OMI Finder

D3: Component Diagram, Class Diagram, OCL

Corso di Ingegneria del Software [145829]

Conti Marco	182438	[marco.conti-1@studenti.unitn.it]
Mongera Davide	209273	[davide.mongera@studenti.unitn.it]
Pugliese Simone	209036	[simone.pugliese@studenti.unitn.it]
Tomaselli Elia	209613	[elia.tomaselli@studenti.unitn.it]

University of Trento
Via Sommarive 14, 80123
Povo (TN), Italy

Diagramma di contesto **3**

Analisi dei Componenti **4**
Definizione dei componenti 4
Database 4
Web Server [Backend] 4
Web Server [Frontend] 4
Downloader 5
API Service 5
Tabella riassuntiva dei livelli di coesione dei componenti 5
Diagramma dei componenti 6
Database 6
Web Server [Backend] 6
Web Server [Frontend] 7
Downloader 7
API Service 7
Livello di accoppiamento dei componenti 8
Discussione livello di accoppiamento dei componenti 8
Database - Web Server [Backend] – Livello Data 8
Database - Downloader – Livello Common 8
Database - API Service – Livello Stamp 8
Web Server [Backend] - Web Server [Frontend] – Livello Data 8
Web Server [Backend] - API Service – Livello Stamp 9

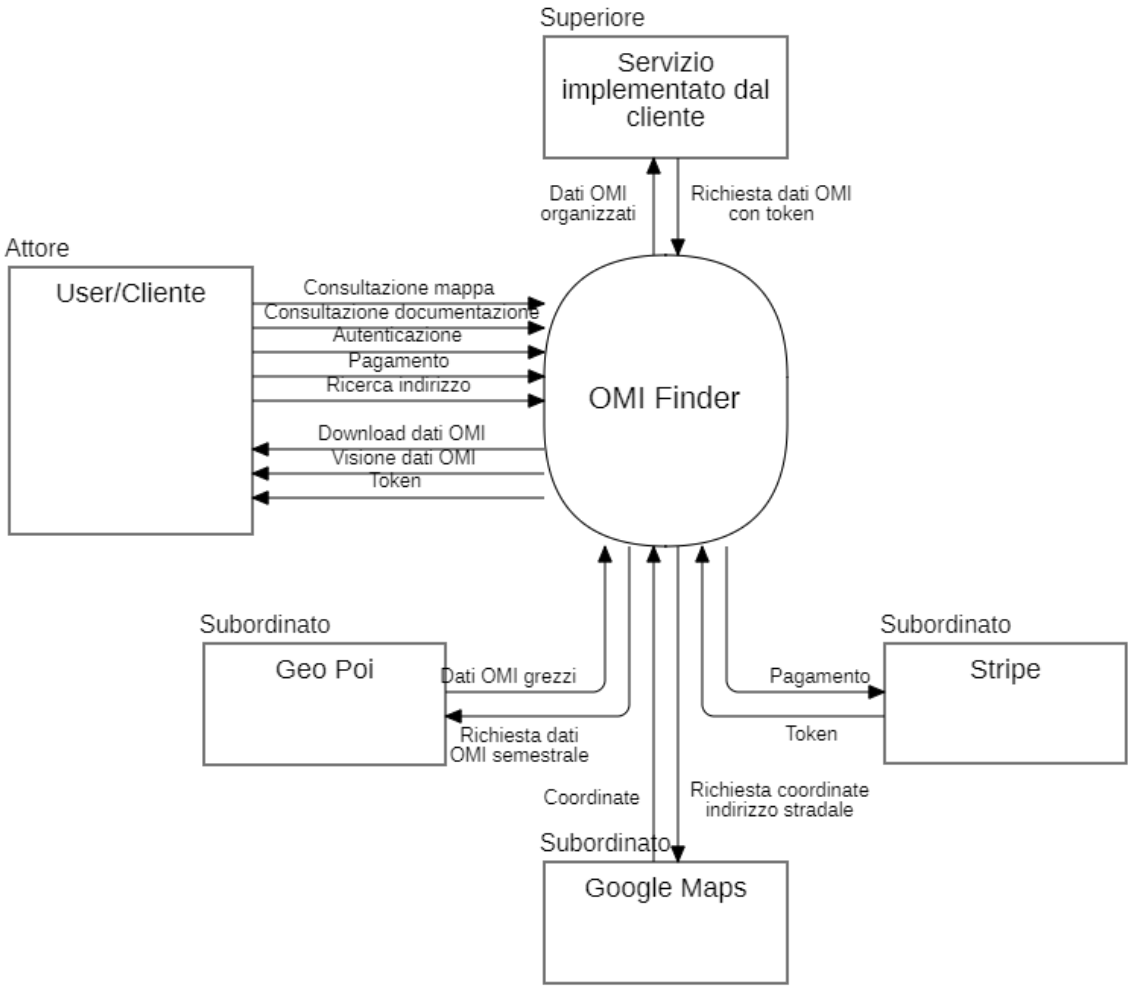
Diagramma delle classi **10**
Diagramma delle classi complessivo 10
Downloader 11
Zona OMI 12
API Service 13
Backend 14
Frontend 15
Utente e sottoclassi 16

Codice in Object Constraint Language **17**
Frontend 17
Backend 18

Diagramma delle classi con codice OCL **19**

Diagramma di contesto

La figura sottostante mostra le principali interazioni tra attori o sistemi esterni e l'applicazione OMI Finder. Prendendo in considerazione le diverse funzionalità ed interazioni del sistema, verranno presentati successivamente i relativi diagrammi dei componenti e delle classi necessari alla realizzazione del progetto.



Analisi dei Componenti

In seguito verranno spiegati i diversi componenti del sistema OMI Finder che sono stati individuati sulla base dei requisiti funzionali e non funzionali, e del diagramma di contesto. Tutti i componenti sono stati pensati per ottenere il grado di coesione più basso possibile. Le interazioni con sistemi esterni ad OMI Finder sono state generalizzate con delle porte che rappresentano la comunicazione con tali sistemi esterni.

Definizione dei componenti

In questa sezione vengono definiti i componenti, insieme ai loro livelli di coesione.

Database

Motivazione. Dati i RF2, RF3, RF5, RF9, il software deve poter interrogare un database con all'interno sia i dati delle zone OMI che i dati utente, con il possibile token di avvenuto pagamento. È stato quindi identificato un componente Database, che si interfaccerà con il Downloader, API Service e il Web Service, gestendo tutte le richieste di dati.

Coesione: livello **7** - Funzionale
Spiegazione: tutti gli elementi del componente sono indipendenti da quelli degli altri componenti.

Web Server [Backend]

Motivazione. Dati i RF2, RF6, e RF9, il software deve poter filtrare i dati su richiesta, oltre ad autorizzare il login utente e verificare l'avvenuto pagamento del pacchetto Premium. È stato quindi identificato un componente Web Server [Backend], che si interfaccerà con il componente Web Server [Frontend], Database e API Service per gestire le richieste specifiche.

Coesione: livello **6** - Informazionale
Spiegazione: per alcuni elementi è necessario richiedere dati al Database, mentre la restante rimane indipendente da quelli degli altri componenti.

Web Server [Frontend]

Motivazione. Dati i RF8, RF10 e RF11, il software deve fornire un'interfaccia web utente che permetta singole query ad utenti che si collegano alla piattaforma. È stato quindi identificato un componente Web Server per il Frontend, che si interfaccerà con il componente Web Server [Backend] e con il componente l'API Service per richieste specifiche.

Coesione: livello **7** - Funzionale
Spiegazione: tutti gli elementi del componente sono indipendenti da quelli degli altri componenti.

Downloader

Motivazione. Dati i RF1 e RF7, il software deve poter raccogliere i dati delle Quotazioni OMI dal servizio GeoPOI offerto dall'Agenzia delle Entrate. È stato quindi identificato un componente Downloader, che si interfaccerà con il sistema di Agenzia delle Entrate tramite apposite richieste, gestendo il download dei dati pubblici e l'upload degli stessi nel Database.

Coesione: livello **7** - Funzionale
Spiegazione: tutti gli elementi del componente sono indipendenti da quelli degli altri componenti.

API Service

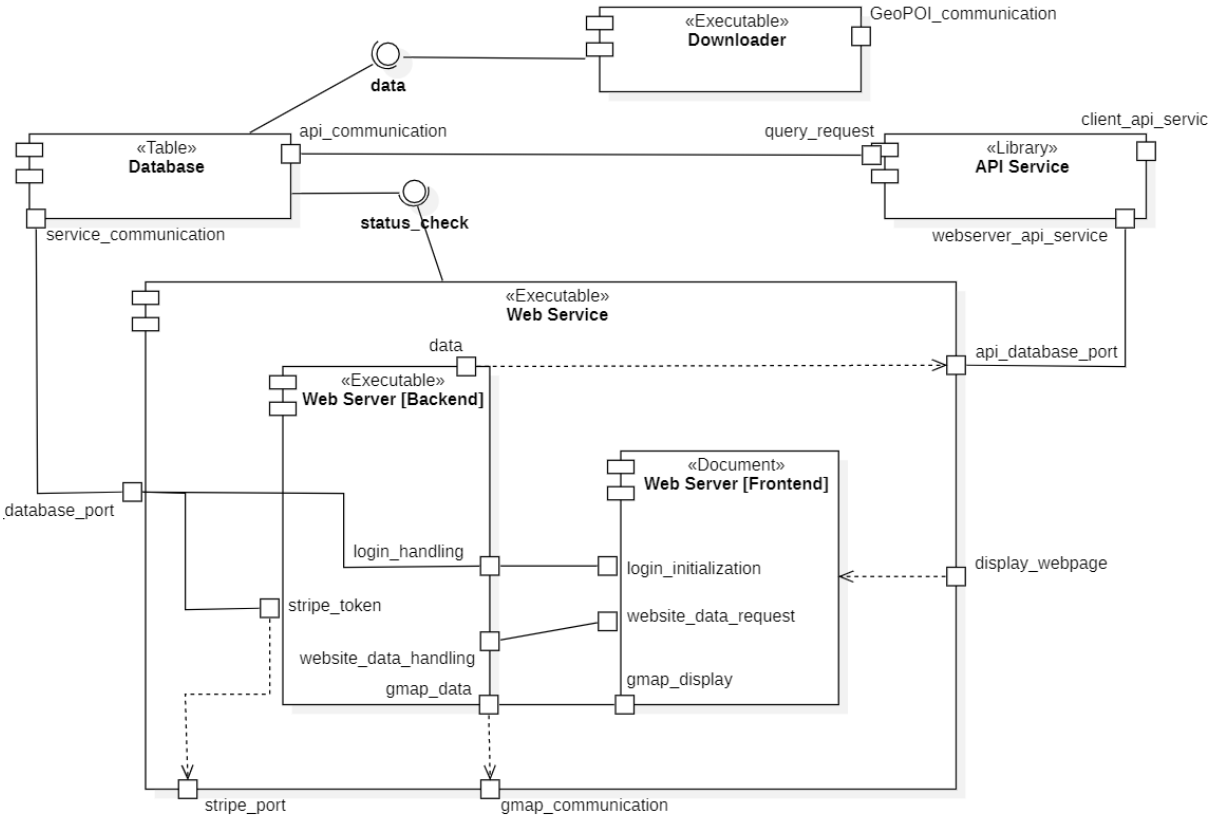
Motivazione. Dati i RF4 e RF5, il software deve fornire appropriate API a software terzi che intendono fare uso dei dati meglio organizzati. È stato quindi identificato un componente API Service, che si interfaccerà con il componente Web Service e con il componente Database per richieste specifiche.

Coesione: livello **7** - Funzionale
Spiegazione: tutti gli elementi del componente sono indipendenti da quelli degli altri componenti.

Tabella riassuntiva dei livelli di coesione dei componenti

Componenti \ Livello di coesione	1 casuale	2 logica	3 temporale	4 procedurale	5 comunicazionale	6 informazionale	7 funzionale
Database							X
Web Server [Backend]						X	
Web Server [Frontend]							X
Downloader							X
API Service							X

Diagramma dei componenti



La figura sopra rappresenta i diversi componenti del sistema OMI Finder. Seguirà una descrizione dei ruoli e compiti di ognuno di essi.

Database

Componente che si occupa di memorizzare i dati permanenti che servono per il funzionamento di OMI Finder. Tra i dati memorizzati ci sono credenziali, token, e dati OMI formattati.

Interfaccia richiesta - data: interfaccia di comunicazione usata per ricevere i dati OMI dal Downloader che devono essere memorizzati all'interno del database

Porta di comunicazione - api_communication: Porta di comunicazione con l'API Service, in base alle query ricevute il database risponde con i dati corrispondenti.

Porta di comunicazione - service communication: Porta di comunicazione con il Web Service usata per gestire credenziali e token

Web Server [Backend]

Componente che si occupa principalmente della comunicazione tra i diversi componenti di OMI Finder. Interagisce con tutti i componenti del sistema eccetto per il Downloader. Il suo ruolo è quello di comunicare i dati delle credenziali, token, dati di geolocalizzazione, e dati dell'API agli altri componenti di OMI Finder che richiedono queste informazioni..

Porta di comunicazione - data: Porta di comunicazione tra Backend e API Service, serve per poter ottenere i dati OMI formattati su richiesta al Backend da fornire al Frontend.
Porta di comunicazione - login_handling: Porta di comunicazione tra Backend e Frontend per gestire le credenziali di accesso al sito.
Porta di comunicazione - stripe_token: Porta di comunicazione utilizzata per ottenere il token di Stripe che serve per effettuare le query con l'API Service.
Porta di comunicazione - website_data_handling: Porta di comunicazione generica tra Frontend e Backend che viene utilizzata per fornire i dati OMI, i diversi token provenienti dal Database e qualunque dato venga mostrato dalla pagina web.
Porta di comunicazione - gmap_data: Porta di comunicazione con Google Maps API che consente di effettuare la geolocalizzazione.

Web Server [Frontend]

Componente che gestisce le richieste fatte da interfaccia web. Fornisce il sito web di OMI Finder e comunica con il Web Server [Backend] per verificare o aggiornare le credenziali, per ottenere i dati OMI da visualizzare sulla mappa e per gestire i diversi token che servono per effettuare le query e ottenere i dati OMI.

Porta di comunicazione - display_webpage: Porta che descrive l'accesso alla pagina web del sito di OMI Finder tramite browser.
Porta di comunicazione - login_initialization: Porta che viene usata per comunicare al Backend le credenziali di login/registrazione inserite nel sito web.
Porta di comunicazione - website_data_request: Porta di comunicazione tra Backend e Frontend che ha la funzione di comunicare le richieste e ottenere i dati relativi alle zone OMI e ai token.
Porta di comunicazione - gmap_display: Una volta inserito un indirizzo da ricercare nel sito web di OMI Finder, i dati vengono comunicati su questa porta di comunicazione, e una volta ottenuta una risposta l'indirizzo digitato viene visualizzato sulla mappa.

Downloader

Componente il cui unico scopo è quello di ottenere i dati OMI provenienti da GeoPOI. I dati vengono formattati per poi essere forniti al database, tutto questo con una frequenza di 6 mesi, cioè non appena i dati che si trovano su GeoPOI vengono aggiornati.

Porta di comunicazione - GeoPOI_communication: Questa porta ottiene i dati provenienti dal database dei dati OMI ufficiale di GeoPOI.
Interfaccia fornita - data: Interfaccia fornita al Database, questa interfaccia viene utilizzata quando i dati di GeoPOI vengono aggiornati per caricare i dati più recenti sul database.

API Service

Componente usato per fornire l'interfaccia di comunicazione tramite API per fare diverse query sui dati OMI che si trovano sul Database.

Porta di Comunicazione - client_api_service. I clienti paganti possono inviare richieste all'API Service direttamente e ricevere risposta senza dover passare dall'intermediario del sito. Questo rende possibile integrare le nostre API in servizi esterni.
Porta di Comunicazione - webserver_api_service. Il Backend del Webserver per conto suo o del Frontend può fare chiamate all'API Service per ottenere i dati necessari dal database.
Porta di Comunicazione - query_request. L'API Service comunica con il database per ricevere i dati necessari a soddisfare le richieste che gli sono state fatte.

Livello di accoppiamento dei componenti

	Database	Web Server [Backend]	Web Server [Frontend]	Downloader	API Service
Database		5 Data	X	2 Common	4 Stamp
Web Server [Backend]			5 Data	X	4 Stamp
Web Server [Frontend]				X	X
Downloader					X
API Service					

Discussione livello di accoppiamento dei componenti

Di seguito vengono spiegati i livelli di accoppiamento tra i vari componenti.

Database - Web Server [Backend] – Livello Data

Spiegazione: I due moduli si scambiano informazioni (mail,password,token) per autenticare, registrare o verificare l'esistenza di un Utente.

Database - Downloader – Livello Common

Spiegazione: Il downloader scarica i dati dalla piattaforma GeoPoi in autonomia ogni 6 mesi e li carica sul Database.

Database - API Service – Livello Stamp

Spiegazione: Le API recuperano le informazioni dal Database e le filtra in base alle richieste

Web Server [Backend] - Web Server [Frontend] – Livello Data

Spiegazione: Il Backend ed il Frontend si scambiano i dati per visualizzare il contenuto delle pagine e per processare il contenuto dei vari form (es. Login,Registrazione,Valutazione allindirizzzo x, ecc..)

Web Server [Backend] - API Service – Livello Stamp

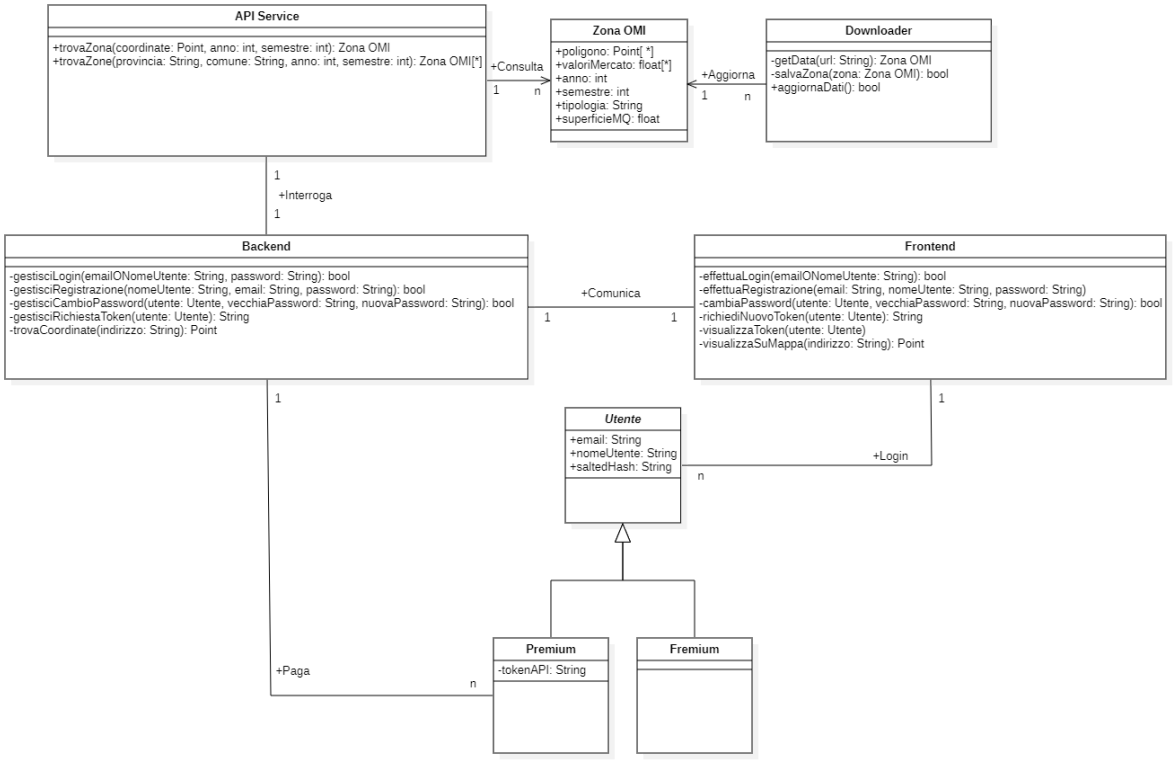
Spiegazione: Il Backend formatta le informazioni della Zona OMI, richieste tramite il Frontend, attraverso le API e le formatta per essere visualizzate.

Diagramma delle classi

Di seguito vengono presentate le classi individuate del sistema OMI Finder. Le diverse classi sono state create sulla base del diagramma dei componenti e del più generale diagramma di contesto. Seguirà il diagramma complessivo delle classi e più avanti una spiegazione dettagliata di ogni classe.

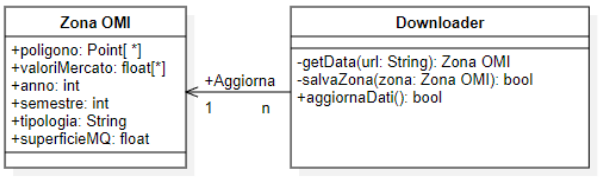
Diagramma delle classi complessivo

Qui viene presentato il diagrammi delle classi che racchiude tutte le classi necessarie individuate nel sistema. Le classi individuate sono molto simili ai componenti che appaiono nel diagramma dei componenti, in particolare troviamo le classi per il Donwloader, per l'API Service, per il Backend, e infine per il Frontend. Le classi Premium, Freemium e Zona OMI sono classi che vivono all'interno del Database.



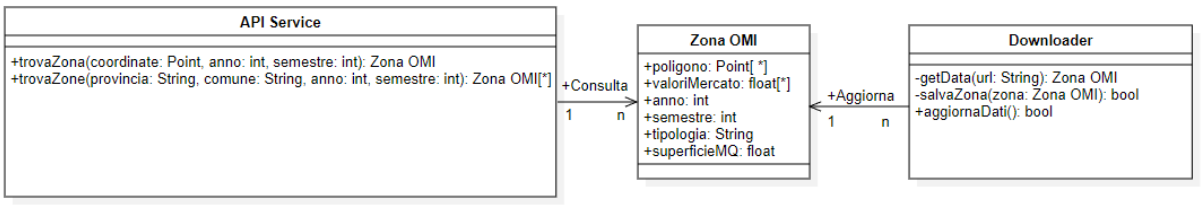
Downloader

La classe del Downloader corrisponde all'omonimo componente del diagramma dei componenti e svolge la funzione di aggiornare le zone OMI semestralmente su Database qualora tali dati vengano modificati nella banca dati di GeoPOI. Il downloader ottiene i dati delle zone OMI tramite il metodo "getData", può salvare una Zona OMI sul database grazie al metodo "salvaZona", e infine i dati possono essere aggiornati con il metodo pubblico "aggiornaDati".



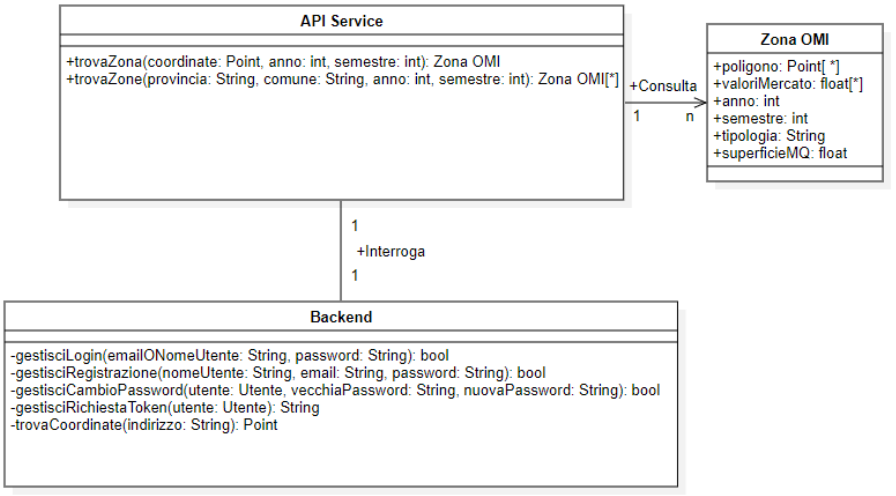
Zona OMI

Questa classe rappresenta una generica zona OMI. Una zona OMI ha molteplici attributi pubblici, e in ordine sono: i punti (coordinate) del poligono della zona OMI, i suoi valori sul mercato in €/mq, l'anno e il semestre in cui è stato valutato il terreno, la tipologia di terreno, e la superficie in metri quadrati. Le zone OMI risiedono sul Database, vengono modificate dal Downloader e vengono consultate dall'API Service.



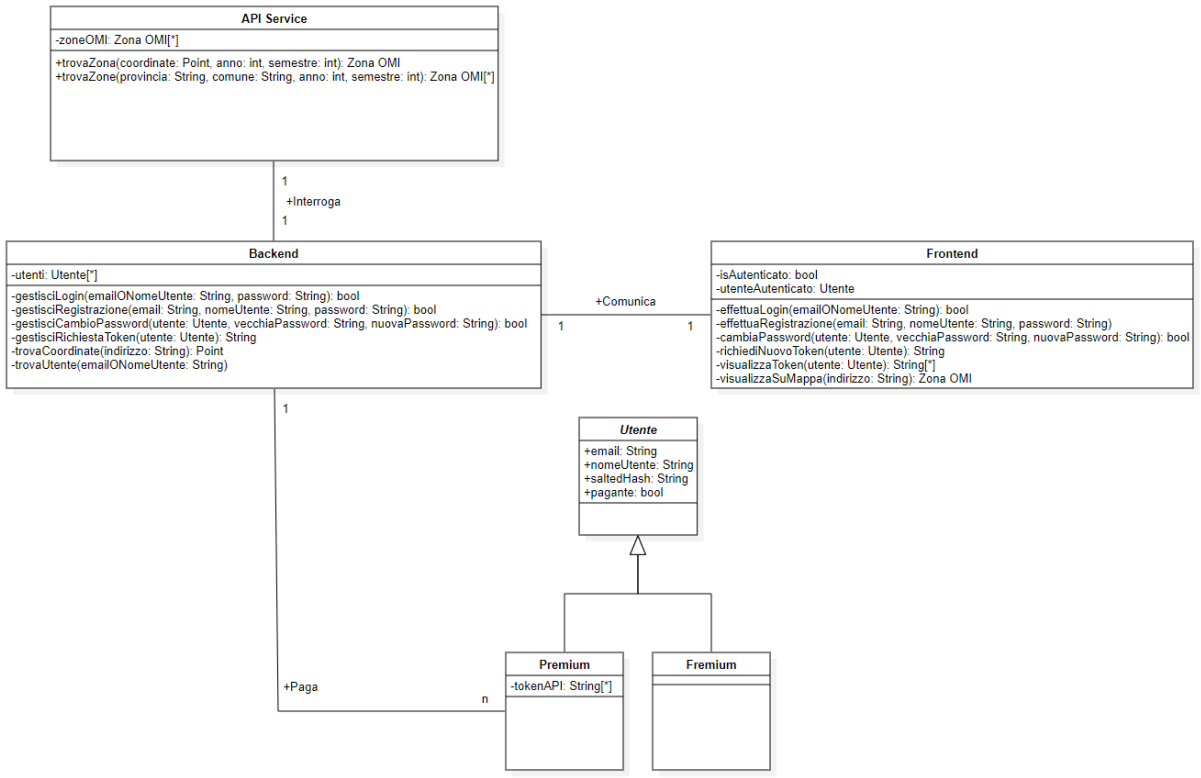
API Service

L'API Service si occupa di gestire le principali funzionalità dell'API. Per il momento sono state individuati due principali utilizzi, il primo consiste nell'individuare una Zona OMI a partire da delle coordinate, un anno e un semestre, la Zona OMI ottenuta sarà quindi la Zona OMI in un particolare anno e semestre che contiene le coordinate. Nel secondo caso invece viene passata una provincia, un comune, un anno e un semestre, e le Zone OMI così ottenute saranno tutte quelle all'interno del comune in quel particolare periodo.



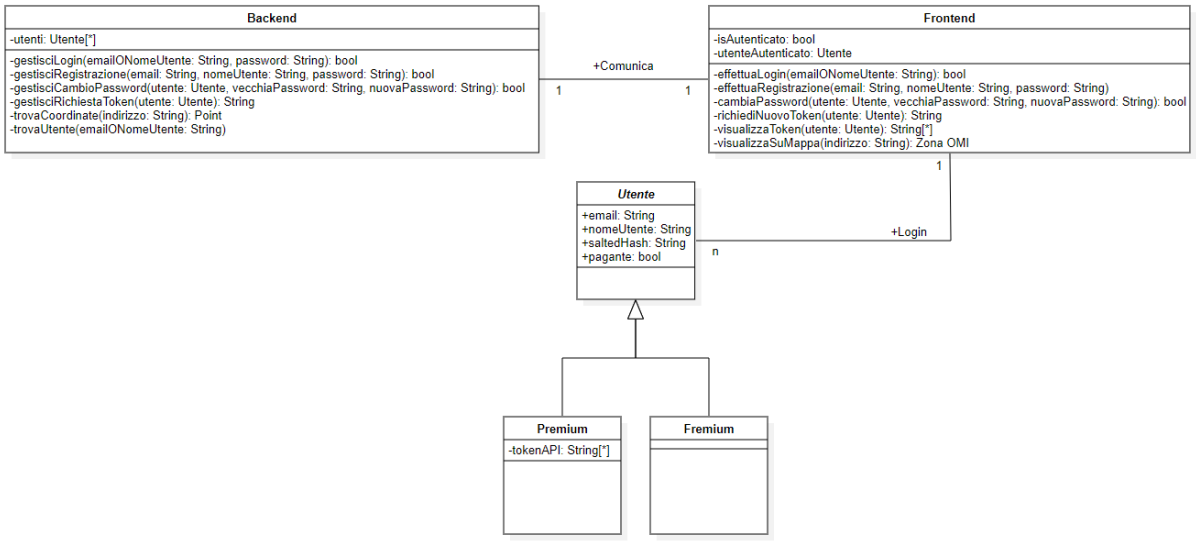
Backend

Il ruolo della classe Backend è quello di gestire la comunicazione tra il Frontend da una parte, e il Database e l'API Service dall'altra. I primi due metodi "gestisciLogin" e "gestisciRegistrazione" vengono chiamati quando un utente richiede di effettuare il login o la registrazione dal Frontend. Analogo è il comportamento di "gestisciCambioPassword". Il metodo "gestisciRichiestaToken" viene chiamato quando un utente ha intenzione di pagare per ottenere un token da usare con l'API, quando il pagamento va a buon fine il token arriva da Stripe al backend e viene salvato sul Database. Per concludere "trovaCoordinate" ha lo scopo di ottenere latitudine e longitudine di un indirizzo tramite delle API di geolocalizzazione, questo metodo serve qualora un utente voglia consultare la mappa dal Frontend.



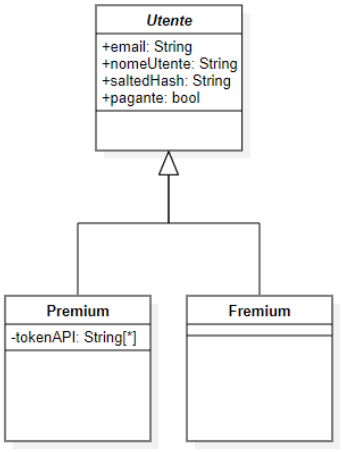
Frontend

La classe Frontend si occupa di gestire tutte le principali funzionalità del Frontend. I metodi “effettuaLogin”, “effettuaRegistrazione” e “cambiaPassoword” sono metodi che vengono invocati quando l’utente richiede una di queste funzioni tramite il Frontend. “richiediNuovoToken” invece serve per ottenere un token per effettuare query tramite l’API, e si può ottenere tramite un pagamento che viene effettuato su Stripe tramite un reindirizzamento. “visualizzaToken” serve per ottenere i token già associati all’utente che sta navigando il sito. Per ultimi rimangono i metodi “visualizzaSuMappa” che serve per visualizzare una Zona OMI sulla mappa, e “trovaUtente” che ha la funzione di trovare i dati dell’utente associato ad una email o nome utente.



Utente e sottoclassi

Le ultime classi che rimangono da analizzare sono quella dedicata all'Utente Premium e Freemium. Intuitivamente un Utente Premium e un Utente Freemium sono entrambi degli Utenti, quindi è logico creare una classe astratta "Utente" che possiede gli attributi "email", "nomeUtente" e "saltedHash" (la password con salt a cui è stato fatto l'hashing) che verrà ereditata da "Premium" e "Freemium". Un utente ha anche un attributo "pagante" per verificare che il pagamento su Stripe sia andato a buon fine. Per il momento "Freemium" è una classe vuota, ma questo potrebbe cambiare in futuro, il codice risulta più ordinato e mantenibile nel caso in cui un giorno si vogliano effettuare delle modifiche ed aggiungere delle proprietà a "Freemium" che non si vogliono far avere a "Premium". "Premium" invece possiede un vettore di token, coi quali può effettuare diverse query tramite l'API.



Codice in Object Constraint Language

Sono state individuate delle parti logiche non esprimibili solamente attraverso il linguaggio UML utilizzato fino ad ora. Queste parti verranno espresse in Object Constraint Language o OCL. Le parti coinvolte sono il Backend e il Frontend di OMI Finder.

Frontend

Per quanto riguarda il Frontend sono alcuni casi particolari accadono quando un utente autenticato prova ad eseguire il login oppure quando uno non autenticato prova a cambiare la password. Qui sotto verranno elencati uno ad uno tutti i casi descritti in OCL che possono generare dei problemi.

```
context Frontend::effettuaLogin(emailONomeUtente:
String)
pre: Frontend::isAutenticato = false and
Frontend::utenteAutenticato = null
post: Frontend::isAutenticato = true
```

Nella classe Frontend bisogna accertarsi che un utente che esegue il login non sia già autenticato con un altro account, e questo OCL esprime questa condizione.

```
context Frontend::effettuaRegistrazione(email: String,
nomeUtente: String, password: String)
pre: Frontend::isAutenticato = false and
Frontent::utenteAutenticato = null
post: Frontend::isAutenticato = true
```

Lo stesso identico comportamento accade anche per la registrazione.

```
context Frontend::richiediNuovoToken(utente: Utente)
pre: Frontend::isAutenticato = true
```

Questo OCL in un modo simile ai precedenti si accerta che un utente che sta cercando di effettuare una richiesta per un nuovo token sia autenticato.

```
context Frontend::cambiaPassword(utente: Utente,
vecchiaPassword: String, nuovaPassword: String)
pre: Frontend::isAutenticato = true
```

Quando si vuole cambiare password bisogna accertarsi che l'utente che vuole cambiare password sia autenticato.

```
context Frontend::richiediNuovoToken(utente: Utente)
pre: Utente::pagante = true
```

Quando un utente richiede un nuovo token bisogna controllare che abbia effettuato il pagamento su Stripe, questo viene fatto tramite il controllo di una variabile "pagante" della classe Utente.

Backend

Il Backend deve controllare che gli accessi e le autenticazioni avvengano correttamente. Per esempio devono esserci certi controlli per verificare che non ci siano duplicazioni di nomi utente o email all'interno del database quando un utente si vuole registrare. In seguito tutti gli OCL necessari.

```
context Backend::gestisciRegistrazione(email: String,
nomeUtente: String, password: String)
pre: Backend::utenti ->
excludes(Backend::trovaUtente(email)) and
Backend::utenti ->
excludes(Backend::trovaUtente(nomeUtente))
```

Questo OCL come spiegato precedentemente si occupa di verificare che non vi siano duplicazioni di nomi utente o email quando un utente prova ad effettuare la registrazione.

```
context Backend::gestisciCambioPassword(utente:
Utente, vecchiaPassword: String, nuovaPassword:
String)
pre: Backend::utenti -> includes(utente)
```

Per gestire il cambio delle password è essenziale controllare prima che l'utente che vuole effettuare la modifica compaia nel database degli utenti dell'applicazione.

```
context Backend::gestisciRichestaToken(utente:
Utente)
pre: Backend::utenti -> includes(utente)
```

L'ultimo OCL del Backend si assicura che la richiesta di un token sia associata ad un utente esistente nel Database.

Diagramma delle classi con codice OCL

Riportiamo il diagramma delle classi complessivo fino ad ora presentate ed il codice OCL individuato.