



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI FIRENZE

CORSO DI INGEGNERIA INFORMATICA

PROGETTO SIM

APPLICATIVO PER GESTIONE E CONSULTAZIONE DEI SERVIZI DI INFOMOBILITÀ



Authors:

Marco LOSCHIAVO

Giulio CALAMAI

Teacher:

Prof. Enrico VICARIO

Collaborators:

Ing. Jacopo PARRI

Ing. Samuele SAMPIETRO

ANNO ACCADEMICO 2019-2020

INDICE

1	Introduzione	3
2	Contesto applicativo	4
3	Documentazione	5
3.1	Architettura SIM Backend	6
3.2	Endpoint di SIM Backend	7
3.2.1	User Account Endpoint	8
3.2.2	Smart Station Endpoint	10
3.2.3	Infomobility Service Endpoint	13
3.2.4	Municipality Endpoint	16
3.2.5	Authentication Endpoint	18
3.2.6	Admin Authorization Endpoint	19
3.3	Modello di dominio di SIM Backend	20
3.3.1	Entità coinvolte	21
3.4	Casi d'uso	25
3.5	Tecnologie adottate	27
3.6	Implementazione	27
3.6.1	Frontend	28
3.6.2	Backend	29
3.7	Code Snippet	32
4	Panoramica dell'applicazione	34
5	Conclusioni	40
A	Appendice	41
	Riferimenti bibliografici	42

INTRODUZIONE

Negli ultimi anni si è sentito ormai parlare molto di Smart City. Con quest'ultimo termine si intende una "città intelligente" che sa stare al passo con le innovazioni e con la rivoluzione digitale ed ha come obiettivo quello di migliorare la qualità della vita delle persone che vivono nella città, abilitando la creazione di servizi innovativi per la mobilità, sicurezza, il turismo, le imprese, la sanità, etc.

Uno degli aspetti cruciali nel processo di crescita e di progresso di una Smart City è la mobilità. È evidente, infatti, che gli spostamenti di cose e persone all'interno delle città, sono sempre più frequenti e numerosi, con un impatto notevole sia dal punto di vista della qualità dell'aria che della vivibilità e fruibilità delle nostre città. Alle Smart city, dunque, è demandato il difficile compito di individuare un nuovo modo di concepire e organizzare la mobilità e i trasporti, dando vita a una mobilità intelligente, molto diversa da quella incentrata su vetture private alimentate a combustibili fossili e un apporto residuale del trasporto pubblico locale. Molto concretamente, questo significa attivare nelle città servizi che favoriscano la condivisione dei mezzi (car sharing, car pooling, bike sharing, ecc), nonché di soluzioni digitali che permettano di usufruire più agevolmente dei mezzi pubblici (ad esempio tramite app, accesso ad access point, etc). In questo elaborato si propone una soluzione pratica per l'accesso ai servizi di mobilità presenti all'interno delle città attraverso delle "stazioni intelligenti", dette Smart Station, dislocate nella città, a cui un cittadino può accedere per usufruire di un particolare servizio di mobilità.

Per la realizzazione ci è stata fornita una documentazione di partenza che noi, nel ruolo di developer, abbiamo sviluppato ed ampliato. Non avevamo nessuna esperienza nello sviluppo frontend e backend di un'applicazione, perciò è stata per noi una buona occasione per affinare le nostre conoscenze sull'argomento. Nel secondo capitolo, viene descritto più nello specifico l'architettura utilizzata e l'obiettivo della parte backend. Il terzo capitolo tratta la documentazione originale e le modifiche apportate in fase di sviluppo, assieme alle tecnologie scelte. Nel capitolo quattro viene illustrato il funzionamento dell'applicazione finale, lato frontend. Infine, nel capitolo cinque si trattano le conclusioni e le considerazioni.

CONTESTO APPLICATIVO

L'obiettivo di questo elaborato è quindi l'implementazione lato backend di questi domini e le relazioni tra essi. In particolare, si vuole modellare la gestione di tutti i fornitori di servizi di infomobilità presenti nella città di varia tipologia (e. g. bike sharing, car sharing, etc.) e come questi possono essere utilizzati da un cittadino. Inoltre, un altro obiettivo di questo elaborato è lo sviluppo, lato frontend, di un applicativo web. Attraverso di esso è possibile accedere sia come amministratore per la gestione delle Smart Station, sia come utente per la fruizione del servizio.

I punti fondamentali sono:

1. Fornire tutti i servizi di mobilità ad un utente viaggiatore in modo da facilitargli gli spostamenti.
2. Consentire agli amministratori semplici la gestione di stazioni intelligenti e servizi di infomobilità. Essi hanno la possibilità di creare, modificare e disabilitare le smart stations presenti nelle città alle quali sono autorizzati. Inoltre possono aggiungere o rimuovere i servizi di infomobilità ai quali sono autorizzati. Quest'ultimi possono a loro volta essere abilitati, disabilitati e modificati.
3. Consentire agli amministratori di sistema (alto livello) di autorizzare gli amministratori semplici ad operare su determinate città e servizi. Possono inoltre aggiungere nuovi infomobility service provider oltre alle operazioni già citate.

DOCUMENTAZIONE

L'architettura utilizzata per la realizzazione è di tipo RESTful. Tale architettura è *service oriented*, ovvero orientata ai servizi, e tende a mantenere separati i moduli frontend e backend. Inoltre espone delle REST API nel backend che sono fruibili da diversi consumatori.

Il frontend lato client definisce delle "viste" che vengono popolate dopo aver ricevuto dal backend una response in formati *human-readable* e *machine-readable* (e.g. JSON, XML, etc.). Il web browser interpreta i documenti HTML+CSS dopo un pre-processing tramite un linguaggio lightweight client-side (e.g. JavaScript), e si aggiorna con HTTP request che inducono una *partial-refresh*.

Il backend è quella parte di applicazione che risiede strettamente lato Server e che espone servizi tramite specifiche REST API. In particolare consente le operazioni CRUD (CREATE, RETRIEVE, UPDATE and DELETE) su delle risorse.

Nel nostro progetto abbiamo un DBMS che abilita gli utenti a definire, creare, mantenere e controllare gli accessi ad un database per le seguenti informazioni:

- **Municipality:** rappresentano le città dove è presente il servizio, nelle quali sono presenti più stazioni.
- **Smart Stations:** rappresentano la stazione che ospita i servizi di mobilità.
- **Infomobility Services:** sono i servizi esposti in una stazione (e.g. Car2Go, Enjoy etc.).
- **Users:** sono gli utenti amministratori adibiti alla gestione della piattaforma e degli elementi che offre.
- **Authorizations:** rappresentano le autorizzazioni fornite agli **users** per gestire i servizi.

Oltre questo, sono stati utilizzati servizi offerti da *service provider* esterni (STINGRAY), adattati e utilizzati per le nostre necessità. Il primo fornisce dati ambientali (temperatura

3.2 ENDPOINT DI SIM BACKEND

I servizi esposti da SIM Backend sono consumabili direttamente dall'applicazione client di frontend, differenziandosi in base a condizioni di autenticazione.

Nel caso in cui un utente risulti:

- **autenticato**, equivale ad un utente con ruolo “SUPER ADMIN” o “ADMIN”. Tale utente è dotato di maggiori diritti rispetto agli utenti semplici non autenticati;
- **non autenticato**, equivale ad un utente senza alcun ruolo, ossia un viaggiatore che ha diritti di fruizione ridotti rispetto ad un amministratore.

Gli endpoint previsti sono i seguenti:

- **User Account Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo “account utente”.
Tale endpoint predispone le funzionalità basilari per compiere casi d’uso CRUD (Create Retrieve Update Delete) su tali entità. Le funzionalità “in scrittura” di gestione degli account saranno permesse solamente a utenti con ruolo “SUPER ADMIN”;
- **Smart Station Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo “stazione intelligente”.
Tale endpoint predispone le funzionalità basilari per compiere casi d’uso CRUD (Create Retrieve Update Delete) su tali entità, permettendo anche agli utenti con ruolo “ADMIN” di associare l’elenco di servizi di infomobilità attivi per essa;
- **Infomobility Service Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo “servizio esterno di infomobilità”.
Tale endpoint predispone le funzionalità basilari per compiere casi d’uso CRUD (Create Retrieve Update Delete) su tali entità. Le specifiche funzionalità per la gestione delle configurazioni sono permesse a utenti con ruolo “ADMIN”;
- **Municipality Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo “municipalità”.
Tale endpoint predispone le funzionalità basilari per compiere casi d’uso CRUD (Create Retrieve Update Delete) su tali entità. Le specifiche funzionalità per la gestione delle configurazioni sono permesse solo ad utenti con ruolo “SUPER ADMIN”;

- **Authentication Endpoint:** servizio REST dedicato all' "autenticazione".
Tale endpoint predispone la funzionalità basilare di autenticare un utente, generando un token JWT con auth0.
- **Admin Authorization Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo "autorizzazione".
Tale endpoint predispone le funzionalità basilari per compiere casi d'uso CRUD (Create Retrieve Update Delete) su tali entità. Le specifiche funzionalità per la gestione delle configurazioni sono permesse solo ad utenti con ruolo "SUPER ADMIN";

Tra gli endpoint evidenziati, 3 sono stati aggiunti successivamente, ovvero **Municipality, Authentication** e **AdminAuthorization**. Questa decisione è stata presa al fine di semplificare la gestione via frontend della web app, consentendo il login degli amministratori, operazioni CRUD sulle municipalità e l'autorizzazione degli ADMIN a determinati servizi.

Nei sotto-paragrafi che seguono vengono descritti i dettagli di ogni servizio previsto. Con la denominazione **SIM_BACKEND_URI**, si intende l'URI riferita al dominio applicativo del sotto-sistema SIM backend.

3.2.1 USER ACCOUNT ENDPOINT

Questo endpoint è referenziato dalla URI:

`SIM_BACKEND_URI/users`

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-USR-01 [List Users]	
Verbo HTTP	GET
URI	/users
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista degli utenti dotati di account

SIM-B-USR-02 [Create Users]	
Verbo HTTP	POST
URI	/users
Path Params	/
Query Params	/
Request Body	JSON contenente una rappresentazione di un account utente da registrare all'interno del sistema SIM Backend
Response Body	JSON contenente una rappresentazione del nuovo account utente, successivamente all'operazione di persistenza

SIM-B-USR-03 [Retrieve User]	
Verbo HTTP	GET
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione dell'account utente richiesto

SIM-B-USR-04 [Update Users]	
Verbo HTTP	PUT
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	JSON contenente una rappresentazione di un account utente da modificare
Response Body	JSON contenente una rappresentazione dell'account utente, successivamente all'operazione di modifica

SIM-B-USR-05 [Delete User]	
Verbo HTTP	DELETE
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione dell'account utente, antecedente all'operazione di cancellazione

3.2.2 SMART STATION ENDPOINT

Questo endpoint è referenziato dalla URI:

`SIM_BACKEND_URI/smart-stations`

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-SS-01 [List Smart Stations]	
Verbo HTTP	GET
URI	/smart-stations
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle stazioni intelligenti presenti all'interno del sistema SIM Backend

SIM-B-SS-02 [Create Smart Station]	
Verbo HTTP	POST
URI	/smart-stations
Path Params	/
Query Params	/
Request Body	JSON contenente una rappresentazione di una stazione intelligente da creare, con l'elenco di servizi di infomobilità attivi per essa
Response Body	JSON contenente una rappresentazione della nuova stazione intelligente, successivamente all'operazione di persistenza

SIM-B-SS-03 [Retrieve Smart Station]	
Verbo HTTP	GET
URI	/smart-stations/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente la stazione intelligente richiesta

SIM-B-SS-04 [Update Smart Station]	
Verbo HTTP	PUT
URI	/smart-stations/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	JSON contenente una rappresentazione di una stazione intelligente da modificare, con l'eventuale indicazione dell'elenco di servizi di infomobilità attivi per essa
Response Body	JSON contenente la stazione intelligente, successivamente all'operazione di modifica

SIM-B-SS-05 [Enable/Disable Smart Station]	
Verbo HTTP	PATCH
URI	/smart-stations/{uuid}?enable={en}
Path Params	uuid: stringa in formato UUIDv4
Query Params	<p>en: booleano. Se assume valore:</p> <ul style="list-style-type: none"> • true, attiva il processo di ri-abilitazione • false, attiva il processo di disabilitazione <p>Si intuisce come una ri-abilitazione abbia senso solamente quando lo stato precedente della stazione risulti già disabilitato (e viceversa).</p>
Request Body	/
Response Body	JSON contenente una rappresentazione della stazione intelligente che è stata appena "ri-abilitata/disabilitata" all'interno del sistema SIM Backend

Vista la necessità, sono stati aggiunti altri servizi:

SIM-B-SS-06 [List Filter Smart Stations]	
Verbo HTTP	GET
URI	/smart-stations/municipality/{uuid}
Path Params	uuid: stringa in formato UUIDv4 riferita alla municipalit� di interesse
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle stazioni intelligenti in una municipalit�, presenti all'interno del sistema SIM Backend

SIM-B-SS-07 [List Smart Stations for Admin]	
Verbo HTTP	GET
URI	/smart-stations/getAllAuthorized
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle stazioni intelligenti gestibili da un ADMIN, presenti all'interno del sistema SIM Backend

SIM-B-SS-08 [List Smart Stations by Infomobility]	
Verbo HTTP	GET
URI	/smart-stations/associable-infomobility-services/{uuid}
Path Params	uuid: stringa in formato UUIDv4 riferita all'infomobility di interesse
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle stazioni intelligenti associabili ad un servizio di mobilit� da un ADMIN, presenti all'interno del sistema SIM Backend

SIM-B-SS-09 [Retrieve Ambiental Data for Smart Station]	
Verbo HTTP	GET
URI	/smart-stations/ambientalData/{cmad_mac_address}
Path Params	cmad_mac_address: stringa identificativa di un dispositivo di acquisizione e trasmissione dati
Query Params	/
Request Body	/
Response Body	JSON contenente i dati ambientali per la stazione intelligente richiesta

SIM-B-SS-10 [Delete Smart Station]	
Verbo HTTP	DELETE
URI	/smart-stations/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione della stazione intelligente, antecedente all'operazione di cancellazione

3.2.3 INFOMOBILITY SERVICE ENDPOINT

Questo endpoint è referenziato dalla URI:

SIM_BACKEND_URI/infomobility-services

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-IS-01 [List Infomobility Services]	
Verbo HTTP	GET
URI	/infomobility-services
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista dei servizi di mobilità presenti all'interno del sistema SIM Backend

SIM-B-IS-02 [List Infomobility Services by Smart Station]	
Verbo HTTP	GET
URI	/infomobility-services/smart-station/{uuid}
Path Params	uuid: stringa in formato UUIDv4 riferita alla Smart Station di interesse
Query Params	/
Request Body	/
Response Body	JSON contenente la lista dei servizi di mobilità in una stazione, presenti all'interno del sistema SIM Backend

SIM-B-IS-03 [Create Infomobility Service]	
Verbo HTTP	POST
URI	/infomobility-services
Path Params	/
Query Params	/
Request Body	JSON contenente una rappresentazione di un servizio di infomobilità da creare, con l'elenco delle smart station associate
Response Body	JSON contenente una rappresentazione del nuovo servizio di infomobilità, successivamente all'operazione di persistenza

SIM-B-IS-04 [Retrieve Infomobility Service]	
Verbo HTTP	GET
URI	/infomobility-services/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione del servizio di infomobilità richiesto

SIM-B-IS-05 [Update Infomobility Service]	
Verbo HTTP	PUT
URI	/infomobility-services/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	JSON contenente una rappresentazione di un servizio di infomobilità da modificare, con l'eventuale elenco delle smart station associate
Response Body	JSON contenente una rappresentazione del servizio di infomobilità, successivamente all'operazione di modifica

SIM-B-IS-06 [Enable/Disable Infomobility Service]	
Verbo HTTP	PATCH
URI	/infomobility-services/{uuid}?enable={en}
Path Params	uuid: stringa in formato UUIDv4
Query Params	<p>en: booleano. Se assume valore:</p> <ul style="list-style-type: none"> • true, attiva il processo di ri-abilitazione • false, attiva il processo di disabilitazione <p>Si intuisce come una ri-abilitazione abbia senso solamente quando lo stato precedente del servizio di infomobilità risulti già disabilitato (e viceversa).</p>
Request Body	/
Response Body	JSON contenente una rappresentazione del servizio di infomobilità che è stato appena “ri-abilitata/disabilitata” all’interno del sistema SIM Backend

SIM-B-IS-07 [Get Available Mobility Vehicles]	
Verbo HTTP	GET
URI	/infomobility-services/{uuid}/vehicles?ssuuid={ssuuid}
Path Params	<p>uuid: stringa in formato UUIDv4 riferita al servizio di infomobilità di interesse</p> <p>ssuuid: stringa in formato UUIDv4 riferita alla Smart Station di interesse</p>
Query Params	/
Request Body	/
Response Body	JSON contenente un elenco di mezzi di mobilità disponibili per il service provider associato al servizio di infomobilità associato, nei pressi della Smart Station di riferimento

Anche in questo endpoint, sono stati aggiunti nuovi servizi:

SIM-B-IS-08 [List Infomobility Service for Admin]	
Verbo HTTP	GET
URI	/infomobility-services/getAllAuthorized
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista degli infomobility service gestibili da un ADMIN, presenti all'interno del sistema SIM Backend

SIM-B-IS-09 [List Infomobility by Smart Station]	
Verbo HTTP	GET
URI	/infomobility-services/associable-smart-stations/{uuid}
Path Params	uuid: stringa in formato UUIDv4 riferita alla smart station di interesse
Query Params	/
Request Body	/
Response Body	JSON contenente la lista degli infomobility services associabili ad una smart station da un ADMIN, presenti all'interno del sistema SIM Backend

SIM-B-IS-10 [Delete Infomobility Service]	
Verbo HTTP	DELETE
URI	/infomobility-services/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione dell' infomobility-service, antecedente all'operazione di cancellazione

Data l'aggiunta di nuovi endpoints nello svolgimento, riportiamo di seguito anche la documentazione per **Municipality**, **Authentication** e **AdminAuthorization**.

3.2.4 MUNICIPALITY ENDPOINT

Questo endpoint è referenziato dalla URI:

SIM_BACKEND_URI/municipality

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-M-01 [List Municipality]	
Verbo HTTP	GET
URI	/municipality
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle municipalità

SIM-B-M-02 [Create Municipality]	
Verbo HTTP	POST
URI	/municipality
Path Params	/
Query Params	/
Request Body	JSON contenente una rappresentazione di una municipalità da registrare all'interno del sistema SIM Backend
Response Body	JSON contenente una rappresentazione della nuova municipalità, successivamente all'operazione di persistenza

SIM-B-M-03 [Retrieve Municipality]	
Verbo HTTP	GET
URI	/municipality/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione della municipalità richiesta

SIM-B-M-04 [Update Municipality]	
Verbo HTTP	PUT
URI	/municipality/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	JSON contenente una rappresentazione di una municipalità da modificare
Response Body	JSON contenente una rappresentazione della municipalità, successivamente all'operazione di modifica

SIM-B-M-05 [Delete Municipality]	
Verbo HTTP	DELETE
URI	/municipality/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione della municipalità, antecedente all'operazione di cancellazione

SIM-B-M-06 [List Municipality for Admin]	
Verbo HTTP	GET
URI	/municipality/getAllAuthorized
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle municipalità gestibili da un ADMIN, presenti all'interno del sistema SIM Backend

3.2.5 AUTHENTICATION ENDPOINT

Questo endpoint è referenziato dalla URI:

SIM_BACKEND_URI/auth

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-A-01 [Log In Authentication]	
Verbo HTTP	POST
URI	/auth
Path Params	/
Query Params	/
Request Body	JSON contenente nome utente e password da verificare all'interno del sistema SIM Backend
Response Body	JSON contenente un token per la sessione dell'utente, successivamente all'operazione di persistenza

3.2.6 ADMIN AUTHORIZATION ENDPOINT

Questo endpoint è referenziato dalla URI:

SIM_BACKEND_URI/adminAuthorization

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-AA-01 [List Admin Authorization]	
Verbo HTTP	GET
URI	/adminAuthorization
Path Params	/
Query Params	/
Request Body	/
Response Body	JSON contenente la lista di tutte le autorizzazioni

SIM-B-AA-02 [Create Admin Authorization]	
Verbo HTTP	POST
URI	/adminAuthorization
Path Params	/
Query Params	/
Request Body	JSON contenente una rappresentazione di una autorizzazione da registrare per un ADMIN all'interno del sistema SIM Backend
Response Body	JSON contenente una rappresentazione della nuova autorizzazione per un ADMIN, successivamente all'operazione di persistenza

SIM-B-AA-03 [Retrieve Admin Authorization]	
Verbo HTTP	GET
URI	/adminAuthorization/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione dell'autorizzazione richiesta

SIM-B-AA-04 [Update Admin Authorization]	
Verbo HTTP	PUT
URI	/adminAuthorization/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	JSON contenente una rappresentazione di un' autorizzazione da modificare
Response Body	JSON contenente una rappresentazione dell' autorizzazione, successivamente all'operazione di modifica

SIM-B-AA-05 [Delete Admin Authorization]	
Verbo HTTP	DELETE
URI	/adminAuthorization/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	JSON contenente una rappresentazione dell'autorizzazione, antecedente all'operazione di cancellazione

SIM-B-AA-06 [List Authorization for Admin]	
Verbo HTTP	GET
URI	/municipality/user/{uuid}
Path Params	uuid: stringa in formato UUIDv4 riferita all'utente di interesse
Query Params	/
Request Body	/
Response Body	JSON contenente la lista delle autorizzazioni di un ADMIN, presenti all'interno del sistema SIM Backend

3.3 MODELLO DI DOMINIO DI SIM BACKEND

Il diagramma UML delle classi, riportato in Figura 2, mostra la logica di dominio concettuale del sotto-sistema SIM Backend.

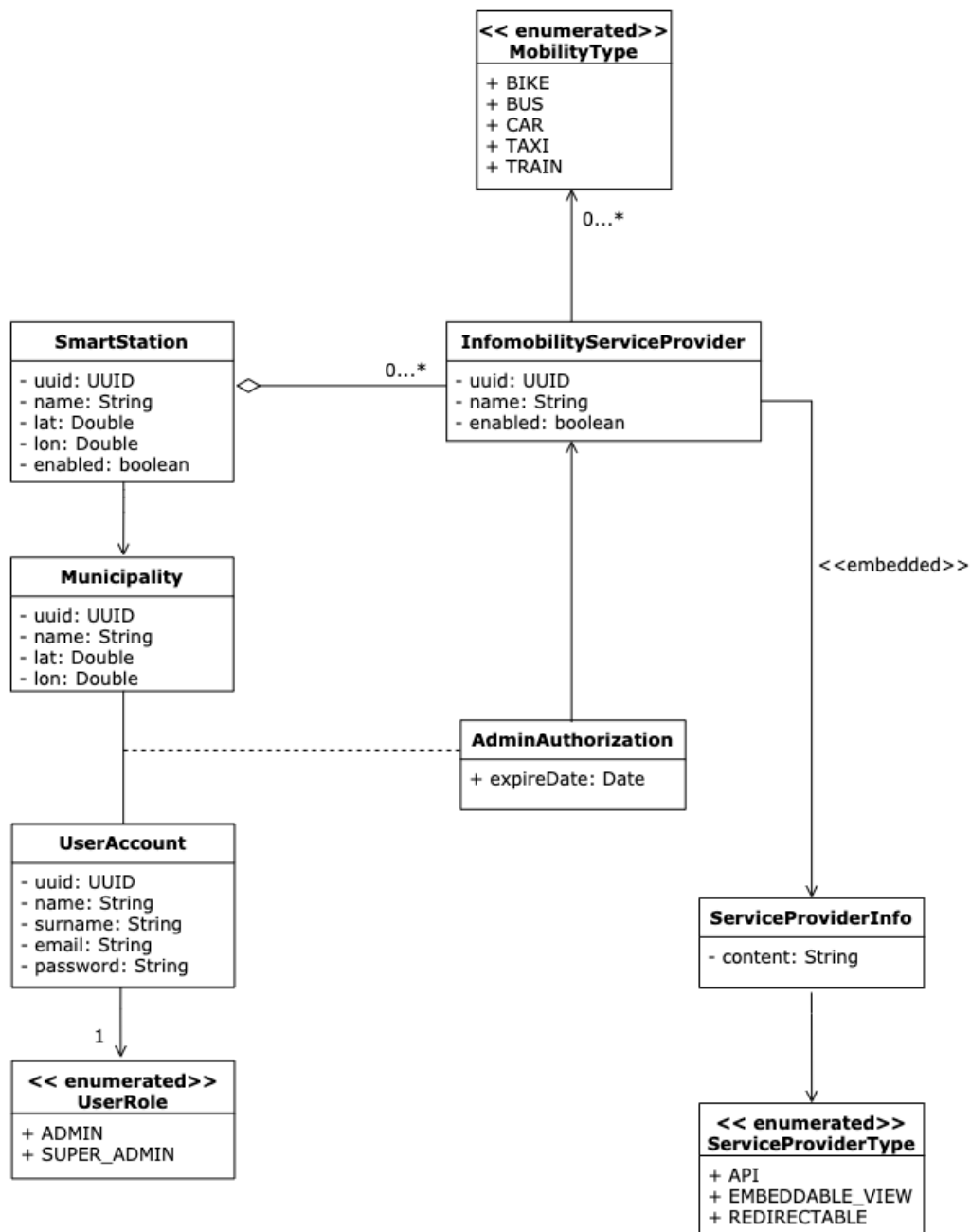


Figura 2: Diagramma UML delle classi relative al modello di dominio del sotto-sistema SIM Backend

3.3.1 ENTITÀ COINVOLTE

Il modello di dominio possiede varie entità di interesse.

UserAccount: entità che rappresenta un account utente, ossia di un utente identificabile negli attori SUPER ADMIN o ADMIN.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e auto-generato;
 - **email**, stringa human-readable, riferita all'indirizzo di posta elettronica dell'utente. Tale indirizzo verrà impiegato anche come "username" ai fini dell'autenticazione;
 - **password**, stringa cifrata secondo una codifica (e.g. MD5, SHA), riferita al "segreto" adottato dall'utente in fase di autenticazione;
 - **name**, stringa human-readable, riferita al nome anagrafico dell'utente;
 - **surname**, stringa human-readable, riferita al cognome anagrafico dell'utente.
-

UserRole: enumerazione che rappresenta un possibile ruolo utente.

L'enumerazione prevede questi valori:

- **SUPER_ADMIN**, che rappresenta l'omonimo attore;
 - **ADMIN**, che rappresenta l'omonimo attore;
-

AdminAuthorization: classe di associazione che definisce per ogni istanza di utente (i.e. UserAccount), abilitato ad operare su una o più istanze di municipalità (i.e. Municipality), quali fornitori di servizi di infomobilità è effettivamente autorizzato a gestire in configurazione.

Municipality: entità che rappresenta una municipalità, ossia un Comune.

Gli attributi di maggior rilievo:

- **uuid**, identificativo di sistema univoco e auto-generato;
 - **name**, stringa human-readable, riferita al nome geografico della municipalità;
 - **lat**, valore double, riferito alla coordinata di latitudine;
 - **lon**, valore double, riferito alla coordinata di longitudine.
-

Smart Station: entità che rappresenta una stazione intelligente, ossia una Smart Station abilitata a erogare servizi agli utenti.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e auto-generato;
 - **name**, stringa human-readable, riferita al nome ufficiale della stazione;
 - **lat**, valore double, riferito alla coordinata di latitudine;
 - **lon**, valore double, riferito alla coordinata di longitudine.
 - **enabled**, valore booleano, che indica a seconda del valore assunto se l'entità stazione risulta abilitata (true) o disabilitata (false) all'interno del sistema.
-

InfomobilityServiceProvider: entità che rappresenta un servizio di infomobilità (esterno), ossia un fornitore di servizi tra quelli scelti come fruibili dagli utenti in prossimità delle Smart Station visitate.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e autogenerato;
- **name**, stringa human-readable, riferita al nome ufficiale del fornitore di servizi;
- **enabled**, valore booleano, che indica a seconda del valore assunto se l'entità servizio di infomobilità risulta abilitato (true) o disabilitato (false) all'interno del sistema.

ServiceProviderInfo: entità che rappresenta un contenitore di informazioni fondamentali di un fornitore di servizi (esterni).

L'attributo di rilievo è:

- **content**, contenuto informativo fornito da un Service Provider.

ServiceProviderType: enumerazione che rappresenta i tipi di contenuto informativo.

L'enumerazione prevede, attualmente, questi valori:

- **ApiServiceProvider**, che rappresenta un provider i cui servizi sono consumabili tramite API (in questo caso, si rendono necessari alcuni componenti adattatori, esterni a SIM);
- **EmbeddableViewServiceProvider**, che rappresenta un provider i cui servizi sono offerti tramite incapsulamento di codice preesistente (e.g. tramite HTML o web view);
- **RedirectableServiceProvider**, che rappresenta un provider i cui servizi sono fruibili solamente tramite reindirizzamento su applicazioni esterne non integrabili direttamente (e.g. app mobili pubblicate in store specifici).

MobilityType: enumerazione che rappresenta una possibile tipologia di fornitore di servizi di infomobilità (i.e. InfomobilityServiceProvider).

L'enumerazione prevede questi valori:

- **BIKE**, che rappresenta un fornitore di servizi per mezzi di tipo bicicletta;
- **BUS**, che rappresenta un fornitore di servizi per mezzi di tipo autobus;
- **CAR**, che rappresenta un fornitore di servizi per mezzi di tipo macchina;
- **TAXI**, che rappresenta un fornitore di servizi per mezzi di tipo taxi;

- **TRAIN**, che rappresenta un fornitore di servizi per mezzi di tipo treno;
 - **POLY**, che rappresenta un fornitore di servizi per più tipologie di mezzi (e.g. bici e automobili).
-

3.4 CASI D'USO

In questa sezione vengono definiti i casi d'uso, che coinvolgono il sotto-sistema SIM Backend (si veda Figura 3).



Figura 3: Diagramma UML dei casi d'uso relativi al sotto-sistema SIM Backend

3.5 TECNOLOGIE ADOTTATE

Le tecnologie adottate sono le seguenti:

- **Java Enterprise Edition**, con le specifiche:
 - **CDI** (Context and Dependency Injection)
 - **JPA** (Java Persistence API)
 - **JAX-RS** (Java API for RESTful Web Services)
- **MySQL Server** come DBMS relazionale
- **JBoss Wildfly** come Application Server per il deployment
- **Maven** come dependencies manager
- Repository di sviluppo fornito da STLAB
- **Postman** per il test dei contratti dei dati di ogni servizio esposto da SIM Backend
- **Java JWT** per la generazione del token durante la fase di login dell'utente
- Framework **Angular 2+** con linguaggio **Typescript** per lo sviluppo frontend

3.6 IMPLEMENTAZIONE

Analizziamo adesso, a livello implementativo, i due elementi cardine del progetto:

3.6.1 FRONTEND

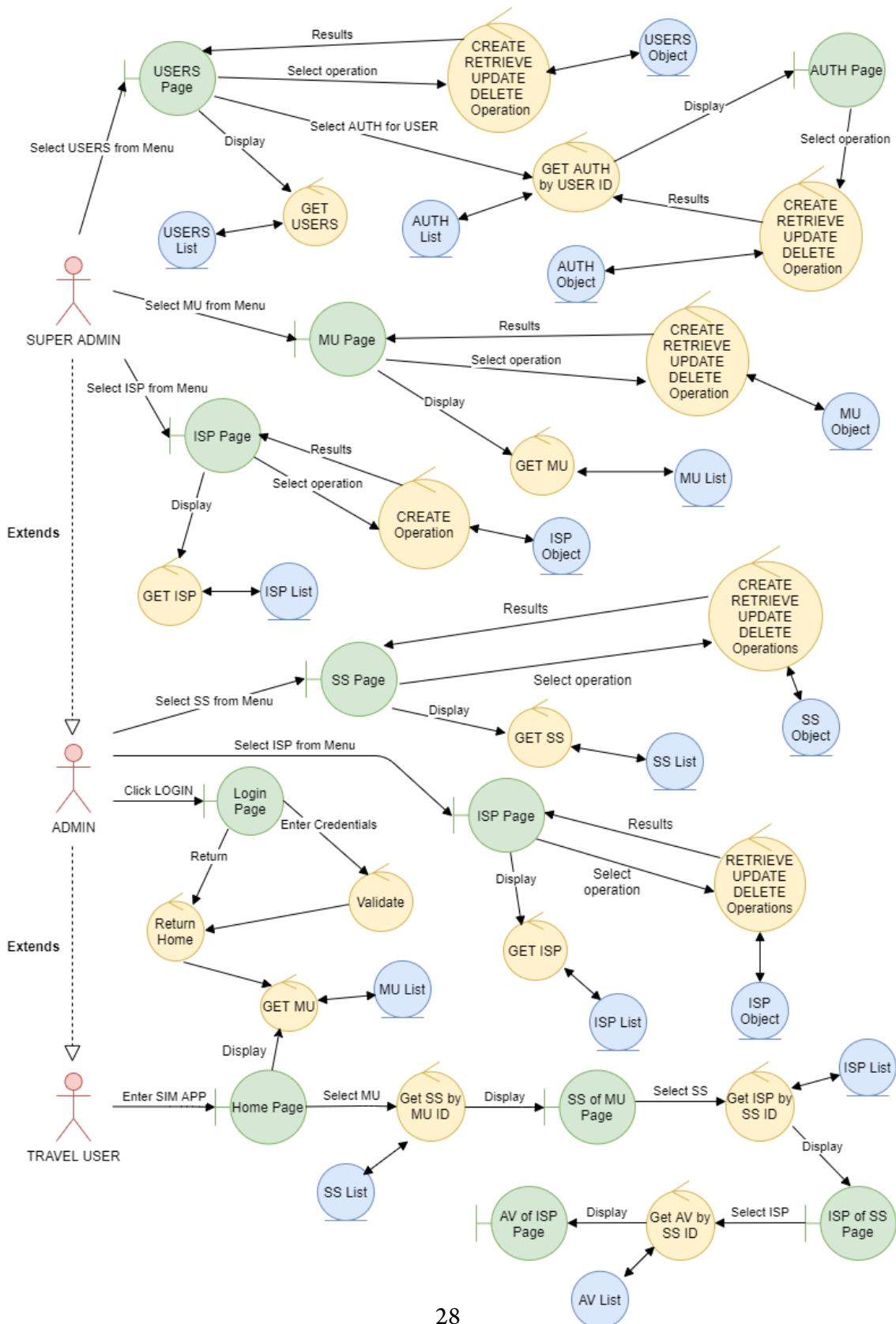


Figura 4: Diagramma di robustezza relativo alla progettazione del sistema Frontend

La figura 4 mostra un diagramma di robustezza che rappresenta tutte le possibili azioni effettuabili da un utente (a seconda del suo "grado") e la navigazione tra le pagine dell'app. Gli elementi dello schema sono stati associati a dei colori per differenziarne l'utilizzo:

- rosso per l'utente
- verde per la pagina web (boundary)
- blu per l'oggetto (entity)
- giallo per i controllori (controller)

3.6.2 BACKEND

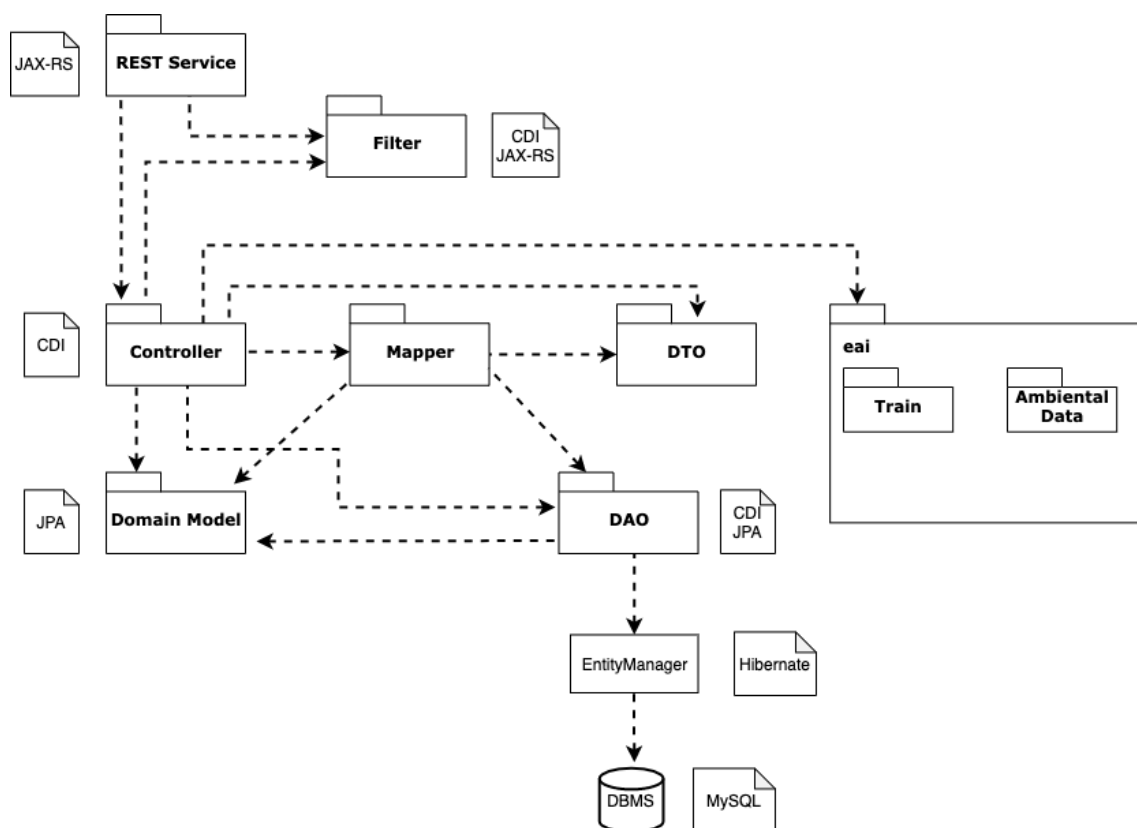


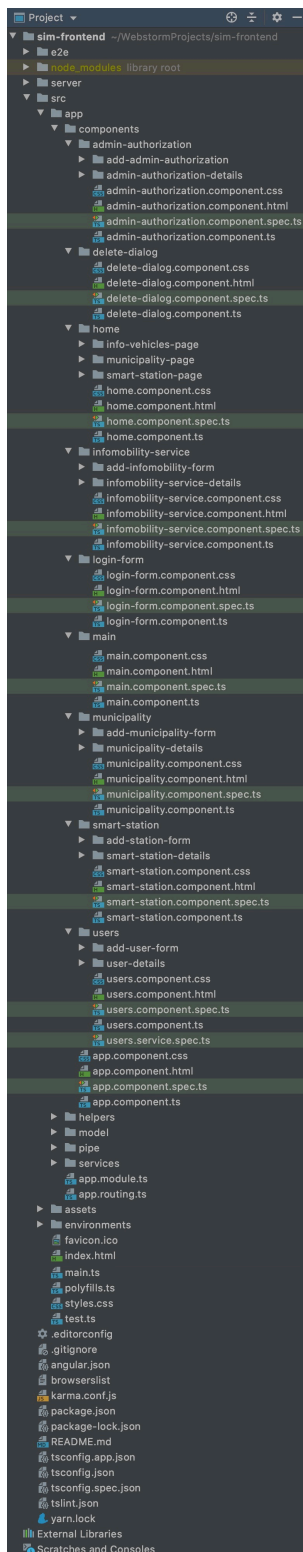
Figura 5: Diagramma UML dei package relativi al sotto-sistema SIM Backend

La figura 5 mostra uno schema logico/architetturale del backend:

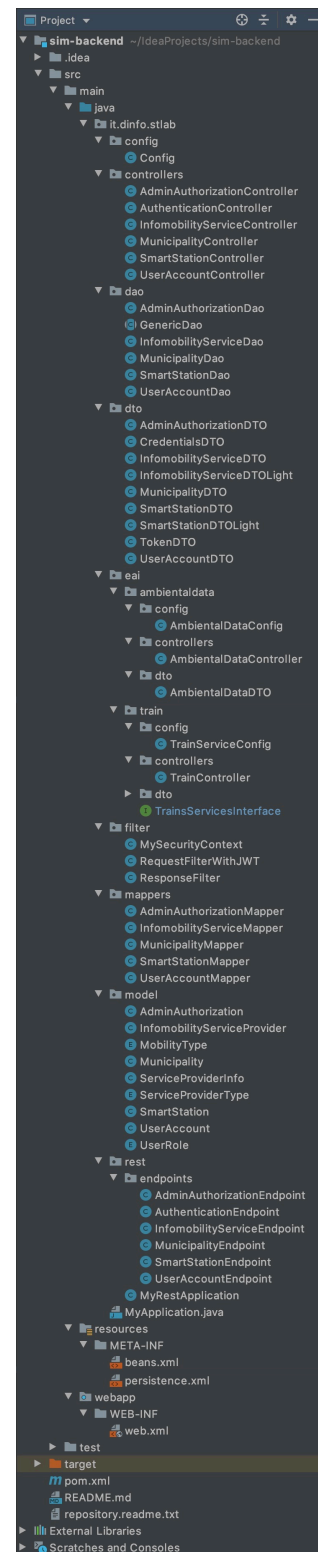
- **Rest Service:** package contenente tutti gli endpoint messi a disposizione dal BE. Per la loro creazione sono state utilizzate le specifiche di JAX-RS. Alcuni servizi degli endpoint sono protetti e possono essere usufruiti solamente da personale autorizzato (SUPER

ADMIN, ADMIN). La loro protezione è stata effettuata tramite tecniche di autenticazione(JWT) e di autorizzazioni basate sul ruolo(Intercettori di azioni tramite SecurityContext JAX-RS) e su delle condizioni da rispettare.

- **Filter:** contiene tutta una serie di impostazioni riguardanti le Request e le Response. In particolare si configurano le impostazioni per l'ACAO(access-control-origin-allow) e per la protezione degli endpoint.
- **Controller:** contiene una serie Controller in grado di assolvere i casi d'uso invocando od implementando direttamente i metodi caratterizzanti. Essi manipolano entità del Domain Model per intermediazione del DAO o direttamente, ed elaborano ed interpretano istanze di DTO, solitamente per intermediazione dei Mapper.
- **Mapper:** contiene una serie di mapper che si occupano di fare la conversione bidirezionale tra entità proprie del modello di dominio e oggetti di trasferimento (i.e. DTO).
- **DTO:** contiene una serie di DTO(Data Transfer Object) i quali rappresentano una versione "semplificata" o "adeguata" di oggetti riferiti al modello di dominio. Essi svolgono il ruolo di "trasportatore di dati" tra processi comunicanti(ie Front-end e Back-end).
- **eai -Enterprise Application Integration-**: contiene una serie di package inerenti all'utilizzo di servizi esterni. Vengono integrati i servizi esterni offerti da Stingray (trenti ed informazioni ambientali). Tramite JAX-RS client vengono contattati i servizi i quali rispondono con formato XML. Viene successivamente effettuata una conversione da XML a JSON tramite libreria Jackson ed inviato al frontend.
- **Domain Model:** contiene tutte le classi del modello di dominio. Si utilizza la specifica JPA con le annotazioni offerte per fare il mapping sul DB delle entità.
- **DAO:** contiene una serie di DAO(Data Access Object) che forniscono un'interfaccia astratta verso il livello di persistenza. Svolgono il ruolo di intermediario tra le entità del modello di dominio (in relazione 1-to-1) e le "tabelle reali" del database avvalendosi del supporto di un EntityManager. Quest'ultimo consente di effettuare delle operazioni sulle entità(creare, rimuovere e trovare) e interrogare entità mediante specifiche query utilizzando sia Java Persistence Query Language (JPQL) che normali query SQL.



(a) Struttura del codice SIM Frontend



(b) Struttura del codice di SIM Backend

Figura 6: Struttura Frontend e Backend

3.7 CODE SNIPPET

In questa sezione mostriamo alcuni parti di codice.

- Consumo lato front-end di un servizio offerto dal back-end [Fig. ??]

```
@JacksonXmlRootElement(localName = "DeparturesResponse") //Define root element name in XML. localName is the name of the XML root element
public class TrainsDeparturesResponseDTO {

    @JacksonXmlElementWrapper(localName = "Departures") //Define wrapper to use for collection types. localName is the name of the XML wrapper element
    @JacksonXmlProperty(localName = "Departure") //Define XML property, can be attribute or element. localName is the name of the XML element
    @JsonProperty("departures") //Nome di uscita in json. Altrimenti il nome sarebbe quello della variabile (departures)
    private List<Departure> departures;
    public List<Departure> getDepartures() { return departures; }

    public void setDepartures(List<Departure> departures) { this.departures = departures; }

    static class Departure {

        @JacksonXmlProperty(localName = "Cancelled")
        @JsonProperty("cancelled")
        private Boolean cancelled;
    }
}
```

(a) Frontend-consume update isp service

```
/**
 * @URI: SIM_BACKEND_URI/infomobility-services/{uuid}
 * @PathParams: uuid: stringa in formato UUIDv4 riferita all'infomobility di interesse
 * @RequestBody: JSON contenente una rappresentazione dell'infomobility service da modificare
 * @ResponseBody: JSON contenente l'infomobility service modificato */
@PUT
@Path("/{uuid}")
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
@RolesAllowed({"SUPER_ADMIN", "ADMIN"})
public Response update(@PathParam("uuid") String uuid, InfomobilityServiceDTO infomobilityDTOReceived) {
    infomobilityServiceController.update(uuid, infomobilityDTOReceived);
    InfomobilityServiceDTO dto = infomobilityServiceController.getById(uuid);
    return Response.status(Response.Status.OK).entity(dto).build();
}
```

(b) Backend- update isp service

Figura 7: Code Snippet 1

- Richiesta al servizio esterno di Stingray adattamento della risposta [Figure ??]


```

public TrainsDeparturesResponseDTO getDepartureInfo(String placeId) throws IOException {
    String path = TrainServiceConfig.TRAIN_SERVICE_ENDPOINT;

    //Si utilizza la specifica
    Client client = ClientBuilder.newClient();
    String xml = client.target(path)
        .path("GetDepartures")
        .queryParams( s: "PlaceId", placeId)
        .request(MediaType.APPLICATION_XML)
        .get(String.class);

    //JACKSON
    XmlMapper xmlMapper = new XmlMapper();
    xmlMapper.disable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES);
    TrainsDeparturesResponseDTO trainsDeparturesResponseDTO = xmlMapper.readValue(xml, TrainsDeparturesResponseDTO.class);
    return trainsDeparturesResponseDTO;
}

```

(a) Client request to STINGRAY service

```

@JacksonXmlRootElement(localName = "DeparturesResponse") //Define root element name in XML. localName is the name of the XML root element
public class TrainsDeparturesResponseDTO {

    @JacksonXmlElementWrapper(localName = "Departures") //Define wrapper to use for collection types. localName is the name of the XML wrapper element
    @JacksonXmlProperty(localName = "Departure") //Define XML property, can be attribute or element. localName is the name of the XML element
    @JsonProperty("departures") //Nome di uscita in json. Altrimenti il nome sarebbe quello della variabile (departures)
    private List<Departure> departures;
    public List<Departure> getDepartures() { return departures; }

    public void setDepartures(List<Departure> departures) { this.departures = departures; }

    static class Departure {

        @JacksonXmlProperty(localName = "Cancelled")
        @JsonProperty("cancelled")
        private Boolean cancelled;
    }
}

```

(b) Adapter XML response to JSON

Figura 8: Code Snippet 2

PANORAMICA DELL'APPLICAZIONE

In questo capitolo verranno mostrati alcuni screenshot dell'app che ne evidenziano le principali funzionalità, in riferimento ai casi d'uso (fig. 3) ed al diagramma di robustezza (fig. 4).

Un utente **viaggiatore** ha accesso unicamente alla home page nella quale può scegliere la municipalità (fig. 9), la stazione (fig. 10), il servizio (fig. 11) ed infine i veicoli disponibili (fig. 12), sfruttando anche un filtro di ricerca.

Il pulsante in alto a destra reindirizza alla pagina di login (figura 13), nella quale è possibile inserire le credenziali (email e password) per accedere come utente per la gestione. In particolare:

- Un **SUPER ADMIN** può visualizzare e modificare le stazioni, i servizi di infomobilità, le municipalità e gli utenti senza restrizioni. Per quest'ultimi può fornire delle autorizzazioni per limitarne il "raggio d'azione" (figura 14).
- Un **ADMIN** ha accesso alle pagine di smart stations e infomobility services. In entrambi i casi saranno mostrate solo quelle per le quali ha un'autorizzazione fornita da un SUPER ADMIN, così come le operazioni CRUD su di esse. (figura 15).

Le immagini 16 e 17 raffigurano rispettivamente i form per l'aggiornamento e la creazione di smart station, azioni estendibili anche agli altri elementi dell'applicativo (Infomobility Services, Municipality, Users, Authorization).

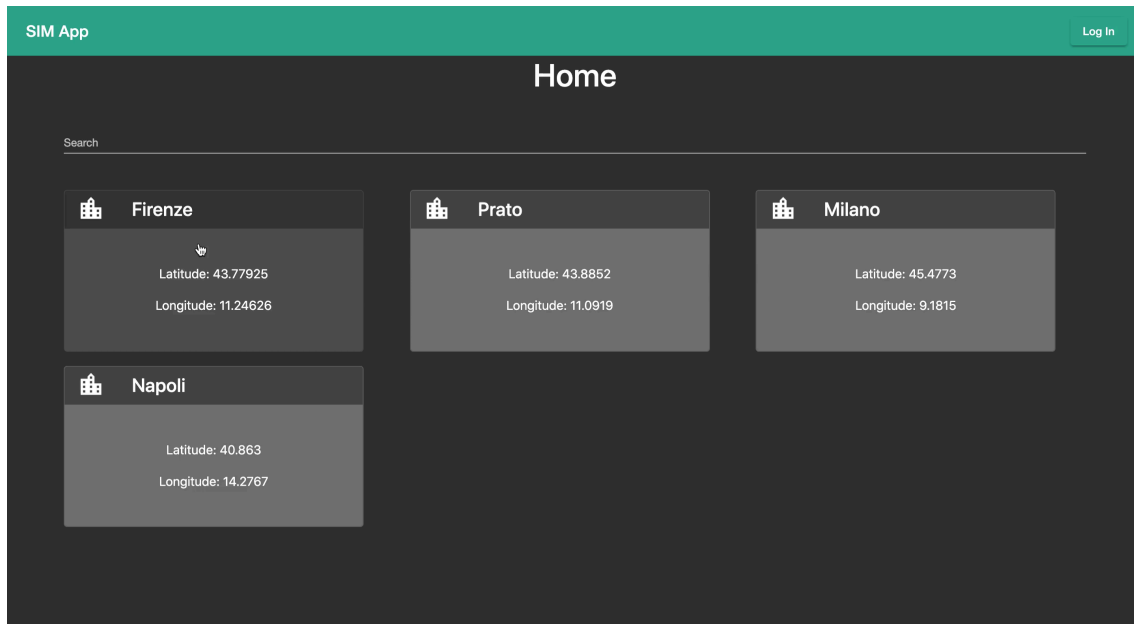


Figura 9: Homepage che mostra l'elenco di tutte le municipalità registrate nell'applicazione

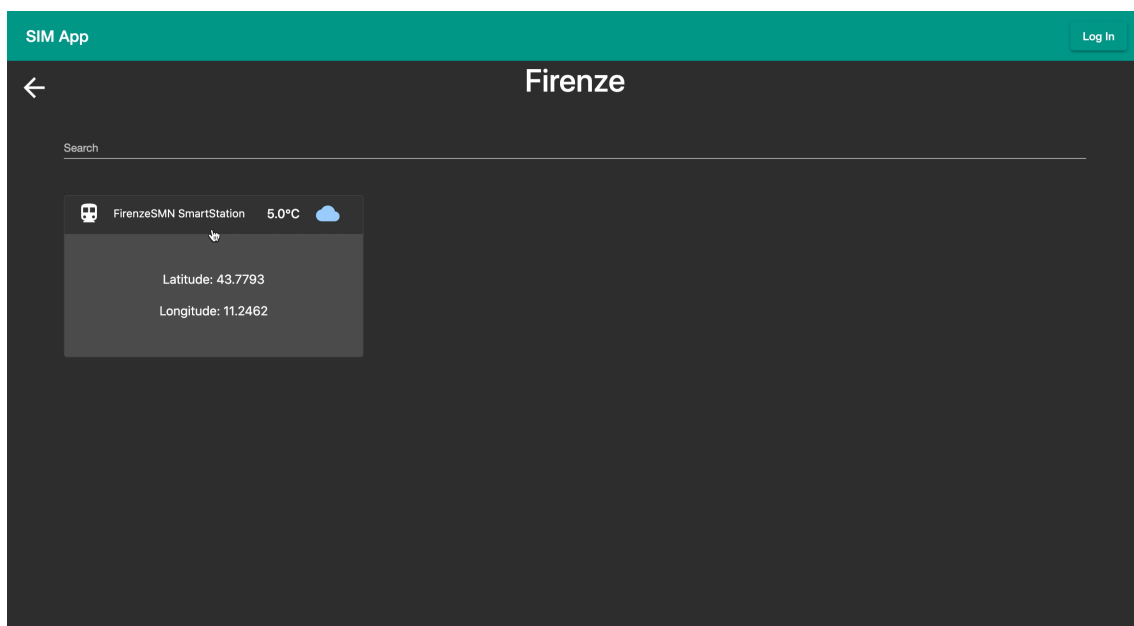


Figura 10: Stazioni intelligenti presenti in una municipalità

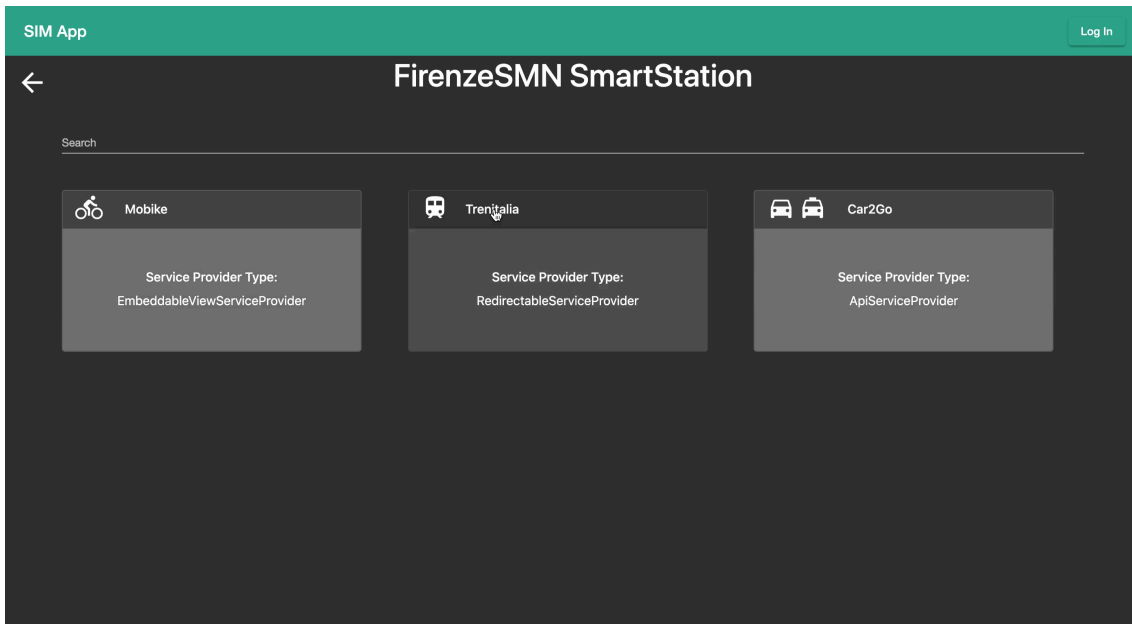


Figura 11: Servizi di infomobilità presenti in una stazione intelligente

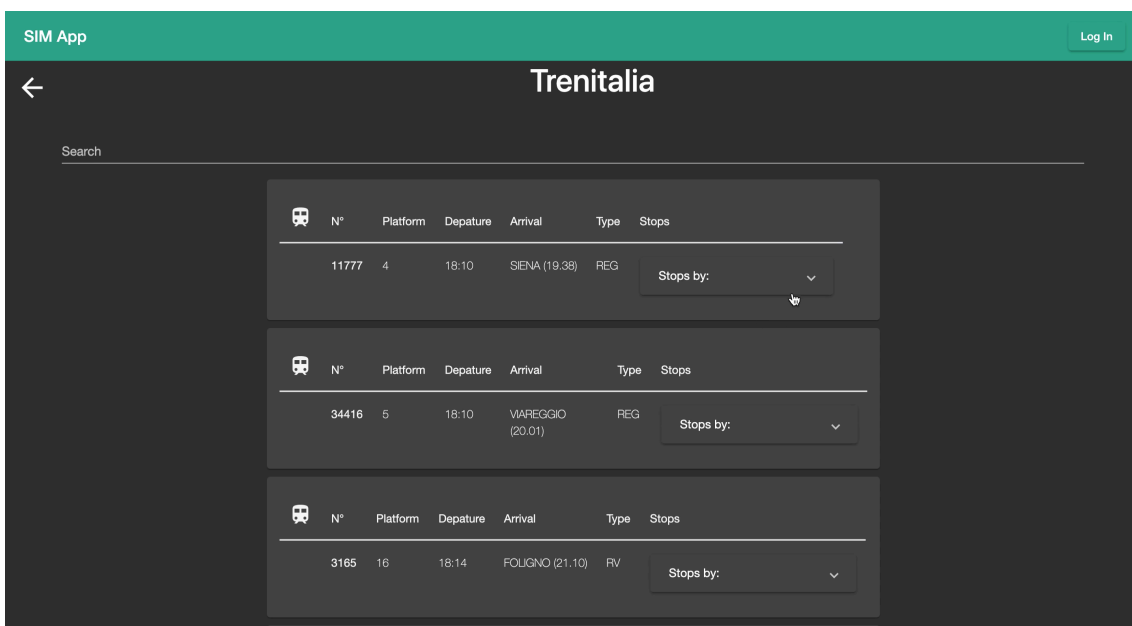


Figura 12: Veicoli disponibili di un servizio di infomobilità

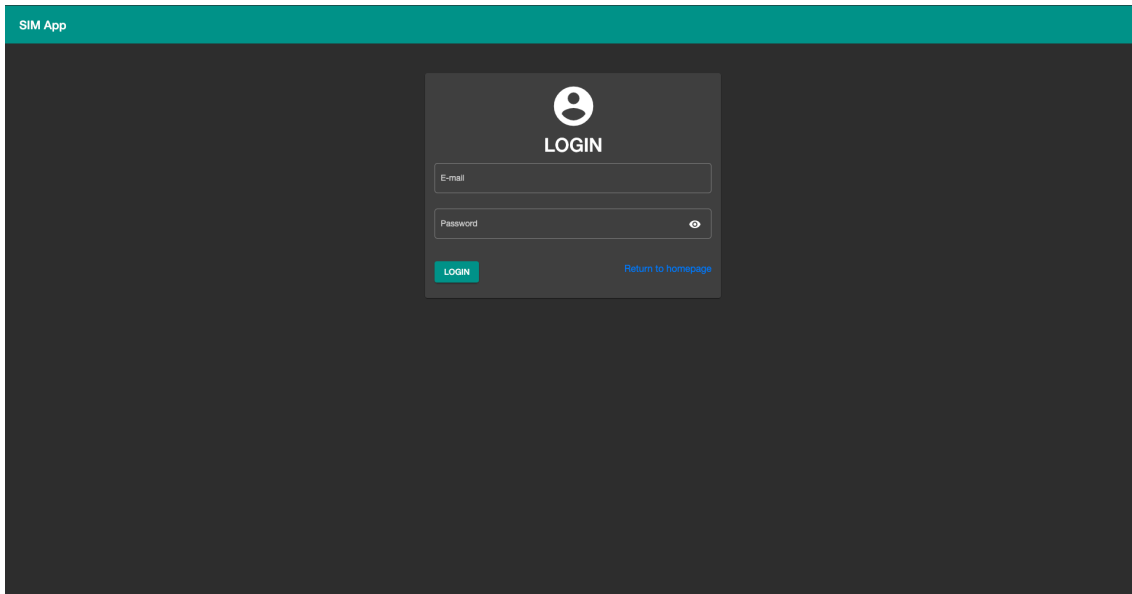


Figura 13: Pagina di login per utenti SUPER ADMIN o ADMIN

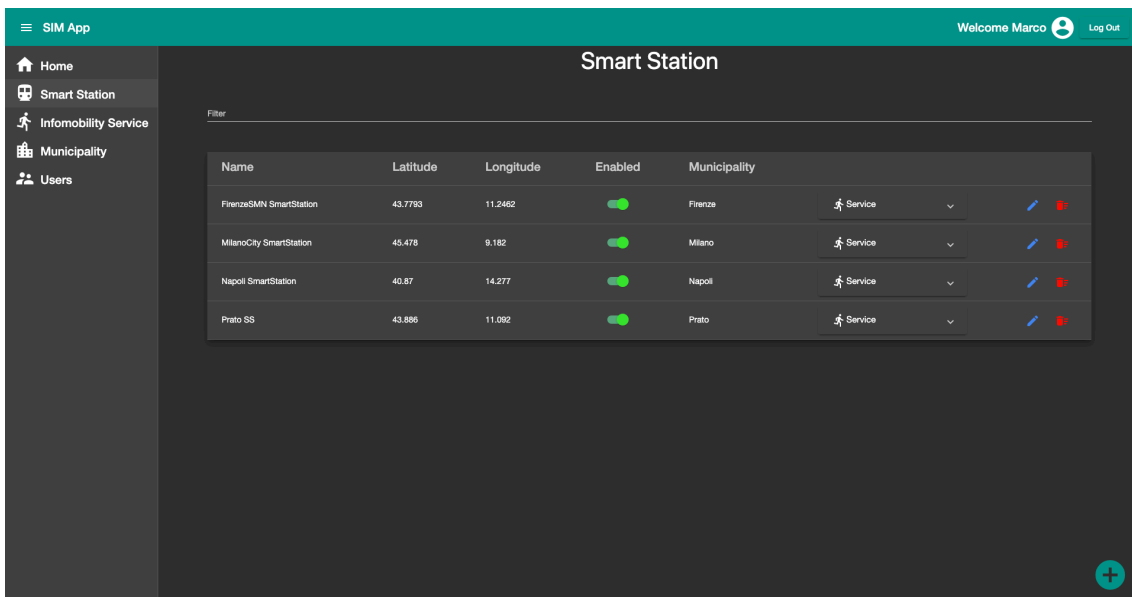


Figura 14: Pagina delle smart station vista come SUPER ADMIN

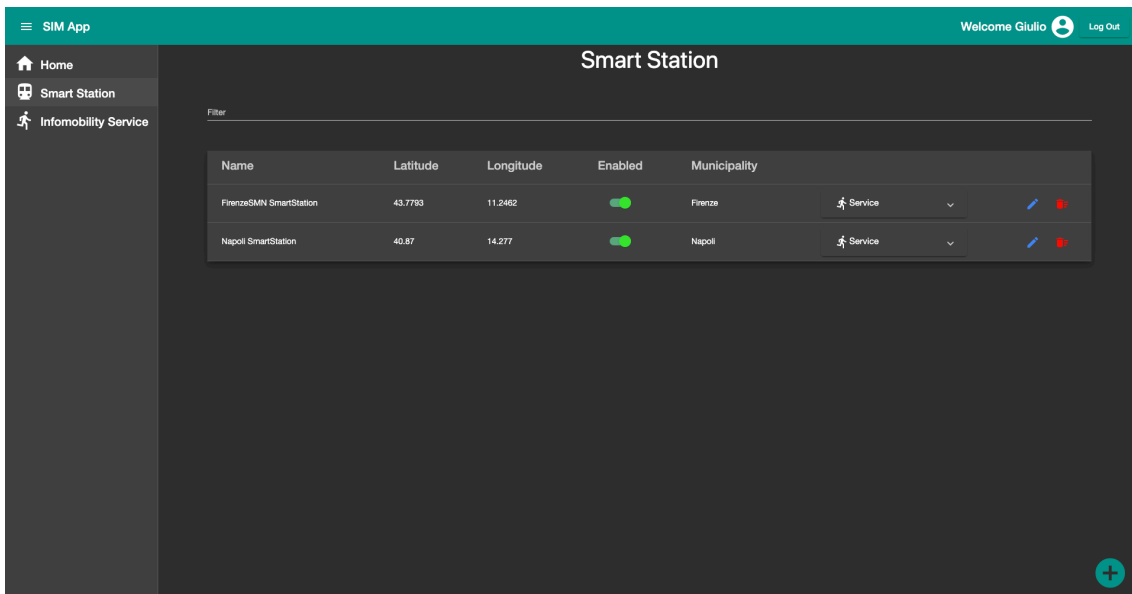


Figura 15: Pagina delle smart station viste come ADMIN

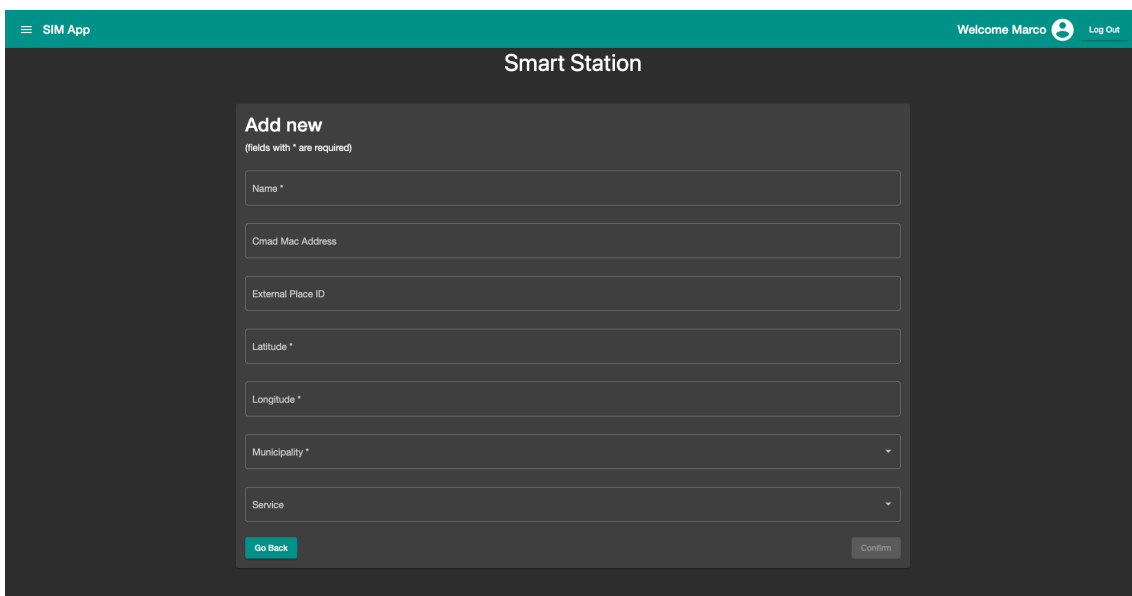



Figura 16: Pagina di aggiunta di un nuovo elemento

SIM App

Welcome Marco  Log Out

Smart Station

FirenzeSMN SmartStation

(fields with * are required)

Name *

FirenzeSMN SmartStation

Cmacd Mac Address

fffe000009f8

External Place ID

1325

Latitude *

43.7793

Longitude *

11.2462

Service

Mobike, Trenitalia, Car2Go

Municipality *

Firenze

Go Back

Confirm

Figura 17: Pagina di modifica di un elemento esistente

CONCLUSIONI

Questo elaborato si è focalizzato sulla realizzazione di un applicativo per l'accesso a servizi di mobilità e la possibilità, per chi è autorizzato, di gestire le strutture. Il risultato ottenuto è un backend con tutte le funzionalità prefissate ed alcuni miglioramenti aggiunti durante lo sviluppo. Uno dei più importanti è sicuramente la realizzazione di una web app lato client, che ci ha consentito di dare forma ad un progetto più completo.

L'esperienza è stata molto preziosa per noi, benchè non avessimo nessuna conoscenza pratica sull'argomento. Le nozioni teoriche del corso e l'aiuto da parte dei nostri collaboratori dell'STLAB sono serviti a farci comprendere come si realizza e gestisce effettivamente una applicazione web.

Nel complesso ci riteniamo molto soddisfatti del percorso svolto, poichè è servito a farci maturare professionalmente ed a destare un discreto interesse nei confronti di tale ambito.



APPENDICE

Progettazione di avvio

Elaborato SWAM

Obiettivo dell'elaborato: implementare tramite le Specifiche Java EE il sotto-sistema SIM Backend, rispettando la documentazione di progetto assegnata. NON è richiesto il Frontend.

1. Architettura di SIM Backend

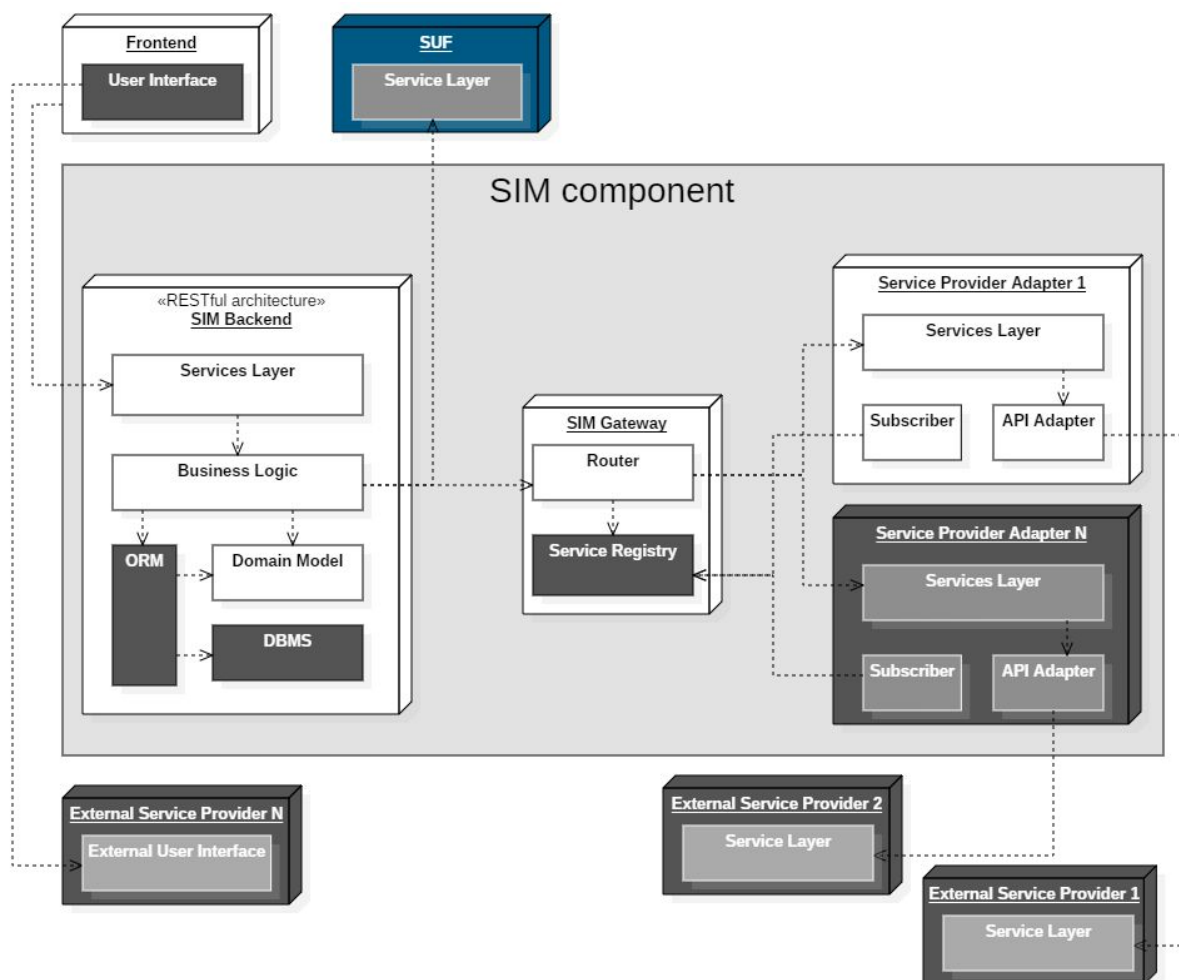


Fig. 1 - Diagramma di deployment UML, relativo all'architettura complessiva in cui si vede il sotto-sistema SIM Backend

Il componente SIM espone un insieme di servizi, nella forma di endpoint REST.

Tali endpoint sono tutti implementati all'interno del sotto-sistema denominato *SIM Backend* e sono collocati all'interno del suo *Services Layer*.

Tale sotto-sistema è stato progettato come architettura RESTful.

2. Endpoint di SIM Backend

I servizi esposti da *SIM Backend* sono consumabili direttamente dall'applicazione client di frontend, differenziandosi in base a condizioni di autenticazione.

Nel caso in cui un utente risulti:

- **autenticato**, equivale ad un utente con ruolo "SUPER ADMIN" o "ADMIN".
Tale utente è dotato di maggiori diritti rispetto agli utenti semplici non autenticati;
- **non autenticato**, allora equivale ad un utente senza alcun ruolo, ossia un viaggiatore che ha diritti di fruizione ridotti rispetto ad un amministratore.

Gli endpoint previsti sono i seguenti:

- **User Account Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo "account utente".
Tale endpoint predispone le funzionalità basilari per compiere casi d'uso CRUD (Create Retrieve Update Delete) su tali entità. Le funzionalità "in scrittura" di gestione degli account saranno permesse solamente a utenti con ruolo "SUPER ADMIN";
- **Smart Station Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo "stazione intelligente".
Tale endpoint predispone le funzionalità basilari per compiere casi d'uso CRUD (Create Retrieve Update Delete) su tali entità, permettendo anche agli utenti con ruolo "ADMIN" di associare l'elenco di servizi di infomobilità attivi per essa;
- **Infomobility Service Endpoint:** raccogliatore di servizi REST dedicati alla risorse di tipo "servizio esterno di infomobilità".
Tale endpoint predispone le funzionalità basilari per compiere casi d'uso CRUD (Create Retrieve Update Delete) su tali entità. Le specifiche funzionalità per la gestione delle configurazioni sono permesse a utenti con ruolo "ADMIN";

Nei sotto-paragrafi che seguono vengono descritti i dettagli di ogni servizio previsto.

Con la denominazione **SIM_BACKEND_URI**, si intende l'URI riferita al dominio applicativo del sotto-sistema SIM backend.

2.1 User Account Endpoint

Questo endpoint è referenziato dalla URI:

SIM_BACKEND_URI/users

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-USR-01 [List Users]	
Verbo HTTP	GET
URI	/users
Path Params	/
Query Params	/
Request Body	/
Response Body	documento JSON contenente la lista degli utenti dotati di account

SIM-B-USR-02 [Create User]	
Verbo HTTP	POST
URI	/users
Path Params	/
Query Params	/
Request Body	documento JSON contenente una rappresentazione di un account utente da registrare all'interno del sistema SIM Backend
Response Body	documento JSON contenente una rappresentazione del nuovo account utente, successivamente all'operazione di persistenza

SIM-B-USR-03 [Retrieve User]	
Verbo HTTP	GET
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	documento JSON contenente una rappresentazione dell'account utente richiesto

SIM-B-USR-04 [Update User]	
Verbo HTTP	PUT
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	documento JSON contenente una rappresentazione di un account utente da modificare
Response Body	documento JSON contenente una rappresentazione dell'account utente, successivamente all'operazione di modifica

SIM-B-USR-05 [Delete User]	
Verbo HTTP	DELETE
URI	/users/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	documento JSON contenente una rappresentazione dell'account utente, antecedente all'operazione di cancellazione

2.2 Smart Station Endpoint

Questo endpoint è referenziato dalla URI:

`SIM_BACKEND_URI/smart-stations`

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-SS-01 [List Smart Stations]	
Verbo HTTP	GET
URI	<code>/smart-stations</code>
Path Params	/
Query Params	/
Request Body	/
Response Body	documento JSON contenente la lista delle stazioni intelligenti presenti all'interno del sistema SIM Backend

SIM-B-SS-02 [Create Smart Station]	
Verbo HTTP	POST
URI	<code>/smart-stations</code>
Path Params	/
Query Params	/
Request Body	documento JSON contenente una rappresentazione di una stazione intelligente da creare, con l'elenco di servizi di infomobilità attivi per essa
Response Body	documento JSON contenente una rappresentazione della nuova stazione intelligente, successivamente all'operazione di persistenza

SIM-B-SS-03 [Retrieve Smart Station]	
Verbo HTTP	GET
URI	<code>/smart-stations/{uuid}</code>
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/
Response Body	documento JSON contenente la stazione intelligente richiesta

SIM-B-SS-04 [Update Smart Station]	
Verbo HTTP	PUT
URI	/smart-stations/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	documento JSON contenente una rappresentazione di una stazione intelligente da modificare, con l'eventuale indicazione dell'elenco di servizi di infomobilità attivi per essa
Response Body	documento JSON contenente la stazione intelligente, successivamente all'operazione di modifica

SIM-B-SS-05 [Enable/Disable Smart Station]	
Verbo HTTP	PATCH
URI	/smart-stations/{uuid}?enable={en}
Path Params	uuid: stringa in formato UUIDv4
Query Params	<p>en: booleano</p> <p>se assume valore:</p> <ul style="list-style-type: none"> • <code>true</code>, attiva il processo di ri-abilitazione • <code>false</code>, attiva il processo di disabilitazione <p>Si intuisce come una ri-abilitazione abbia senso solamente quando lo stato precedente della stazione risulti già disabilitato (e viceversa).</p>
Request Body	/
Response Body	documento JSON contenente una rappresentazione della stazione intelligente che è stata appena "ri-abilitata/disabilitata" all'interno del sistema SIM Backend

2.3 Infomobility Service Endpoint

Questo endpoint è referenziato dalla URI:

`SIM_BACKEND_URI/infomobility-services`

Ogni suo servizio sarà quindi esposto in subpath.

SIM-B-IS-01 [List Infomobility Services]	
Verbo HTTP	GET
URI	<code>/infomobility-services?ssuuid={uuid}</code>
Path Params	/
Query Params	uuid: stringa in formato UUIDv4 riferita alla Smart Station di interesse <ul style="list-style-type: none"> parametro opzionale che delimita la ricerca ai servizi di infomobilità attivi per la stazione intelligente associata all'id
Request Body	/
Response Body	documento JSON contenente la lista dei servizi di infomobilità disponibili all'interno del sistema SIM Backend

SIM-B-IS-02 [Create Infomobility Service]	
Verbo HTTP	POST
URI	<code>/infomobility-services</code>
Path Params	/
Query Params	/
Request Body	documento JSON contenente una rappresentazione di un servizio di infomobilità da creare
Response Body	documento JSON contenente una rappresentazione del nuovo servizio di infomobilità, successivamente all'operazione di persistenza

SIM-B-IS-03 [Retrieve Infomobility Service]	
Verbo HTTP	GET
URI	<code>/infomobility-services/{uuid}</code>
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	/

Response Body	documento JSON contenente una rappresentazione del servizio di infomobilità richiesto
----------------------	---

SIM-B-IS-04 [Update Infomobility Service]	
Verbo HTTP	PUT
URI	/infomobility-services/{uuid}
Path Params	uuid: stringa in formato UUIDv4
Query Params	/
Request Body	documento JSON contenente una rappresentazione di un servizio di infomobilità da modificare
Response Body	documento JSON contenente una rappresentazione del servizio di infomobilità, successivamente all'operazione di modifica

SIM-B-IS-05 [Enable/Disable Infomobility Service]	
Verbo HTTP	PATCH
URI	/infomobility-services/{uuid}?enable={en}
Path Params	uuid: stringa in formato UUIDv4
Query Params	<p>en: booleano</p> <p>se assume valore:</p> <ul style="list-style-type: none"> • true, attiva il processo di ri-abilitazione • false, attiva il processo di disabilitazione <p>Si intuisce come una ri-abilitazione abbia senso solamente quando lo stato precedente del servizio di infomobilità risulti già disabilitato (e viceversa).</p>
Request Body	/
Response Body	documento JSON contenente una rappresentazione del servizio di infomobilità che è stato appena "ri-abilitata/disabilitata" all'interno del sistema SIM Backend

SIM-B-IS-06 [Get Available Mobility Vehicles]	
Verbo HTTP	GET
URI	/infomobility-services/{uuid}/vehicles?ssuuid={ssuuid}
Path Params	uuid: stringa in formato UUIDv4

Query Params	ssuuid: stringa in formato UUIDv4 riferita alla Smart Station di interesse
Request Body	/
Response Body	documento JSON contenente un elenco di mezzi di mobilità disponibili per il service provider associato al servizio di infomobilità associato, nei pressi della Smart Station di riferimento

3. Modello di dominio di SIM Backend

Il diagramma UML delle classi, riportato in Figura 2, mostra la logica di dominio concettuale del sotto-sistema *SIM Backend*.

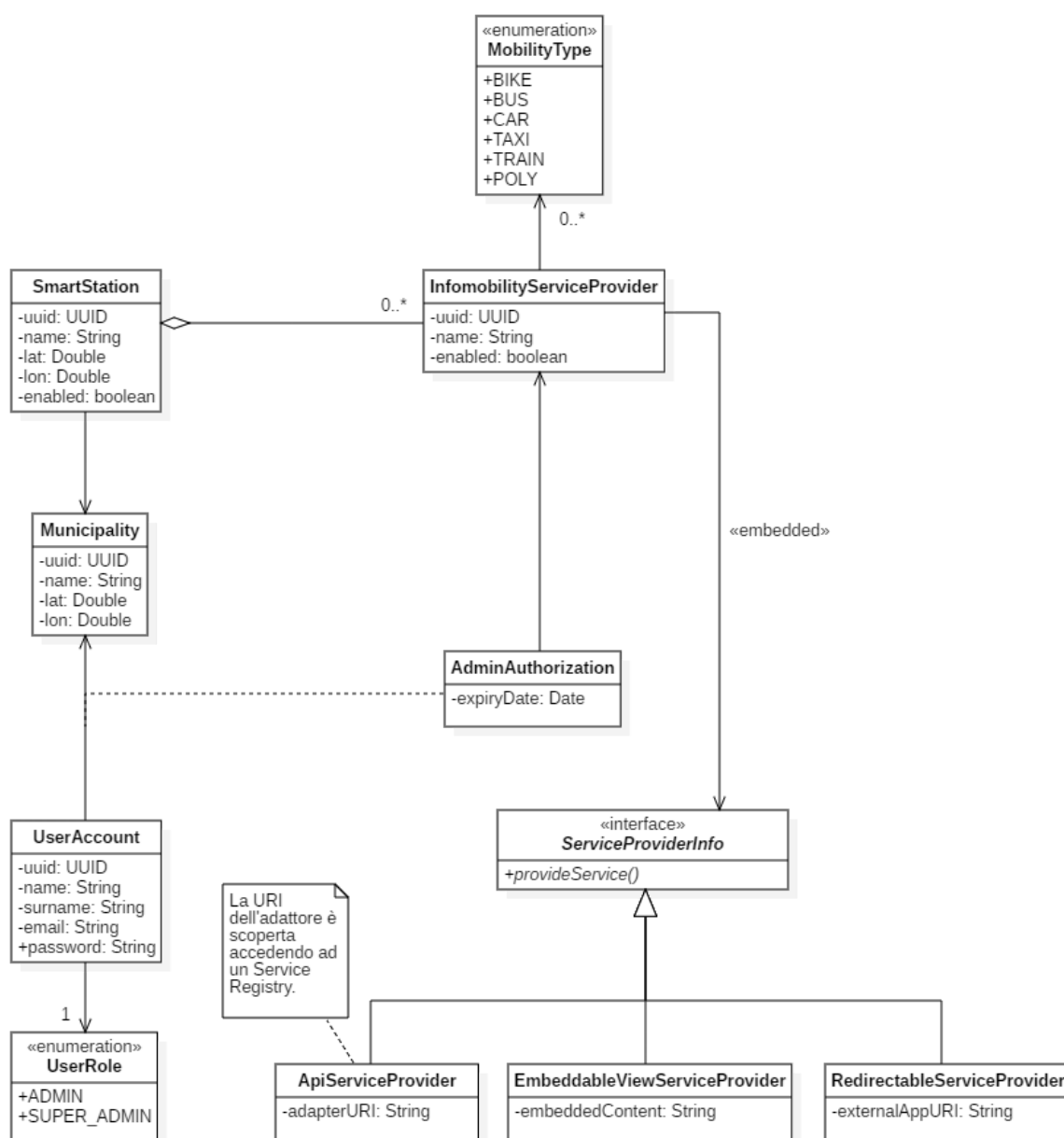


Fig. 2 - Diagramma UML delle classi relative al modello di dominio del sotto-sistema SIM Backend

3.1 Entità coinvolte

Il modello di dominio possiede varie entità di interesse.

UserAccount: entità che rappresenta un account utente, ossia di un utente identificabile negli attori SUPER ADMIN o ADMIN.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e autogenerato;
 - **email**, stringa *human-readable*, riferita all'indirizzo di posta elettronica dell'utente. Tale indirizzo verrà impiegato anche come "username" ai fini dell'autenticazione;
 - **password**, stringa cifrata secondo una codifica (e.g. MD5, SHA), riferita al "segreto" adottato dall'utente in fase di autenticazione;
 - **name**, stringa *human-readable*, riferita al nome anagrafico dell'utente;
 - **surname**, stringa *human-readable*, riferita al cognome anagrafico dell'utente.
-

UserRole: enumerazione che rappresenta un possibile ruolo utente.

L'enumerazione prevede, attualmente, questi valori:

- **SUPER_ADMIN**, che rappresenta l'omonimo attore;
 - **ADMIN**, che rappresenta l'omonimo attore.
-

AdminAuthorization: classe di associazione che definisce per ogni istanza di utente (i.e. *UserAccount*), abilitato ad operare su una o più istanze di municipalità (i.e. *Municipality*), quali fornitori di servizi di infomobilità è effettivamente autorizzato a gestire in configurazione.

Municipality: entità che rappresenta una municipalità, ossia un Comune.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e autogenerato;
 - **name**, stringa *human-readable*, riferita al nome geografico della municipalità;
 - **lat**, valore double, riferito alla coordinata di latitudine;
 - **lon**, valore double, riferito alla coordinata di longitudine.
-

SmartStation: entità che rappresenta una stazione intelligente, ossia una Smart Station abilitata a erogare servizi agli utenti.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e autogenerato;
 - **name**, stringa *human-readable*, riferita al nome ufficiale della stazione;
 - **lat**, valore double, riferito alla coordinata di latitudine;
 - **lon**, valore double, riferito alla coordinata di longitudine;
 - **enabled**, valore booleano, che indica a seconda del valore assunto se l'entità stazione risulta abilitata (*true*) o disabilitata (*false*) all'interno del sistema.
-

InfomobilityServiceProvider: entità che rappresenta un servizio di infomobilità (esterno), ossia un fornitore di servizi tra quelli scelti come fruibili dagli utenti in prossimità delle Smart Station visitate.

Gli attributi di maggior rilievo sono:

- **uuid**, identificativo di sistema univoco e autogenerato;
 - **name**, stringa *human-readable*, riferita al nome ufficiale del fornitore di servizi;
 - **enabled**, valore booleano, che indica a seconda del valore assunto se l'entità servizio di infomobilità risulta abilitato (*true*) o disabilitato (*false*) all'interno del sistema.
-

ServiceProviderInfo: interfaccia che rappresenta un contenitore di informazioni fondamentali di un fornitore di servizi (esterni).

Le implementazioni previste per questa interfaccia sono:

- **ApiServiceProvider**, che rappresenta un provider i cui servizi sono consumabili tramite API (in questo caso, si rendono necessari alcuni componenti adattatori, esterni a *SIM Backend*);
 - **EmbeddableViewServiceProvider**, che rappresenta un provider i cui servizi sono offerti tramite incapsulamento di codice preesistente (e.g. tramite HTML o web view);
 - **RedirectableServiceProvider**, che rappresenta un provider i cui servizi sono fruibili solamente tramite reindirizzamento su applicazioni esterne non integrabili direttamente (e.g. app mobili pubblicate in store specifici).
-

MobilityType: enumerazione che rappresenta una possibile tipologia di fornitore di servizi di infomobilità (i.e. *InfomobilityServiceProvider*).

L'enumerazione prevede, attualmente, questi valori:

- **BIKE**, che rappresenta un fornitore di servizi per mezzi di tipo bicicletta;
- **BUS**, che rappresenta un fornitore di servizi per mezzi di tipo autobus;

- **CAR**, che rappresenta un fornitore di servizi per mezzi di tipo automobile;
 - **TAXI**, che rappresenta un fornitore di servizi per mezzi di tipo taxi;
 - **TRAIN**, che rappresenta un fornitore di servizi per mezzi di tipo treno;
 - **POLY**, che rappresenta un fornitore di servizi per più tipologie di mezzi (e.g. bici e automobili).
-

4. Casi d'uso

In questa sezione vengono definiti i casi d'uso, che coinvolgono il sotto-sistema *SIM Backend* (si veda Figura 3).

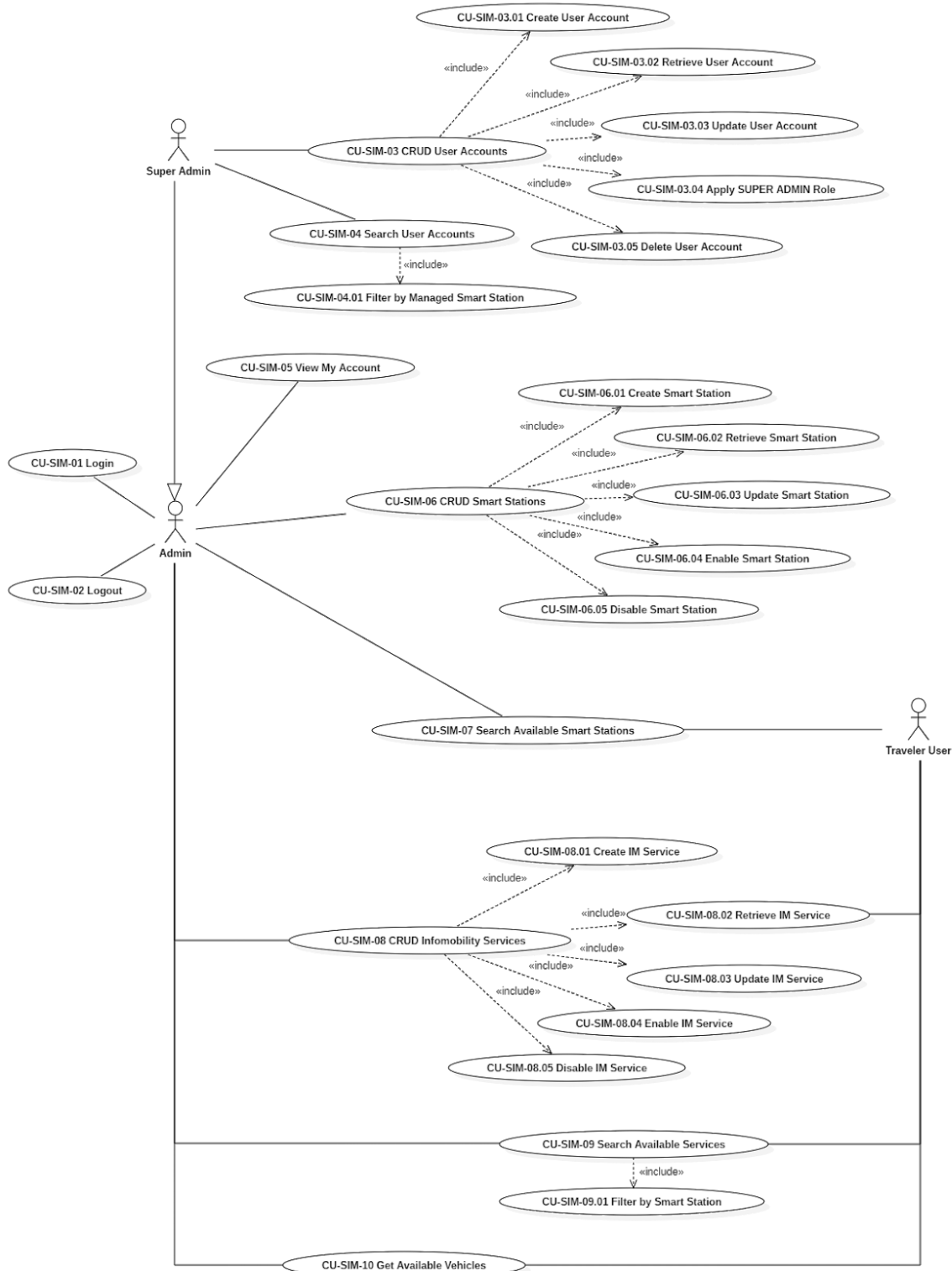


Fig. 3 - Diagramma UML dei casi d'uso relativi al sotto-sistema *SIM Backend*

5. Tecnologie da adottare

Le tecnologie da adottare sono le seguenti:

- **Java Enterprise Edition** (a seguire le Specifiche utili)
 - **CDI** (Contexts and Dependency Injection)
 - **JPA** (Java Persistence API)
 - **JAX-RS** (Java API for RESTful Web Services)
- **MySQL Server** come DBMS relazionale
- **JBoss Wildfly** come Application Server per il deployment
- **Maven** come dependencies manager
- Il repository di sviluppo è quello fornito da STLAB, al quale dovrete interfacciarvi tramite **GIT**
- **Postman** per il test dei contratti dei dati di ogni servizio esposto da SIM Backend

5.1 Tecnologie per JWT

Per quanto riguarda il JWT vi diamo qualche dritta nelle varie prospettive:

- **prospettiva teorica**, se non avete chiaro in mente come funzioni il processo di autenticazione JWT, potete leggere <https://jwt.io/>
Sinteticamente, il workflow generale tra client e server è il seguente.
 - (log-in) si invia una richiesta HTTP contenente username e password
 - si riceve un token dal Backend
 - salvando tale token in un sistema di data storage lato client
 - e.g. Local Storage, Session Storage, IndexedDB, etc.
 - si inoltra tale token in ogni successiva richiesta HTTP
 - all'interno dell'header HTTP `Authorization`
 - (log-out) si rimuove il token dallo storage
- **prospettiva di backend**, come avrete capito, il backend lavora su 2 aspetti principali:
 - Al termine della fase di log-in il backend deve generare un JWT valido e comunicarlo al frontend nel body della response riferita al servizio rest di login.
 - Nelle richieste successive, per identificare, autenticare e autorizzare gli utenti, il backend deve leggere il token inviato nell'header dal client, validarne l'integrità e interpretare i dati contenuti nel JSON dei claims.

- ciò può essere fatto implementando l'interfaccia `ContainerRequestFilter` e definendo il metodo `filter()` (SCREEN-SHOT 1)
- quindi potete distinguere quale risorsa/servizio state contattando, al fine di non comportarsi nell'implementazione del metodo `filter()` sempre allo stesso modo (SCREEN-SHOT 2)
- quindi estraete il token JWT dall'header della request e verificatelo come spiegato al link (<https://github.com/auth0/java-jwt>) (SCREEN-SHOT 3 - ma con modifiche)
- In sintesi, in Java per implementare l'algoritmo di generazione del token, la validazione e l'interpretazione potreste adottare questa libreria:
 - <https://github.com/auth0/java-jwt>
- **prospettiva di frontend**, in Angular potete salvare il token JWT che ricevete dalla response del Login nel Local storage (e ri-leggerlo da lì). Potete poi usare un HTTP Interceptor di Angular per decorare l'header delle richieste con questo token (se presente nel Local storage, altrimenti gestite il routing verso la pagina di login).
 - guardate (SCREEN-SHOT 4) per fare il vostro intercettore
 - guardate (SCREEN-SHOT 5) per implementare un servizio che legge dal Local storage e decodifica i claims
 - **Nota:** attenzione perché modificare l'header di una request in angular non è permesso (sono **immutabili**), va quindi clonato l'header per fare ciò (troverete diverse informazioni in rete).

```
23  @Provider
24  @Priority(Priorities.AUTHENTICATION)
25  public class SecurityRequestFilter implements ContainerRequestFilter {
26
27      @Context
28      UriInfo uriInfo;
29
30
31      @Inject
32      UserDao userDao;
33
34
35      @Override
36  public void filter(ContainerRequestContext requestContext) throws IOException {
```

Screen-shot 1

```
// PUBLIC SERVICES
String uriPath = requestContext.getUriInfo().getPath();
if(uriPath.startsWith("/users/login")) {
    if(StringUtils.isBlank(authorizationHeader))
        return;
    else
        requestContext.abortWith(Response.status(Response.Status.BAD_REQUEST).build());
}
if(uriPath.startsWith("/test")) {
    return;
}
```

Screen-shot 2

```
52
53 // AUTHENTICATED SERVICES
54 String principal;
55 if(authorizationHeader != null) {
56
57     String token = authorizationHeader.substring("Bearer".length()).trim();
58     try {
59         Key key = KeyGenerator.generateKey();
60         principal = Jwts.parser().setSigningKey(key).parseClaimsJws(token).getBody().getSubject();
61         cambia di sicuro il modo in cui si verifica e interpreta
62
63         requestContext.setSecurityContext(new SecurityContext() {
64
65             @Override
66             public Principal getUserPrincipal() {
67                 return new Principal() {
68                     @Override
69                     public String getName() { return principal; }
70                 };
71             }
72
73             @Override
74             public boolean isUserInRole(final String role) {
75                 return (userDao.getUserRole(principal).toString().equals(role));
76                 //return false;
77             }
78
79             @Override
80             public boolean isSecure() {
81                 return requestContext.getSecurityContext().isSecure();
82                 //return false;
83             }
84
85             @Override
86             public String getAuthenticationScheme() {
87                 return SecurityContext.BASIC_AUTH;
88                 //return null;
89             }
90         });
91     } catch (Exception e) {
92         // Imposto l'utente nel contesto per riprenderlo con
93         // @Context SecurityContext securityContext;
94         // User user = userDao.findByIdNumber(securityContext.getUserPrincipal().getName());
95         requestContext.setProperty("user", userDao.findByIdNumber(principal));
96         boolean test = requestContext.getSecurityContext().isUserInRole("ADMIN");
97         return;
98     }
99 }
```

Screen-shot 3

```

1  import { JwtStorageService } from '../storages/jwt-storage.service';
2  import { Observable } from 'rxjs';
3  import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';
4  import { Injectable } from '@angular/core';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class JwtInterceptorService implements HttpInterceptor {
10
11    constructor(private jwtStorageService: JwtStorageService) {}
12
13    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
14
15      console.log("JWT Interceptor: ", this.jwtStorageService.read());
16
17      request = request.clone({
18        setHeaders: {
19          Authorization: `Bearer ${this.jwtStorageService.read()}`
20        }
21      });
22
23      return next.handle(request);
24    }
25  }

```

Screen-shot 4

```

1  import { Injectable } from '@angular/core';
2  import * as jwt_decode from 'jwt-decode';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class JwtStorageService {
8    static JWT_KEY: string = 'jwt';
9
10   constructor() {}
11
12   store(jwt: string): void {
13     localStorage.setItem(JwtStorageService.JWT_KEY, jwt);
14   }
15
16   read(): string {
17     return localStorage.getItem(JwtStorageService.JWT_KEY);
18   }
19
20   remove(): void {
21     localStorage.removeItem(JwtStorageService.JWT_KEY);
22   }
23
24   getUsername(): string {
25     var decoded = jwt_decode( this.read() );
26     console.log('jwt decoded', decoded);
27     return decoded['username'];
28   }
29
30   getRole(): string {
31     var decoded = jwt_decode( this.read() );
32
33     return decoded ? decoded['role'] : undefined;
34   }
35 }
36

```

Screen-shot 5

RIFERIMENTI BIBLIOGRAFICI

- [1] E.Vicario, S.Mattolini, J.Parri, S.Sampietro *Slides per il corso di Software Architecture and Methodologies*.
- [2] Christian Bower, Gavin King, *Java Persistence with Hibernate*. Casa editrice Manning.
- [3] Oracle, *Java EE*, <https://javaee.github.io/javaee-spec/javadocs/>
- [4] Google, *Angular*, <https://angular.io/>
- [5] Google, *Angular Material*, <https://material.angular.io/>
- [6] Greg Lim, *Beginning with Angular 2 with Typescript*. Prima Edizione febbraio 2017.