

# BIKE SHARING

Manuele Lucchi - 922866

**Nota:** Nel caso i diagrammi non fossero sufficientemente chiari all'interno del documento, si può trovare un'immagine ad alta risoluzione nella cartella images e il loro sorgente nel file apribile con StarUML "BikeSharing.mdj"

## 1 Descrizione del problema

### 1.1 Analisi e specifica dei requisiti

Il progetto consiste in un sistema di Bike Sharing.

Concretamente si tratta di un applicativo da installare nei **totem**. Altri attori fisici con cui si può interagire sono le **biciclette** e le **morse**. Le biciclette sono di tre tipi:

- Normale
- Elettrica
- Elettrica con sellino

E per ciascun tipo di bici c'è un tipo di morsa in cui va inserita la bici corrispondente. Gli attori principali sono gli utenti, che possono essere anche specificazioni come studenti e admin.

I totem (dotati di schermo touch) sono il centro dell'interazione con l'utente che può, tramite i totem stessi, iscriversi, noleggiare bici, segnalare problemi e vedere statistiche.

L'interfaccia grafica è quindi di dimensioni generose e pensata per un pannello touch screen. Sono inoltre connessi ai sistemi delle morse, per poterle aprire e chiudere su necessità

Per utilizzare il servizio è necessario sottoscrivere un **abbonamento** (tra un giorno, una settimana e un anno di durata) e opzionalmente si può specificare se si è studenti per non pagare nulla. Solo in caso sia abbonamento **annuale** viene attivato subito.

Il costo dell'abbonamento dipende dal tipo e se si è uno studente è completamente gratuito. L'utente deve fornire i dati di una **carta di credito** valida e una password oltre al tipo dell'abbonamento e in caso di successo, verrà restituito un codice personale.

Utilizzando il codice fornito e la medesima password, è possibile quindi fare il login per tutta la durata della sottoscrizione.

Senza aver bisogno di registrarsi o fare il login, dalla pagina principale è possibile andare in una pagina dedicata alla segnalazione di difetti alle bici (indicandone l'id che si presume essere scritto su ogni bici) oppure andare nella pagina che mostra tutte le statistiche relative al singolo totem o alla rete di tutti i totem.

Si **suppone** infatti che il database locale e condiviso tra le istanze sia in futuro trasformato in un server remoto ed è quindi stato astratto a sufficienza in modo che l'interazione tra UI e dati sia aggiornabile senza modifiche alla parte non direttamente influenzata. Dopo login o registrazione, si può scegliere la bici da noleggiare (solo nel caso non si abbia già fatto un noleggio o non si abbia restituito una bici da meno di cinque minuti) oppure, nel caso si sia **admin** (che si presume siano già registrati di default e in futuro via server o un'interfaccia dedicata) è anche accessibile una pagina che permette di aggiungere e togliere biciclette, morsa (si suppone ci sia una sola rastrelliera per totem e che sia componibile aggiungendo e togliendo morsa). Dopo un noleggio, all'utente viene mostrata la posizione della bici e può ancora scegliere se annullarlo oppure tornare alla homepage. In caso la tipologia di bici richiesta non sia disponibile in quel totem, l'utente finirà in una pagina

che indica il totem più vicino. Le varie pagine presentano dei menu espliciti di simulazione per poter ad esempio cambiare totem o simulare gli eventi di ritiro bici e riconsegna bici

## 1.2 Glossario

Admin: personale addetto alla gestione delle bici

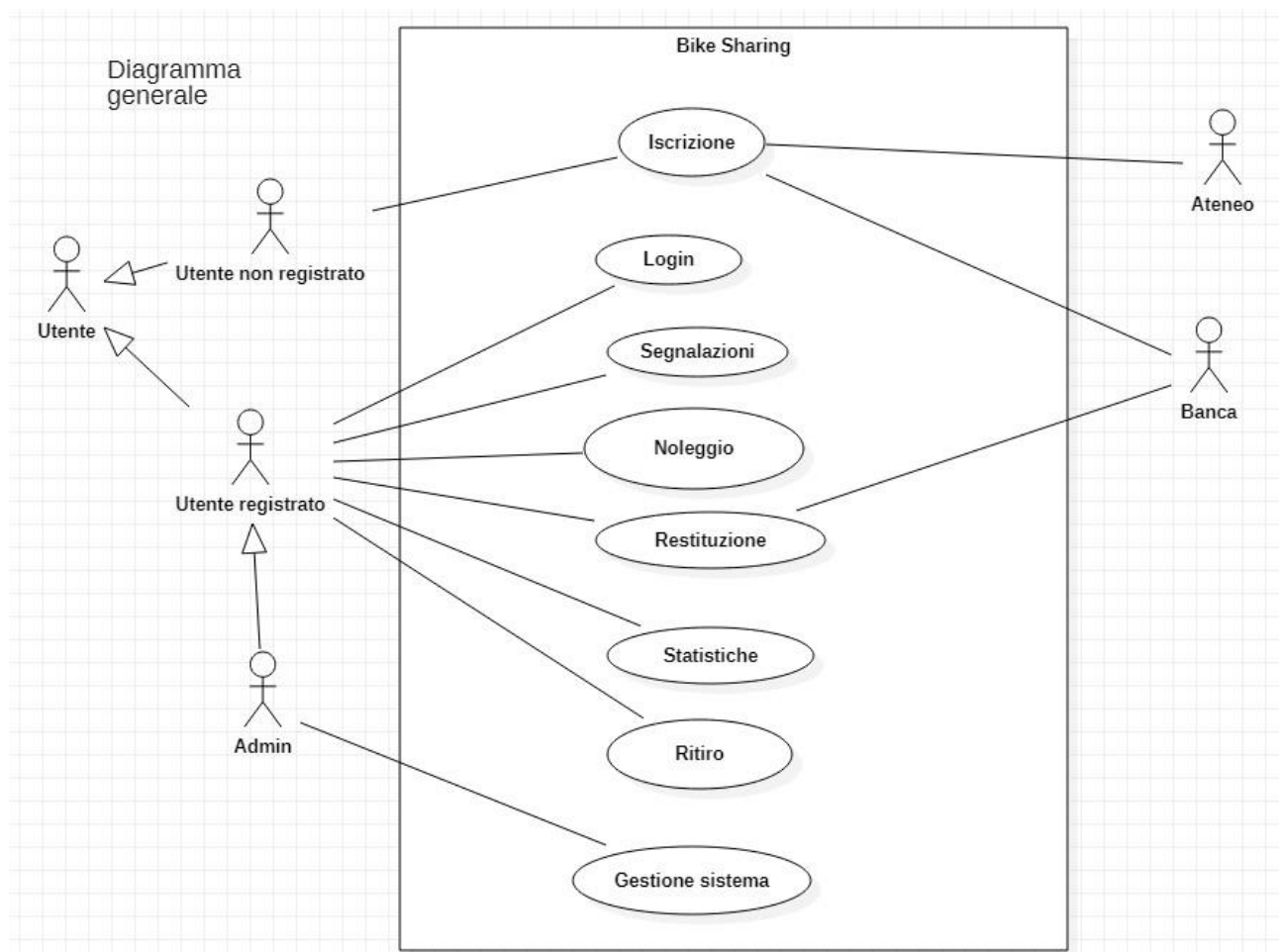
UI: User Interface (Interfaccia Grafica)

ORM: Object-Relational Mapping

## 2 Progettazione del Sistema

### 2.1 Diagramma dei casi d'uso

I sorgenti dei diagrammi si trovano in BikeSharing.mdi nella sezione Bike Sharing > Use Cases oppure si possono trovare immagini ad alta risoluzione nella cartella images e che iniziano con il prefisso "Use Case"



Di seguito una breve descrizione per ciascuno scenario, con alcune specificazioni

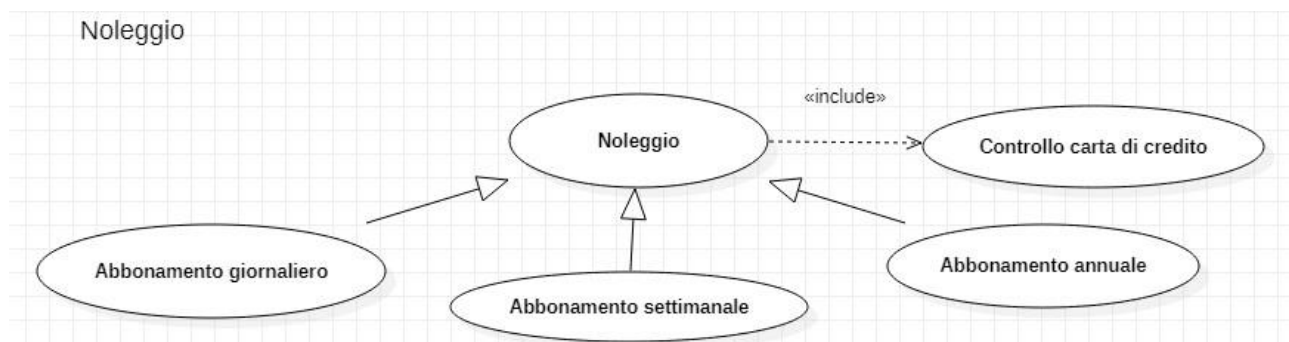
**Iscrizione:** L'utente deve iscriversi per utilizzare il sistema. L'iscrizione ha un costo che varia a seconda della durata e può dichiararsi studente per non dover pagare (verrà effettuato un controllo con il servizio dell'ateneo). L'utente deve inoltre inserire i dati di una carta valida (che

verrà controllata dall'apposito servizio della banca). In caso di registrazione riuscita (e quindi pagamento), l'utente riceverà un codice

**Login:** Il login può effettuarlo solo un utente registrato, che dovrà inserire il codice fornitogli e la password. In caso di successo, gli sarà permesso proseguire dal sistema

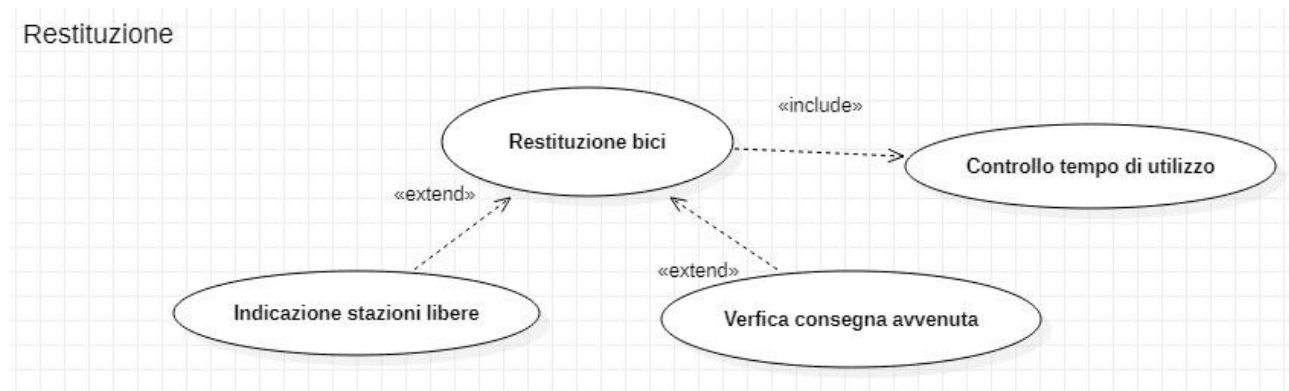
**Segnalazione:** Un utente qualsiasi può segnalare una bici danneggiata inserendo il codice di quest'ultima nell'apposita schermata

**Noleggio:** Un utente loggato può scegliere il tipo di bici da noleggiare e noleggiarla, a quel punto il sistema sbloccherà la morsa e potrà ritirarla.



**Restituzione:** Un utente per restituire la bici deve inserirla in una morsa, a quel punto il sistema registrerà la restituzione e avverrà il pagamento. Il costo varia in base

- Alla bici scelta
- Al tempo per cui è stata noleggiata
- Se il noleggio supera le 2 ore (Ammonizione)
- Se il noleggio supera le 24 ore (Penale)
- 



**Statistiche:** Un utente può vedere le statistiche del sistema entrando nell'apposita schermata dal menu principale

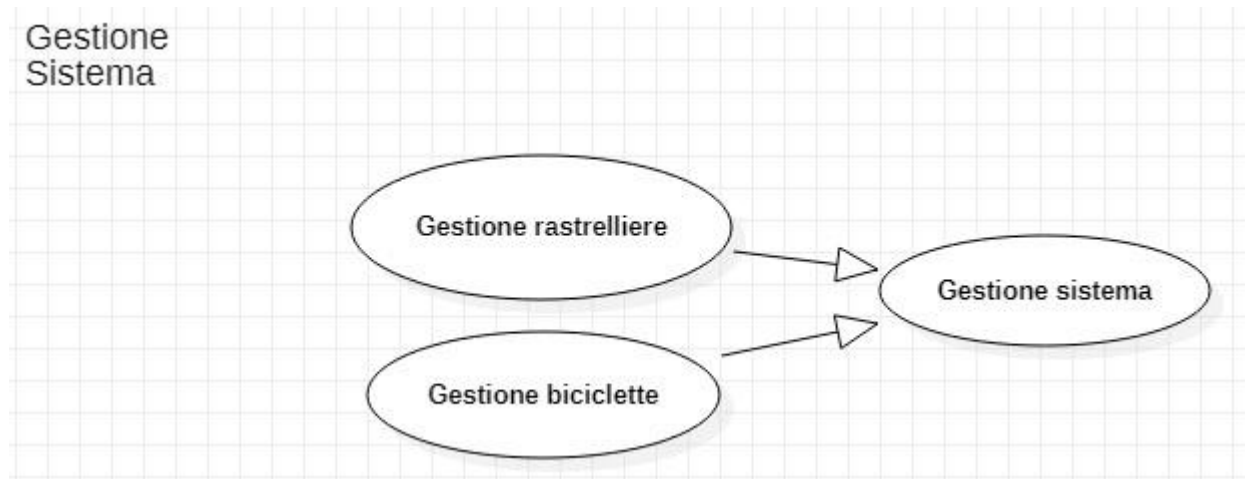
**Ritiro bici:** A noleggio riuscito della bici, il sistema sbloccherà la morsa e l'utente potrà rimuovere la bici

**Aggiunta Bici:** Un admin può aggiungere una bici al sistema che prima non era registrata

**Rimozione bici:** Un admin può rimuovere una bici dal sistema

**Aggiunta morsa:** Un admin può aggiungere una morsa che prima non esisteva alle rastrelliere

**Rimozione morsa:** Un admin può rimuovere una morsa da una rastrelliera



## 2.2 Descrizione degli scenari

Nome	Iscrizione
Scopo	Iscrivere l'utente al sistema creando un abbonamento
Attore/i	Utente
Pre-condizioni	Utente non iscritto, carta valida
Trigger	Utente che fa gli step per registrarsi
Descrizione sequenza eventi	<ol style="list-style-type: none"><li>1. L'utente entra nella pagina di registrazione</li><li>2. L'utente sceglie il tipo di abbonamento</li><li>3. L'utente inserisce i dati della carta</li><li>4. L'utente inserisce la password</li><li>5. L'utente conferma la registrazione</li><li>6. L'utente riceve il codice</li></ol>
Alternativa/e	<ol style="list-style-type: none"><li>2a. L'utente si registra come studente</li><li>3a. L'utente inserisce dati non validi</li></ol>
Post-condizioni	Abbonamento valido per tutta la durata programmata

Nome	Login
Scopo	Entrare nel totem con il proprio abbonamento
Attore/i	Utente, Admin
Pre-condizioni	Attore iscritto, abbonamento non scaduto
Trigger	Utente che fa gli step per fare il login
Descrizione sequenza eventi	<ol style="list-style-type: none"><li>1. L'utente entra nella pagina di login</li><li>2. L'utente inserisce la password</li><li>3. L'utente inserisce il codice</li><li>3. L'utente preme il pulsante di login</li></ol>
Alternativa/e	<ol style="list-style-type: none"><li>2a. Password errata</li></ol>
Post-condizioni	Utente ha accesso al noleggio

Nome	Noleggio
------	----------

<b>Scopo</b>	Noleggiare una bici del tipo specificato
<b>Attore/i</b>	Utente, Admin
<b>Pre-condizioni</b>	Attore loggato, carta valida
<b>Trigger</b>	L'utente noleggia una bici
<b>Descrizione sequenza eventi</b>	1. L'utente sceglie il tipo di bici 2. L'utente conferma 3. L'utente ritira la bici
<b>Alternativa/e</b>	2a. Bici non disponibile
<b>Post-condizioni</b>	L'utente ha una bici in custodia

<b>Nome</b>	<b>Gestione Sistema</b>
<b>Scopo</b>	Gestire bici e morsa di uno specifico totem
<b>Attore/i</b>	Admin
<b>Pre-condizioni</b>	Admin loggato
<b>Trigger</b>	L'admin preme il pulsante Admin Panel
<b>Descrizione sequenza eventi</b>	1. L'admin sceglie la bici da aggiungere 2. L'admin conferma la sua scelta
<b>Alternativa/e</b>	1a. L'admin sceglie la bici da rimuovere 1b. L'admin sceglie la morsa da aggiungere 1c. L'admin sceglie la morsa da rimuovere
<b>Post-condizioni</b>	-

<b>Nome</b>	<b>Statistiche</b>
<b>Scopo</b>	Visualizzare le statistiche del sistema
<b>Attore/i</b>	Utente
<b>Pre-condizioni</b>	Essere nella pagina home
<b>Trigger</b>	Utente clicca sul pulsante statistiche
<b>Descrizione sequenza eventi</b>	1. L'utente apre la schermata delle statistiche 2. L'utente guarda le statistiche
<b>Alternativa/e</b>	Login, Iscrizione, Segnalazione
<b>Post-condizioni</b>	-

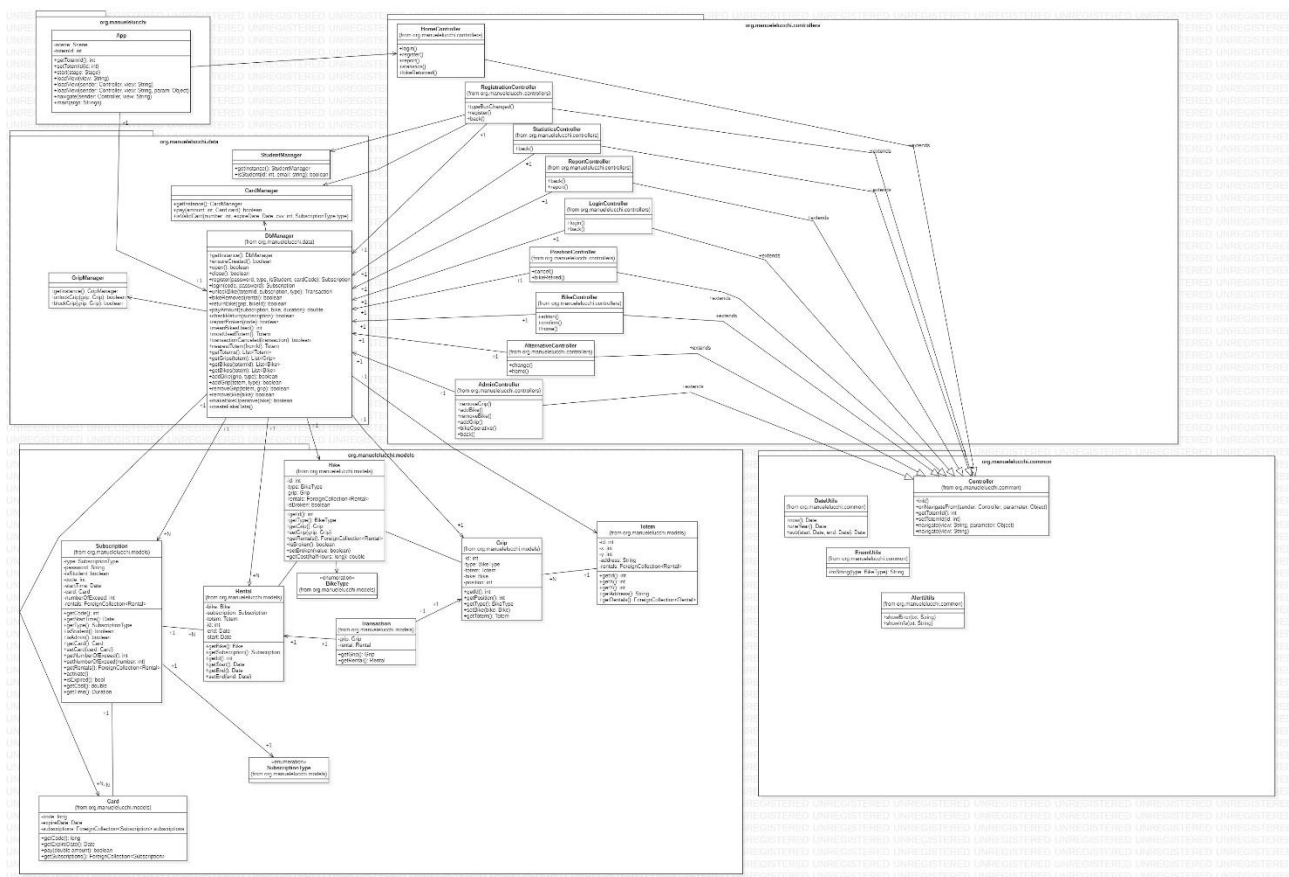
<b>Nome</b>	<b>Restituzione</b>
<b>Scopo</b>	Restituire una bici noleggiata
<b>Attore/i</b>	Utente, Admin
<b>Pre-condizioni</b>	Bici noleggiata
<b>Trigger</b>	L'utente inserisce una bici noleggiata nella morsa del tipo corretto
<b>Descrizione sequenza eventi</b>	1. L'utente inserisce la bici nella morsa 2. Viene effettuato il pagamento
<b>Alternativa/e</b>	2a. Se l'utente è in ritardo di più di 24 ore viene segnalato 2b. Se l'utente ha 3 segnalazioni viene annullato l'abbonamento
<b>Post-condizioni</b>	L'utente non ha più bici noleggiate

<b>Nome</b>	<b>Segnalazione</b>
<b>Scopo</b>	Segnalare una bici danneggiata
<b>Attore/i</b>	Utente, Admin
<b>Pre-condizioni</b>	L'utente si trova in home page

<b>Trigger</b>	L'utente segnala una bici specifica
<b>Descrizione sequenza eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente apre la pagina di segnalazione</li> <li>2. L'utente inserisce il codice di una bici</li> <li>3. L'utente preme conferma</li> </ol>
<b>Alternativa/e</b>	2a. L'utente inserisce un codice non valido
<b>Post-condizioni</b>	L'utente non ha più bici noleggiate

## 2.3 Diagramma delle classi (modello di programma)

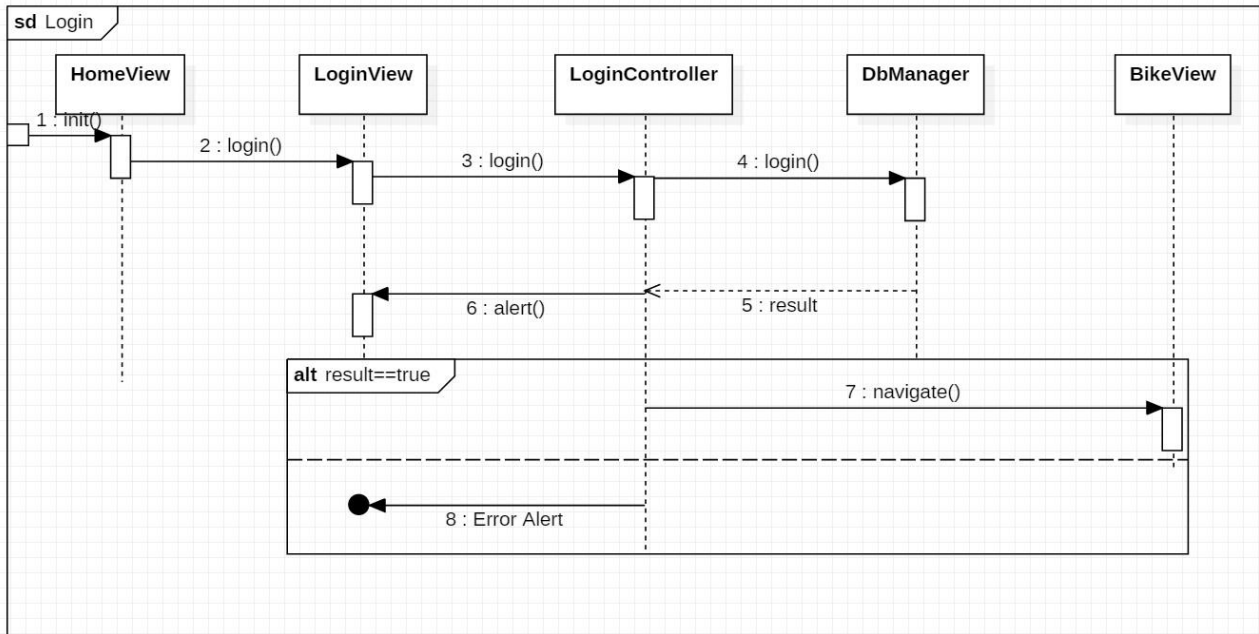
È stato implementato un unico diagramma delle classi (di programma). Il diagramma esclude i campi del DbManager relativi alle classi fornite da ORMLite e i riferimenti a controlli di JavaFX nei controller, riportando solo gli eventi relativi a certe azioni della UI. Per quanto riguarda il dominio del progetto e il data layer, è stato fatto uno scarso uso dell'ereditarietà in quanto poco compatibile con la relazionalità del database. Viene invece utilizzata nei controllers per la navigazione. Le classi che terminano con "Utils" sono classi statiche utilizzate in vari punti del progetto, di conseguenza le relazioni con le altre classi sono state omesse per questioni di chiarezza



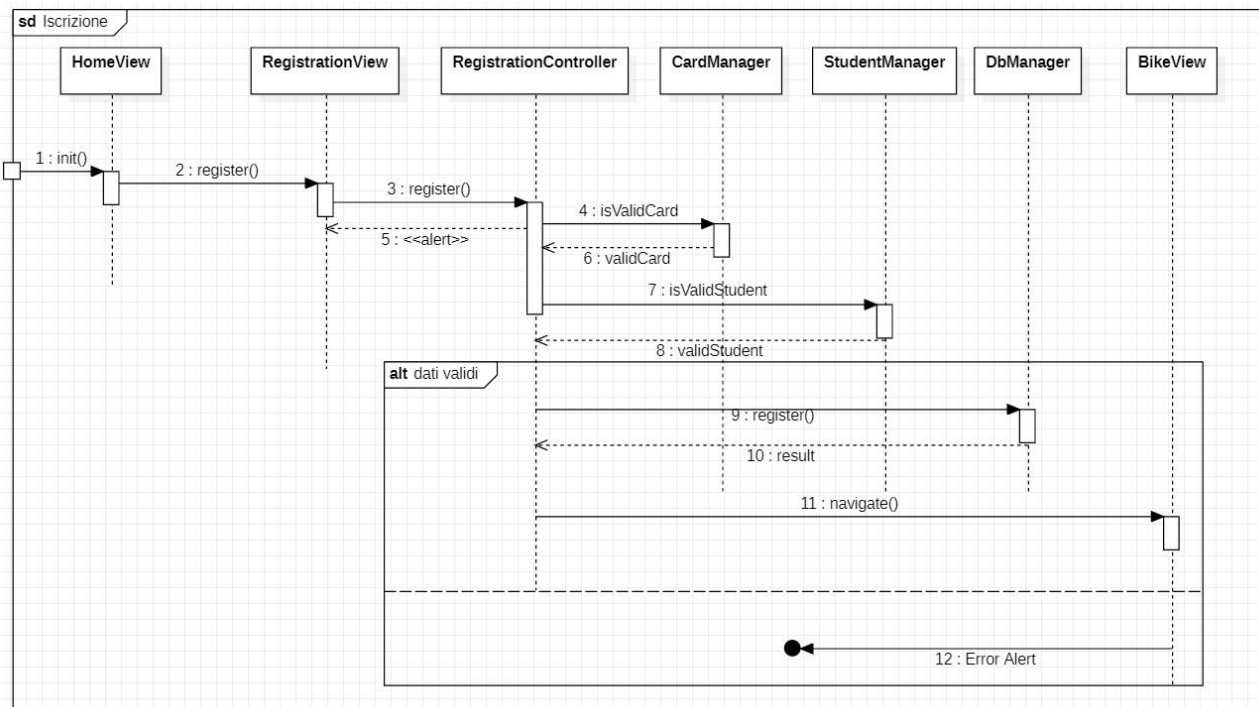
Il sorgente del diagramma di classe si trova in Bike Sharing > Class Diagram > Class oppure nel file nella cartella images "Class.jpg" per un'immagine ad alta risoluzione

## 2.4 Diagrammi di sequenza

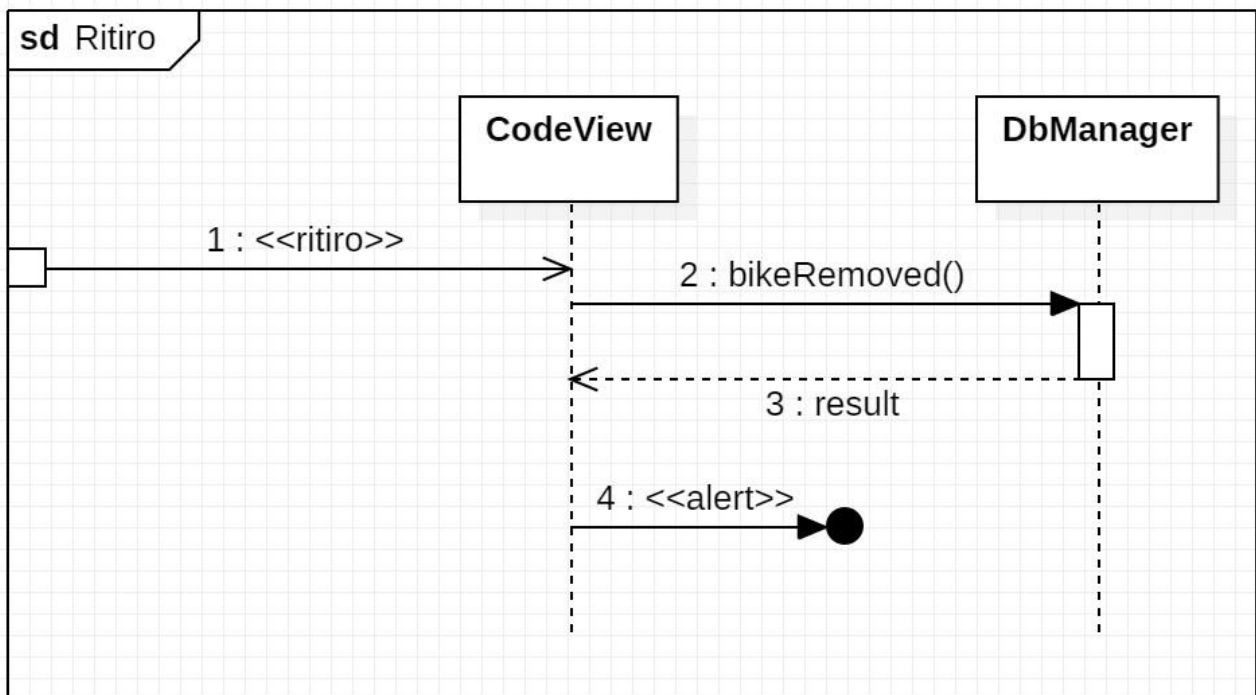
### 2.4.1 Login



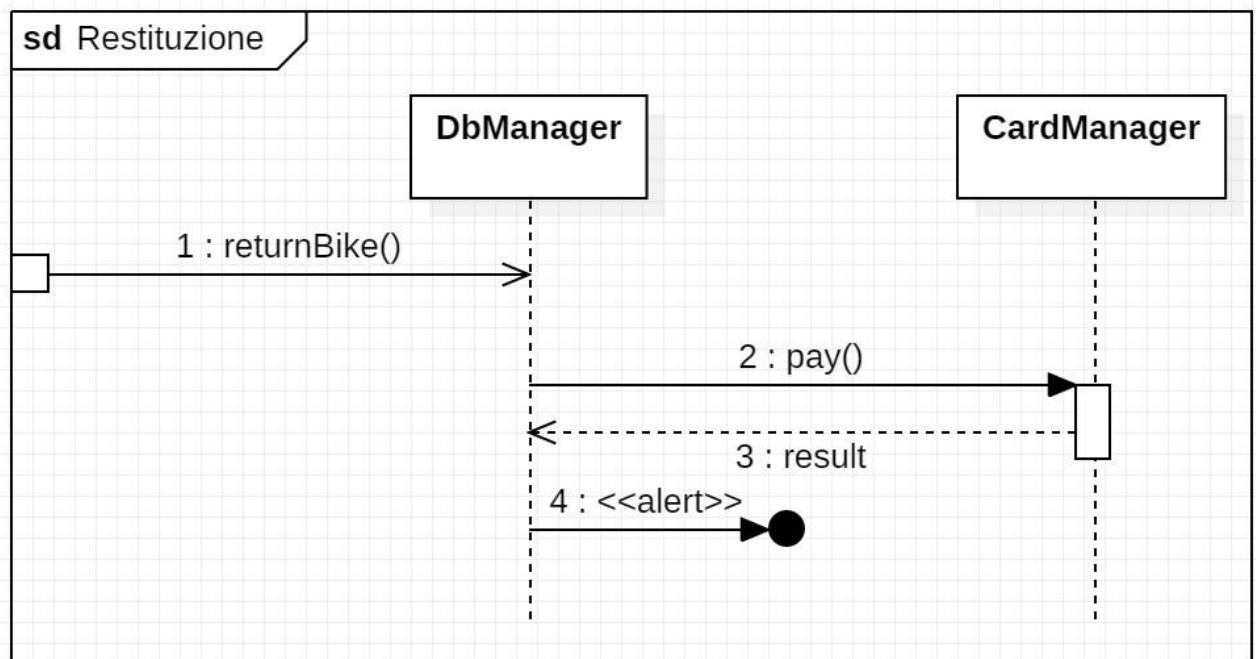
### 2.4.2 Iscrizione



### 2.4.3 Ritiro

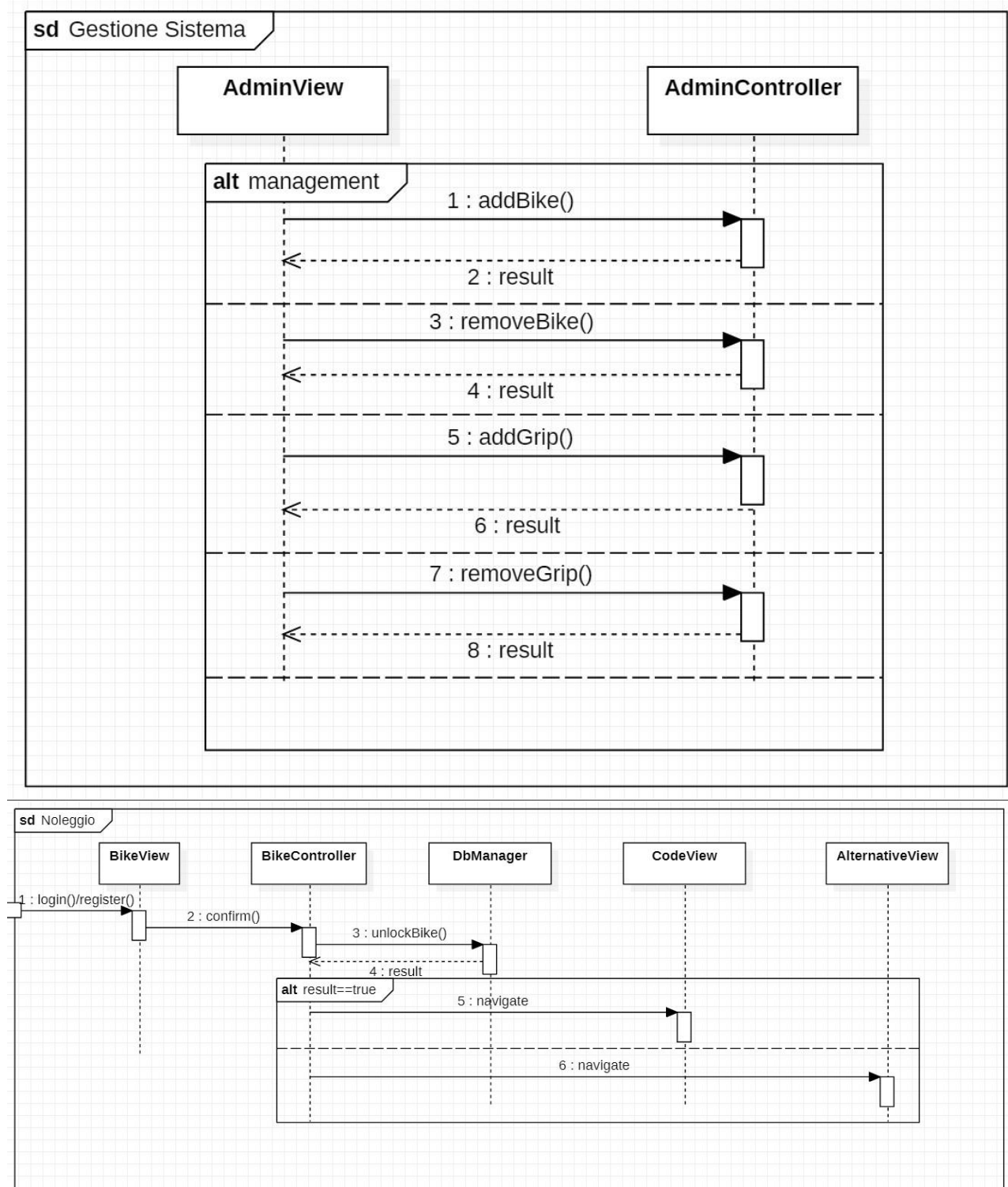


### 2.4.4 Restituzione

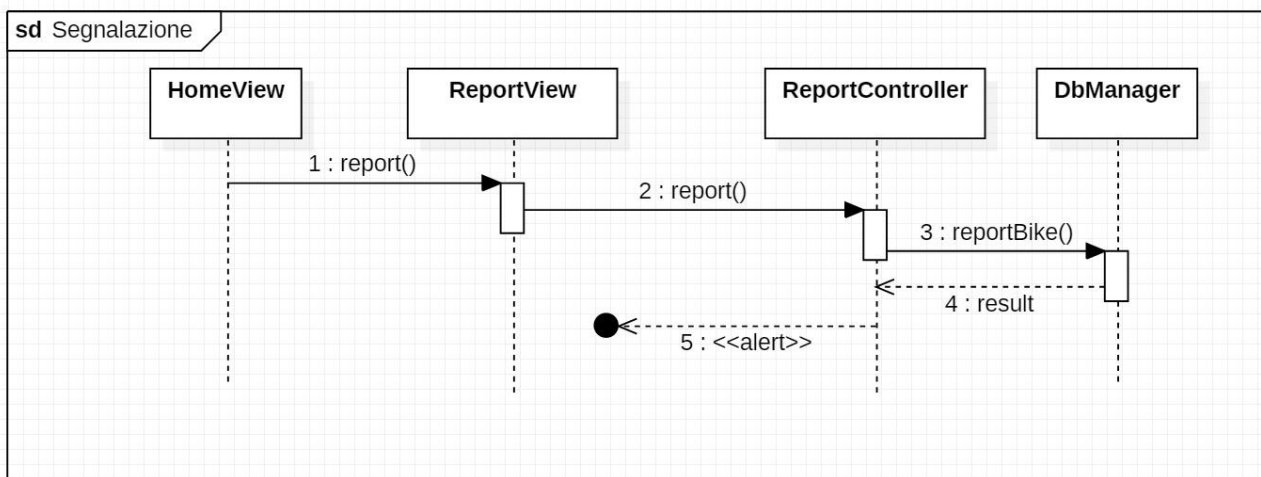




## 2.4.5 Gestione Sistema



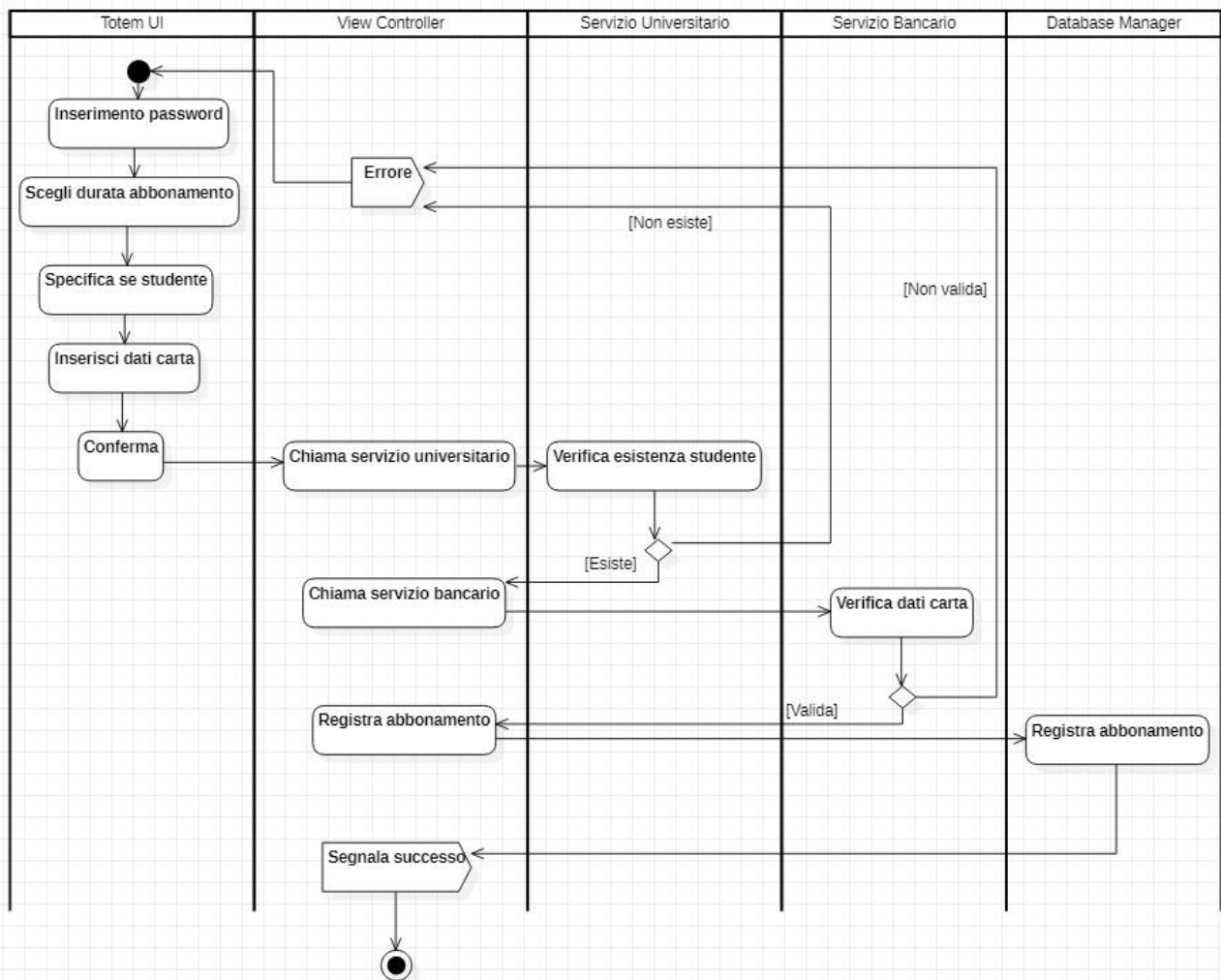
## 2.4.6 Segnalazione



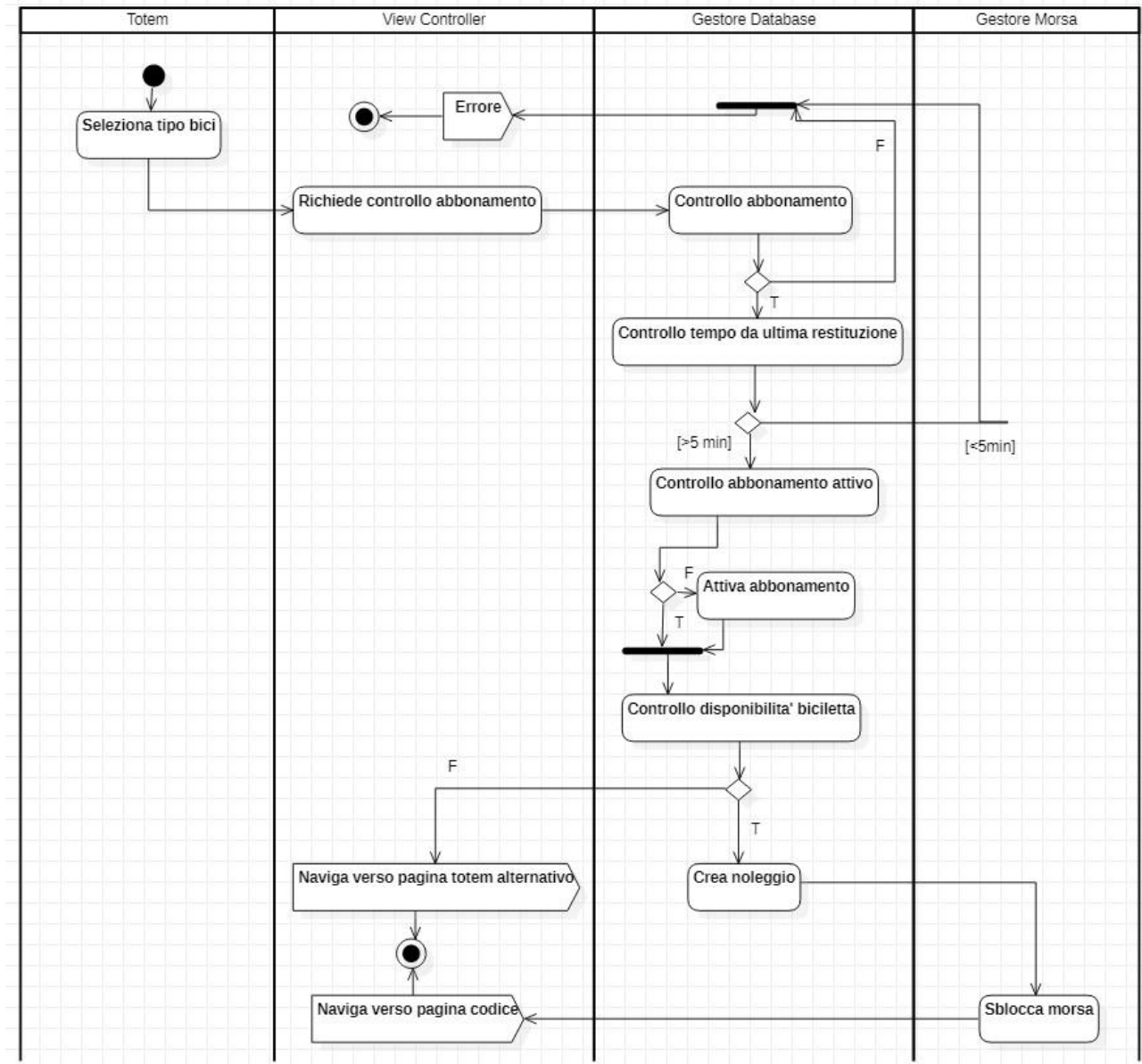
I sorgenti dei diagrammi di sequenza si trovano nel file BikeSharing.mdj sotto la sezione Bike Sharing > Sequence Diagrams oppure nella cartella images che iniziano con il prefisso “Sequence” si possono trovare immagini ad alta risoluzione

## 2.5 Diagrammi delle attività

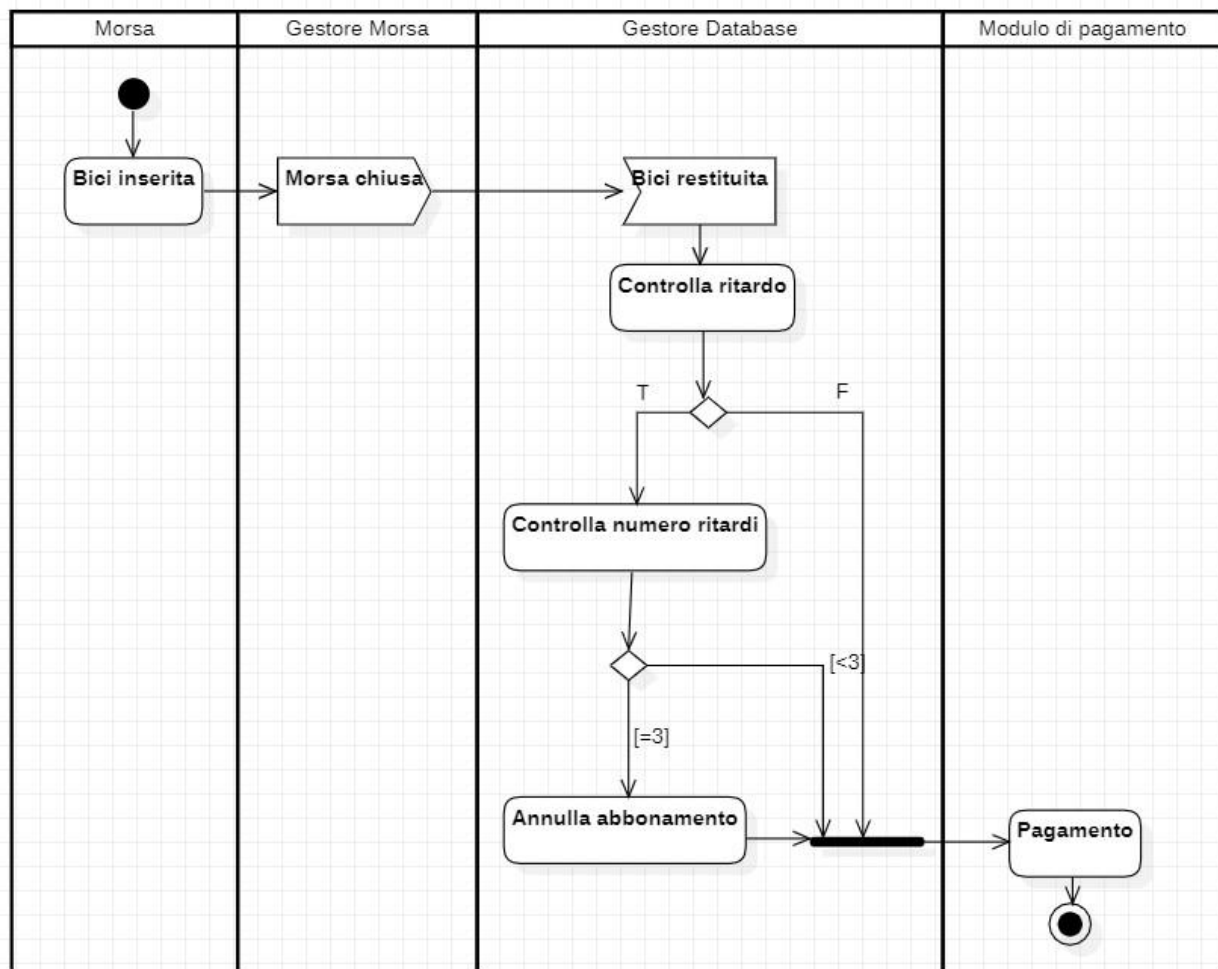
### 2.5.1 Abbonamento



### 2.5.2 Totem



### 2.5.3 Morsa

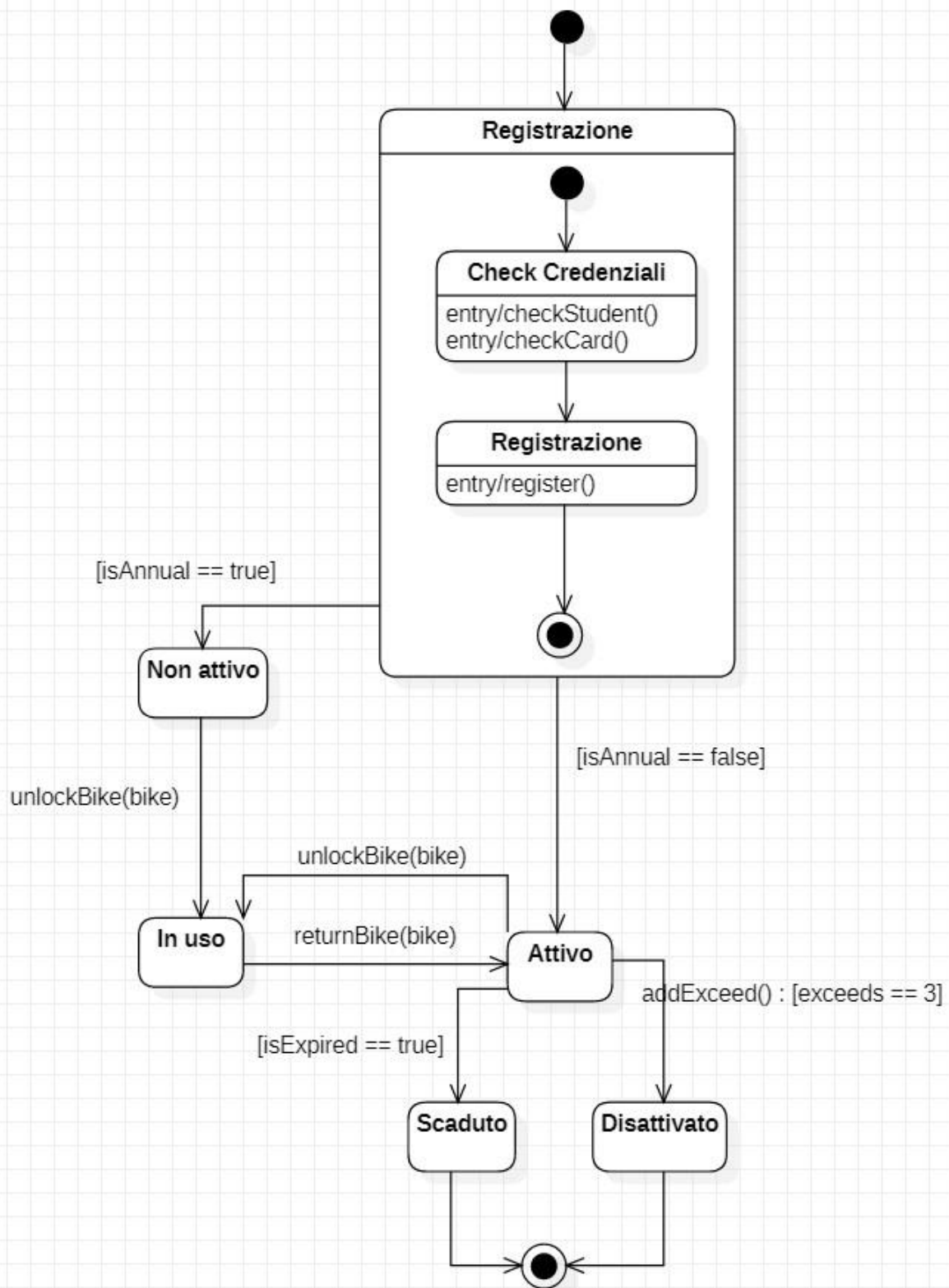


I sorgenti dei diagrammi delle attività si trovano nel file BikeSharing.mdi nella sezione Bike Sharing > Activity Diagram oppure nella cartella images si possono trovare immagini ad alta risoluzione che iniziano con il prefisso “Activity”

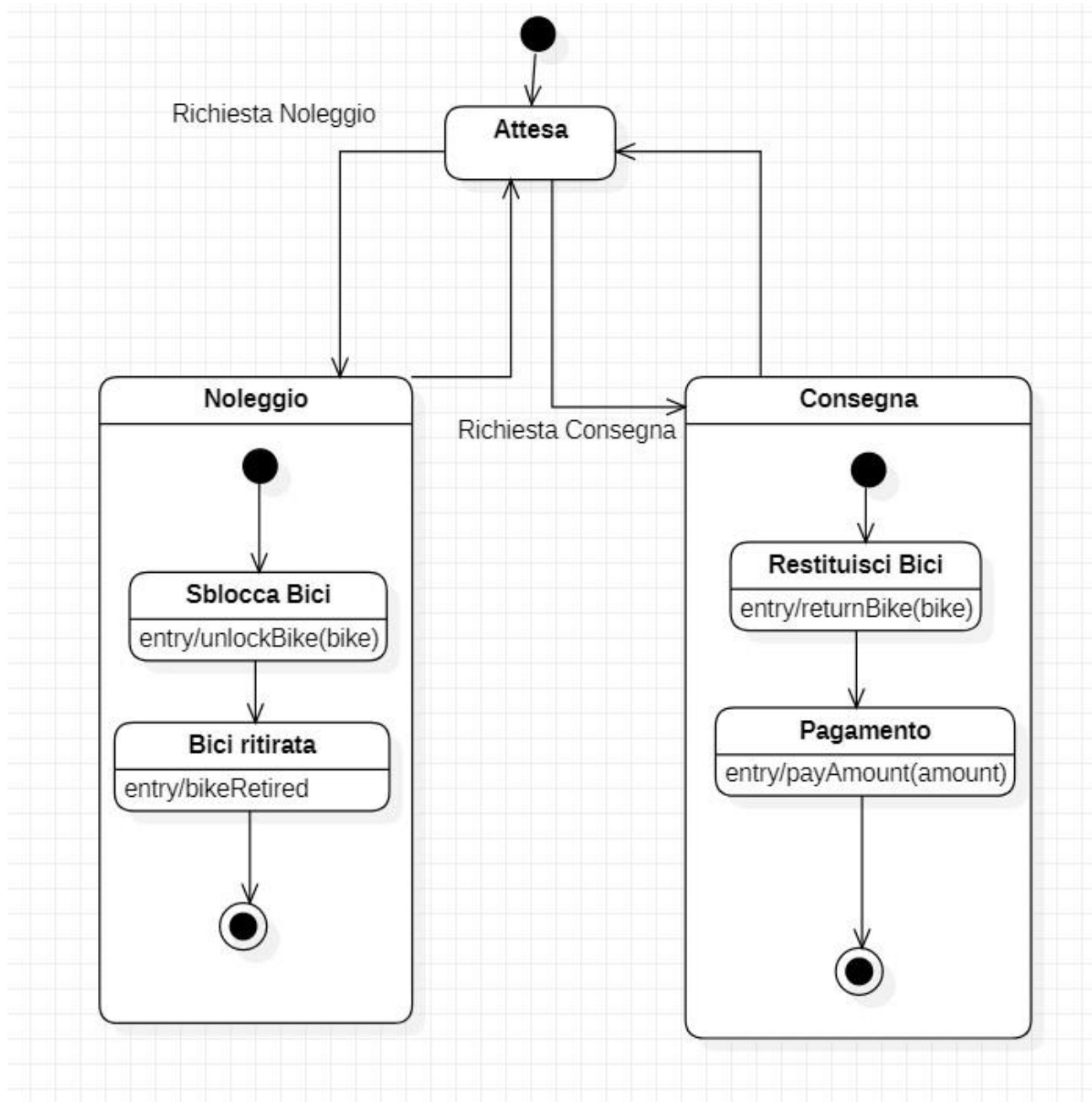
### 2.6 Macchine di stato

Ci sono quattro macchine di stato, una per ogni macroarea in cui agisce il programma, ossia la fase di creazione dell’abbonamento, la fase di noleggio e restituzione bici (quindi l’interazione con il totem) e la gestione di morse e bici

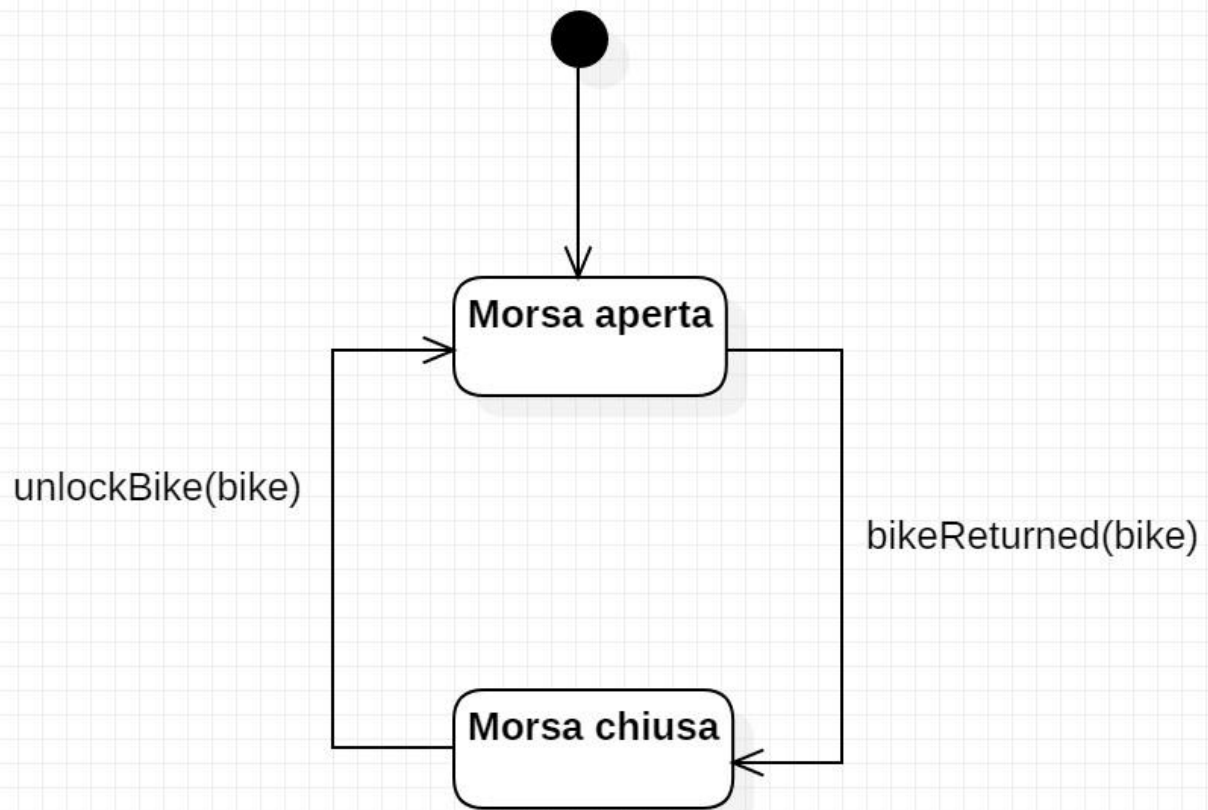
### 2.6.1 Registrazione



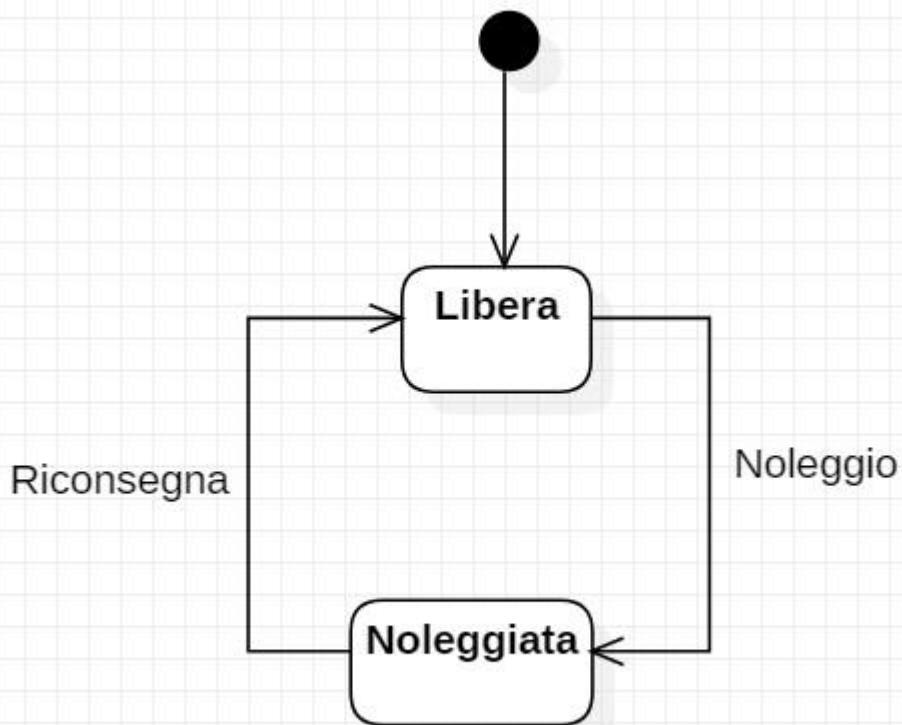
### 2.6.2 Totem



### 2.6.3 Morsa



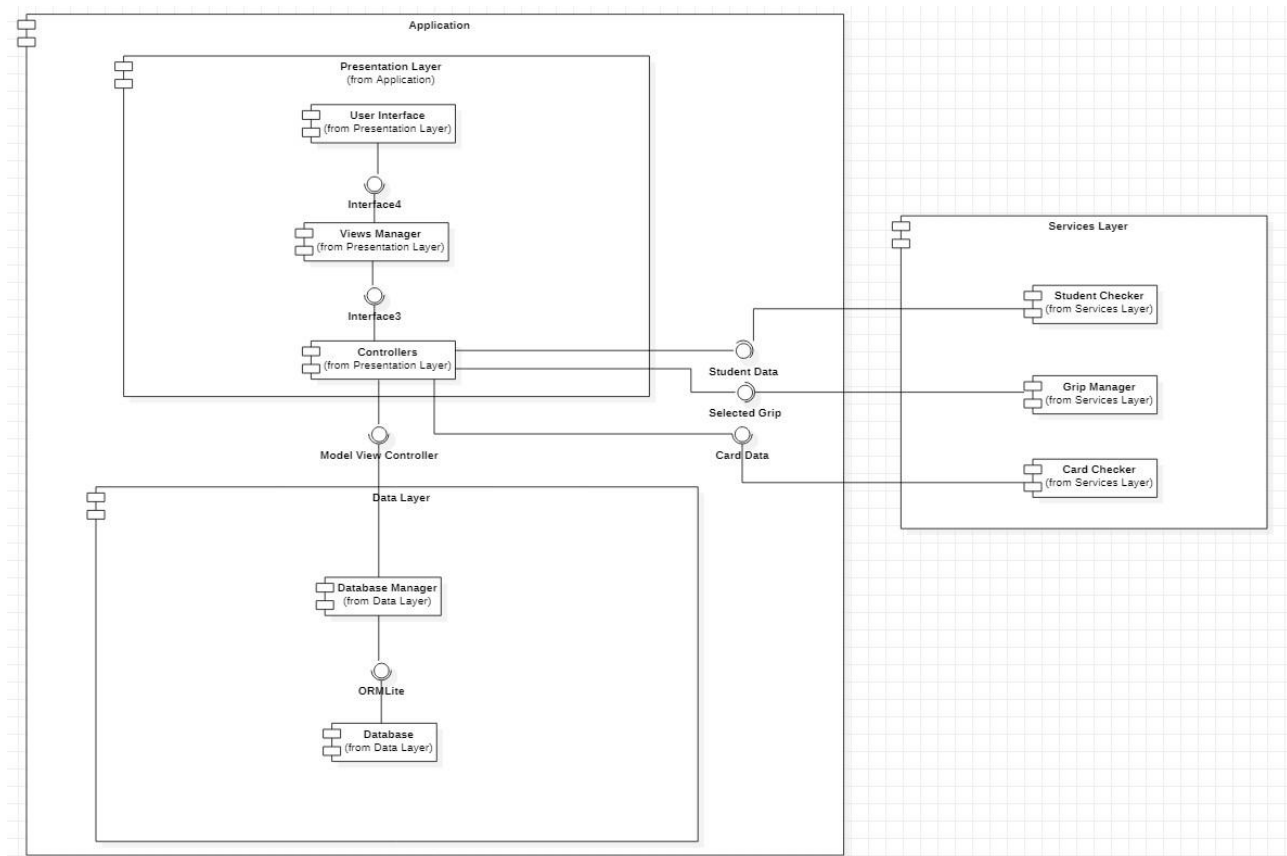
### 2.6.4 Bici



I sorgenti dei digrammi delle macchine di stato si trovano nella sezione Bike Sharing > State Machines del file BikeSharing.mdi oppure nei file .jpg della cartella images si possono trovare immagini ad alta risoluzione che iniziano con il prefisso “FSM”

## 2.7 Diagramma dei componenti

Il diagramma dei componenti divide il sistema in due macro-componenti: servizi e applicazione. Nel caso dei servizi abbiamo alcuni servizi che ipoteticamente saranno web (un check degli studenti e uno delle carte di credito) e uno hardware, ossia il gestore delle morse che devono aprirsi e chiudersi a comando. L'applicazione è divisa in due aree, il Presentation Layer (contenente la UI vera e propria, il gestore delle view e della navigazione e i controllers) e un data layer, con il DbManager che fa da astrazione all'effettivo database. Sono connessi attraverso l'interfaccia fornita da ORMLite. Si suppone che in futuro il Database venga sostituito da un servizio web centralizzato e quindi i metodi del DbManager vadano reimplementati, senza però necessita' di modificare l'interfaccia in sé.



Il sorgente del diagramma si può trovare nel file “BikeSharing.mdi” oppure una sua immagine ad alta risoluzione nel file “Component.jpg” della cartella images

## 3 Implementazione del sistema

### 3.1.1 Discussione dei Design Pattern utilizzati

**MVC:** Il design pattern più incisivo utilizzato è sicuramente il model view controller. Infatti, l'intera applicazione è stata divisa principalmente in tre packages, quello che contiene i modelli, quello che contiene i controllers, ciascuno dei quali eredita la classe Controller e le view, che sono dei file FXML interpretati da JavaFX. Tutto il progetto è stato costruito con questo pattern in mente, in



modo da dividere la dichiarazione della UI dalla manipolazione di essa e dal dominio, che può essere riutilizzato non avendo dipendenze.

**DAO:** Il secondo pattern più influente è il DAO, Data Access Object, che astrae il database. In questo caso non è stato necessario implementarlo direttamente, in quanto l'ORM utilizzato (ORMLite) propone già un oggetto `Dao<T>` per l'utilizzo.

**Singleton:** Un altro pattern utilizzato è il singleton, utilizzato ad esempio per il `DbManager`, classe di astrazione delle query del database che non ha necessità di multiple istanze che anzi potrebbero rivelarsi dannose nel caso il database utilizzato non supportasse multiple connessioni concorrenti. È stato quindi reso privato il costruttore e la singola istanza disponibile è accessibile tramite un metodo statico `getInstance()`. Altre classi che lo utilizzano sono `StudentManager`, `CardManager` e `GripManager`

**Adapter:** Molti controllers fanno utilizzo della `ChoiceBox<T>`, un riferimento della UI parametrico che prende in input una lista di `T` e mostra nella UI una stringa per ciascun elemento della lista. In questi casi è stato usato uno `StringConverter<T>` per convertire dal tipo alla stringa ed è a tutti gli effetti un pattern adapter

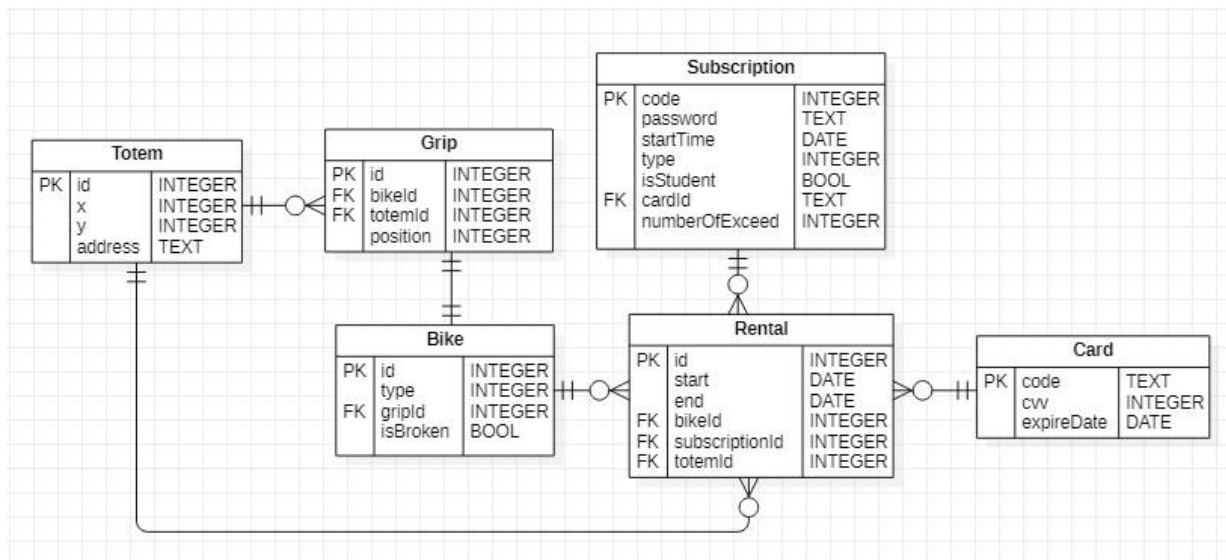
**Observer:** JavaFX utilizza un pattern observable per la gestione degli eventi, che vengono registrati e sollevati in caso di interazione con la UI

### 3.2 Gestione dei dati persistenti

Il database è composto da sei tabelle (e rispettive classi):

- **Bike:** Rappresenta una bici discriminata da un campo tipo e da un id. È in relazione 1 a 1 con `Grip` e N a M con `Subscription`
- **Subscription:** Rappresenta una sottoscrizione, discriminata da un campo tipo che ne determina la durata e un Id. È in relazione N a 1 con `Card` e N a M con `Bike`
- **Rental:** Una tabella intermedia per formalizzare la relazione N a M tra `Bike` e `Subscription`, rappresenta un singolo noleggio
- **Totem:** Un totem fisico, in relazione 1 a N con `Grip`
- **Grip:** Una morsa contenente al più una bicicletta (1 a N con `Bike`) e appartenente ad un Totem (N a 1)
- **Card:** Una tabella per le carte usate nei pagamenti. Ogni `Subscription` può avere una sola Card e una Card può avere più `Subscription`

La gestione è stata affidata all'ORM chiamato ORMLite, che crea le query SQL in automatico attraverso le Stream API di Java. Si occupa inoltre di caricare in automatico le relazioni tra le tabelle (e quindi gli oggetti). Il DBMS utilizzato è SQLite, ma grazie all'ORM può essere facilmente sostituito con altri DBMS



Il sorgente dello schema ER del database può essere trovato in “BikeSharing.mdi” nella sezione Bike Sharing > Database > Database o un’immagine a più alta risoluzione nel file “ER.jpg” nella cartella images

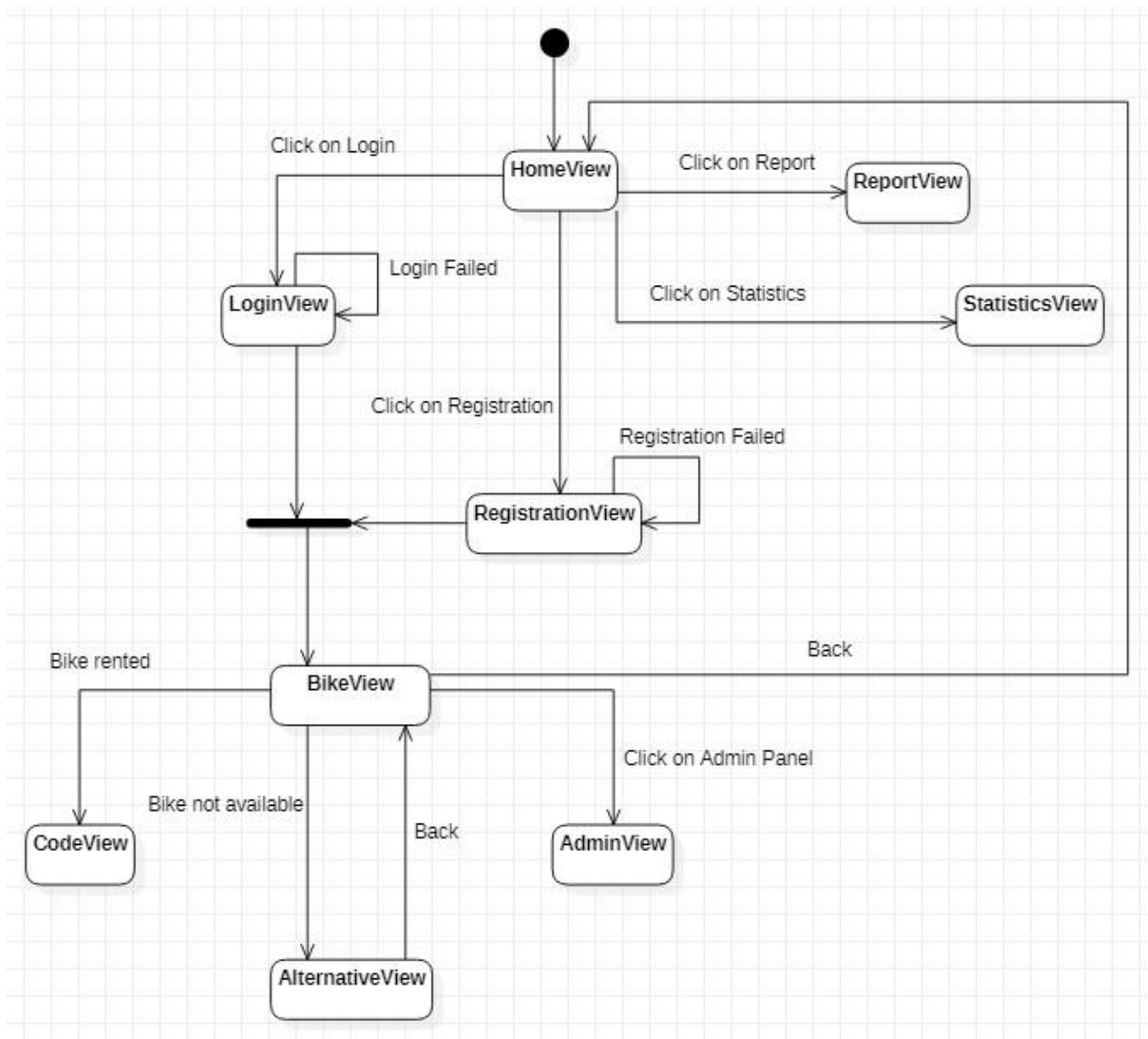
### 3.3 Descrizione dell’Interfaccia Grafica

La UI è composta da svariate pagine, all’interno di una singola finestra. La navigazione è stata implementata manualmente, attraverso un metodo navigateTo nella base class Controller. Quando una pagina viene caricata, viene anche richiamato l’evento onNavigateFrom per poter ottenere parametri.

In qualsiasi pagina è sempre possibile tornare alla pagina precedente o alla home. La UI è stata costruita con lo schermo touch del Totem in mente, di conseguenza gli oggetti sono di grandi dimensioni per facilitare l’utilizzo.

La validità degli input è sempre controllata e in caso di errori interni o dovuti a dati errati, verrà mostrato un alert di errore

La navigazione è descritta dalla Navigation Map seguente

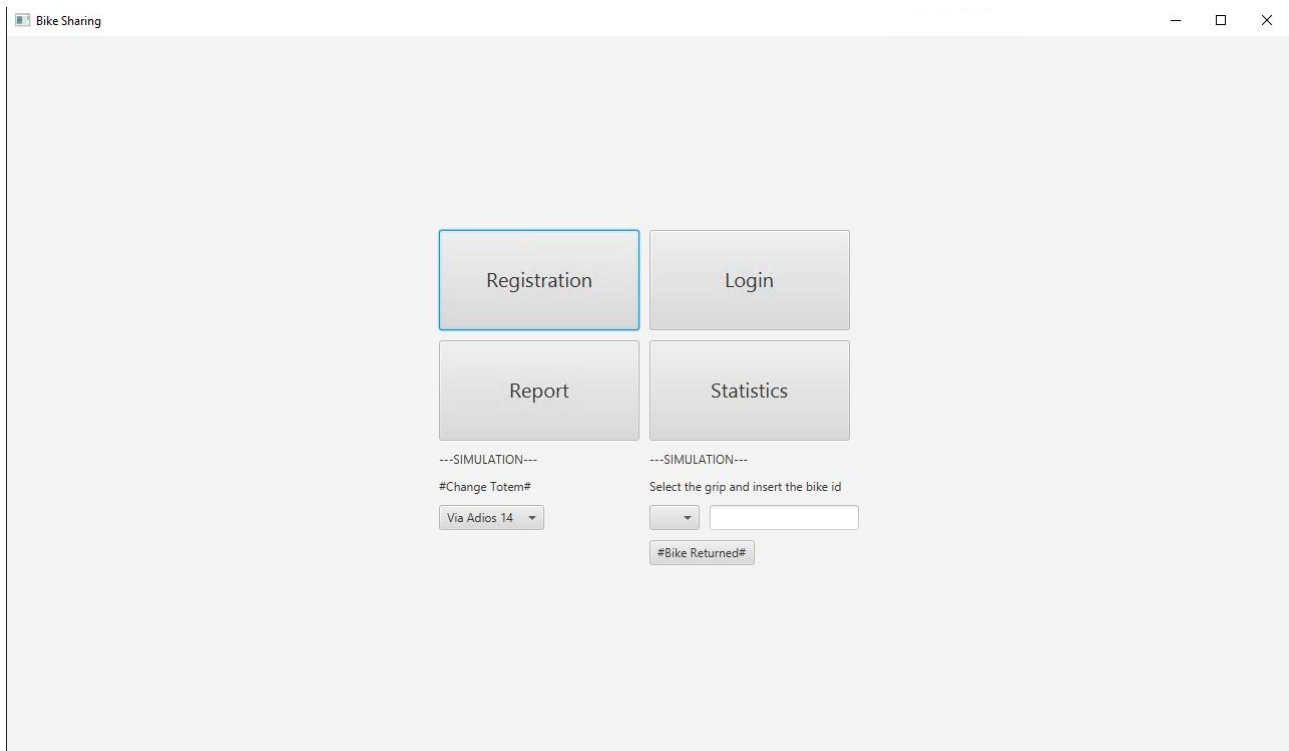


Il sorgente della Navigation Map può essere trovato nel file “BikeSharing.mdi” nella sezione Navigation Map. Un’immagine ad alta risoluzione si trova nel file “Navigation Map.jpg” nella cartella images

Gli screenshot ad alta risoluzione possono essere trovati nella cartella images con l’iniziale “UI ”

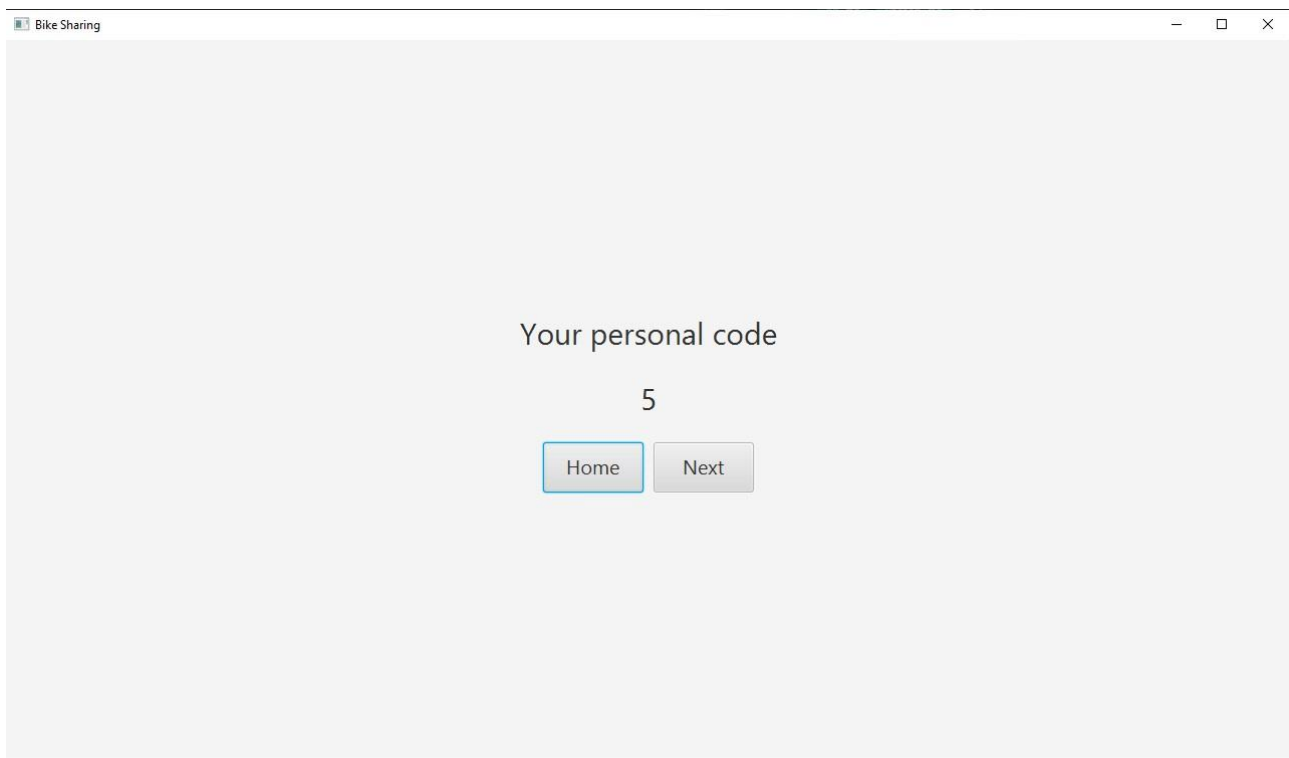
### 3.3.1 Home

La Home page presenta 4 pulsanti + 2 sezioni di simulazione (che quindi nella versione da rilasciare in produzione connessa all’hardware non saranno presenti), i primi che permettono di andare a fare login, registrazione, vedere le statistiche e segnalare un problema e le seconde che permettono di cambiare il totem in cui si trova (nella simulazione) e simulare la restituzione di una bici.



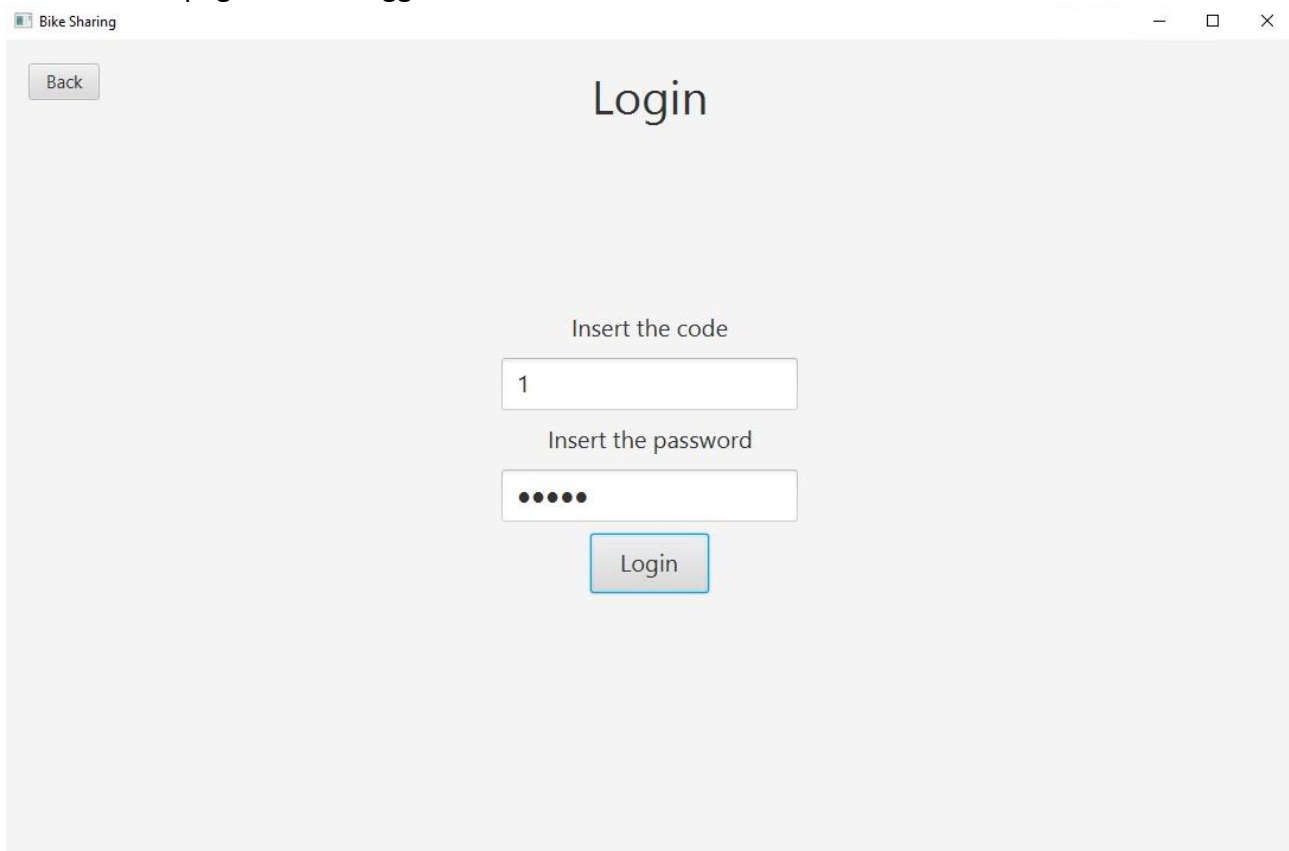
### 3.3.2 Registrazione

La pagina di registrazione richiede l'inserimento di una password, dei dati della carta ed eventualmente quelli da studente. In caso di avvenuta registrazione navigherà in una pagina che mostra il codice utente. Il controllo di dati di carta e utente vengono simulati da apposite classi di servizi e prima ancora che siano validi (ad esempio il CVV di 3 cifre e che il codice della carta sia un numero di 16 cifre)



### 3.3.3 Login

Inserendo codice utente e password e premendo il pulsante di login, in caso di successo si potrà accedere alla pagina del noleggio



### 3.3.4 Segnalazioni

In questa pagina l'utente potrà inserire il codice di una bicicletta e segnalarla come difettosa, rendendola impossibile da noleggiare

Bike Sharing

Back

# Report

Insert the bike code

2

Report

### 3.3.5 Statistiche

In questa pagina verrà visualizzato un insieme di statistiche relative al sistema

Bike Sharing

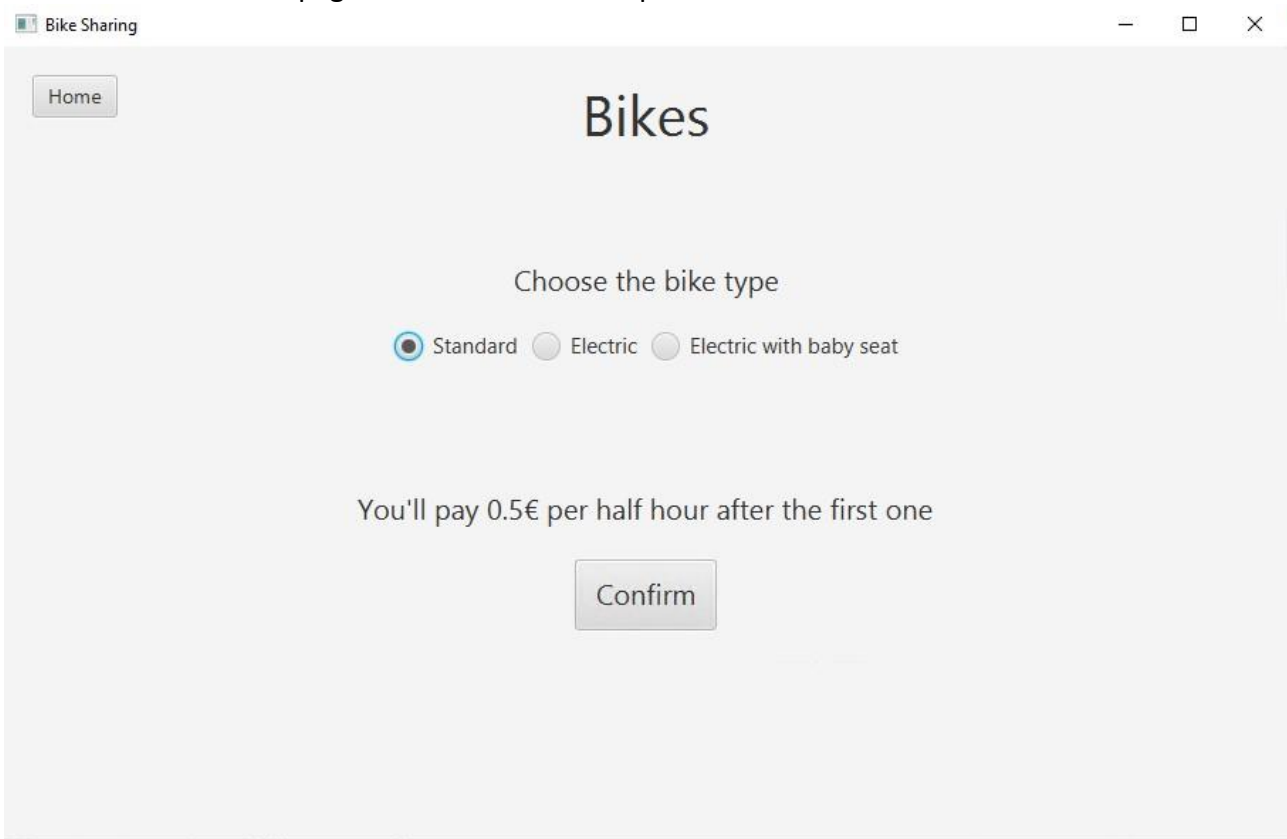
Back

# Statistics

Mean bikes used per day: 0  
Most used Totem: Via Adios 14

### 3.3.6 Noleggio

Nella pagina di noleggio, nel caso l'utente sia admin, avrà a disposizione un bottone in più per navigare alla pagina di gestione delle bici. Una volta scelta la bici (non è necessario controllo degli input in quanto tutti gli input sono tra scelte finite e non nulle) e confermato il noleggio, l'utente viene mandato in una pagina nella quale viene indicata la posizione della bici da ritirare e in cui è sempre possibile annullare l'ordine. Nel caso la bici del tipo richiesto non sia disponibile, si viene invece mandati in una pagina che indica il totem più vicino.



The screenshot shows a web browser window titled "Bike Sharing". The page has a light gray background. In the top left corner, there is a "Home" button. The main heading "Bikes" is centered at the top. Below it, the text "Choose the bike type" is centered. There are three radio button options: "Standard" (selected), "Electric", and "Electric with baby seat". Below these options, the text "You'll pay 0.5€ per half hour after the first one" is centered. At the bottom, there is a "Confirm" button.

Home

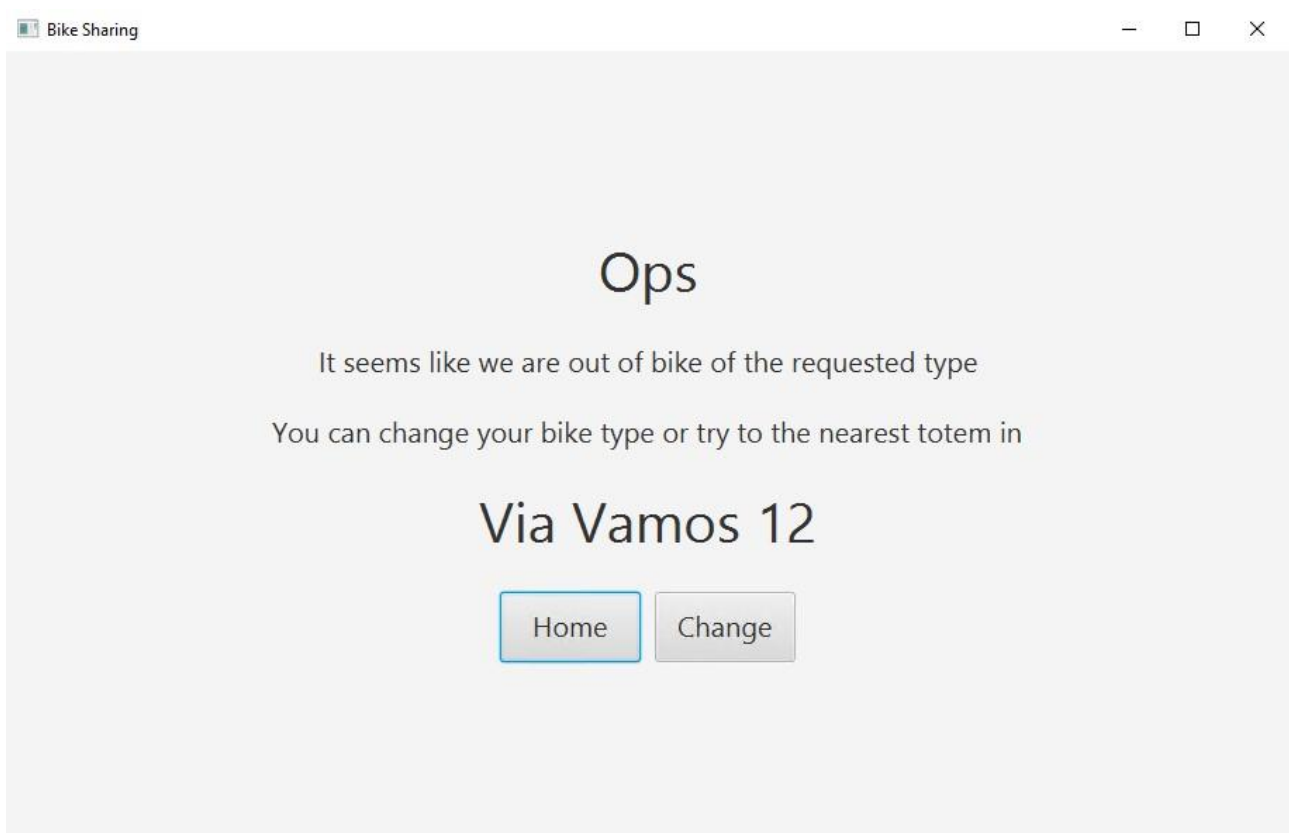
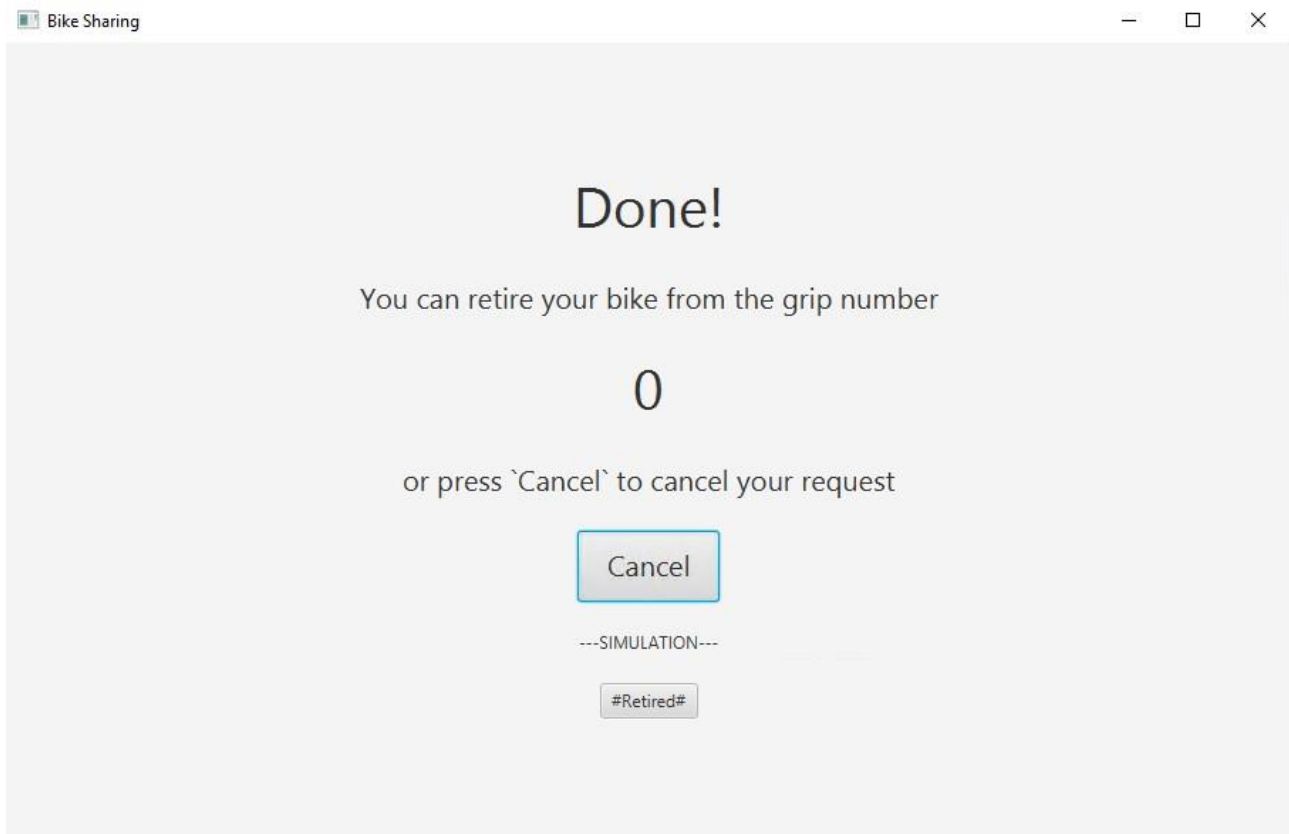
# Bikes

Choose the bike type

☒ Standard ☐ Electric ☐ Electric with baby seat

You'll pay 0.5€ per half hour after the first one

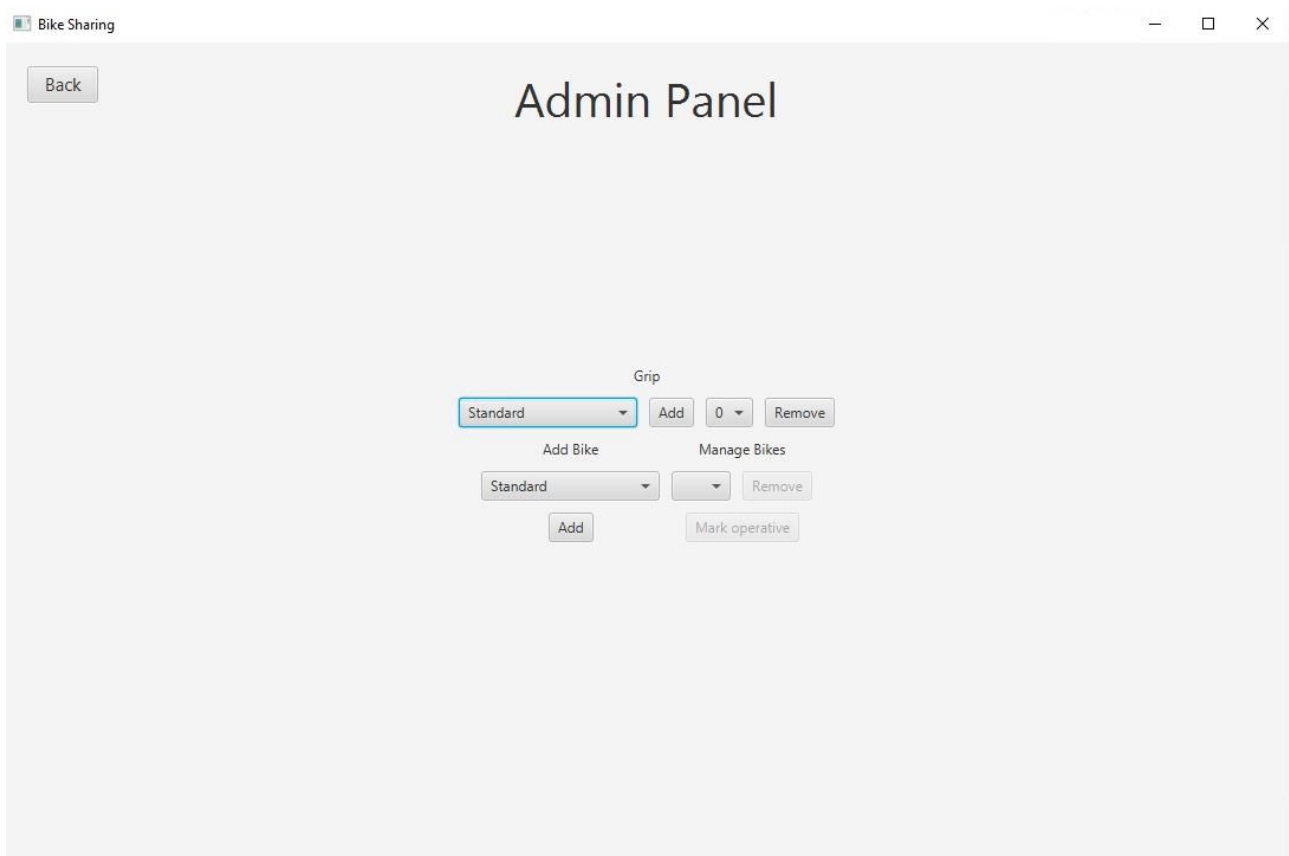
Confirm



### 3.3.7 Admin

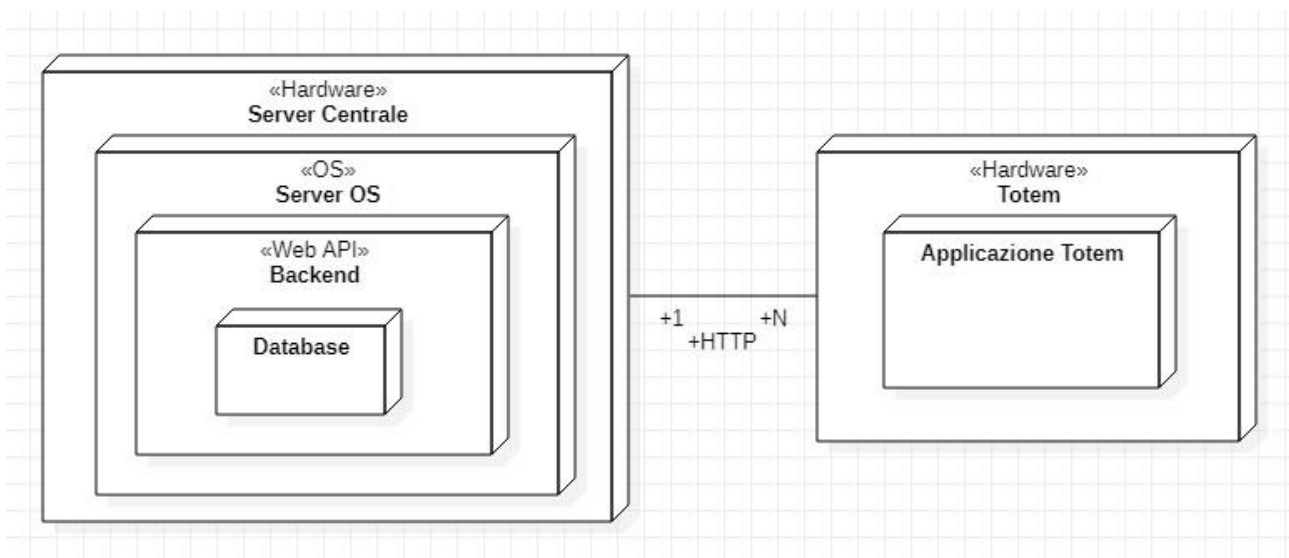
Questa pagina consente agli operatori del servizio di aggiungere e rimuovere bici o aggiungere e rimuovere morse





### 3.4 Diagramma di deployment

Il diagramma di deployment in questo caso è molto semplice, modellando i totem che si connettono ad un generico “backend”, sia esso un database condiviso o come spiegato in altri punti, un futuro server centrale



Il sorgente del diagramma di deployment si trova in Bike Sharing > Deployment > Deployment oppure un’immagine ad alta risoluzione si trova nella cartella images col nome “Deployment.jpg”

### 3.5 Specifica e verifica dei vincoli

Di seguito i vincoli OCL di alcune funzioni. Il corrispettivo in JML si trova direttamente applicato alle stesse sul codice sorgente sotto forma di commento

## **Abbonamento**

```
{
context Subscription
Inv: Subscription.allInstances.isUnique(id)
Inv: password.length >= 4 and password.length <=20
Inv: numberOfExceed >=0
Inv: numberOfExceed = 3 implies isExpired() = true
}
```

## **Morsa**

```
{
context Grip
Inv: Grip.allInstances.isUnique(id)
Inv: position > 0
}
```

## **Totem**

```
{
context Totem
Inv: Totem.allInstances.isUnique(id)
Inv: Totem.allInstances.isUnique(address)
}
```

## **Card**

```
{
Context Card
Inv: Card.allInstances.isUnique(id)
Inv: code.length = 16
Inv code.ToCharArray().all.isDigit() == true
}
```

## **Bike**

```
{ context Bike
Inv: Bike.allInstances.isUnique(id)
}
```

## **DbManager**

```
context DbManager::register(password:String, type:SubscriptionType, isStudent:Boolean,
cardCode: int, cardExpireDate:Date) pre:
password<>null and cardExpireDate<>null and type <> null
```

```
context DbManager::login(code:int, password:String) pre:
```

password<>null

## CardManager

context CardManager::isValidCard(code:String, expireDate:Date, cvv:int, type:SubscriptionType):Boolean pre:  
expireDate <> null and cvv > 100 and cvv < 999 and code.length = 16

context CardManager::pay(card:Card, amount:double): Boolean pre:  
card <> null and amount > 0

### 3.6 Descrizione del testing

I test consistono in delle classi istanziate da JUnit che prima dei test si assicurano che il database sia creato e contenga dei record, **cancellando il precedente**.

I test implementati riguardanti il database sono:

- Test sul login di un account già esistente
- Test sulla registrazione di un nuovo account
- Test sul report di una bicicletta esistente come danneggiata
- Test sul rendere di nuovo operativa una bicicletta disabilitata
- Test per bloccare una morsa
- Test per sbloccare una morsa

Per quanto riguarda il dominio, ci sono dei test per verificare gli stati

- Test sul costo della sottoscrizione
- Test sul costo della bici
- Test sulla disattivazione della sottoscrizione

I test utilizzano il branch coverage

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

L'ambiente di sviluppo è Visual Studio Code (ultima versione stabile). Le estensioni utilizzate sono incluse nel [Java Extension Pack](#), la versione consigliata di Java è la 11 (in particolare è stata usata la OpenJDK di Red Hat). Per far partire l'applicazione dal codice sorgente basta aprire la cartella principale del progetto con Visual Studio Code configurato e premere F5, oppure cliccare il pulsante "run" che apparirà sopra il main in App.java. Il database di test può essere generato eseguendo lo script "data.sql" con Sqlite. In particolare va eseguito, attraverso la CLI di sqlite3 il comando "sqlite3 PATH/data.db" e in seguito ".read PATH/data.db" (PATH sarà la path in cui viene eseguito il progetto). Nel caso ci dovessero essere problemi sarà comunque presente un file .db contenente gli stessi record. È importante che il db sia nella stessa working directory da cui viene lanciato il programma

Viene fornito nella cartella compressa "BikeSharingBinaries.rar" i binari e una copia del database già pronti che possono essere avviati con il file run.bat (su Windows, su Mac/Linux andrebbe adattato leggermente il run.bat)

Il database di test contiene tre utenti di prova:

- Codice: 1, Password: admin
- Codice: 2, Password: normal

- Codice: 3, Password: student
- Codice: 4, Password: expired

I primi tre hanno le capacità descritte dalla password, mentre il quarto è una sottoscrizione scaduta e quindi non utilizzabile. Sono disponibili due totem (si può simulare in quale totem si è dalla schermata home), 6 morse (3 per totem) e 3 bici (tutte sulle morse del primo totem)