Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

---

Tesi di Laurea Magistrale in Ingegneria Informatica

Microservices

Candidato
Lorenzo Niccolai

Relatore
Prof. Enrico Vicario

Correlatore
???

Anno Accademico 2019

Indice

1 Modellazione dell'incrocio **1**

1.1 Il grafo dei trasporti 1

1.2 Configurazione 3

2 Applicazione **5**

2.1 Architettura 6

2.1.1 Architettura esagonale 6

Bibliografia **9**

Capitolo 1

Modellazione dell'incrocio

Interesse di questo lavoro è lo studio del traffico nei pressi di un incrocio tra auto e tramvia. Per supportare una visione più ampia, ed altre eventuali tipologie di analisi, sarà proposto un modello in grado di descrivere una rete di trasporti generica.

1.1 Il grafo dei trasporti

Si può pensare di modellare una rete stradale con un grafo in cui gli archi rappresentano le strade ed i nodi rappresentano i punti di congiunzione: questi potranno poi essere specializzati a seconda del tipo di incrocio. Come osservabile in figura 1.1, al centro di tutto vi sono gli elementi *RelevantPoint* e *LaneSegment*, che potranno essere legati tra loro in vario modo a seconda della rete.

Per poter rappresentare un singolo frammento di rete stradale sono stati introdotti due tipi di punti speciali che modellano entrata e uscita dal sistema (1.2).

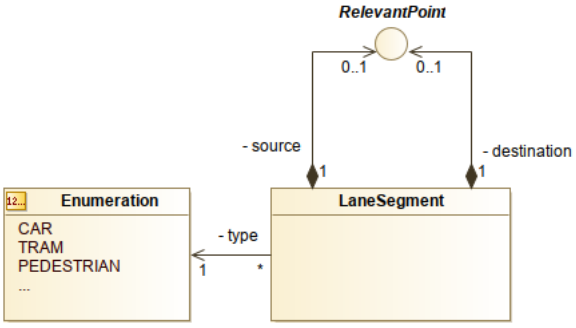

Figura 1.1: Rappresentazione di un sistema di incroci

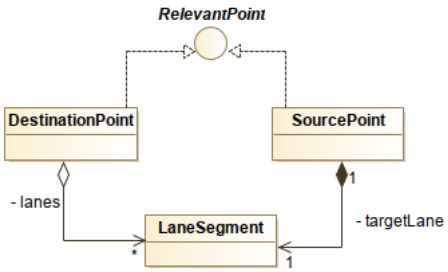

Figura 1.2: Sorgenti e destinazioni

I nodi interni alla rete invece sono realizzati da istanze della classe *CrossingPoint*, che fornisce il supporto di base alla modellazione di un incrocio generico, definendo una mappa che associa ad ogni linea in entrata una o più linee in uscita: in questo modo sono coperti numerosi casi reali, tra cui:

- Impossibilità di accedere ad una strada a partire da una certa corsia.
- Impossibilità di effettuare inversioni ad U.
- Impostare alcune tratte come più o meno trafficate.

Oltre alla versione base di *CrossingPoint*, che rappresenta un incrocio non arbitrato, è stata definita una versione *TrafficLightCrossingPoint* che invece introduce il concetto di semaforo (*TrafficLight*). Diventa possibile associare

ad ogni linea in ingresso all'incrocio un semaforo che la gestisce, in particolare sono state implementate le varianti temporizzate e a sensore: quest'ultimo caso è utile quando per esempio una linea automobilistica è interrotta da una linea tramviaria. Vedi figura 1.3.

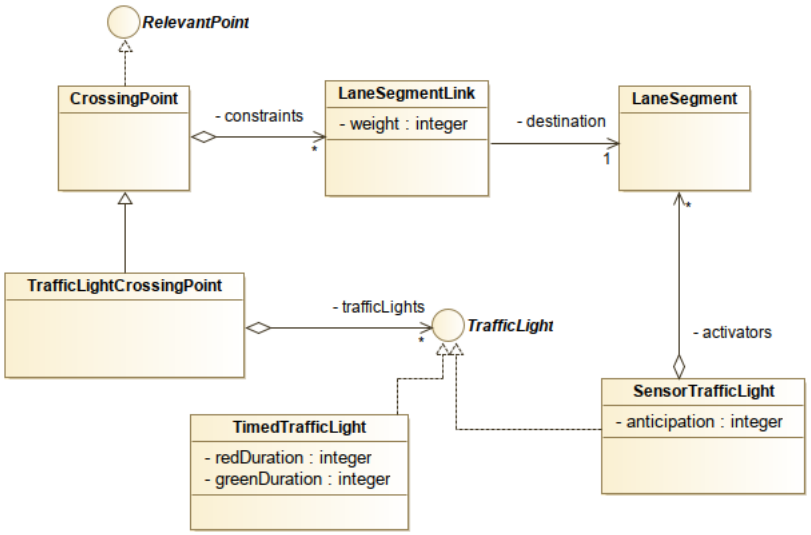

Figura 1.3: Modellazione di un incrocio

1.2 Configurazione

Per gli scopi a cui è destinato, il modello mostrato non prevede particolari funzionalità: il suo scopo è principalmente quello di *descrivere* una rete stradale. Per poter facilmente estendere il software (per esempio con un modulo user friendly per la creazione della rete o nuove analisi non ancora previste), le entità precedentemente descritte hanno la possibilità di essere configurate con una serie arbitraria di proprietà tipizzate chiave-valore (*Property*). Sarà possibile da parte dell'utente creare una serie di configurazioni predefinite da applicare poi ad una eventuale rete. Vedi figura 1.4. Lo scopo di questo

livello intermedio è quello di scaricare l'utente dalla necessità di conoscere tutte le possibili proprietà di configurazione.

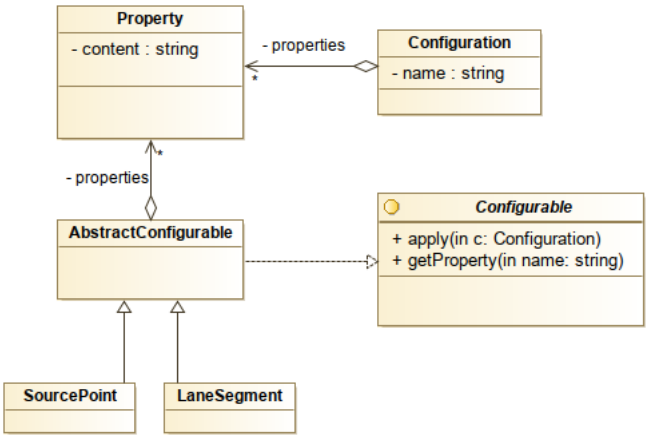

Figura 1.4: Configurazione

Capitolo 2

Applicazione

Per poter creare, configurare ed analizzare i modelli di incrocio è stata realizzata un'applicazione Web monolitica di esempio che implementa alcuni dei casi d'uso prima descritti, nonchè alcune funzionalità di base come la gestione utenti.

Sono stati individuati tre tipi di utente (vedi figura 2.1):

- Amministratore
- Esperto
- Cliente

Gli utenti di tipo amministratore hanno la possibilità di gestire i profili degli utenti aggiungendone di nuovi o eliminandone di esistenti (figura 2.2). Sarà lo stesso amministratore a decidere le password iniziali.

Gli utenti esperti conoscono a fondo il dominio e sono in grado di configurare un modello specificando le varie proprietà. Per questo hanno il permesso di creare configurazioni per gli utenti meno esperti (figura 2.3).

Gli utenti semplici possono invece creare dei progetti. Ogni progetto è associato ad una topologia di rete ed è di proprietà dell'utente che lo ha

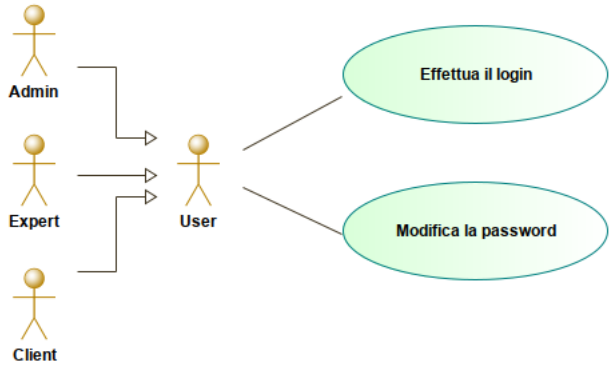

Figura 2.1: Classificazione degli utenti

Figura 2.2: Casi d'uso di un amministratore

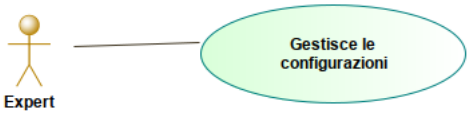

Figura 2.3: Casi d'uso di un utente esperto

creato. Una volta configurati i vari elementi del modello è possibile lanciare delle analisi, che rimarranno legate al progetto e delle quali sarà possibile monitorare lo stato ed accedere ai risultati.

2.1 Architettura

2.1.1 Architettura esagonale

Una delle alternative all'architettura organizzata a *layers* è l'architettura esagonale.

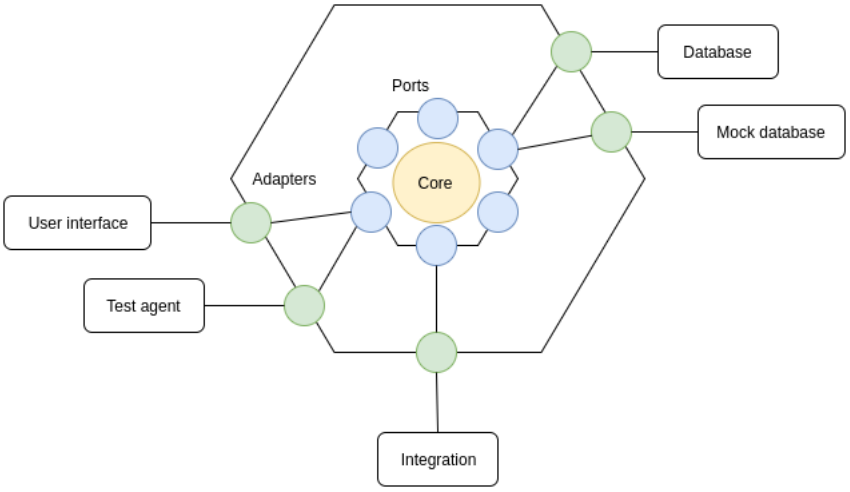

Figura 2.4: Modulo ad architettura esagonale

L'idea è quella di identificare con chiarezza il nucleo dell'applicazione e di renderlo completamente indipendente dalle tecnologie di contorno. Le comunicazioni di input/output con l'esterno sono guidate da interfacce definite dal nucleo e chiamate *porte*: queste possono essere di tipo *inbound* quando un modulo esterno ha necessità di invocare la logica di dominio, o di tipo *outbound* quando il core ha la necessità di invocare servizi esterni.

Ogni porta è implementata da uno o più adattatori (*adapters*) a seconda delle esigenze: per esempio una *outbound port* per l'accesso ad un repository di dati può essere implementata sia da un adattatore legato ad un DBMS che da un adattatore adatto al testing.

Convertire un'applicazione monolitica realizzata a layers in una aderente all'architettura esagonale non è un'operazione complessa: se per esempio lo stack prevede una serie di servizi REST che invocano la logica di dominio ed uno strato di persistenza (es: JPA), la separazione tra core e adapters è immediata.

Di nuovo, i benefici dell'architettura esagonale diventano evidenti quando

da un'applicazione monolitica si passa ad un'applicazione a microservizi, in quanto è possibile identificare con più chiarezza il compito del microservizio e questo è testabile con maggior facilità.

Bibliografia