuare il pagamento di un noleggio
<b>Attori</b>	Cliente, Banca
<b>Pre-condizioni</b>	Il cliente deve essere abbonato
<b>Trigger</b>	Restituzione bicicletta
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il cliente effettua il pagamento</li> <li>2. Il sistema processa il pagamento e restituisce esito positivo</li> </ul>
<b>Alternative</b>	1a. Se il pagamento non va a buon fine il sistema restituisce un errore
<b>Post-condizioni</b>	Il cliente ha effettuato il pagamento del noleggio

<b>Nome</b>	Comunicazione danni
<b>Scopo</b>	Permettere ai clienti di comunicare dei danni alla bicicletta
<b>Attori</b>	Cliente
<b>Pre-condizioni</b>	Il cliente ha effettuato un <u>ritiro</u>
<b>Trigger</b>	Il cliente fa click su “comunica danni”
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema registra il danno</li> <li>2. Il sistema restituisce un messaggio al cliente nel quale consiglia di riconsegnare la bicicletta</li> </ul>
<b>Alternative</b>	
<b>Post-condizioni</b>	La bicicletta è segnalata come danneggiata

<b>Nome</b>	Eliminazione rastrelliera
<b>Scopo</b>	Permettere ai manutentori di eliminare una rastrelliera
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore fa click sul pulsante elimina relativo di una specifica rastrelliera
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema controlla che la rastrelliera selezionata sia vuota</li> <li>2. Il sistema elimina la rastrelliera selezionata</li> </ul>
<b>Alternative</b>	1a. Se la rastrelliera selezionata non è vuota il sistema restituisce un messaggio di errore
<b>Post-condizioni</b>	La rastrelliera è stata eliminata

<b>Nome</b>	Inserimento rastrelliera
<b>Scopo</b>	Permettere ai manutentori aggiungere una nuova rastrelliera
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore fa click sul pulsante aggiungi rastrelliera nella lista rastrelliere
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il cliente inserisce i campi necessari e conferma</li> <li>2. Il sistema aggiunge la nuova rastrelliera</li> <li>3. Il sistema ritorna alla lista rastrelliere</li> </ul>
<b>Alternative</b>	1a. Se vi sono dati mancanti/errati il sistema restituisce un messaggio di errore e ritorna alla lista rastrelliere
<b>Post-condizioni</b>	E' stata creata una nuova rastrelliera

<b>Nome</b>	Spostamento Bicicletta
<b>Scopo</b>	Permettere ai manutentori spostare una bicicletta da una rastrelliera ad un'altra
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore seleziona una rastrelliera e una bicicletta e fa click su conferma
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema controlla che nella rastrelliera di destinazione siano disponibili posti per la bicicletta selezionata</li> <li>2. Il sistema sgancia la bicicletta dalla rastrelliera attuale e la aggancia alla nuova rastrelliera</li> </ul>
<b>Alternative</b>	1a. Se la rastrelliera di destinazione non ha morse libere compatibili il sistema restituisce un messaggio di errore
<b>Post-condizioni</b>	La bicicletta è stata spostata

<b>Nome</b>	Inserimento bicicletta
<b>Scopo</b>	Permettere ai manutentori di inserire una nuova bicicletta
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore seleziona una rastrelliera e un tipo di bicicletta e fa click su conferma
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema controlla che nella rastrelliera selezionata ci sia un posto disponibile per il tipo di bicicletta scelto</li> <li>2. Il sistema crea la nuova bicicletta e la aggancia alla rastrelliera</li> </ul>
<b>Alternative</b>	1a. Se nella rastrelliera selezionata non c'è un posto disponibile per il tipo di bicicletta scelto il sistema restituisce un messaggio di errore
<b>Post-condizioni</b>	E' stata creata una nuova bicicletta e agganciata ad una rastrelliera

<b>Nome</b>	Eliminazione bicicletta
<b>Scopo</b>	Permettere ai manutentori di eliminare una bicicletta
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore fa click sul pulsante elimina relativo a una specifica bicicletta
<b>Descrizione</b>	<ul style="list-style-type: none"> <li>1. Il sistema sgancia la bicicletta dalla morsa a cui è collegata</li> <li>2. Il sistema elimina la bicicletta selezionata</li> </ul>
<b>Alternative</b>	
<b>Post-condizioni</b>	La bicicletta è stata eliminata

<b>Nome</b>	Riparazione bicicletta
<b>Scopo</b>	Permettere ai manutentori di riparare una bicicletta
<b>Attori</b>	Manutentore
<b>Pre-condizioni</b>	
<b>Trigger</b>	Il manutentore fa click sul pulsante ripara relativo a una specifica bicicletta
<b>Descrizione</b>	<ol style="list-style-type: none"> <li>1. Il sistema controlla che la bicicletta selezionata sia danneggiata</li> <li>2. Il sistema ripara la bicicletta</li> </ol>
<b>Alternative</b>	1a. Se la bicicletta non è danneggiata il caso d'uso termina
<b>Post-condizioni</b>	La bicicletta è stata riparata

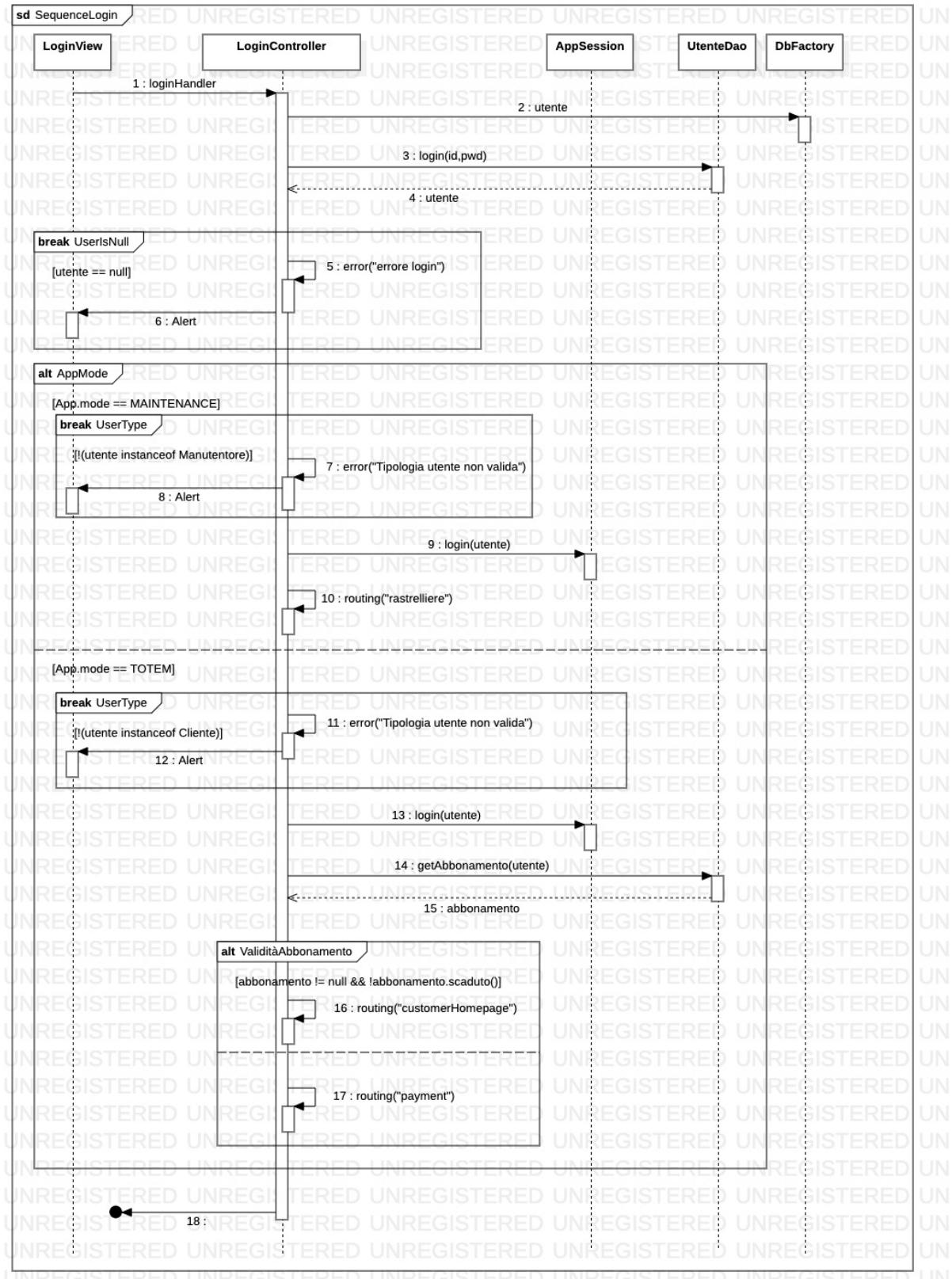
### 2.3. Diagramma delle classi (modello di progetto)

Il diagramma delle classi di programma riportato nella sezione relativa all'implementazione è frutto di diverse modifiche dovute a correzioni e miglioramenti in fasi successive, per questo motivo è stato deciso di riportare solo quello.

## 2.4. Diagrammi di sequenza

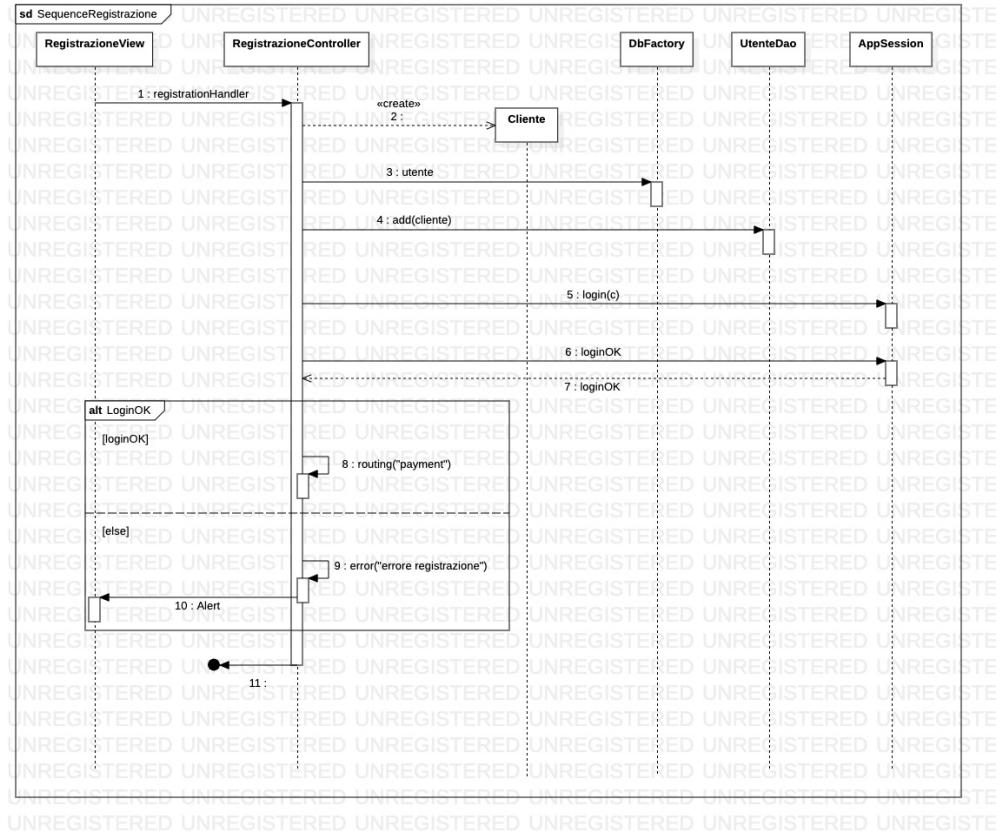
### 2.4.1. Sequenze comuni

#### 2.4.1.1. Login

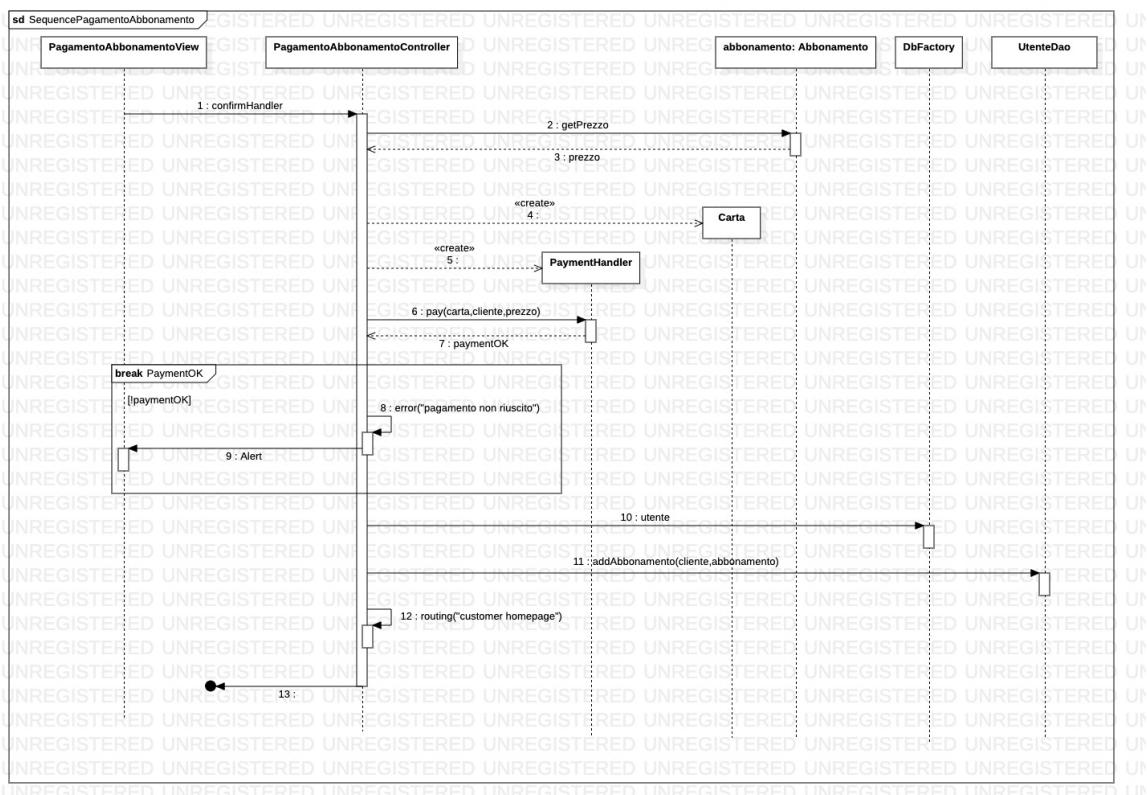


## 2.4.2. Sequenze cliente

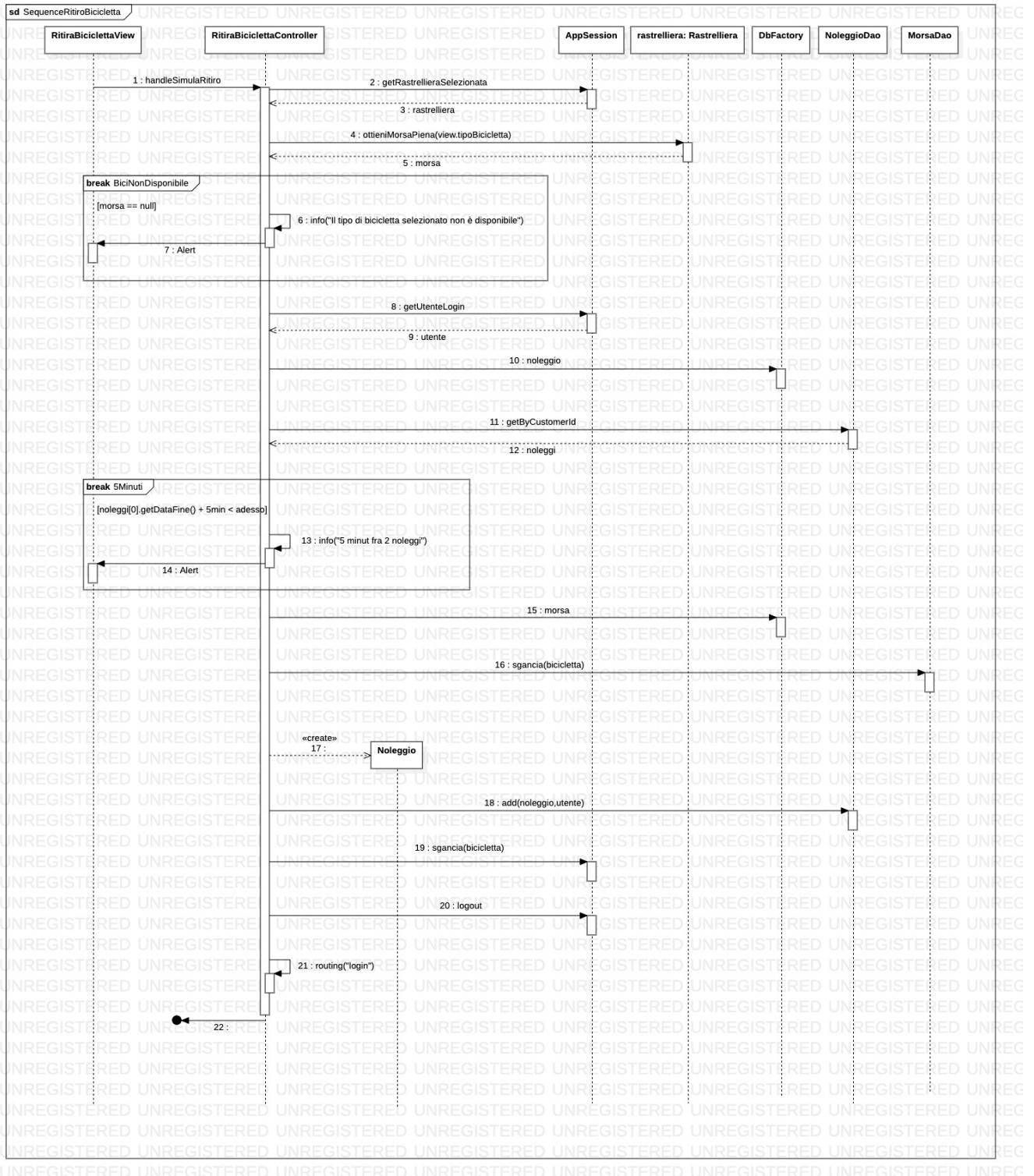
### 2.4.2.1. Registrazione



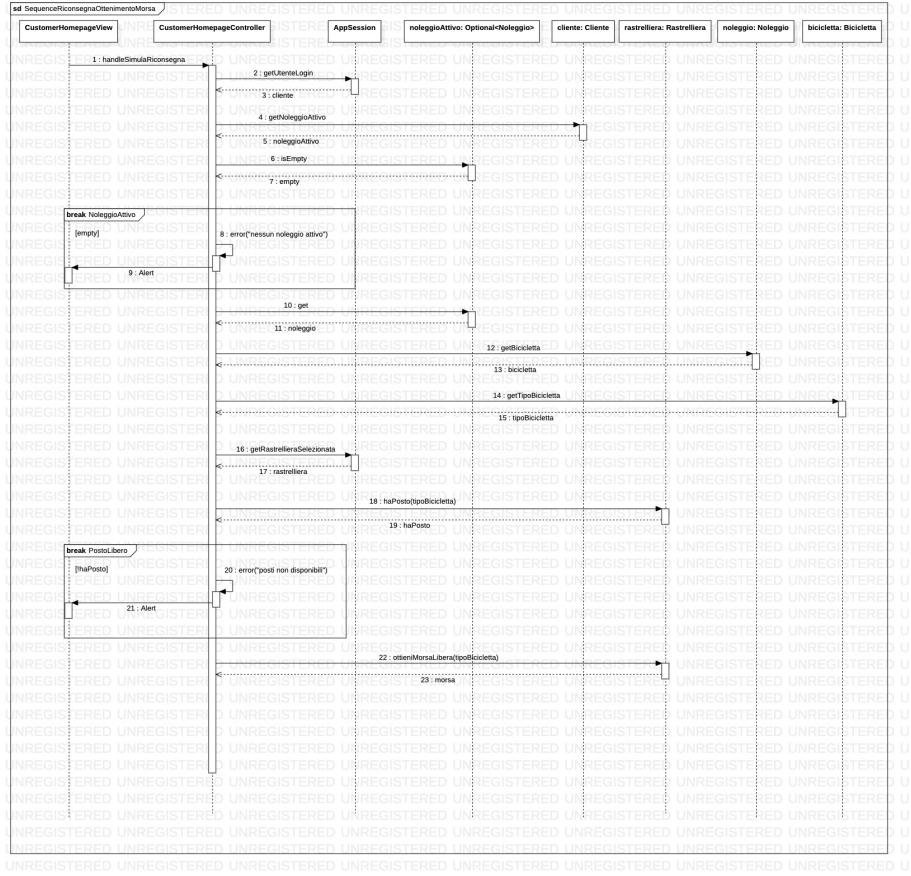
### 2.4.2.2. Pagamento abbonamento



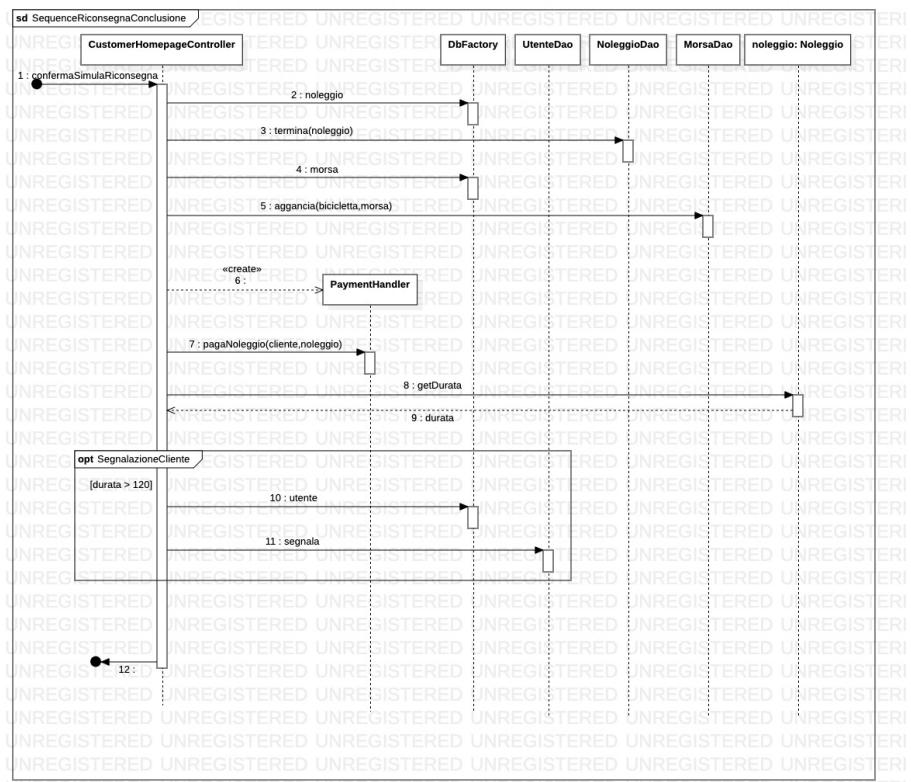
### 2.4.2.3. Ritiro bicicletta



#### 2.4.2.4. Riconsegna - Ottenimento morsa

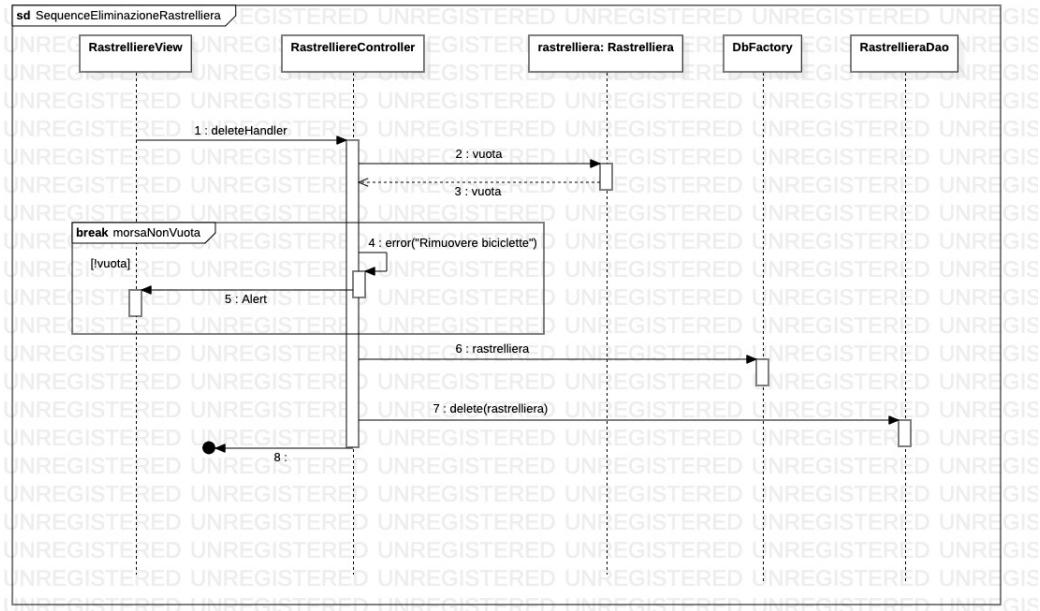


#### 2.4.2.5. Riconsegna - Conclusioni

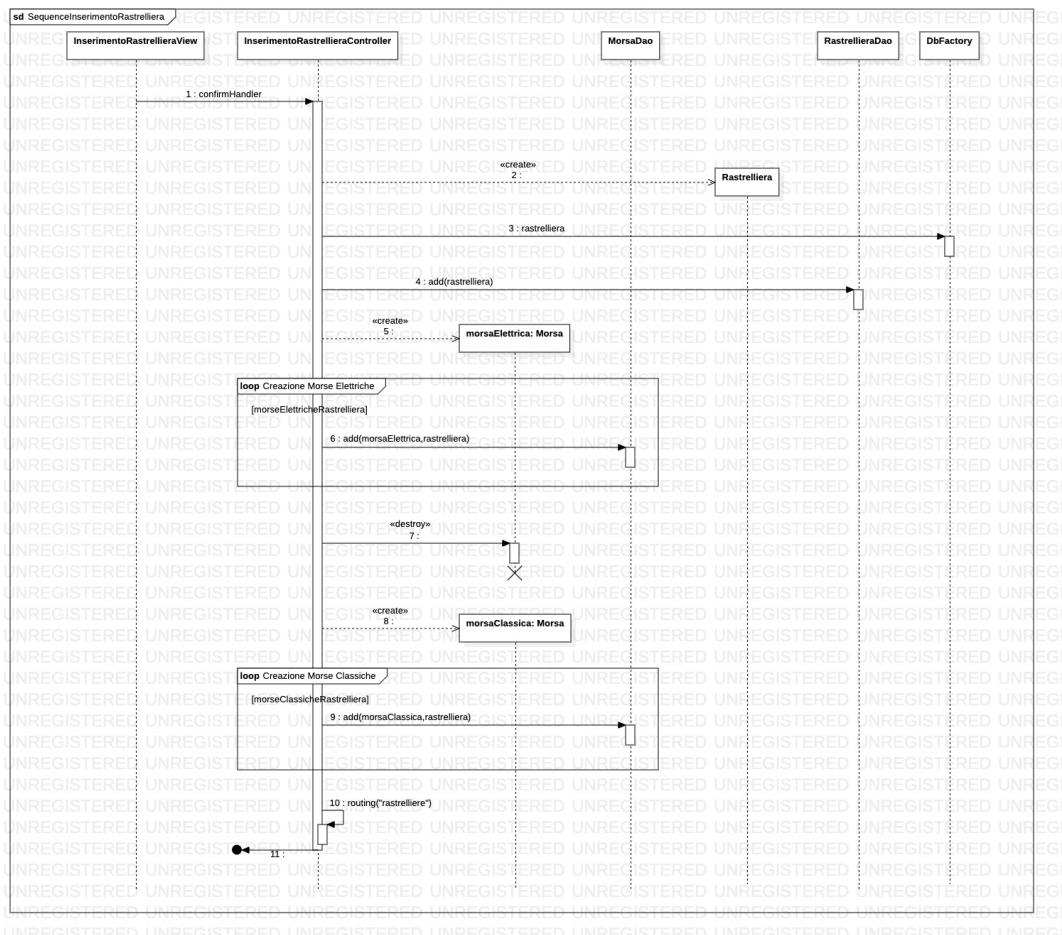


## 2.4.3. Sequenze manutenzione

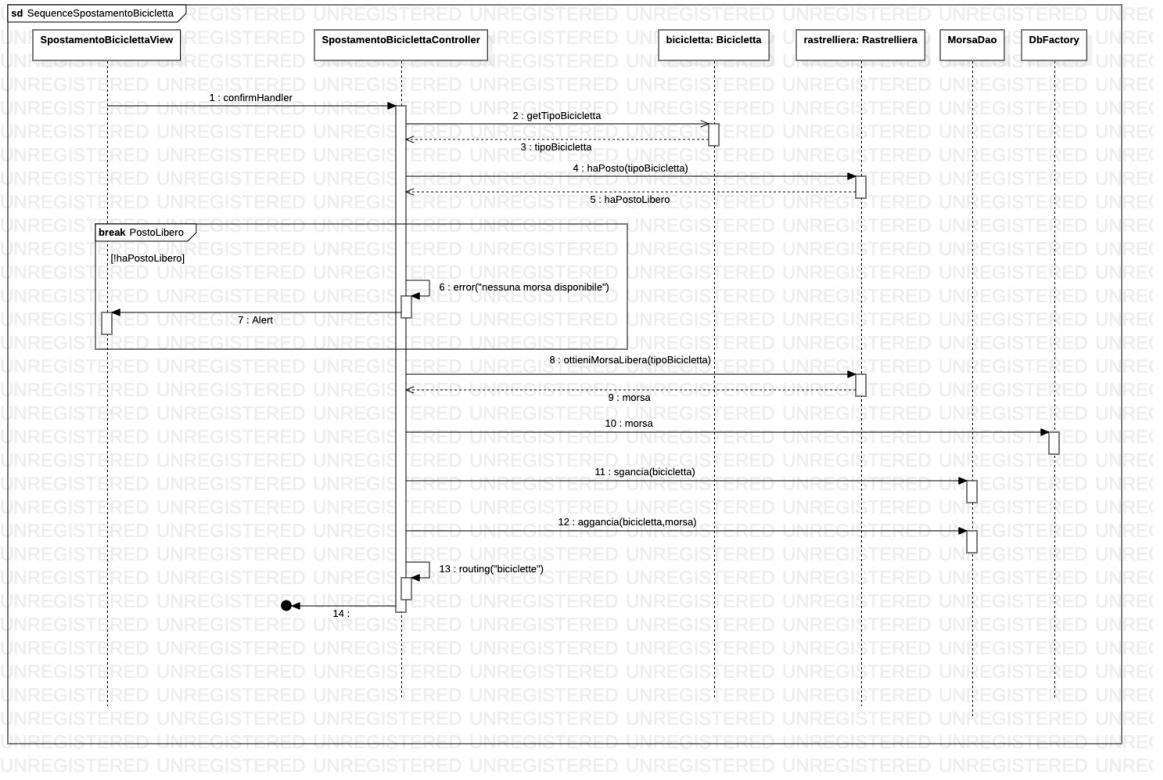
### 2.4.3.1. Eliminazione rastrelliera



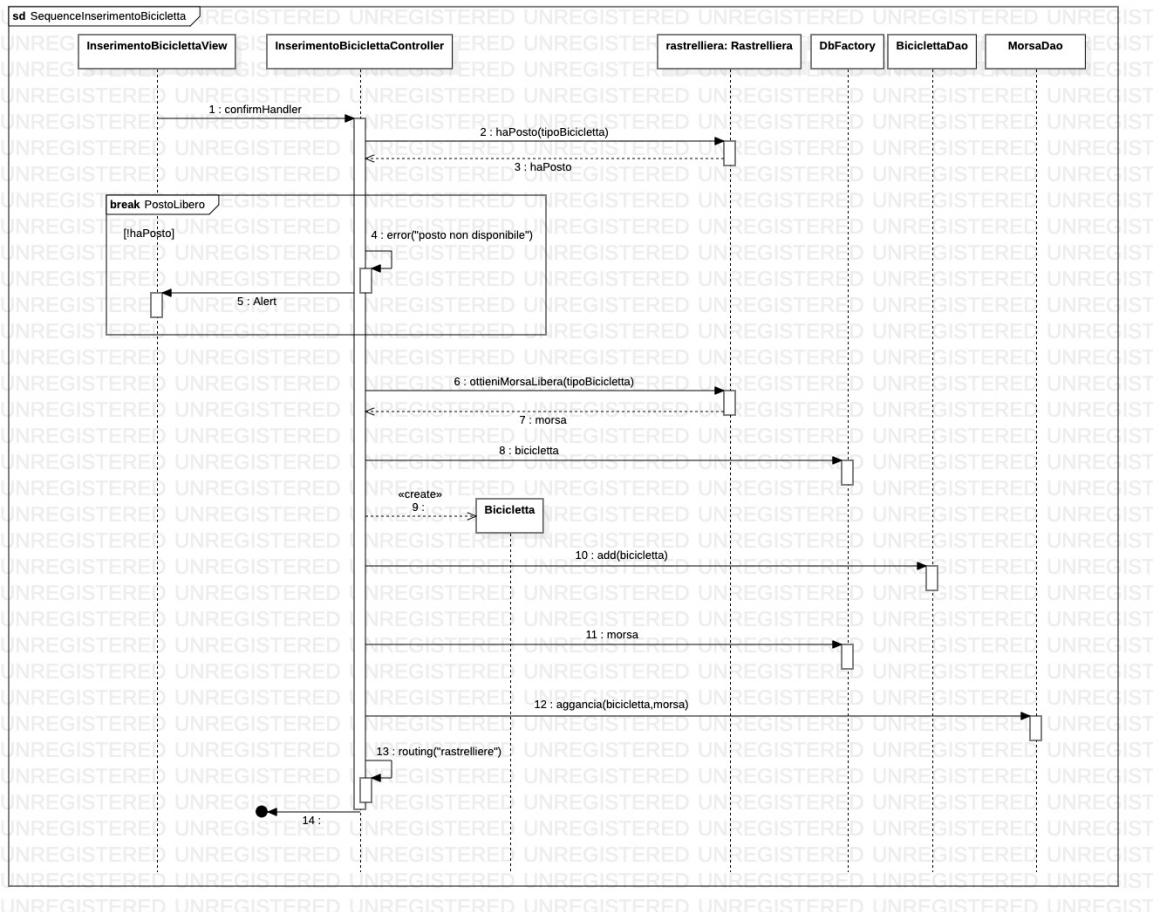
### 2.4.3.2. Inserimento rastrelliera



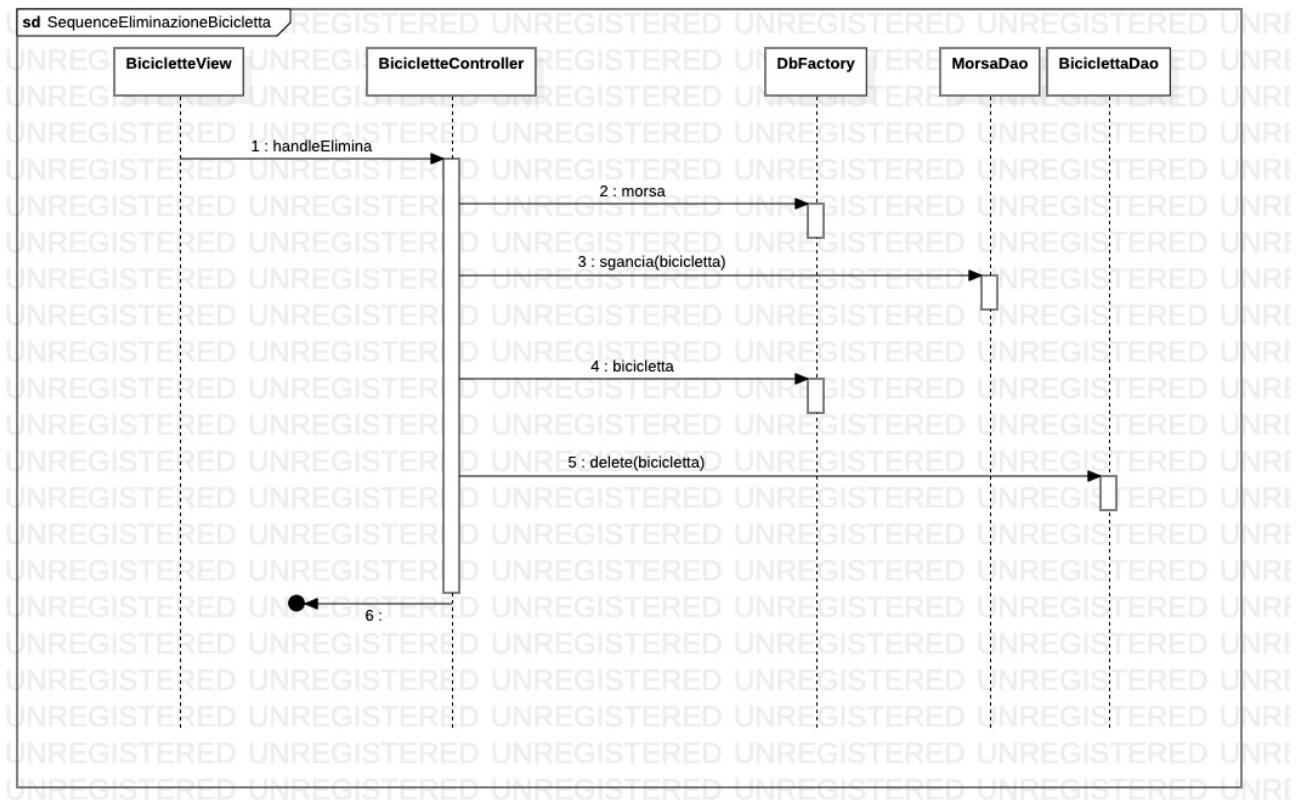
### 2.4.3.3. Spostamento bicicletta



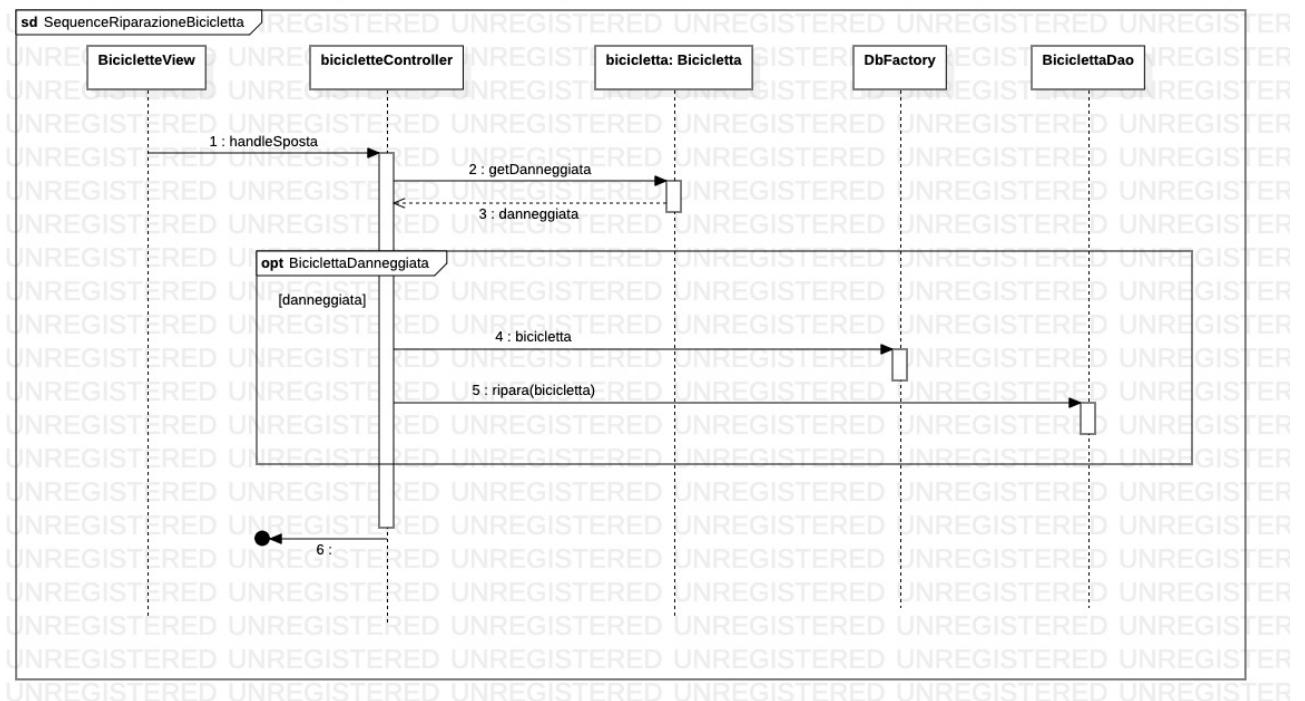
### 2.4.3.4. Inserimento bicicletta



#### 2.4.3.5. Eliminazione bicicletta

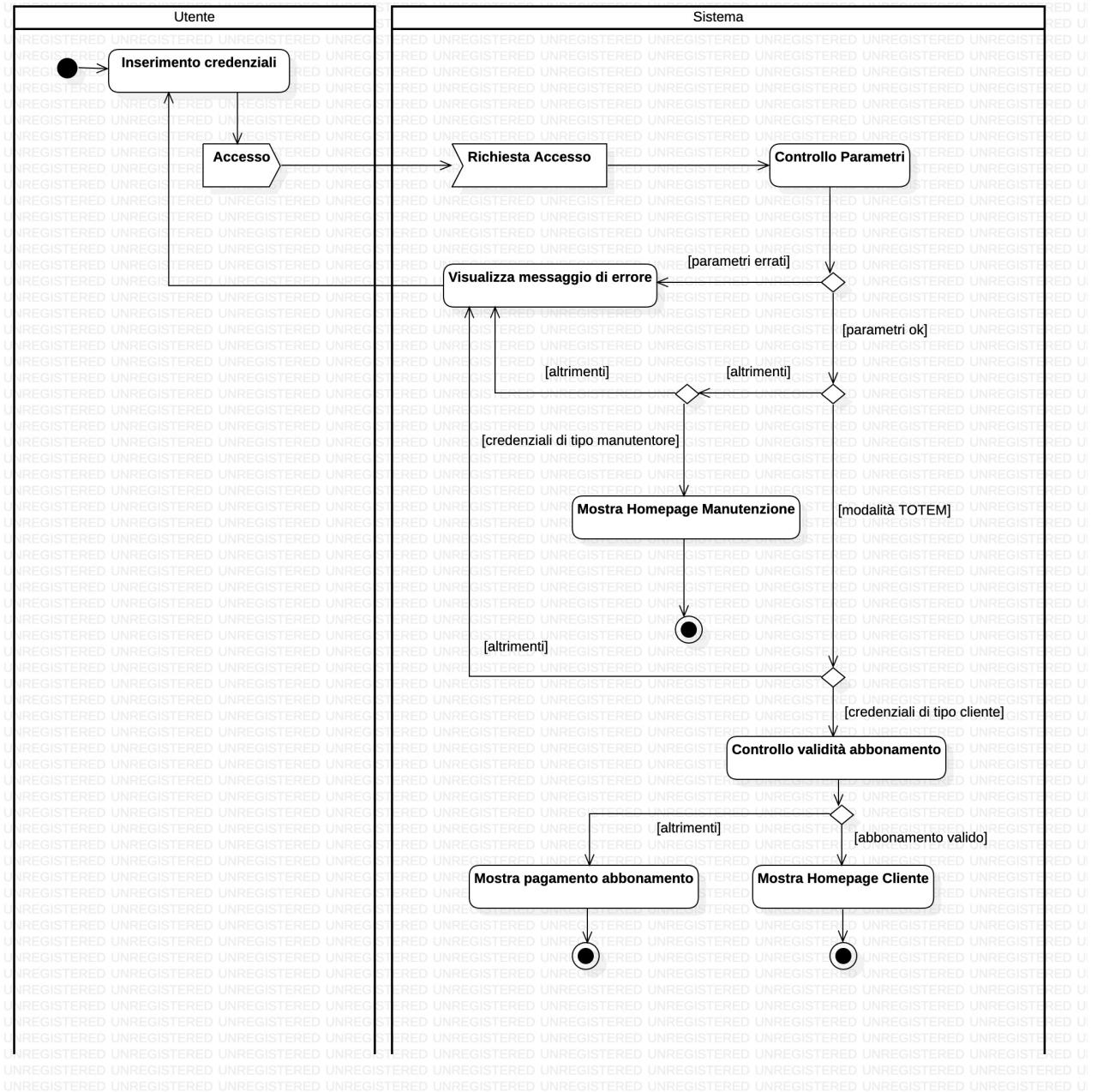


#### 2.4.3.6. Riparazione bicicletta

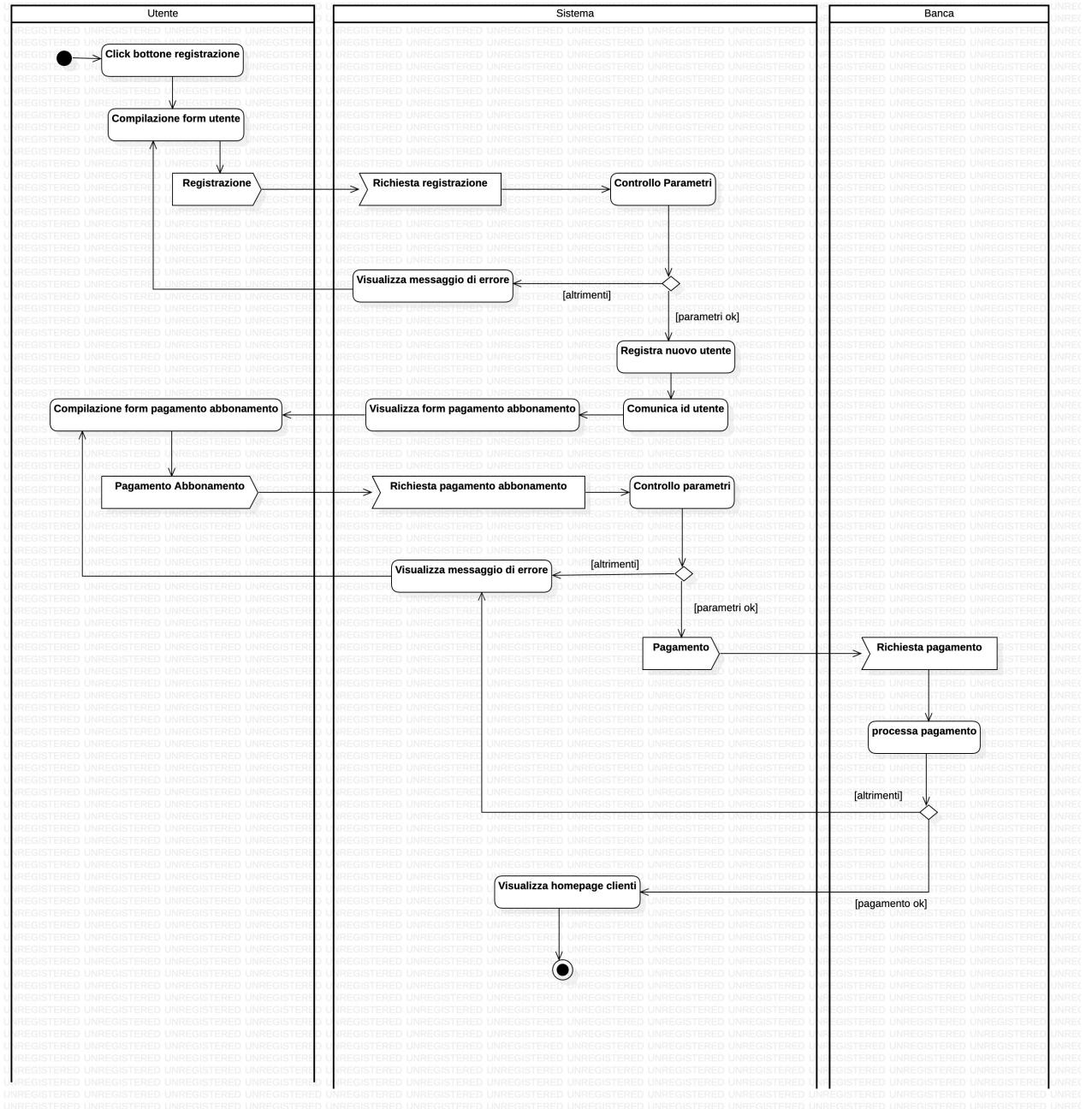


## 2.5. Diagrammi delle attività

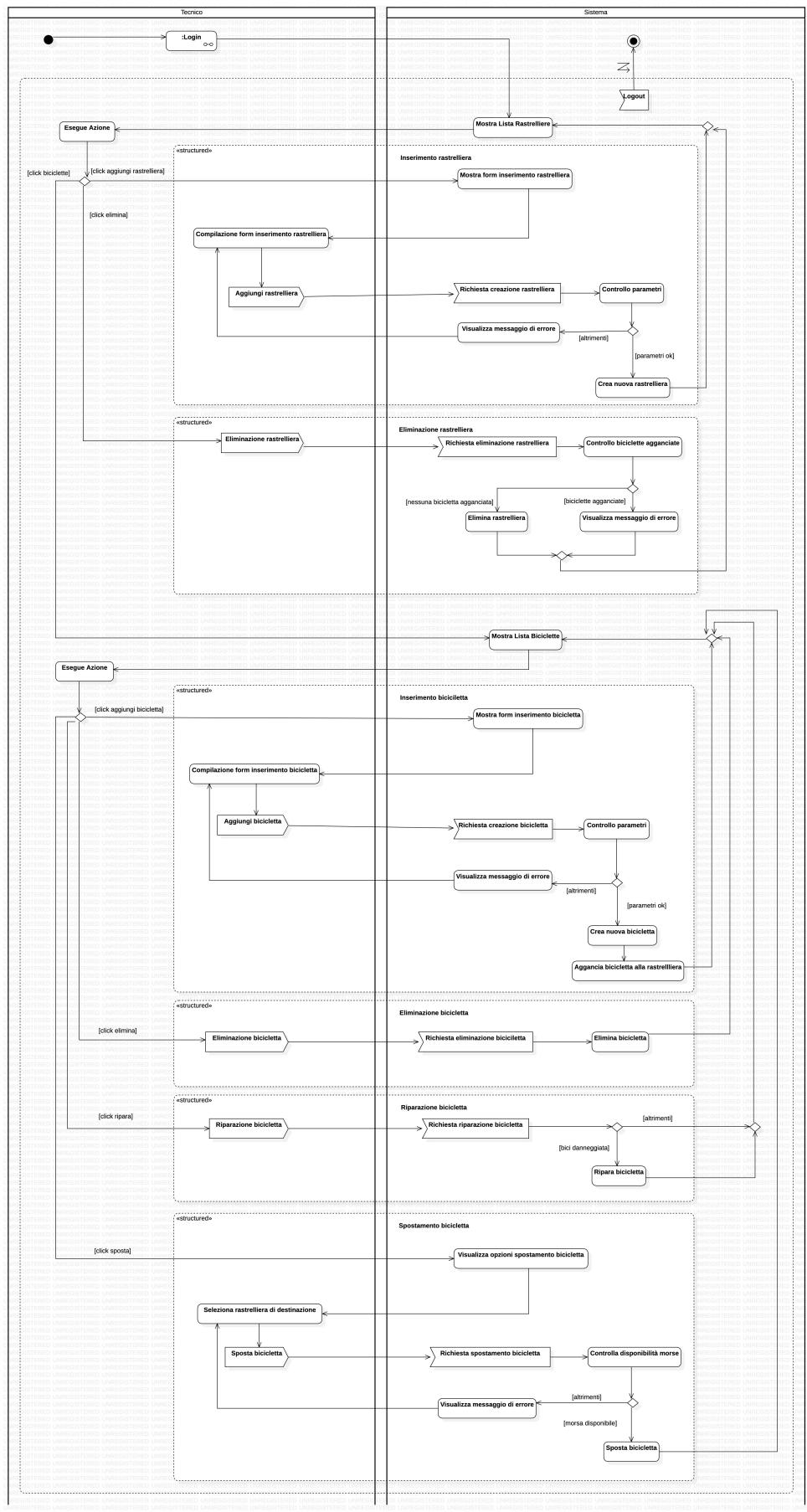
### 2.5.1. Login



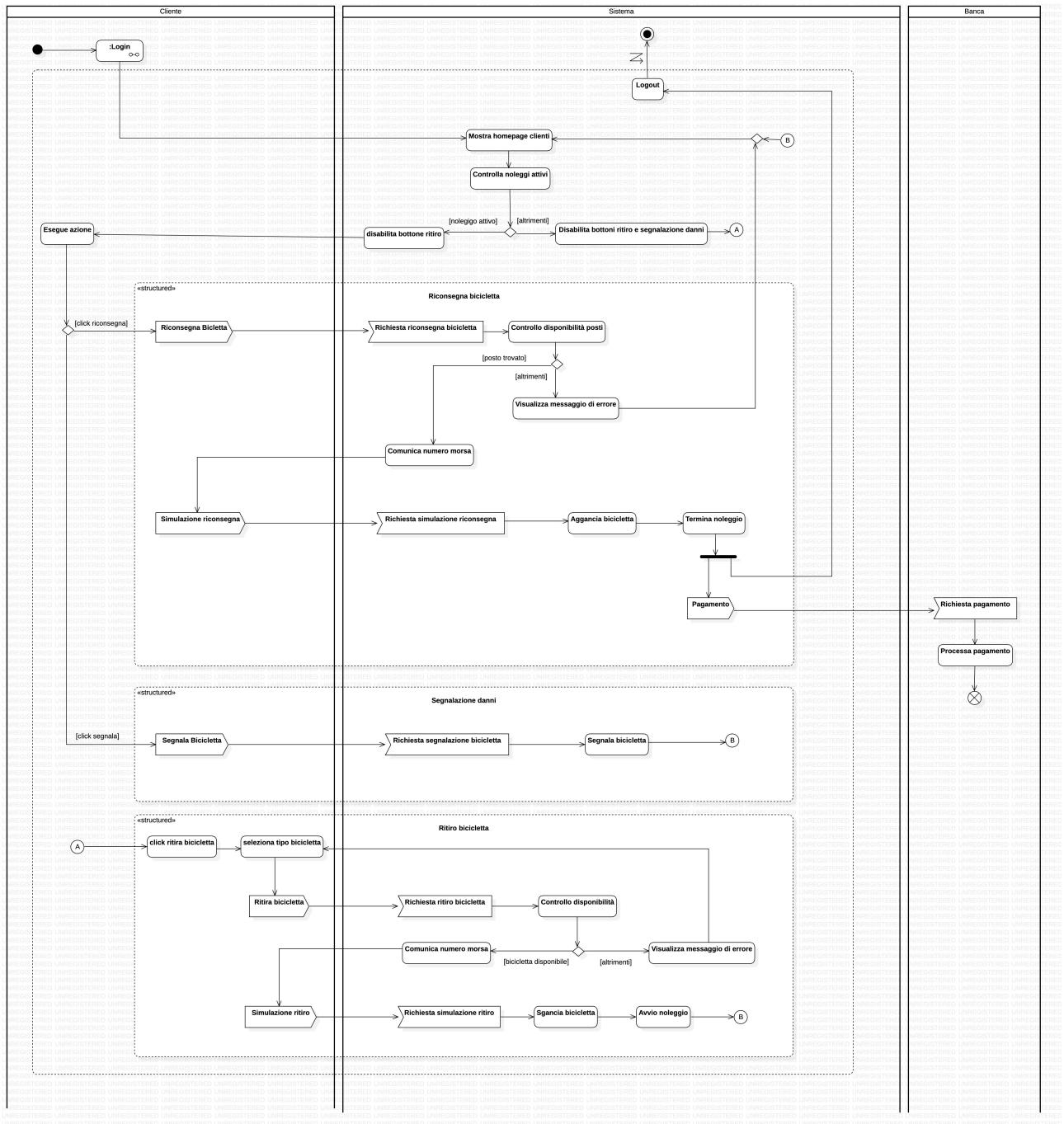
## 2.5.2 Registrazione



### 2.5.3 Manutenzione

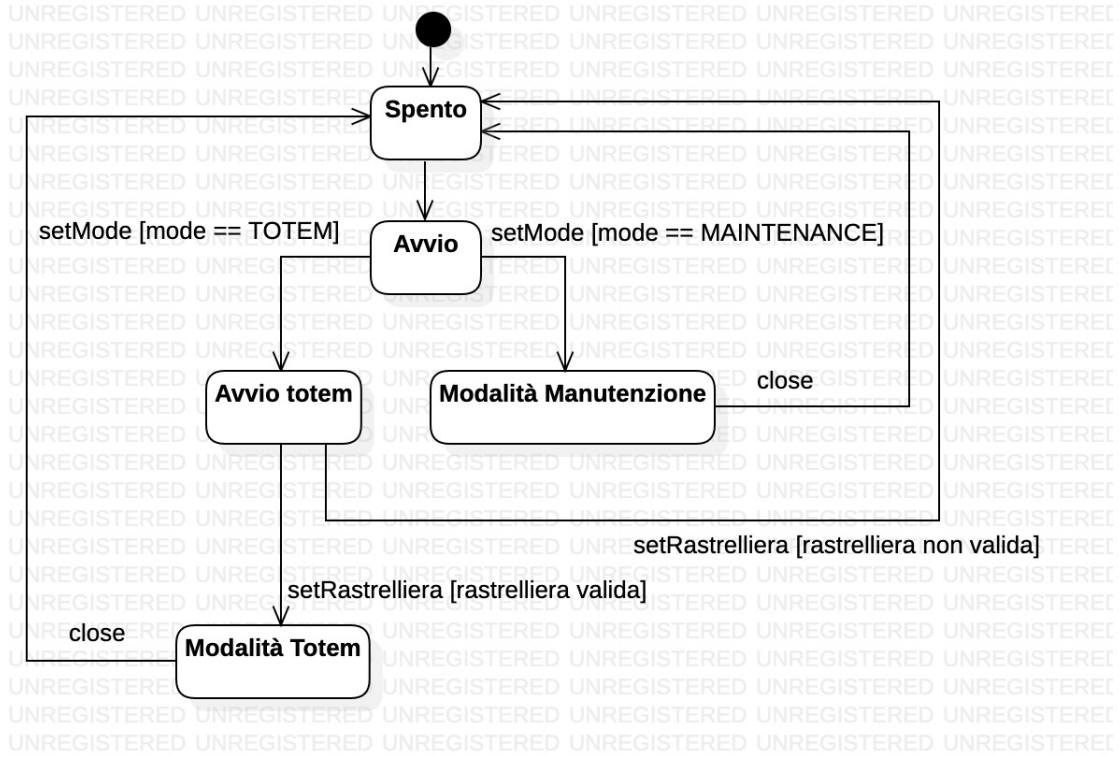


## 2.5.4 Totem

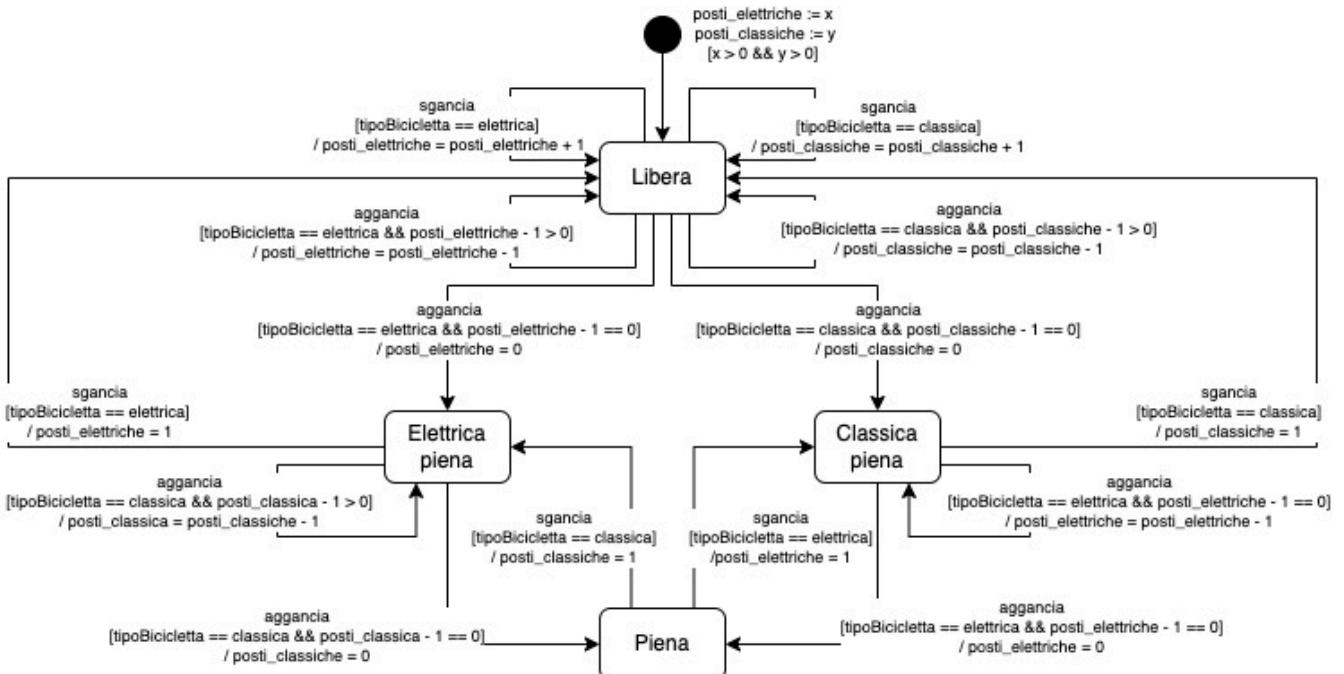


## 2.6. Macchine di stato

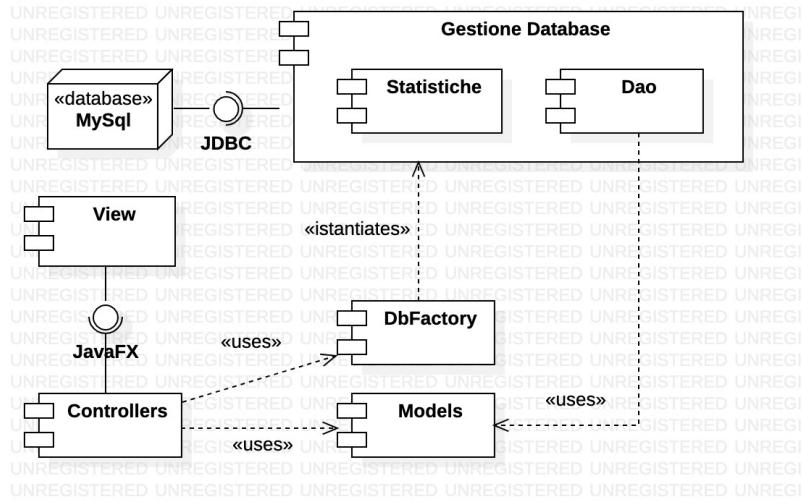
### 2.6.1 Modalità applicazione



### 2.6.2 Stato rastrelliera



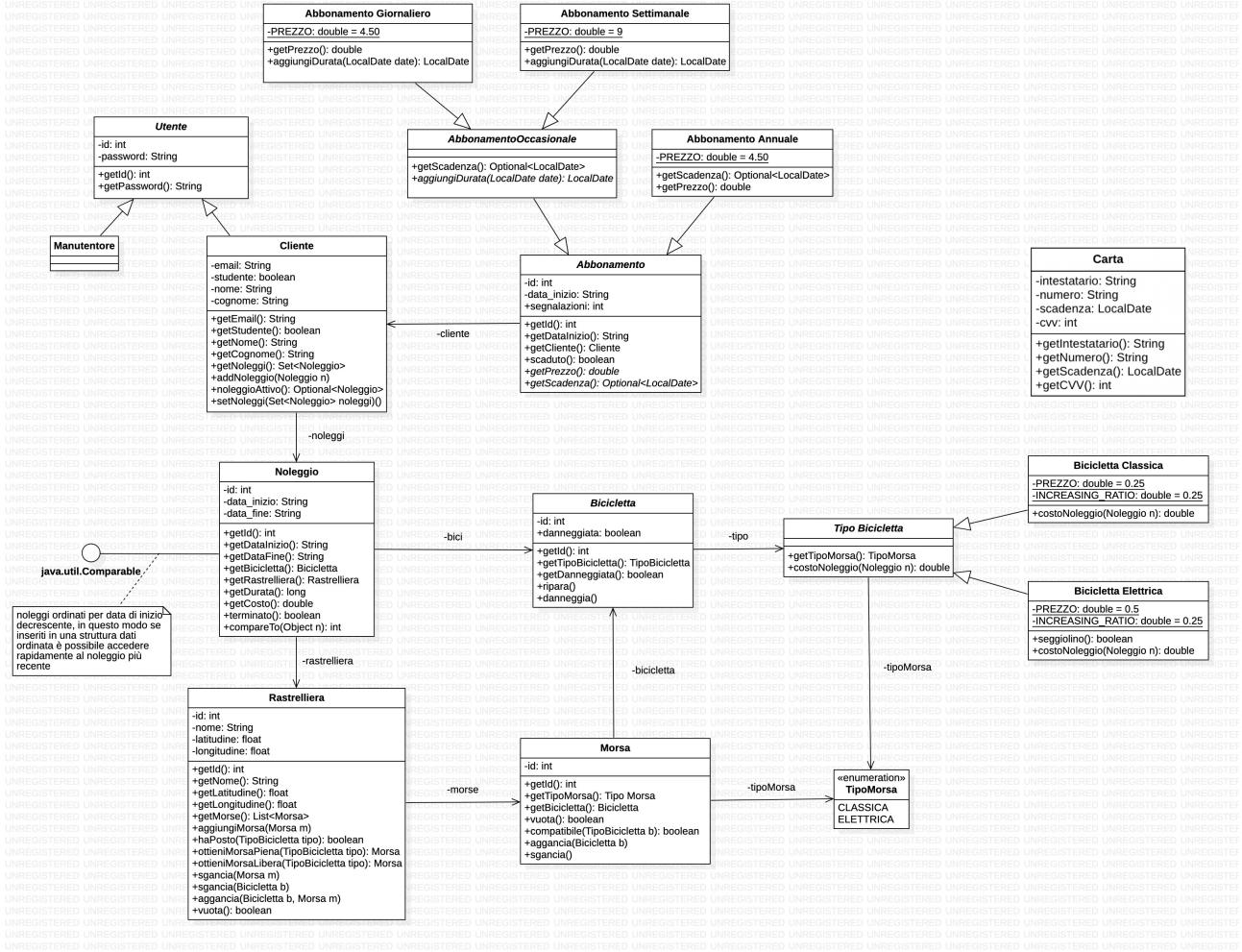
## 2.7. Diagramma dei componenti



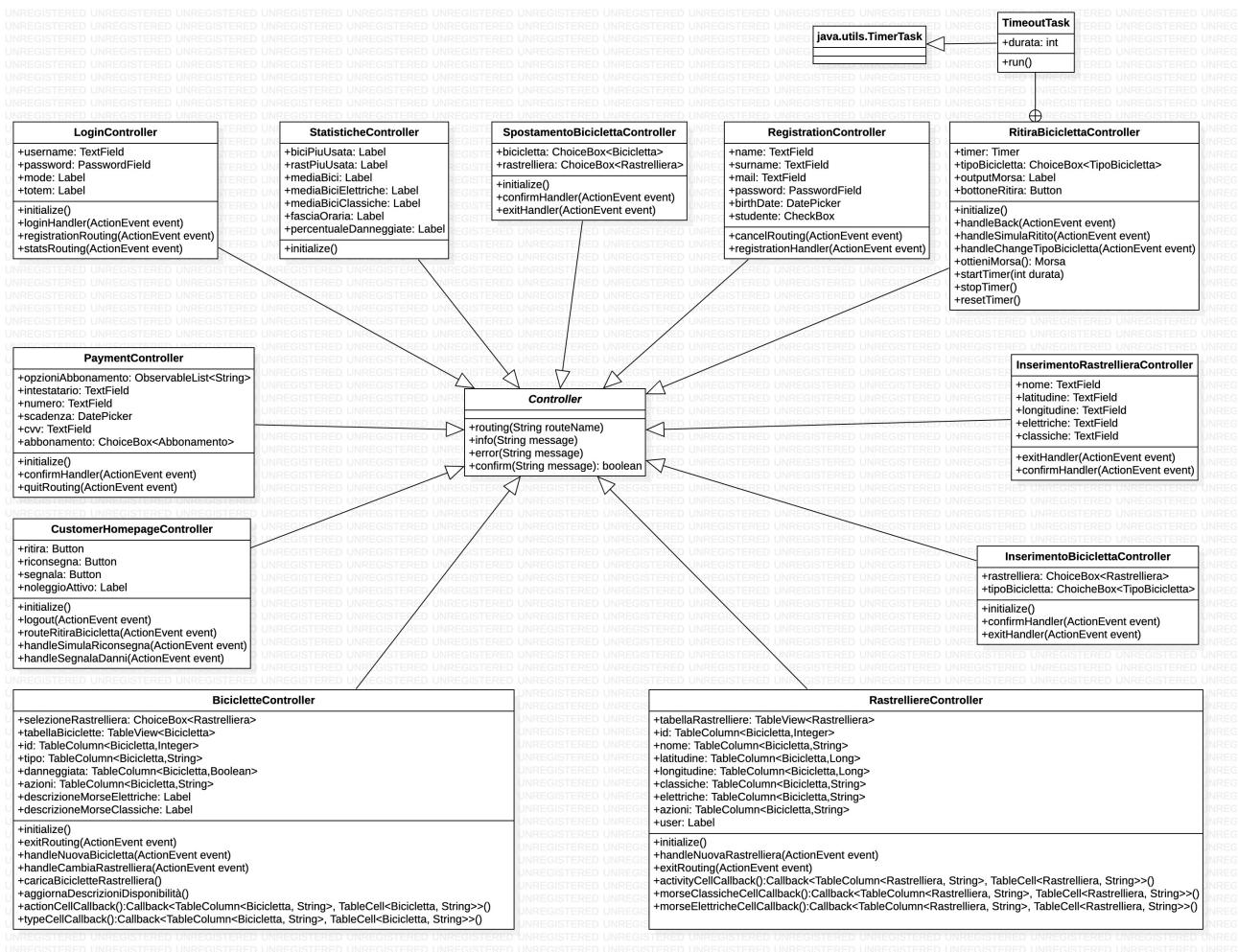
### 3. Implementazione del sistema

### 3.1. Diagramma delle classi (modello di programma)

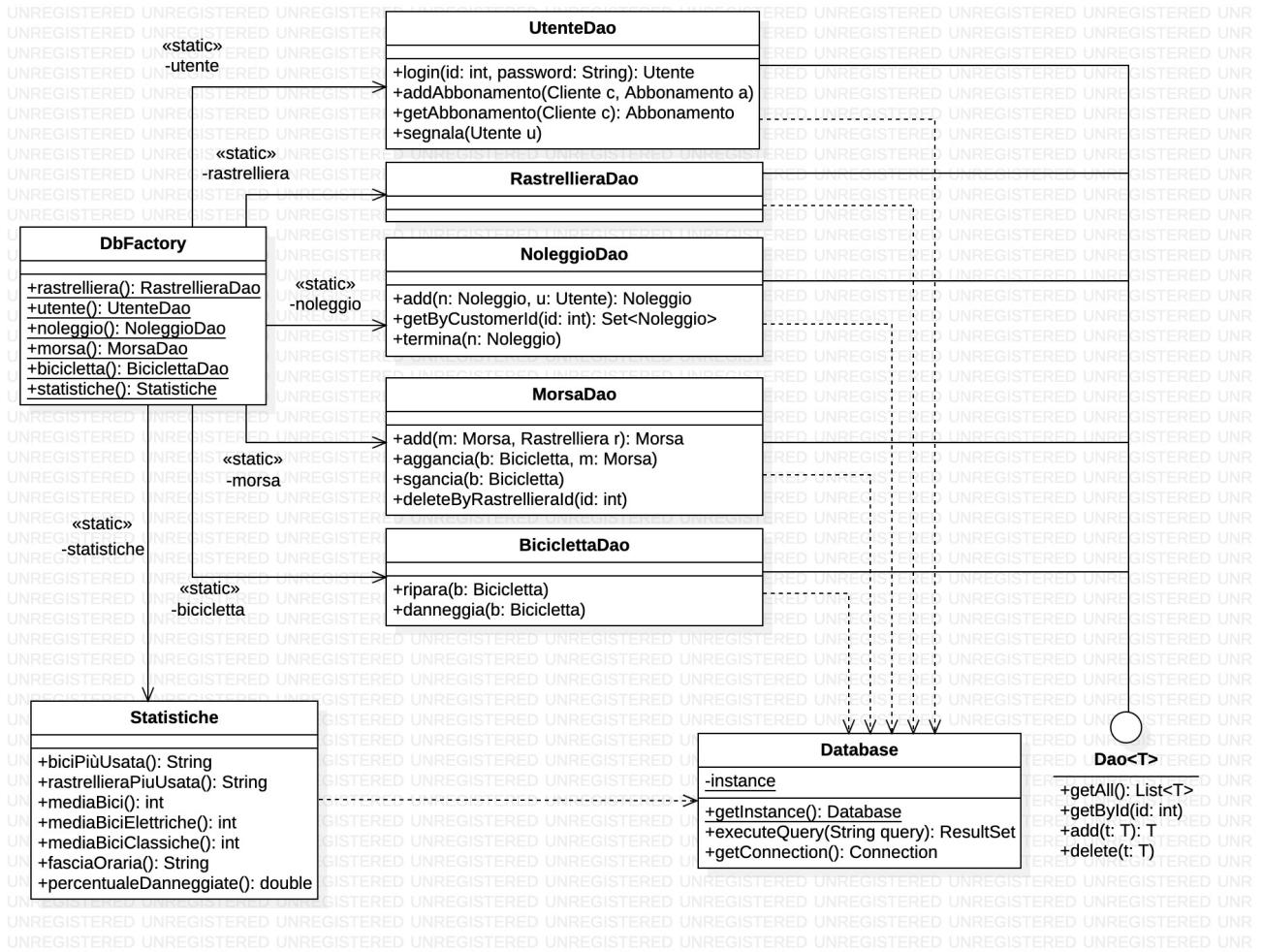
### 3.1.1 Modelli



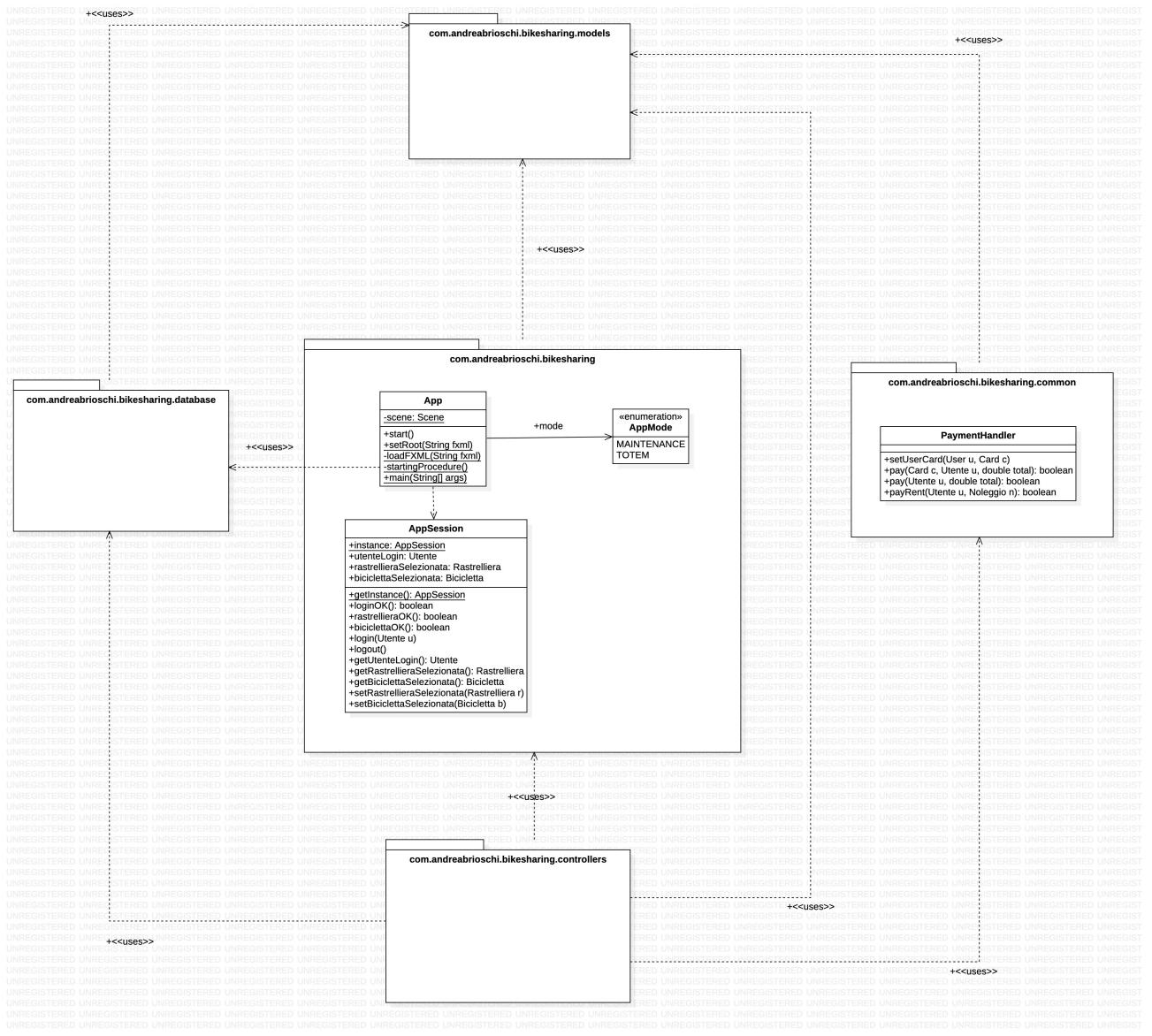
### 3.1.2 Controllers



### 3.1.3 Persistenza



### 3.1.4 Pacchetti



Per rendere facilmente comprensibile il diagramma delle classi riportato è stato scelto di riportare l'interazione fra classi tramite le dipendenze fra i pacchetti che le contengono.

I pacchetti **com.andreabrioschi.bikesharing.database**, **com.andreabrioschi.bikesharing.models** e **com.andreabrioschi.bikesharing.controllers** corrispondono ai diagrammi delle classi presentati nelle pagine immediatamente precedenti a questa.

### 3.1.5 Discussione dei design pattern utilizzati

#### 3.1.5.1 Dao

Il design pattern Data Access Object (DAO) è stato utilizzato per gestire parte della persistenza dei dati utili al programma, il pattern è stato implementato tramite l'interfaccia Dao<T> e le classi che la implementano.

L'interfaccia Dao<T> si occupa di definire le operazioni principali sugli oggetti di tipo T: ottenere tutti gli oggetti, ottenere un oggetto in base al suo id, aggiungere un nuovo oggetto e eliminare un oggetto.

Molte delle classi che implementano l'interfaccia Dao<T> contengono dei metodi aggiuntivi specifici per l'ambito di applicazione.

#### 3.1.5.2 MVC

L'intera applicazione è stata pensata seguendo il pattern Model View Controller (MVC) come è evidente dall'organizzazione dei pacchetti del codice sorgente, in questo modo è stato possibile separare la logica del software dalla gestione dei dati e dalle scelte di visualizzazione.

In particolare, all'interno del pacchetto com.bikesharing.andreabrioschi.models sono presenti tutti i modelli che rappresentano il campo Model del pattern, nel pacchetto com.bikesharing.andreabrioschi.controllers sono presenti tutti i controllori che rappresentano il campo Controller del pattern e il campo View del pattern è implementato tramite file .FXML che vengono interpretati da JavaFX.

#### 3.1.5.3 Singleton

Il design pattern Singleton è stato usato in diverse occasioni all'interno dell'applicazione per garantire la presenza di un'unica istanza di una classe:

Database -> questa classe si occupa della connessione al database e dell'esecuzione dei comandi di basso livello su di esso.

AppSession -> questa classe viene utilizzata come sessione per l'applicazione, al suo interno vengono salvate diverse informazioni come l'utente connesso e la rastrelliera selezionata.

#### 3.1.5.4 Factory

Il design pattern Factory è stato usato per gestire le classi che si occupano dello scambio dei dati con il database ed è stato scelto poiché rende semplice e rapida la creazione di oggetti per accedere al database, oltre a facilitare la manutenzione.

Il pattern è stato implementato tramite la classe DbFactory e i suoi metodi statici.

#### 3.1.5.5 Lazy Loading

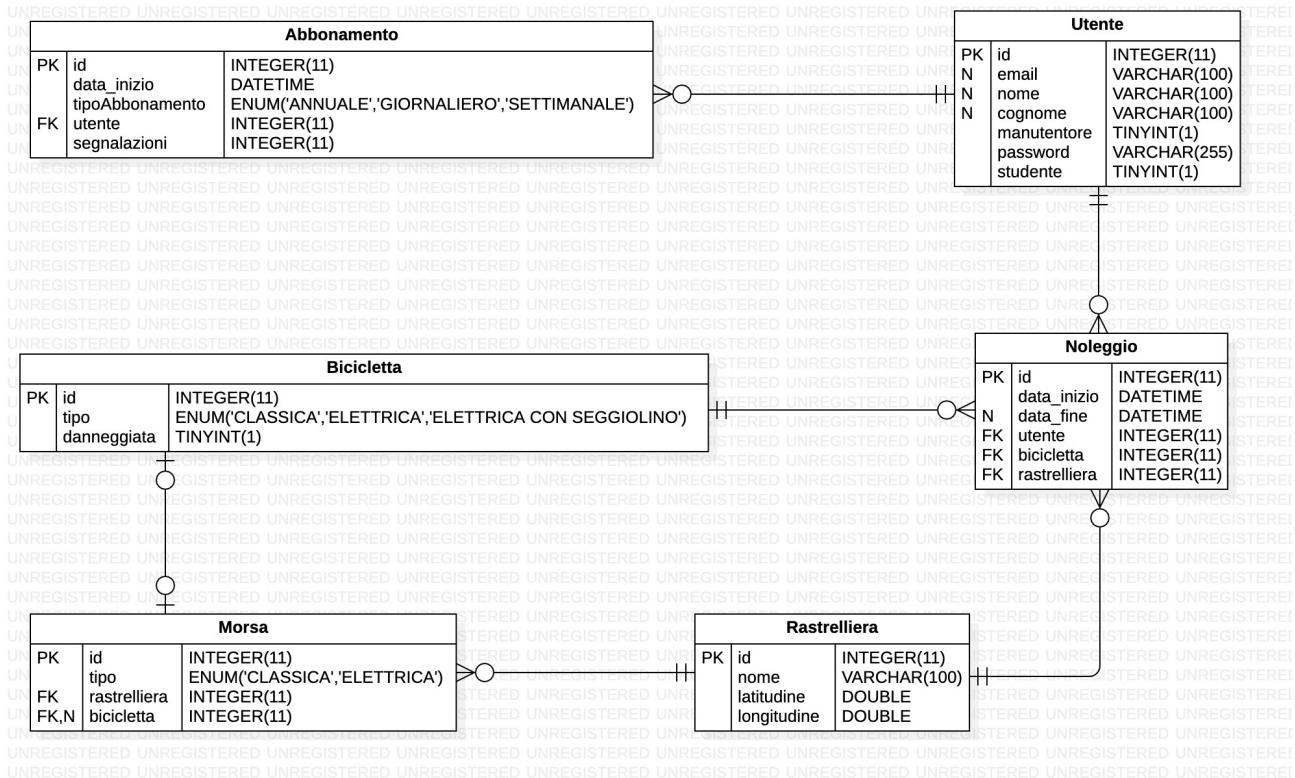
La classe DbFactory utilizza il pattern Lazy Loading per istanziare le classi restituite, in questo modo le istanze vengono create solo quando necessarie e se lo stesso metodo viene chiamato più volte vengono restituite sempre le stesse.

#### 3.1.5.6 Observer

Il pattern Observer viene applicato automaticamente da JavaFX, infatti all'interno del file .FXML che definisce la vista è possibile legare un metodo della classe Controller ad un particolare evento della vista, in questo modo il metodo verrà chiamato automaticamente quando si verifica l'evento.

## 3.2. Gestione dei dati persistenti

### 3.2.1 Diagramma ER



### 3.2.2 Caratteristiche

**Abbonamento:** contiene tutti gli abbonamenti validi o non ancora eliminati, se il numero di segnalazioni è  $\geq 3$  l'abbonamento non è più valido.

**Utente:** contiene sia i clienti che i manutentori, i manutentori sono contrassegnati tramite il campo 'manutentore'. Le password salvate in questa tabella vengono cifrate con un meccanismo di hashing.

**Noleggio:** contiene tutti i noleggi attivi e i noleggi passati, un noleggio è attivo se il campo data\_fine è NULL. All'interno del noleggio viene salvata solamente la rastrelliera di origine, in questo modo è possibile effettuare statistiche sull'utilizzo delle rastrelliere senza mantenere un tracciamento delle tratte percorse dagli utenti durante i noleggi.

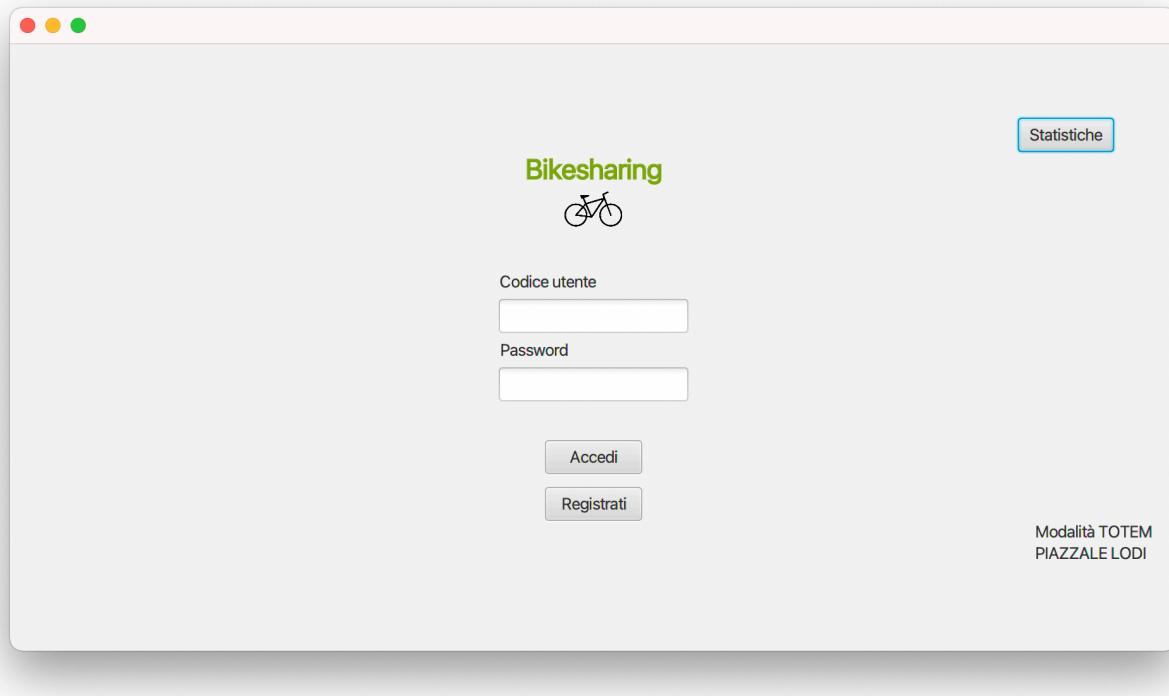
**Bicicletta:** contiene tutte le biciclette e le informazioni sui danni.

**Morsa:** contiene tutte le morsa, se il campo bicicletta è NULL significa che la morsa è vuota.

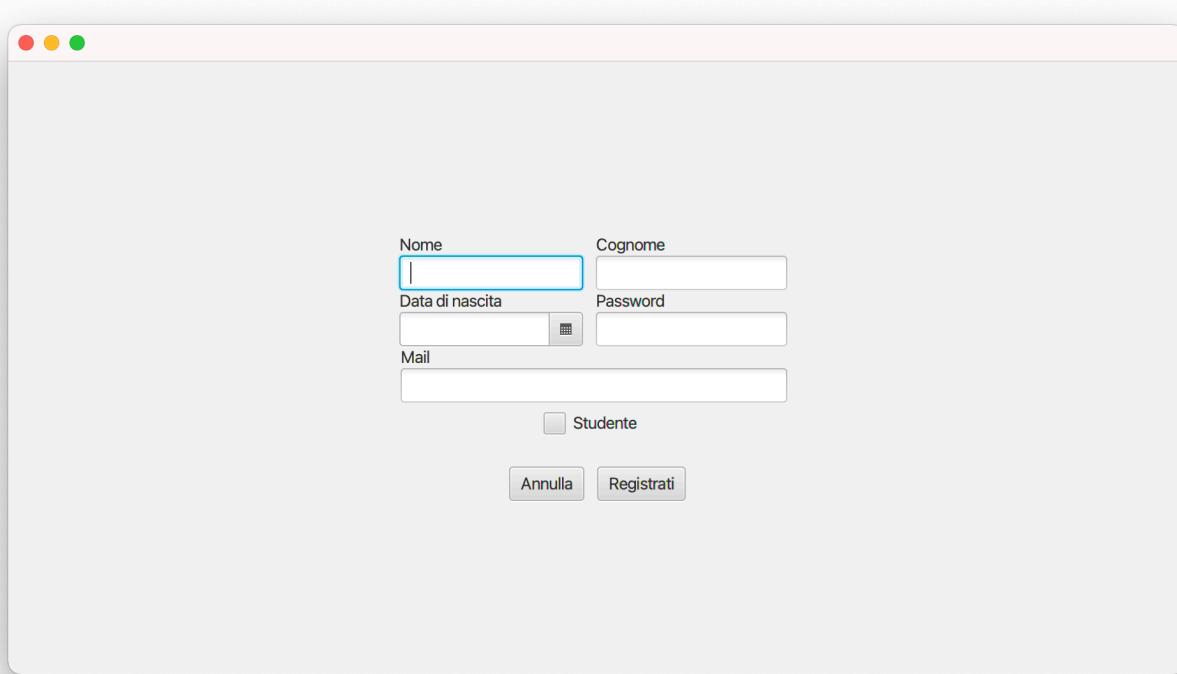
**Rastrelliera:** contiene tutte le rastrelliere e la loro posizione.

### 3.3. Descrizione dell'Interfaccia Grafica

#### 3.3.1 Login



#### 3.3.2 Registrazione



- Nome, cognome, data di nascita, password e mail non devono essere vuoti

### 3.3.3 Pagamento Abbonamento

**Tipo Abbonamento**

**Pagamento**

Intestatario

Numero carta

Scadenza  CVV

Lo stesso metodo di pagamento verrà utilizzato anche per i noleggi

- Intestatario, numero carta, scadenza e cvv non devono essere vuoti.
  - La data di scadenza della carta deve essere minima ad un anno dalla data attuale

### 3.3.4 Rastrelliere

### 3.3.5 Inserimento Rastrelliera

The screenshot shows a Mac OS X style window titled "Creazione Rastrelliera". It contains five input fields: "Nome" (Name), "Latitudine" (Latitude), "Longitudine" (Longitude), "Numero morse classiche" (Number of classic Morse codes), and "Numero morse elettriche" (Number of electric Morse codes). Below the fields are two buttons: "Annulla" (Cancel) and "Aggiungi rastrelliera" (Add rack).

- Nome, latitudine, longitudine, numero morse elettriche e numero morse classiche non devono essere vuoti
- Latitudine, Longitudine, numero morse elettriche e numero morse classiche devono essere di tipo numerico

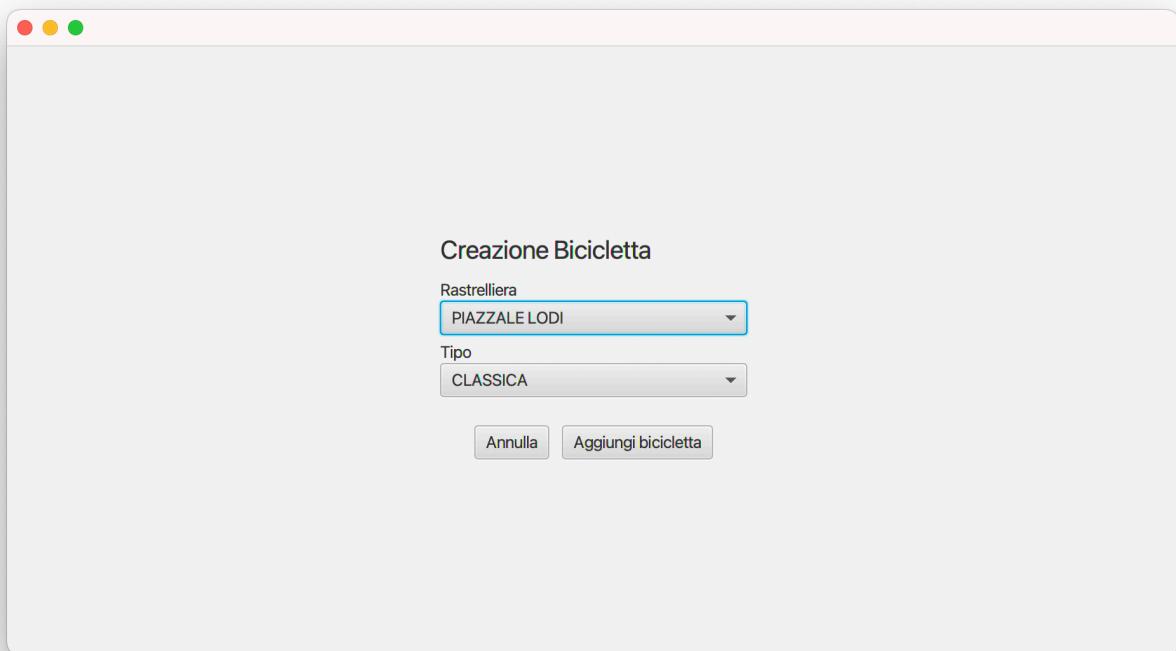
### 3.3.6 Biciclette

The screenshot shows a Mac OS X style window titled "Gestione Biciclette". A dropdown menu labeled "Rastrelliera:" is set to "PIAZZALE LODI". Below it, a message indicates "4 morse elettriche su 5 disponibili" and "8 morse classiche su 10 disponibili". A table lists three bicycles:

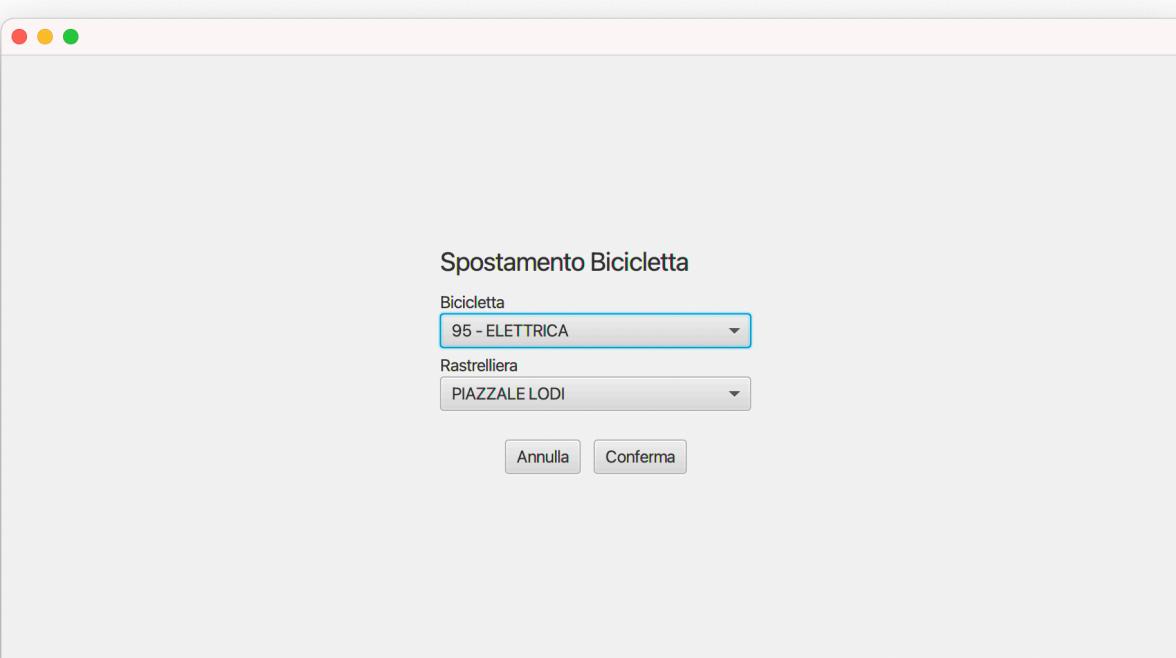
ID	tipo	danneggiata	Azioni		
95	ELETTRICA	false	Sposta	Elimina	Ripara
55	CLASSICA	false	Sposta	Elimina	Ripara
92	CLASSICA	false	Sposta	Elimina	Ripara

At the bottom are two buttons: "Indietro" (Back) and "Aggiungi Bicicletta" (Add Bicycle).

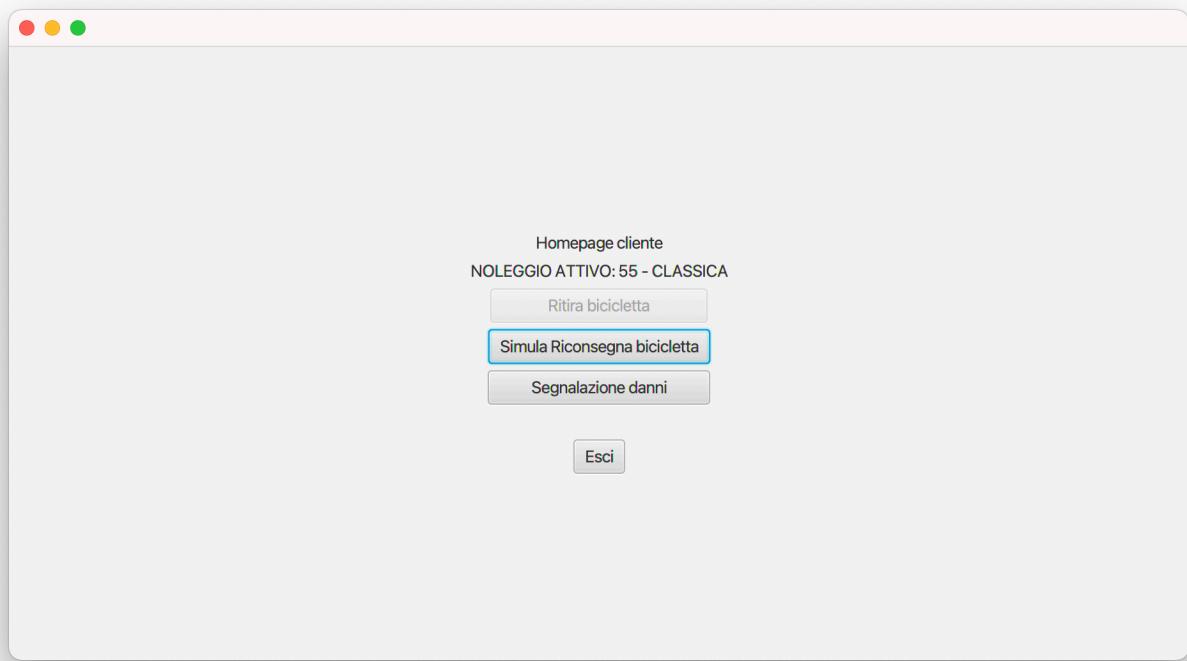
### 3.3.7 Inserimento Bicicletta



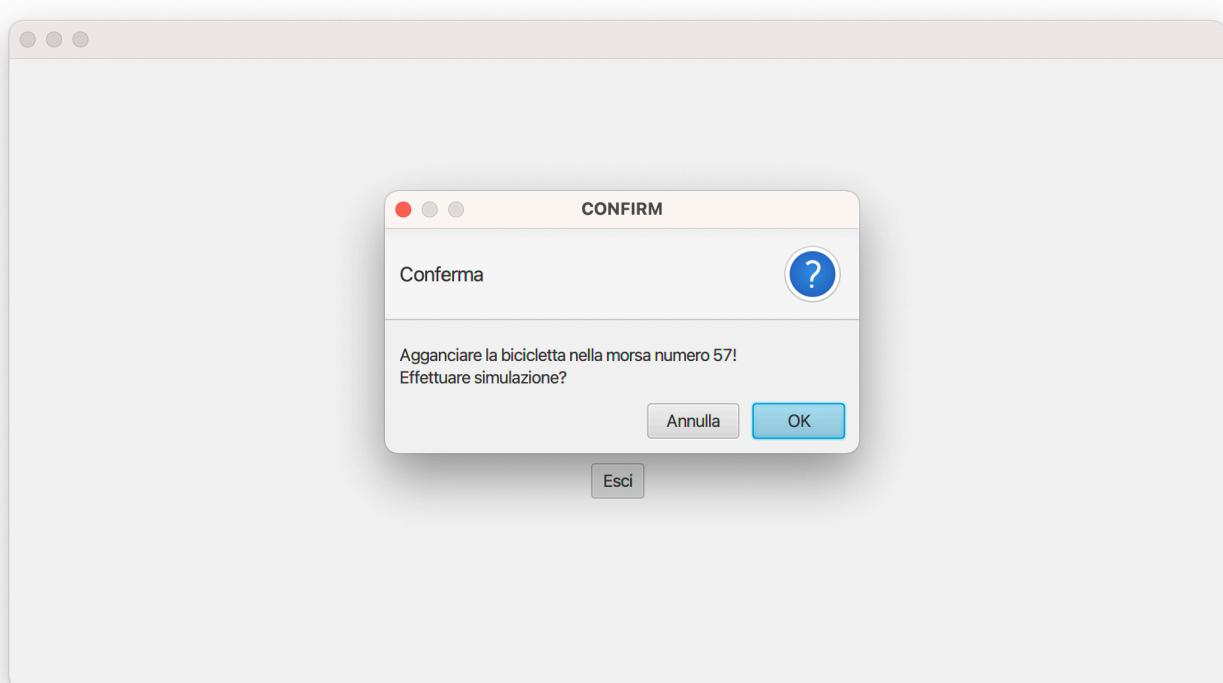
### 3.3.8 Spostamento bicicletta



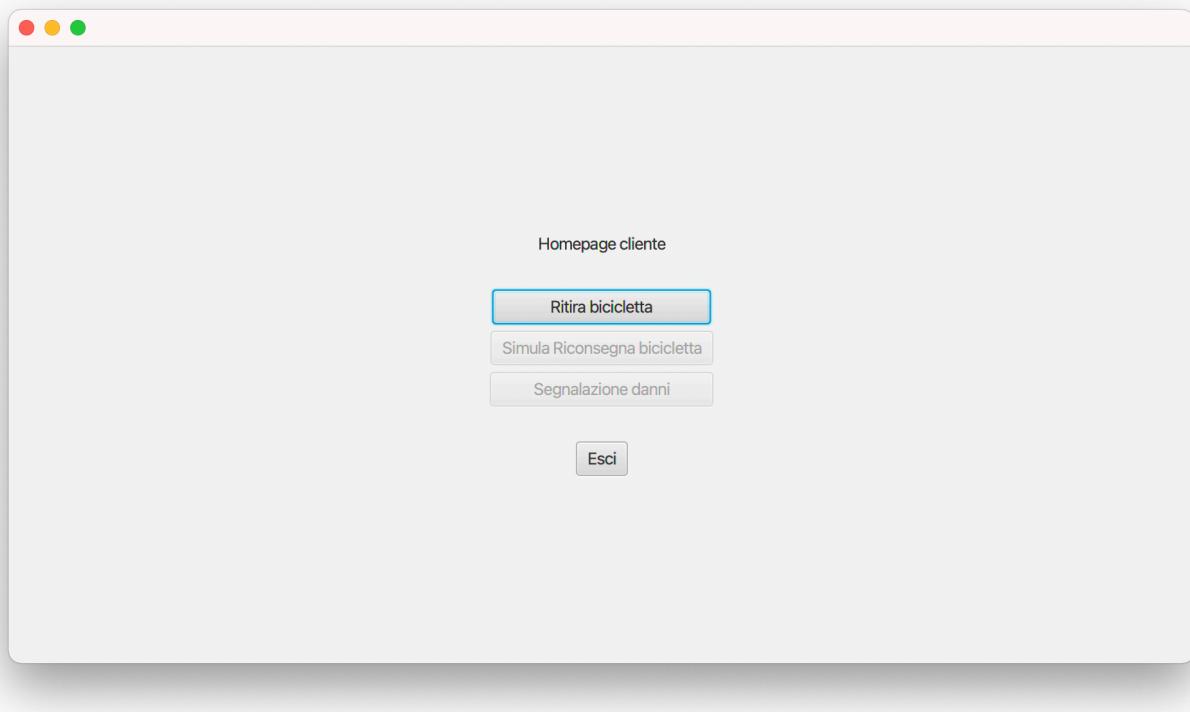
### 3.3.9 Homepage Clienti (Noleggio attivo)



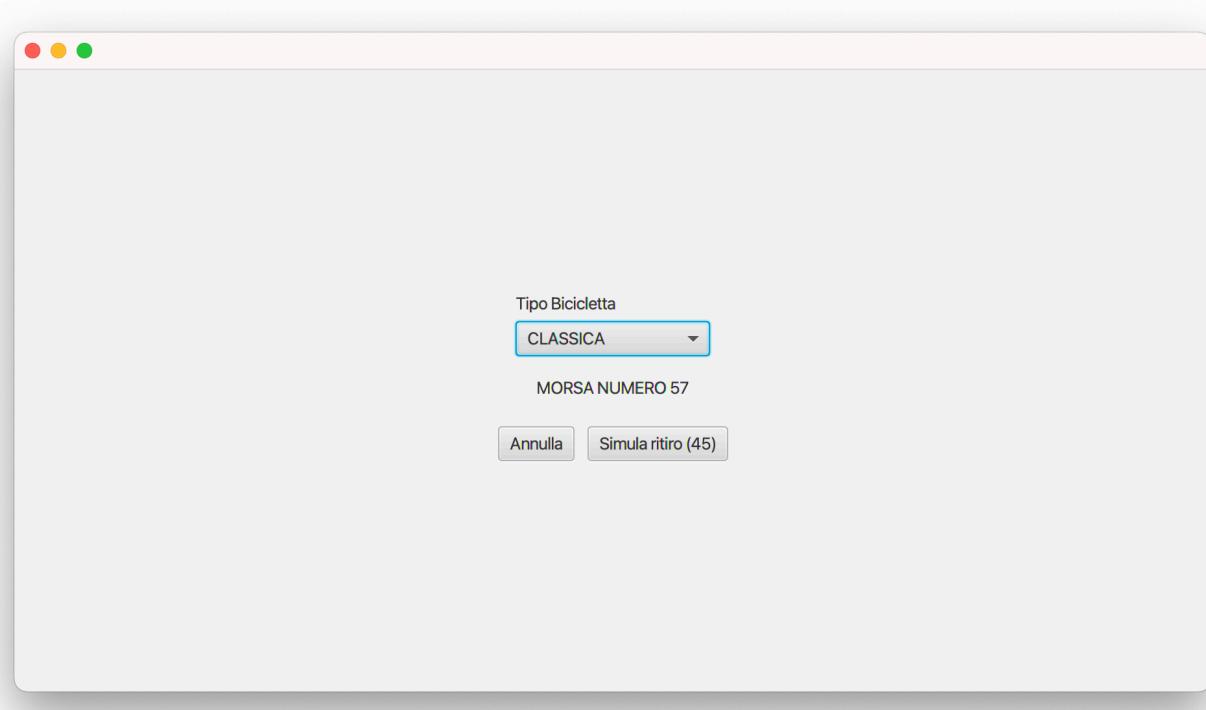
### 3.3.10 Homepage Clienti (Simula riconsegna)



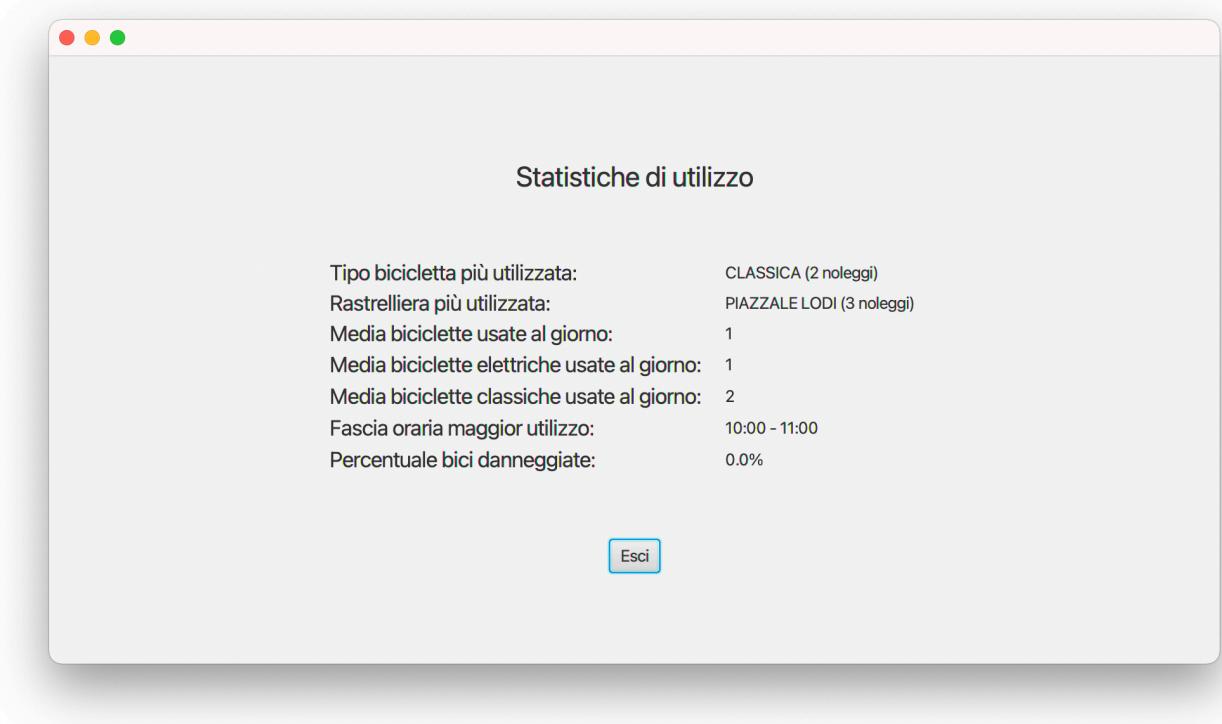
### 3.3.11 Homepage Clienti (Nessun noleggio attivo)



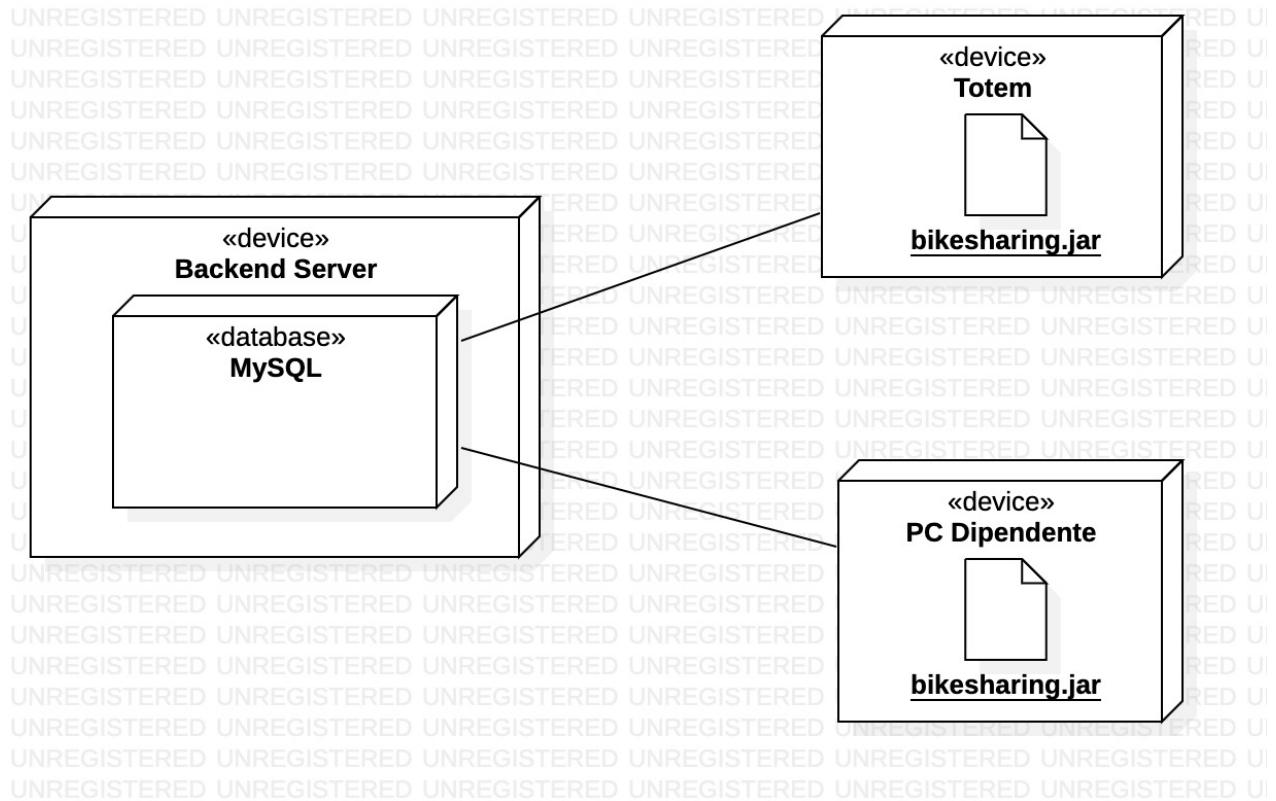
### 3.3.12 Ritiro bicicletta



### 3.3.13 Statistiche



### 3.4. Diagramma di deployment



### 3.5. Specifica e verifica dei vincoli

**context** Noleggio **inv:**

self.bicicletta not null  
self.rastrelliera not null  
self.dataInizio not null

**context** Noleggio::getDurata() **pre:**

self.dataFine not null

**context** Abbonamento **inv:**

self.cliente not null  
self.dataInizio not null  
self.segnalazioni > 0  
self.segnalazioni >= 3 implies self.getScadenza().get() = LocalDate.MIN

**context** AbbonamentoOccasionale **inv:**

self.cliente.getNoleggi().isEmpty() implies self.getScadenza().isEmpty()

**context** Bicicletta **inv:**

self.tipo not null

**context** TipoBicicletta **inv:**

self.morsa not null

**context** Morsa **inv:**

self.tipoMorsa not null  
self.bicicletta = null implies self.vuota()

**context** Morsa::aggancia(b: Bicicletta) **pre:**

self.vuota() and self.compatibile(b.getTipoBicicletta())

**context** Morsa::aggancia(b: Bicicletta) **post:**

self.bicicletta = b

### 3.6. Descrizione del testing

Il testing è stato effettuato con l'ausilio del framework JUnit, sono stati preparati test solo per le componenti più critiche del sistema utilizzando, per tutti i test è stato utilizzato un criterio di copertura delle istruzioni.

Il dump del database fornito può essere utilizzato anche per costruire il database di test, requisito fondamentare per la corretta esecuzione dei test è che i dati presenti nel dump non vengano alterati poichè all'interno dei test sono presenti riferimenti a specifici identificativi di morsa e biciclette.

### 3.6.1 Test connessione al database

I primi test effettuati sono sulla classe che si occupa della connessione al database, sono stati testati i metodi che permettono di:

- Eseguire una semplice query (test metodo Database.executeQuery)
- Ottenere una connessione al database (test metodo Database.getConnection)
- Assicurarsi che venga creata un'unica istanza della classe (test metodo Database.getInstance)

### 3.6.2 Test operazioni su database

Altri test sono stati effettuati sulle classi che si occupano dell'accesso al database, questi test sono fatti in modo da riportare il database di test allo stato iniziale se eseguiti tutti con successo:

- Effettuare il login (test metodo UtenteDao.login)
- Effettuare la registrazione (test metodo UtenteDao.add)
- Eliminare un utente (test metodo UtenteDao.delete)
- Agganciare una bicicletta ad una morsa (test metodo MorsaDao.aggancia)
- Sganciare una bicicletta da una morsa (test metodo MorsaDao.sgancia)

### 3.6.3 Test sui modelli

Ulteriori test sono stati preparati per i modelli utilizzati dall'applicazione:

- Calcolo del totale di un noleggio (test metodo Noleggio.getCosto)
- Validità abbonamento (test metodo Abbonamento.scaduto)
- Prezzo abbonamento (test metodo Abbonamento.getPrezzo)
- Disponibilità posti rastrelliera (test metodo Rastrelliera.haPosto)

## 3.7. Note per l'installazione e l'utilizzo

### 3.7.1 Ambiente di sviluppo

L'ambiente di sviluppo dell'applicazione è Apache NetBeans 12.5 all'interno del quale è stato creato un progetto basato su Apache Maven.

Tramite il plugin Apache Maven Shade Plugin l'applicazione è stata poi compilata in un pacchetto di tipo uber-jar che oltre a contenere la build dell'applicazione contiene anche i moduli mysql-connector e JavaFX, necessari per l'esecuzione.

### 3.7.2 Configurazione database

Allegato al progetto è presente un dump del database, contenuto nel file bikesharing.sql.

Attualmente l'applicazione è configurata per aspettarsi il database in locale sulla porta 3306 con nome bikesharing, e accede con l'utente root con password vuota.

La stringa di connessione utilizzata è la seguente:

*jdbc:mysql://root@localhost:3306/bikesharing*

All'interno del database sono già stati creati tre utenti:

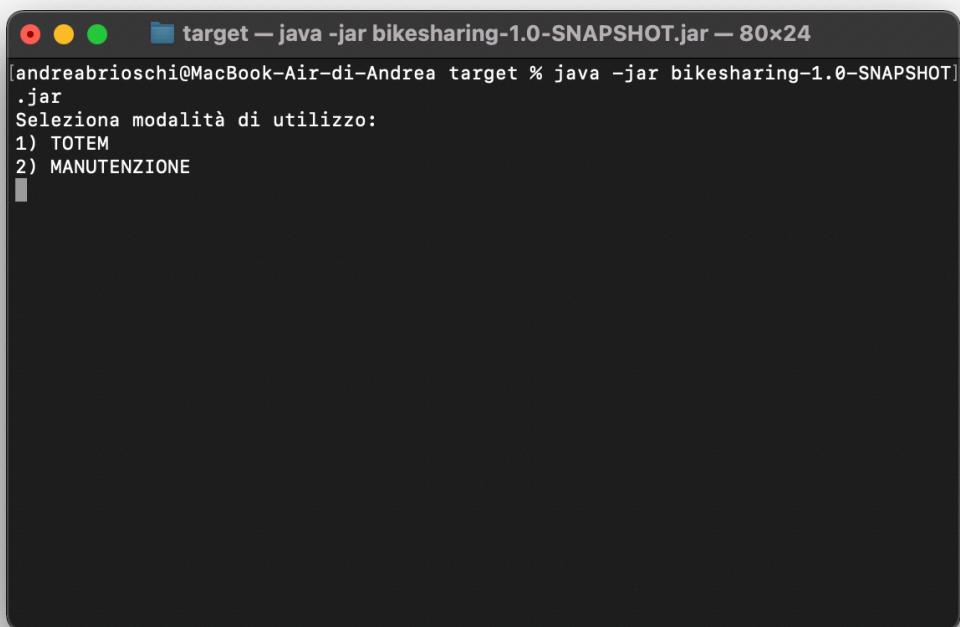
- ID: 1, Password: testPwd, Studente: false, Manutentore: true
- ID: 2, Password: testPwd, Studente: true, Manutentore: false
- ID: 3, Password: a, Studente: false, Manutentore: false

### 3.7.2 Avvio

Dato che mysql-connector e JavaFX sono già compresi nella build è possibile lanciare l'applicazione utilizzando il comando:

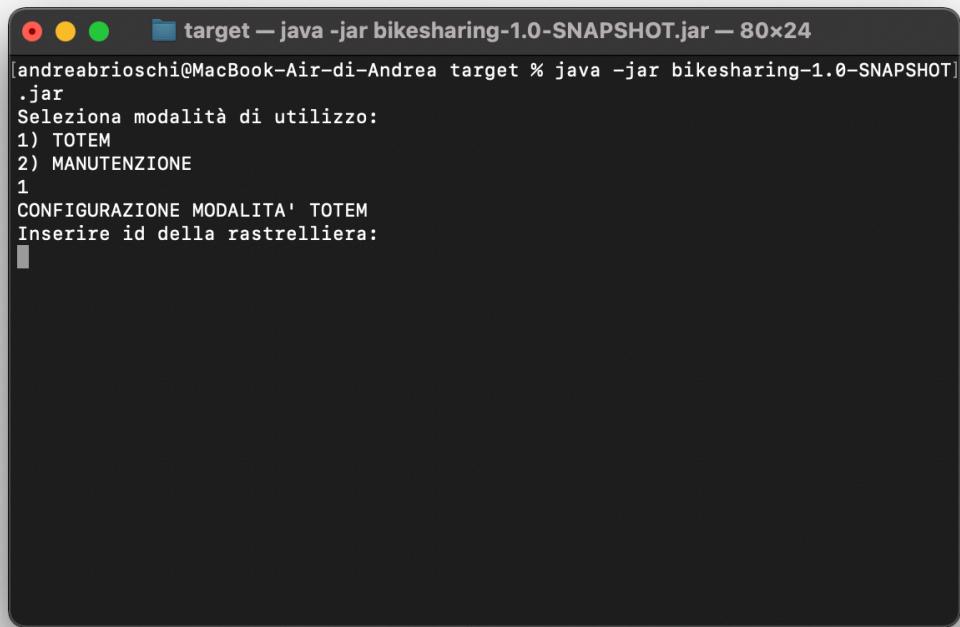
```
java -jar PATH/bikesharing.jar
```

Dopo aver eseguito il comando è necessario effettuare una breve procedura da terminale per configurare la modalità con la quale si desidera eseguire l'applicazione:



```
[andreabrioschi@MacBook-Air-di-Andrea target % java -jar bikesharing-1.0-SNAPSHOT]
.jar
Selezione modalità di utilizzo:
1) TOTEM
2) MANUTENZIONE
```

Nel caso in cui la modalità selezionata sia totem è previsto un ulteriore step di avvio per selezionare la rastrelliera alla quale il totem è legato:



```
[andreabrioschi@MacBook-Air-di-Andrea target % java -jar bikesharing-1.0-SNAPSHOT]
.jar
Selezione modalità di utilizzo:
1) TOTEM
2) MANUTENZIONE
1
CONFIGURAZIONE MODALITA' TOTEM
Inserire id della rastrelliera:
```