# POLITECNICO DI TORINO

Dipartimento di Ingegneria Gestionale e della Produzione

**Corso di Laurea Triennale in Ingegneria Gestionale**
**Classe L-8**

Tesi di Laurea Triennale

## Progettazione e sviluppo di un'applicazione per la gestione di un servizio di bike sharing


**Relatore**
Prof. Fulvio Corno

**Candidato**
Umberto Ferrari
Matricola 246061

AA. 2019/2020

# Indice

# Elenco delle figure



# Elenco delle tabelle

# 1 Proposta di progetto

## 1.1 Titolo della proposta

Applicazione per la gestione di un servizio di bike sharing

## 1.2 Descrizione del problema proposto

La mobilità urbana è un argomento sempre di maggior interesse, da un lato per motivi ecologici e dall'altro, attualmente, a causa dei rischi della pandemia da Covid-19. Infatti, si sta incentivando l'uso di mezzi di trasporto quali il bike sharing in sostituzione al normale trasporto pubblico, perciò si suppone che nei prossimi mesi si assisterà ad un aumento nel numero di utenti di questo tipo di servizi.

L'obiettivo è quello di presentare all'utente una visione d'insieme sull'utilizzo del servizio in caso di aumento dell'utenza e di fornire uno strumento adeguato a pianificare e analizzare i possibili miglioramenti da realizzare.

## 1.3 Descrizione della rilevanza gestionale del problema

L'applicazione intende fornire uno strumento per la gestione di un servizio di bike sharing.

Questo strumento può essere utile al gestore del servizio per metterne in luce i punti critici, per la gestione del sistema e per pianificare eventuali miglioramenti che potrebbero rivelarsi necessari a causa del probabile aumento di noleggi.

## 1.4 Descrizione dei data-set per la valutazione

I dati che verranno utilizzati all'interno dell'applicazione sono relativi al servizio di bike sharing della città di Londra (Santander Cycles):

- **Dati relativi alle stazioni di noleggio (*bike points*):**
  I dati verranno ricavati tramite l'API fornita da TfL (*Transport for London*) dal sito `https://api.tfl.gov.uk/` e verranno importati dal formato JSON in un database SQL tramite una funzione apposita (l'API non verrà integrata nell'applicazione, ma utilizzata una sola volta per creare il dataset).

- **Dati relativi ai noleggi:**
  I dati sui noleggi verranno estratti dal sito di TfL (*Transport for London*) al seguente link `https://cycling.data.tfl.gov.uk/` dove sono disponibili suddivisi per periodo di interesse.
  Essendo di grandi dimensioni, i dati non saranno scaricati integralmente, ma verrà inserito un unico periodo di noleggi nel database. L'applicazione permetterà all'utente di definire un percorso per l'importazione di uno o più file di dati scaricati dal sito sopraindicato, i quali potranno essere sovrascritti o aggiunti ai dati già presenti. Anche in questo caso, i dati verranno importati dal formato CSV in un database SQL tramite una funzione apposita.

Tutti i dati sono rilasciati con licenza *Open Government Licence 2.0* che permette il libero uso dei dati, richiedendo di riconoscere TfL come fonte delle informazioni.

## 1.5 Descrizione preliminare degli algoritmi coinvolti

I principali algoritmi coinvolti saranno:

- Un algoritmo simulativo a eventi discreti che potrà essere eseguito secondo una serie di parametri inseriti dall'utente:
  - Intervallo temporale da simulare.
  - Intervallo dei dati storici da utilizzare per effettuare la simulazione, potrà essere selezionato lo stesso periodo dell'intervallo di simulazione ma dell'anno o mese precedente, oppure si potrà selezionare manualmente l'intervallo di dati da utilizzare.
  - Aumento dell'utenza che usufruisce del servizio, espresso in percentuale.
  - Possibilità di attivare la ridistribuzione notturna delle bici nel sistema.
  - Numero di bici da utilizzare durante la simulazione.
- Un algoritmo che generi, sulla base dei dati "storici" inseriti e dei valori in input, i noleggi del periodo da simulare, in modo che questi ultimi non siano deterministici ma realistici rispetto ai dati inseriti.
  Per esempio: l'aumento del traffico giornaliero inserito porterà all'inserimento di nuovi noleggi nelle ore di punta, oppure nel caso in cui siano

state inserite nuove stazioni di cui non sono disponibili noleggi, l'algoritmo si occuperà di generarli tenendo conto del traffico nelle stazioni vicine.

## 1.6 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione prevede le seguenti funzionalità:

- Nella prima schermata verrà mostrato un riassunto dei dati relativi alle stazioni attualmente presenti nel database, in particolare sarà possibile visualizzare la mappa con la disposizione geografica delle stazioni. Inoltre, da questa schermata sarà possibile effettuare modifiche al sistema: aggiungere, sia manualmente che importando un file in formato JSON, nuove stazioni, modificarne la capienza (intesa come numero di docks) o eliminare stazioni precedentemente aggiunte.
- Nella seconda schermata verrà mostrato un riassunto dei dati relativi ai noleggi attualmente caricati nel database, mostrando i periodi dei gruppi di noleggi attualmente memorizzati. Inoltre sarà possibile aggiungerne di nuovi importando un file in formato CSV.
- Nella terza schermata sarà possibile effettuare la simulazione, eventualmente inserendo i parametri richiesti. Al termine della simulazione, l'utente sarà indirizzato alla quarta schermata, la quale mostrerà i risultati.
- Nella quarta schermata saranno mostrati i risultati relativi all'ultima simulazione eseguita, sarà possibile visualizzare una mappa con indicate le stazioni codificate secondo una scala di colore rispetto al problema più rilevante che vi si è verificato. In particolare, tali problemi possono essere:
  - Stazioni vuote, quindi utenti che non sono riusciti a noleggiare una bicicletta.
  - Stazioni piene in cui gli utenti non sono riusciti a consegnare la bicicletta, dovendosi quindi recare in un'altra stazione per terminare il noleggio.
  - Stazioni ad alto traffico, per cui sarebbe utile un miglioramento.

  Inoltre, da questa schermata sarà possibile visualizzare dei grafici che riassumono i risultati della simulazione.

# 2 Descrizione dettagliata dal problema

## 2.1 Servizi di bike sharing

I servizi di bike sharing sono uno strumento di mobilità alternativa e sostenibile che negli ultimi anni ha visto una sempre maggior diffusione, anche sul territorio italiano.
Questi servizi mettono a disposizione un parco bici da poter essere noleggiato in autonomia dagli utenti.
La necessità di queste tipologie di servizio e il loro miglioramento è dovuto ai sempre maggiori problemi di mobilità presenti ormai nella maggior parte delle città del mondo, anche in ottica ambientale l'ampliamento di questi servizi porterebbe giovamenti rilevanti.
Inoltre, a seguito della pandemia da Covid-19 si sta incentivando l'uso di mezzi di trasporto alternativi, quali bike sharing e car sharing, in sostituzione al normale trasporto pubblico a causa del rischio sanitario legato all'utilizzo di bus, metropolitane e tram.

I servizi di bike sharing si dividono in due categorie fondamentali:

- **Station-based**: il servizio si basa sul posizionamento sul territorio di un certo numero di postazioni (stazioni appunto) in cui noleggiare e riconsegnare i mezzi.

- **Free-floating**: si tratta di servizi a flusso libero, in linea generale permettono di noleggiare e riconsegnare la bici in qualsiasi punto dell'area di copertura del servizio.

La prima tipologia è diffusa ormai da molti anni, mentre la seconda si è sviluppata solo di recente a causa delle maggiori difficoltà tecnologiche e di gestione ad essa connesse.
La città di Torino ne è un esempio: è stato inaugurato il servizio di bike sharing [TO]Bike a postazioni fisse nel 2010, ma soltanto nel 2017 è arrivato il primo servizio free-floating Mobike.

Nello sviluppo di questa applicazione si sono presi in considerazione i servizi della tipologia station-based.

## 2.2 Obiettivi

L'obiettivo di questa applicazione è duplice.
Da un lato, permettere al gestore di un servizio di bike sharing di avere una visione d'insieme sullo stato del servizio e metterne in luce i punti critici.
Dall'altro, l'applicazione potrà essere utilizzata per analizzare le conseguenze dovute a modifiche al sistema, testando, in particolare i possibili benefici che si potrebbero avere a seguito dell'ampliamento del sistema con l'installazione di nuove stazioni o con l'ampliamento del parco mezzi messo a disposizione degli utenti.

Nello specifico, questi obiettivi sono stati realizzati per mezzo di una simulazione dell'utilizzo del sistema da parte degli utenti. Il gestore del servizio potrà agire sui parametri di simulazione o effettuare modifiche al sistema per verificare l'uso del servizio sotto determinate condizioni.
Per esempio, sarà possibile verificare le zone critiche del sistema nel caso di un consistente aumento dell'utenza che usufruisce del servizio, oppure i benefici che si potrebbero avere in caso di installazione di una nuova stazione in una particolare area.

Infine, attivando la funzionalità relativa, sarà possibile visualizzare i benefici legati alla realizzazione di una strategia di ridistribuzione dei mezzi all'interno del sistema durante le ore notturne.

L'applicazione fornirà, oltre che una serie di parametri numerici e grafici, anche una mappa interattiva in cui sarà possibile visualizzare tutte le stazioni del sistema suddivise a seconda del problema più rilevante verificatosi. In particolare, si è deciso di suddividere le stazioni in quattro categorie:

- **Nessun problema rilevante**: una stazione viene inserita in questa categoria se tutti gli indicatori che identificano le tre categorie sottostanti sono al di sotto della media relativa a tutte le stazioni.
- **Stazioni spesso vuote**: stazioni in cui vi è stato un numero di utenti che ha tentato di effettuare un noleggio trovando però la stazione vuota, molto sopra la media di questo stesso numero, calcolata rispetto a tutte le stazioni. E' stato ritenuto il problema più grave che si possa verificare, in quanto oltre che a causare un disservizio per gli utenti, genera anche una perdita economica per il gestore.
- **Stazioni spesso piene**: stazioni in cui vi è stato un numero di utenti che ha tentato di riconsegnare la bici noleggiata trovando però la sta-

zione piena, molto sopra la media di questo stesso numero, calcolata rispetto a tutte le stazioni. Anche in questo caso, si tratta di un problema rilevante, ma che non causa una perdita economica diretta al gestore. In particolare, questa criticità può essere ridotta, almeno in parte, avviando una strategia di ridistribuzione del parco mezzi nel sistema, come dimostrato utilizzando la funzionalità apposita che simula una delle possibili strategie.

- **Stazioni ad alto traffico**: stazioni con un elevato numero di utenti durante il periodo in esame, nelle quali si è verificato almeno un problema in maniera consistente, ma non sufficiente per ricadere in una delle categorie di cui sopra.

## 2.3 Criticità

Le principali criticità sono legate ad alcuni parametri reali di cui non è stato possibile tenere conto.

In primo luogo, a causa della mancanza di informazioni non è stato possibile tenere conto di eventuali danneggiamenti e conseguente riparazione dei mezzi.

Inoltre, molti servizi offrono ai propri utenti minuti di noleggio gratuiti in caso in cui incorrano in una serie di problemi, per esempio una stazione vuota; ovviamente questo fatto aumenta la tendenza degli utenti a recarsi in una nuova stazione per effettuare il noleggio. Per ovvi motivi non è stato possibile tenere conto di questa variabile, si è però tentato di sopperire al problema chiedendo all'utente, in fase di avvio della simulazione, di inserire un parametro che indichi la tendenza degli utenti del servizio a cercare una nuova stazione nel caso in cui incorrano in problemi.

Infine, per quanto riguarda la funzione di ridistribuzione notturna, la strategia implementata è solo una delle possibili e non è di certo universalmente utile. In altre parole, non per tutti i servizi una strategia di questo tipo potrebbe portare reali benefici tali da giustificarne la messa in pratica, infatti ogni gestore, soprattutto sulla base della conformazione dell'area del servizio, elabara una propria strategia di ridistribuzione. Quella che è stata presa in considerazione è la più generale e indipendente dalla conformazione dell'area di copertura. Si è voluto inserire questa funzionalità solo per dare un'idea, per quanto generale, della possibile utilità di azioni di questo tipo.

# 3 Dataset

I dati utilizzati dall'applicazione sono di due tipi principali:

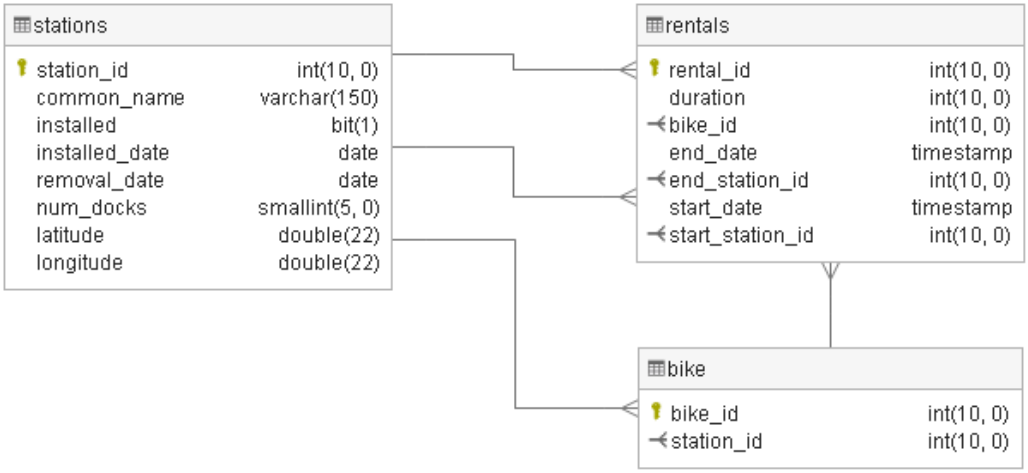
- Dati relativi alle stazioni di noleggio (*stations*)
- Dati relativi ai noleggi (*rentals*)
- In aggiunta, vengono utilizzate le informazioni relative alle biciclette presenti nel sistema, in particolare la sola informazione sull'ID del mezzo, la presenza di queste informazioni è però totalmente facoltativa, l'applicazione è strutturata per funzionare anche nel caso in cui la tabella *bike* sia vuota, provvedendo a generare gli ID dei mezzi a seconda del numero di bici da utilizzare per la simulazione inserito dall'utente.

Il software è stato realizzato per poter funzionare con i dati relativi a qualsiasi servizio, semplicemente importando i dati necessari tramite le funzionalità di caricamento file presenti.

A scopo di esempio e per permettere di testare le funzionalità dell'applicazione è stato scelto di utilizzare i dati relativi al servizio di bike sharing Santander Cycles della città di Londra, dati messi a disposizione dall'ente Trasport for London (*TfL*) e rilasciati con licenza *Open Government Licence 2.0*.

- I dati relativi alle stazioni sono stati ricavati tramite l'API fornita da TfL, è stato estratto il file testuale in formato JSON contenente le informazioni necessarie per poi essere importato nel database SQL.
- I dati relativi ai noleggi sono stati estratti al seguente indirizzo `https://cycling.data.tfl.gov.uk/` dove sono disponibili in formato CSV, per poi essere importati nel database SQL. Al link indicato è possibile, inoltre, scaricare ulteriori dati per poter ampliare il dataset proposto.

Figura 1: Diagramma ER




## 3.1 Descrizione delle tabelle

### 3.1.1 Tabella stations

La tabella *stations* contiene le informazioni relative alle stazioni presenti nel sistema:

- station_id: ID della stazione.
- common_name: nome della stazione.
- installed: parametro di tipo binario, serve a identificare le stazioni ancora installate nel sistema rispetto a quelle ormai rimosse, l'applicazione in fase di simulazione considererà le sole stazioni che presentano questo parametro a 1.
- installed_date e removal_date: rispettivamente data di installazione e di rimozione della stazione, nel caso in cui la stazione sia installata il campo removal_date assume valore *null*.
- num_docks: capienza della stazione intesa come numero di colonnine presenti.
- latitude e longitude: latitudine e longitudine del punto in cui si trova la stazione.

### 3.1.2 Tabella rentals

La tabella *rentals* contiene le informazioni relative ai noleggi che sono stati effettuati dagli utenti del sistema, sulla base di questi dati l'applicazione genererà gli eventi di noleggio per effettuare la simulazione.

- rental_id: ID del noleggio.
- duration: durata del noleggio, espressa in secondi.
- bike_id: ID della bici utilizzata nel noleggio.
- start_date e end_date: campi di tipo timestamp che rappresentano data e ora rispettivamente di inizio e fine noleggio.
- start_station_id e end_station_id: ID della stazione in cui avviene rispettivamente il prelevamento e la riconsegna della bici.

## 3.2 Formato dei file da importare

Tutti i dati possono essere importati tramite file, di seguito si descrive brevemente il formato di questi file per l'utilizzo dell'applicazione con dati relativi ad altri servizi.

Per poter importare dati relativi alle stazioni è necessario realizzare un file testuale in formato JSON con la seguente struttura.

```
{
"stations": [
  {
    "id": "BikePoints_1",
    "commonName": "Nome stazione",
    "additionalProperties": [
      {
        "key": "Installed",
        "value": "true",
      },
      {
        "key": "InstallDate",
        "value": "1278947280000",
      },
      {
```

```
        "key": "RemovalDate",
        "value": "",
      },
      {
        "key": "NbDocks",
        "value": "20",
      }
    ],
    "lat": 51.5,
    "lon": -0.1
  }
]
}
```


Per poter importare dati relativi ai noleggi è necessario realizzare un file in formato CSV utilizzando la virgola come separatore.
L'ordine dei parametri per ogni riga deve essere il seguente.


```
Rental Id,Duration,Bike Id,End Date,EndStation Id,
   EndStation Name,Start Date,StartStation Id,
   StartStation Name
100,60,1,01/01/2020 12:00,5,"Nome stazione arrivo
   ",01/01/2020 13:00,7,"Nome stazione partenza"
```

Tabella 1: Schema dei package

| Nome del package | Numero di classi | Descrizione |
|---|---|---|
| CompassBike | 2 | Avvio dell'applicazione |
| controller | 7 | Controller interfaccia grafica |
| model | 11 | Algoritmi |
| DAO | 4 | Collegamento al database |
| dataImport | 6 | Importazione dei dati |

# 4 Strutture dati e algoritmi utilizzati

## 4.1 Struttura del progetto

L'applicazione è stata sviluppata in linguaggio Java, seguendo i pattern MVC (*Model-View-Controller*) e DAO (*Database Access Object*).

Il progetto dell'applicazione è disponibile nel repository GitHub al seguente link: `https://github.com/TdP-prove-finali/FerrariUmberto`.

Il progetto si compone di 5 package, ognuno dedicato allo svolgimento di determinate funzionalità.

- **CompassBike**: contiene le classi per l'avvio dell'applicazione.
- **controller**: dedicato al controllo dell'interfaccia grafica, è presente una classe *controller* per ogni schermata dell'interfaccia, la classe *ChangePage* permette il cambio di interfaccia e gestisce la schermata di caricamento; infine, la classe *MapsGenerator* si occupa di realizzare le mappe interattive.
- **model**: package principale dell'applicazione, contiene la parte algoritmica e le classi necessarie alla memorizzazione dei dati.
- **DAO**: dedicato alle interazioni dell'applicazione con il database: connessione al database, lettura e modifica dei dati memorizzati.
- **dataImport**: gestisce le funzionalità di importazione dati.

## 4.2 Software e librerie

L'applicazione è stata sviluppata tramite l'IDE Eclipse. Per la realizzazione di parte dell'interfaccia grafica è stato utilizzato il software SceneBuilder.

Il DBMS utilizzato è MariaDB con interfaccia grafica HeidiSQL.
A causa dell'uso di alcuni comandi SQL che presentano notazioni differenti a seconda del DBMS utilizzato, non si assicura il funzionamento dell'applicazione con DBMS diversi da MariaDB.

Per lo sviluppo dell'applicazione sono state utilizzate le seguenti librerie, importate nel progetto, con la sola eccezione dell'ultima sotto riporata, tramite l'uso di *Maven*, inserendo la relativa *dependency* all'interno del file *pom.xml*.

- JSON-Java: utilizzata per facilitare l'importazione di file testuali in formato JSON.
- JGraphT: utilizzata per la realizzazione dei grafi.
- JFoenix: libreria di oggetti JavaFX material design.
- FontAwesomeFX: utilizzata per alcune aggiunte all'interfaccia dell'applicazione.
- MapBox GL: si tratta di una libreria in linguaggio *JavaScript*, è stata utilizzata per la realizzazione delle mappe interattive.

## 4.3 Algoritmi

L'applicazione si compone di due algoritmi principali.

### 4.3.1 Algoritmo di generazione eventi

Lo scopo di questo algoritmo, contenuto nella classe *EventsGenerator*, è quello di generare gli eventi su cui effettuare la simulazione. In particolare, genera gli eventi secondo i parametri inseriti dall'utente e i dati relativi ai noleggi contenuti nel database.

Questo algoritmo utilizza la maggior parte delle funzioni contenute nelle classi del package DAO, per ottenere informazioni aggregate dei dati memorizzate nel database. Sulla base dei dati ottenuti vengono create una serie di strutture dati necessarie all'esecuzione dell'algoritmo, in particolare:

- Un grafo diretto e pesato, i nodi rappresentano le stazioni e gli archi, rappresentati dalla classe *RouteEdge*, contengono le informazioni relativi ai tempi minimi e massimi di percorrenza del tragitto tra le due stazioni connesse dall'arco.
- Una serie di mappe che contengono varie informazioni, ad esempio, la probabilità che un utente del servizio si rechi in una certa stazione per effettuare un noleggio o la distribuzione del numero di noleggi effettuati nei diversi giorni e nei diversi istanti di tempo.

Una seconda parte dell'algoritmo permette di generare i dati sopra descritti anche per le stazioni inserite manualmente dall'utente o comunque per quelle stazioni per cui non sono disponibili dati memorizzati nel database. La generazione di questi parametri si basa sull'analisi dei valori associati alle 5 stazioni più vicine a quella in esame. In particolare, l'inserimento di una stazione in una certa area porterà a un aumento di traffico nella zona, ma anche a un reindirizzamento di una parte delle operazioni effettuate nelle stazioni più vicine nella nuova stazione.

L'algoritmo restituisce una lista di *Event* di tipo *Noleggio*, che rappresentano quindi la volontà di un utente di iniziare un noleggio in una certa stazione in un certo istante di tempo. Questi dati sono scelti casualmente sulla base dei parametri sopra descritti.

Tabella 2: Schema eventi simulazione

| **Tipo** | **Descrizione** |
|---|---|
| Noleggio | Rappresenta la volontà di un utente ad iniziare un noleggio |
| Prelievo | Rappresenta l'inizio effettivo del noleggio con il prelevamento della bici da parte dell'utente |
| Rilascio | Rappresenta il termine del noleggio con la riconsegna della bici |
| Stazione piena | Generato da un evento di *Rilascio* nel caso in cui la stazione di riconsegna scelta non abbia colonnine libere |
| Stazione vuota | Generato da un evento di tipo *Noleggio* nel caso in cui non siano disponibili delle bici nella stazione scelta per il prelevamento |

### 4.3.2 Algoritmo di simulazione

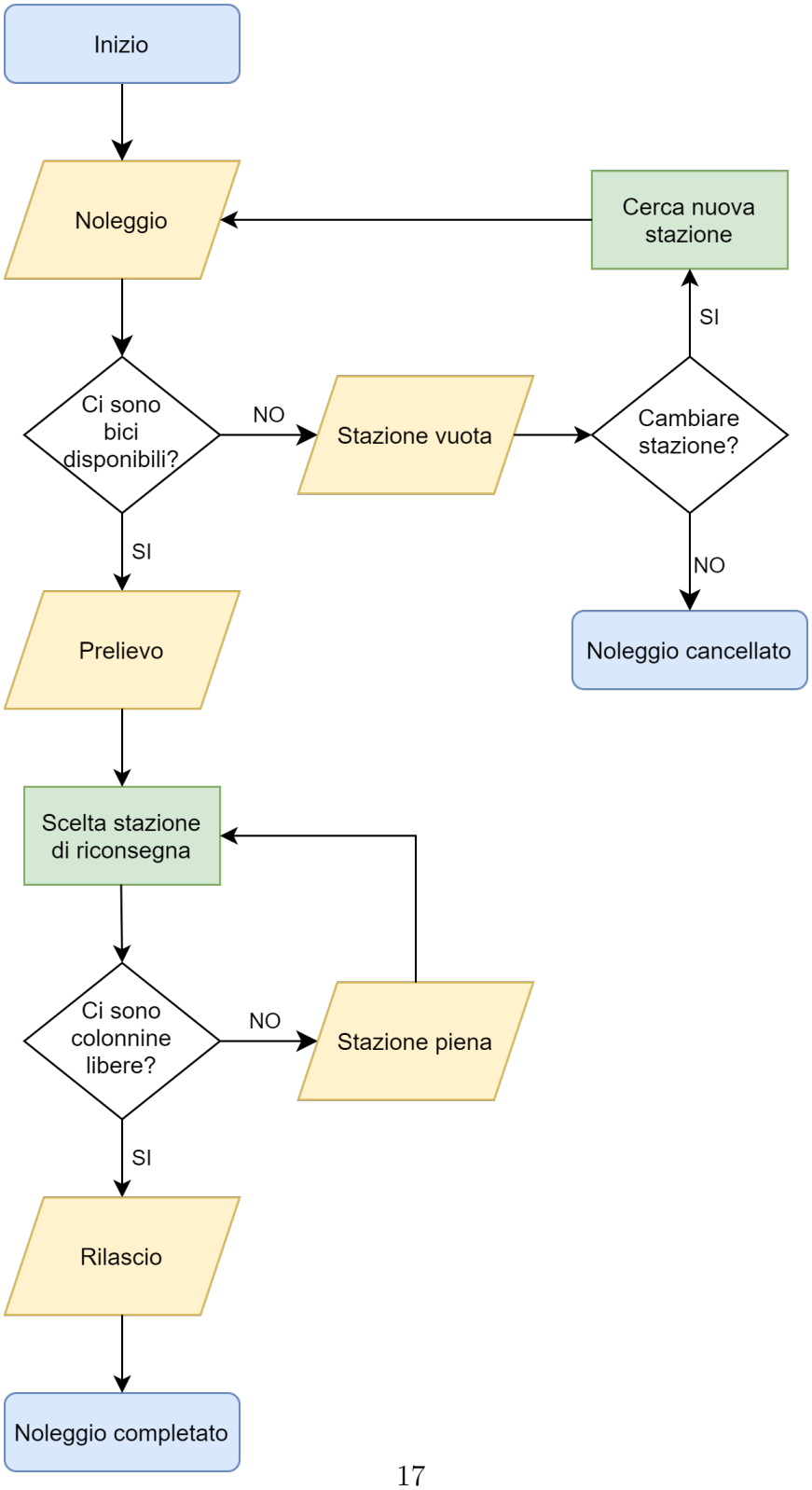
La parte principale dell'applicazione è composta dall'algoritmo di simulazione a eventi discreti, contenuto nella classe *Simulator*, che riceve in input i parametri inseriti dall'utente e la lista di eventi generati e si occupa di simulare tutte le operazioni effettuate nel sistema.

La classe *Simulator*, tramite una funzione apposita, si occupa anche di inizializzare le bici all'interno del sistema. La disposizione avviene in maniera uniforme, tentando di portare tutte le stazioni allo stesso tasso di riempimento, dando però priorità a stazioni a maggior capienza.

Nella tabella 2 è riportata una descrizione delle tipologie di evento utilizzate nell'algoritmo di simulazione.

La figura 2 mostra tramite un flowchart le operazioni gestite dall'algoritmo di simulazione.

Figura 2: Flowchart simulazione



# 5 Diagramma delle classi principali

Figura 3: Diagramma UML package DAO

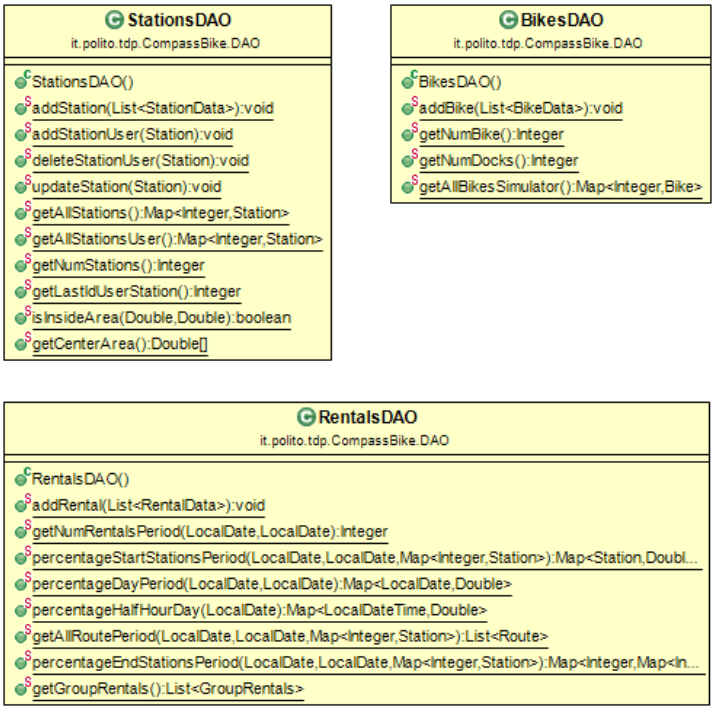



Figura 4: Diagramma UML package dataImport

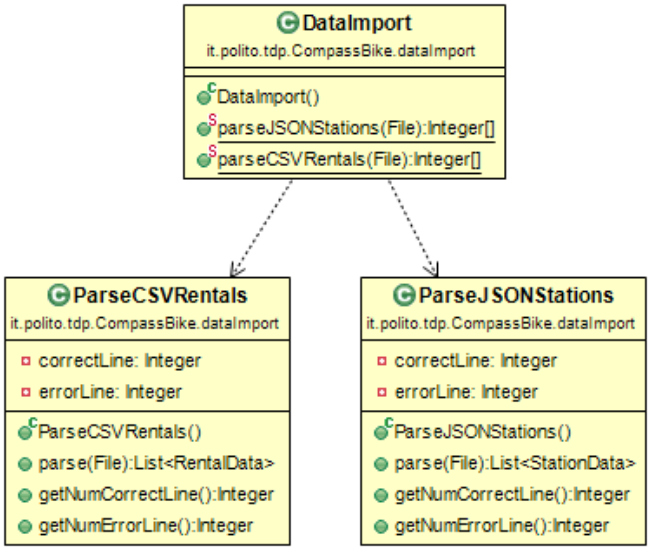
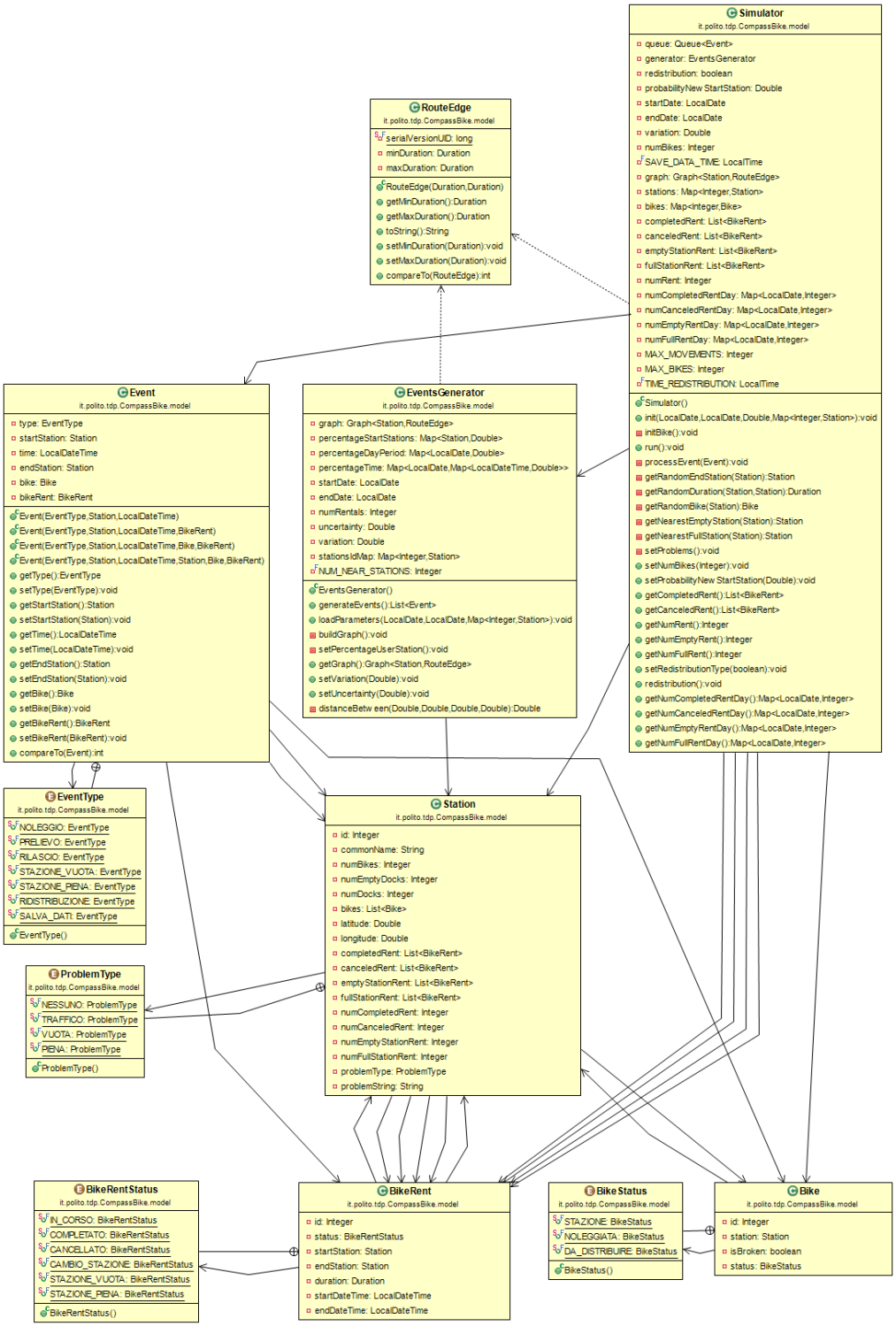


Figura 5: Diagramma UML package model



# 6 Interfaccia dell'applicazione

L'applicazione si compone di 4 schermate, inoltre, tramite un apposito pulsante è possibile aprire il browser per visualizzare le mappe interattive, oppure aprire una nuova finestra che mostra i risultati della simulazione sotto forma di grafici.
E' stato realizzato un video dimostrativo sull'uso dell'applicazione, disponibile al link: `https://youtu.be/wRPsn2E0DYg`.

Figura 6: Schermata inserimento dati stazioni

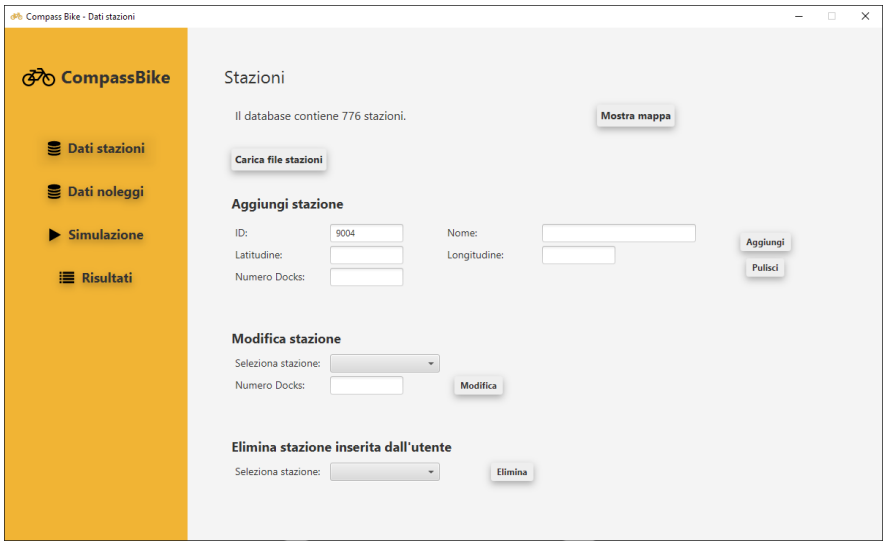



Figura 7: Schermata inserimento dati noleggi

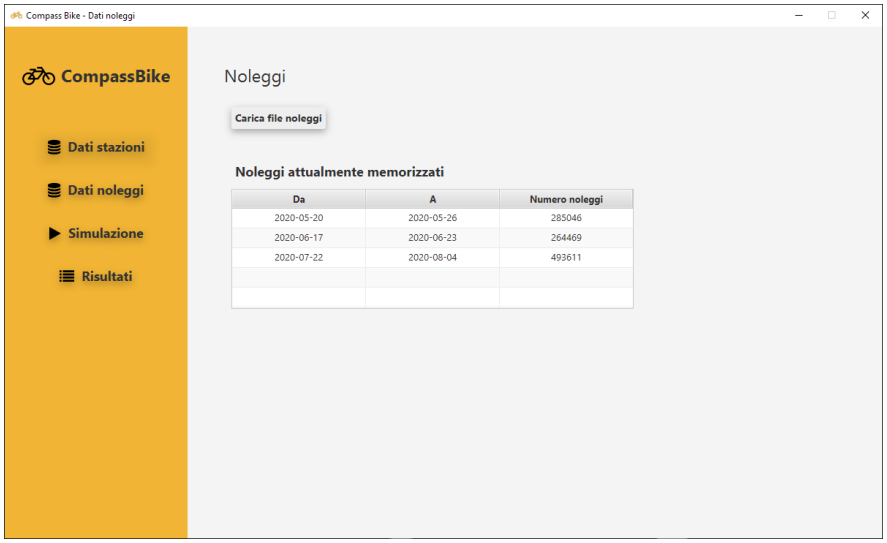


Figura 8: Schermata avvio simulazione

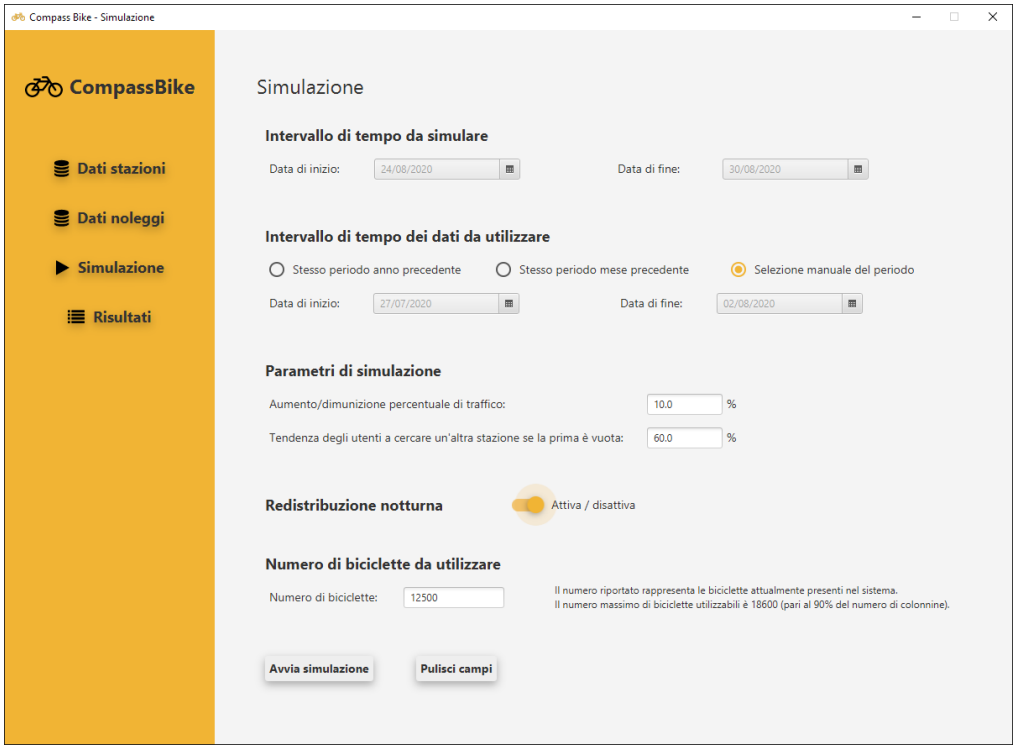



Figura 9: Schermata risultati

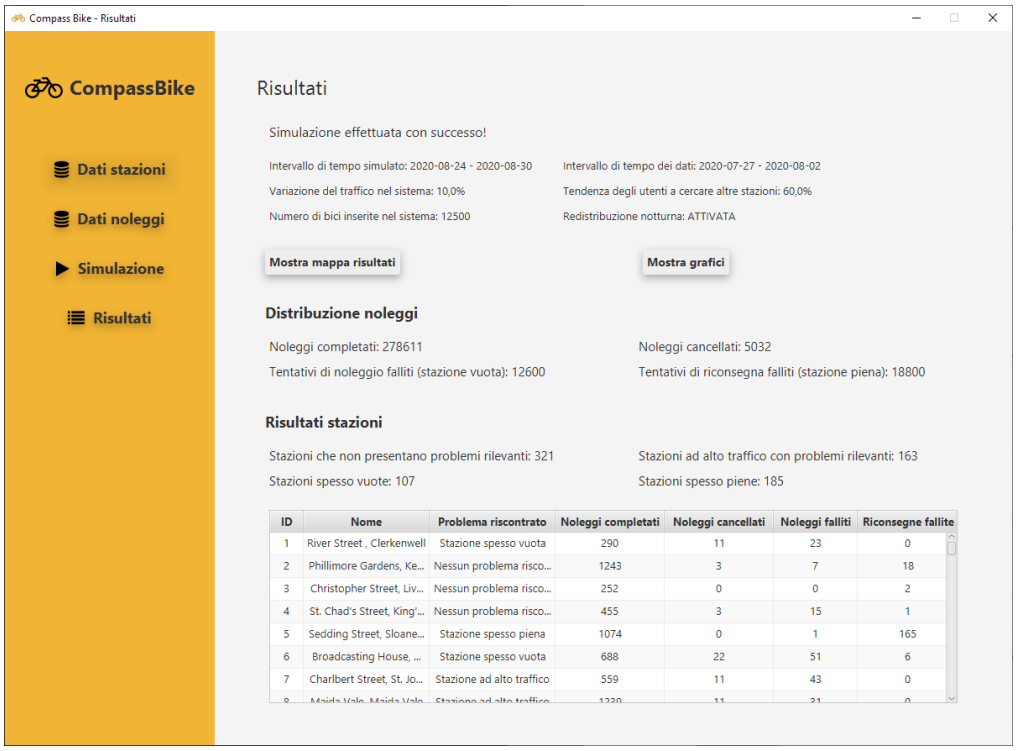


Figura 10: Schermata grafici

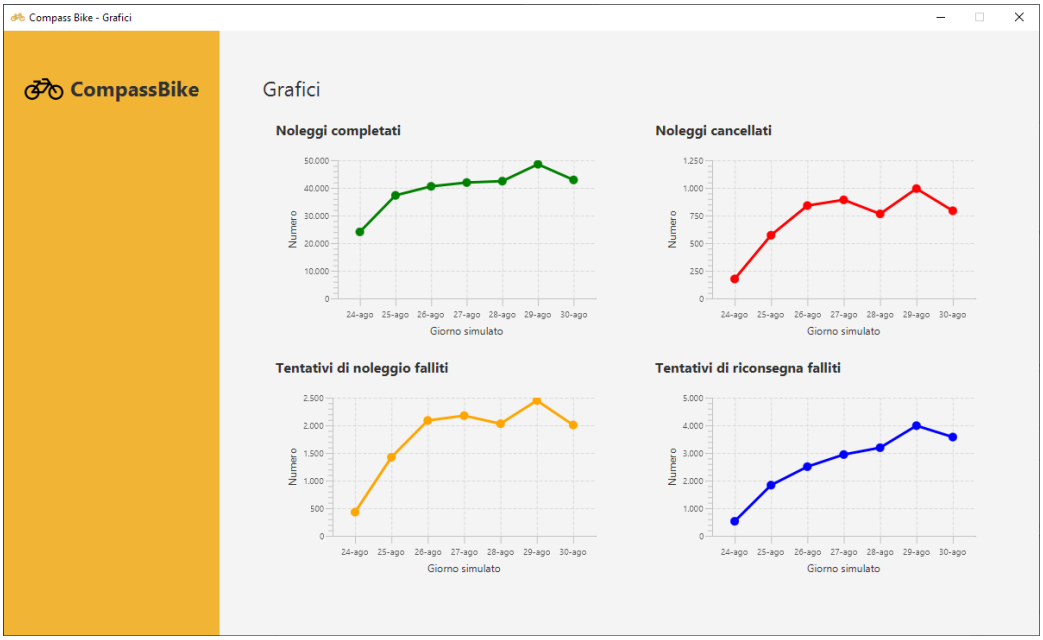
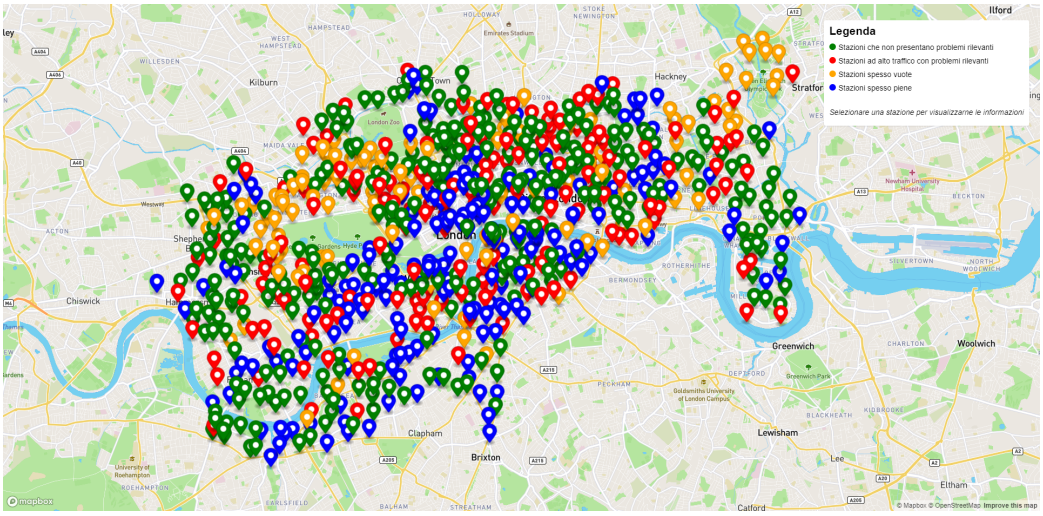



Figura 11: Mappa risultato



# 7 Esempio d'uso e risultati

## 7.1 Parametri

Si riportano i parametri inseriti all'avvio della simulazione utilizzati per gli esempi che seguono.

- Intervallo di tempo da simulare:
  - Data di inizio: 24-08-2020
  - Data di fine: 30-08-2020
- Intervallo di tempo dei dati da utilizzare:
  - Data di inizio: 27-07-2020
  - Data di fine: 02-08-2020
- Aumento o diminuzione percentuale di traffico: 10.0%
- Tendenza degli utenti a recarsi in un'altra stazione: 60.0%
- Numero di biciclette: 12000

Figura 12: Grafici esempio 1

## 7.2 Analisi

La prima simulazione svolta fornirà anche i "valori di controllo" utili ad analizzare le differenze causate dalla modifica dei parametri o del sistema negli esempi successivi.

Osservando i grafici in figura 12 possiamo immediatamente notare alcune informazioni interessanti:

- L'andamento della linea relativa ai noleggi completati è crescente e presenta un picco sabato 29 e domenica 30 agosto.
- Il fatto che l'andamento della linea relativa ai tentativi di noleggio falliti e quella relativa ai noleggi cancellati sia simile è dovuto al fatto che i noleggi cancellati avvengono a seguito di un certo numero di tentativi di noleggio falliti, a causa della mancanza di bici da poter noleggiare nelle stazioni. Inoltre, queste due linee presentano un andamento crescente in quanto i parametri che rappresentano dipendono dal traffico del sistema in un determinato giorno.
- Infine, la linea relativa ai tentativi di riconsegna falliti presenta un andamento crescente molto più netto: una riconsegna fallisce nel caso in cui la stazione scelta dall'utente risulti piena. Questo andamento è molto più interessante dei precedenti in quanto evidenzia come questo parametro sia solo in parte connesso al traffico nel sistema, in altre parole, l'andamento significativamente crescente è dovuto al fatto che il normale utilizzo del servizio da parte degli utenti porta nel lungo periodo ad avere una distribuzione dei mezzi nel sistema non uniforme, portando alla creazione di zone ad alto tasso di biciclette ma in cui il numero di noleggi non è così elevato.
  Questa conclusione ci viene confermata analizzando la mappa in figura 13 dove possiamo chiaramente notare due vaste zone che presentano quasi esclusivamente stazioni spesso piene (codificate in blu).

Figura 13: Mappa esempio 1

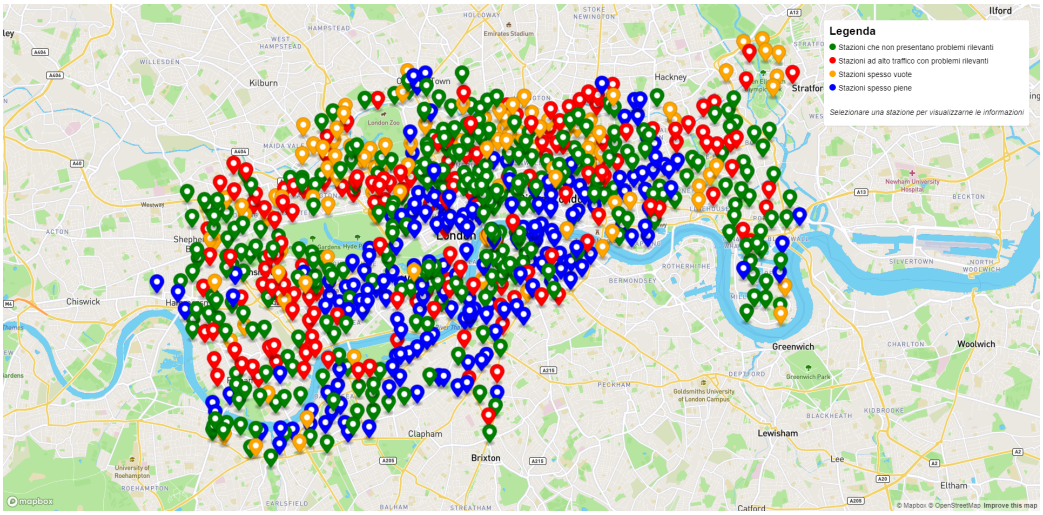
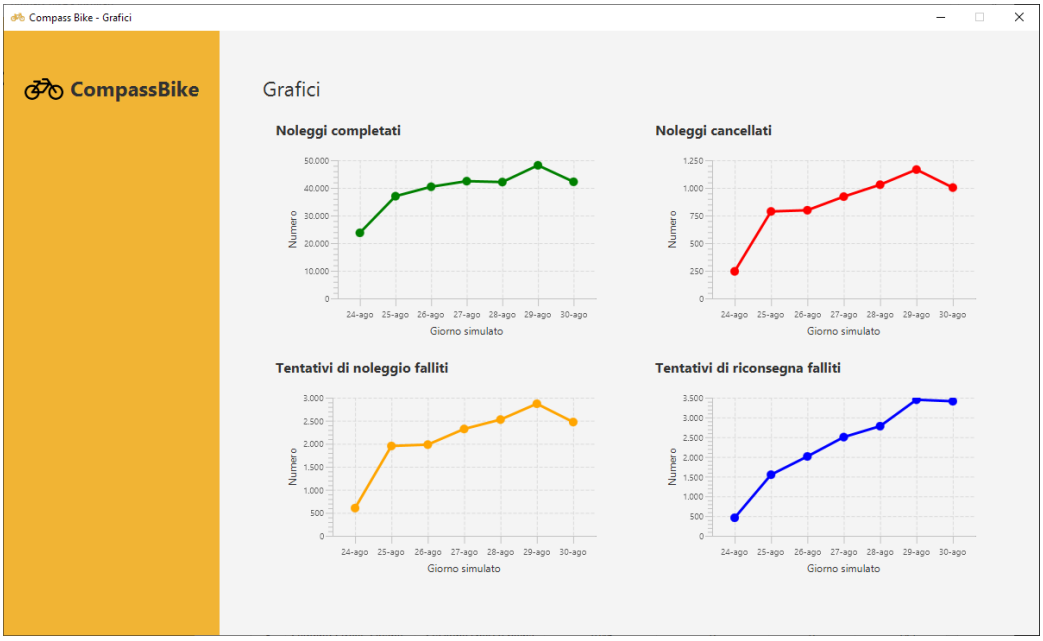



Figura 14: Grafici esempio 2



In figura 14 possiamo vedere i risultati ottenuti con gli stessi parametri in input, ma attivando la ridistribuzione notturna dei mezzi.
Analizzando i grafici, soprattutto quello relativo ai tentativi di noleggio falliti e quello relativo ai tentativi di riconsegna falliti, possiamo vedere prima di tutto come l'entità delle criticità si sia molto ridotta. La linea blu mantiene un andamento crescente, non così netto come il caso precedente, invece, la linea gialla presenta un andamento crescente di minore entità, soprattutto nei giorni centrali del periodo simulato, a dimostrazione di come l'utilizzo di una strategia di ridistribuzione generi un miglior bilanciamento dei mezzi all'interno del sistema.
In tabella 3 sono riportati i valori relativi alla simulazione senza ridistribuzione e in tabella 4 quelli ottenuti con l'attivazione della ridistribuzione.

Tabella 3: Risultati senza ridistribuzione

| **Categoria** | **Numero** |
|---|---|
| Completati | 277811 |
| Cancellati | 7130 |
| Noleggi falliti | 17938 |
| Riconsegne fallite | 21581 |

Tabella 4: Risultati con ridistribuzione

| **Categoria** | **Numero** |
|---|---|
| Completati | 276919 |
| Cancellati | 5950 |
| Noleggi falliti | 14725 |
| Riconsegne fallite | 16367 |

Figura 15: Zona di mappa esempio 3


(a) Stato iniziale

(b) A seguito dell'aggiunta

Per mostrare le funzionalità di test delle modifiche apportate al sistema ci si concentrerà sulla zona visibile in figura 15a estratta dalla mappa in figura 13, notiamo subito che la zona risulta molto critica probabilmente a causa del vicino parco che genera un elevato traffico.

Possiamo vedere in figura 15b come cambi la situazione inserendo una nuova stazione nell'area, oltre che un miglioramento a livello generale, andando a osservare i valori, prima e dopo l'aggiunta, riportati rispettivamente in tabella 5 e 6, relativi alle singole stazioni della zona si può vedere come si è ottenuta una consistente riduzione dell'entità dei problemi presenti.
Parametri stazione inserita:

- Latidutine: 51.539038
- Longitudine: -0.007594
- Numero docks: 30

Tabella 5: Risultati iniziali

| **ID stazione** | **Noleggi falliti** |
|---|---|
| 692 | 83 |
| 783 | 66 |
| 784 | 109 |
| 785 | 365 |
| 786 | 101 |
| 787 | 48 |
| 789 | 165 |
| 790 | 76 |
| 812 | 82 |
| 816 | 32 |
| Totale | 1127 |

Tabella 6: Risultati a seguito dell'aggiunta delle stazioni

| **ID stazione** | **Noleggi falliti** |
|---|---|
| 692 | 73 |
| 783 | 0 |
| 784 | 23 |
| 785 | 21 |
| 786 | 37 |
| 787 | 30 |
| 789 | 6 |
| 790 | 0 |
| 812 | 0 |
| 816 | 0 |
| Aggiunta | 79 |
| Totale | 269 (348) |

# 8 Conclusioni e possibili miglioramenti

In conclusione, l'applicazione sviluppata raggiunge gli obiettivi posti in fase di progettazione seppur con alcune limitazioni già indicate al paragrafo 2.3. Permette sia di ottenere una visione generale sullo stato del sistema, ma anche di analizzare il rendimento della singola stazione.
Inoltre, come mostrato negli esempi, permette, in maniera semplice, di analizzare le conseguenze delle possibili modifiche attuabili sul sistema.

Uno sviluppo interessante potrebbe essere l'aggiunta di indicatori economici che si basino, per esempio, sul listino prezzi del servizio in esame e su eventuali minuti gratuiti concessi agli utenti. Tali indicatori potrebbero essere usati per valutare la perdita economica derivante dai problemi che il sistema presenta e, di conseguenza, per valutare il ritorno economico dovuto a un eventuale risoluzione degli stessi, in modo da permettere la valutazione del vantaggio economico derivato dagli interventi necessari a risolvere tali criticità.

Inoltre, come già detto, potrebbe essere introdotta una funzione che possa gestire il danneggiamento dei mezzi e i conseguenti interventi di riparazione.

Infine, soffermandosi ad analizzare i tempi di esecuzione dell'applicazione, si possono trarre le seguenti conclusioni:

- Per quanto riguarda l'algoritmo di simulazione, il tempo di esecuzione richiesto risulta soddisfacente, considerando il numero di eventi che vengono processati: in media 800000 eventi per un periodo di simulazione pari a una settimana. Questo numero si traduce in circa 12 secondi di esecuzione, tempo che diminuisce drasticamente con la riduzione dell'ampiezza dell'intervallo di simulazione, arrivando a meno di 300ms nelle simulazioni di un singolo giorno.
- Dall'altro lato, invece, l'algoritmo di generazione eventi è meno soggetto a variazioni del tempo di esecuzione dovuto dall'ampiezza dell'intervallo simulato. In questo caso, è richiesto un tempo medio di 11 secondi per la simulazione di una settimane che si riduce, in maniera non molto consistente, a differenza del caso precedente, a circa 4 secondi nelle simulazioni di un singolo giorno. Ciò è dovuto al fatto che questo algoritmo si occupa di effettuare una serie di letture di informazioni aggregate dal database necessarie a inizializzare le strutture dati,

infatti, circa il 60% dei tempi precedentemente indicati è richiesto per effettuare queste letture.

Tenendo conto di queste considerazioni, si ritiene che il margine di miglioramento dei tempi di esecuzione dell'applicazione sia nella possibilità di elaborare query SQL più ottimizzate per estrarre le informazioni dal database.