NetBreak

Progetto API Market

Definizione di Prodotto

Informazioni sul documento

Nome del documento	DefinizioneDiProdotto 2_0_0.pdf
Data di creazione	16 Marzo 2017
Ultima modifica	15 Giugno 2017
Versione	2.0.0
Stato	Approvato
Redatto da	Marco Casagrande Dan Serbanoiu Alberto Nicolè Andrea Scalabrin Nicolò Scapin
Verificato da	Nicolò Scapin Andrea Scalabrin
Approvato da	Davide Scarparo
Uso	Esterno
Distribuzione	NetBreak
Destinato a	Prof. Tullio Vardanega, Prof. Riccardo Cardin, ItalianaSoftware S.r.l.
Email di riferimento	netbreakswe@gmail.com

Abstract

Questo documento descrive la progettazione di dettaglio delle componenti del progetto API Market, definite dal gruppo NetBreak.

Changelog

Versione	Data	Autore	Ruolo	Descrizione
2.0.0	2017-06-15	Davide Scarparo	Responsabile	Approvazione del documento
1.1.0	2017-06-10	Nicolò Scapin	Verificatore	Correzioni alla sezione #3
1.0.2	2017-06-02	Andrea Scalabrin	Verificatore	Correzioni alla sezione #4
1.0.1	2017-05-26	Marco Casagrande	Amministratore	Correzioni alla sezione #5.1
1.0.0	2017-04-26	Davide Scarparo	Responsabile	Approvazione del documento
0.3.0	2017-04-17	Nicolò Scapin	Verificatore	Verifica del documento
0.2.5	2017-04-15	Alberto Nicolè	Progettista	Aggiunta sezione #6.2
0.2.4	2017-04-12	Nicolò Scapin	Verificatore	Aggiunta sezione #6.1
0.2.3	2017-04-11	Alberto Nicolè	Progettista	Correzioni alle sezioni #5.1.2 e #5.1.2 secondo verifica
0.2.2	2017-04-09	Nicolò Scapin	Verificatore	Correzioni alla sezione #5.1.7 secondo verifica
0.2.1	2017-04-07	Alberto Nicolè	Progettista	Correzioni alla sezione #5.1.1 secondo verifica
0.2.0	2017-04-06	Nicolò Scapin	Verificatore	Verifica del documento
0.1.6	2017-03-04	Nicolò Scapin	Progettista	Stesura sezione #5.1.7
0.1.5	2017-04-02	Dan Serbanoiu	Progettista	stesura sezioni #5.1.4, #5.1.5, #5.1.6
0.1.4	2017-03-31	Davide Scarparo	Progettista	Stesura della sezione #4.7, #4.8, #4,9
0.1.3	2017-03-29	Davide Scarparo	Progettista	Inizio stesura della sezione #5 e stesura sezioni #5.1.1, #5.1.2, #5.1.3
0.1.2	2017-03-28	Nicolò Scapin	Progettista	Stesura della sezioni #3.4 e #3.5
0.1.1	2017-03-26	Alberto Nicolè	Progettista	Correzioni secondo la verifica
0.1.0	2017-03-25	Nicolò Scapin	Verificatore	Verifica del documento
0.0.7	2017-03-23	Davide Scarparo	Progettista	Stesura della sezioni #4.4, #4.5, #4,6
0.0.6	2017-03-22	Nicolò Scapin	Progettista	Stesura della sezioni #4.1, #4.2, #4,3
0.0.5	2017-03-21	Davide Scarparo	Progettista	Stesura della sezioni #3.3 e #3.4
0.0.4	2017-03-20	Nicolò Scapin	Progettista	Stesura della sezioni #3.1 e #3.2

Versione	Data	Autore	Ruolo	Descrizione
0.0.3	2017-03-18	Andrea Scalabrin	Progettista	Stesura della sezione #2
0.0.2	2017-03-16	Marco Casagrande	Progettista	Stesura della sezione #1
0.0.1	2017-03-16	Davide Scarparo	Responsabile	Creazione template

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Lo scopo del presente documento è la definizione in dettaglio della struttura e del funzionamento delle componenti relative al prodotto *API Market*$_G$. Questo documento servirà come guida per i *Programmatori* del team *NetBreak* durante le attività di Codifica, fornendo direttive e vincoli per la realizzazione del progetto.

1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un API Market per l'acquisto e la vendita di *microservizi*$_G$. Il sistema offrirà la possibilità di registrare nuove *API*$_G$ per la vendita, permetterà la consultazione e la ricerca di API ai potenziali acquirenti, gestendo i permessi di accesso ed utilizzo tramite creazione e controllo di relative *API key*$_G$. Il sistema, oltre alla web app stessa, sarà corredato di un *API Gateway*$_G$ per la gestione delle richieste e il controllo delle chiavi, e fornirà funzionalità avanzate di statistiche per il gestore della piattaforma e per i fornitori dei microservizi.

1.3 Riferimenti normativi

- NormeDiProgetto 4_0_0.pdf

1.4 Riferimenti informativi

- **Ingegneria del software - Ian Sommerville - 8a edizione:**
  Porzione dedicata alla Progettazione Architetturale (Capitolo 11)
- **Slides del corso - Design patterns strutturali**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E04.pdf`
- **Slides del corso - Design patterns creazionali**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E05.pdf`
- **Slides del corso - Design patterns architetturali**
  - `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E08.pdf`
  - `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E09.pdf`
- **Slides del corso - Diagrammi dei packages**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E02b.pdf`
- **Slides del corso - Diagrammi delle classi**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E02a.pdf`
- **Slides del corso - Diagrammi di sequenza**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E03a.pdf`
- **Slides del corso - Diagrammi di attività**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E03b.pdf`
- **HTML5**
  `https://www.w3.org/TR/html5/`
- **Bootstrap CSS Framework**
  `http://getbootstrap.com/`

- **AngularJS**
  `https://www.angularjs.org/`
- **JQuery JavaScript Library**
  `https://jquery.com/`
- **Jolie Programming Language**
  `http://www.jolie-lang.org/`
- **Java Programming Language**
  `https://www.oracle.com/it/java/index.html`
- **MySQL Database**
  `https://www.mysql.it/`
- **Chris Richardson**
  - `http://microservices.io`
  - `http://microservices.io/patterns/microservices.html`
- **Martin Fowler**
  - `https://martinfowler.com/articles/microservices.html`
  - `https://martinfowler.com/microservices/`
- **IBM**
  `http://www.ibm.com/developerworks/websphere/library/techarticles/1601_clark-trs/1601_clark.html`
- **NGINX**
  - `https://www.nginx.com/blog/introduction-to-microservices/`
  - `https://www.nginx.com/blog/building-microservices-using-an-api-gateway/`

1.5 Glossario

Per semplificare la consultazione e disambiguare alcune terminologie tecniche, le voci indicate con la lettera *G* a pedice sono descritte approfonditamente nel documento Glossario 3_0_0.pdf e specificate solo alla prima occorrenza all'interno del suddetto documento.

2 Architettura

In questa breve sezione, viene fornita una visione ad alto livello dell'intera architettura del prodotto API Market, ovvero come sono strutturati e suddivisi i packages più importanti, mettendo in evidenza la separazione tra *front-end*$_G$ e *back-end*$_G$.

2.1 Visione ad alto livello

Il software API Market implementa una classica applicazione *Client-Server*, caratterizzata da un lato *front-end*$_G$ (Client), il quale si occuperà di fornire all'utente della piattaforma l'interfaccia web su cui poter interagire, e un lato *back-end*$_G$ (Server) che avrà il compito di reperire, salvare e fornire dati, e, tramite l'opportuna componente *API Gateway*, si occuperà della gestione delle chiamate ai microservizi disponibili. Le basi di dati utilizzate, si occuperanno della raccolta di dati sensibili dell'utenza, della gestione dei microservizi e delle relative chiavi e, infine, dell'elaborazione dei dati statistici in merito alla *SLA*$_G$. Per il lato front-end, verrà utilizzato il framework *AngularJS* $_G$ *(Versione 1)*, unito ai comuni linguaggi *HTML5*$_G$, *CSS3*$_G$ e *JavaScript*$_G$. Per il lato back-end, invece, verrà utilizzato il linguaggio Jolie per la realizzazione di microservizi per l'interfacciamento con le basi di dati e con l'API Gateway; quest'ultimo, infine, sarà implementato tramite i linguaggi Jolie e Java.

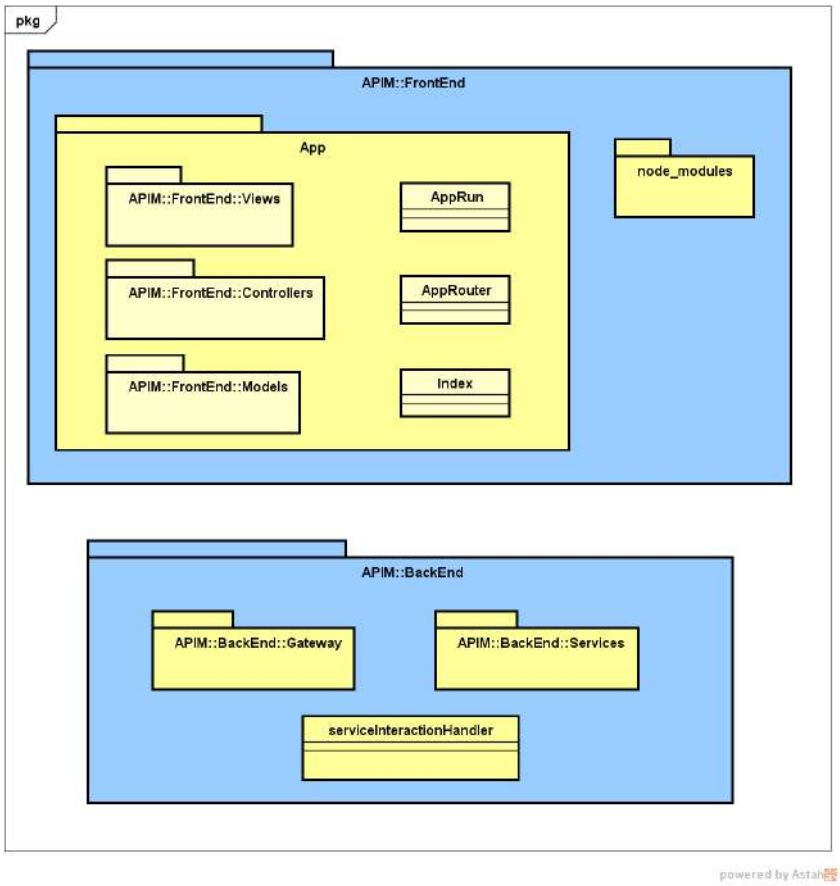

Figura 1: Architettura generale

2.1.1 Informazioni generali

Di seguito, sono raccolte le informazioni generali dello schema presentato precedentemente:

- **Descrizione:** architettura ad alto livello della piattaforma API Market.

- **Packages contenuti:**
  - APIM::FrontEnd: package contenente tutti i packages che compongono la parte di front-end dell'applicazione;
  - APIM::BackEnd: package contenente tutti i packages che compongono la parte di back-end dell'applicazione.

3 Specifica Front-End

3.1 APIM::FrontEnd

3.1.1 Informazioni generali

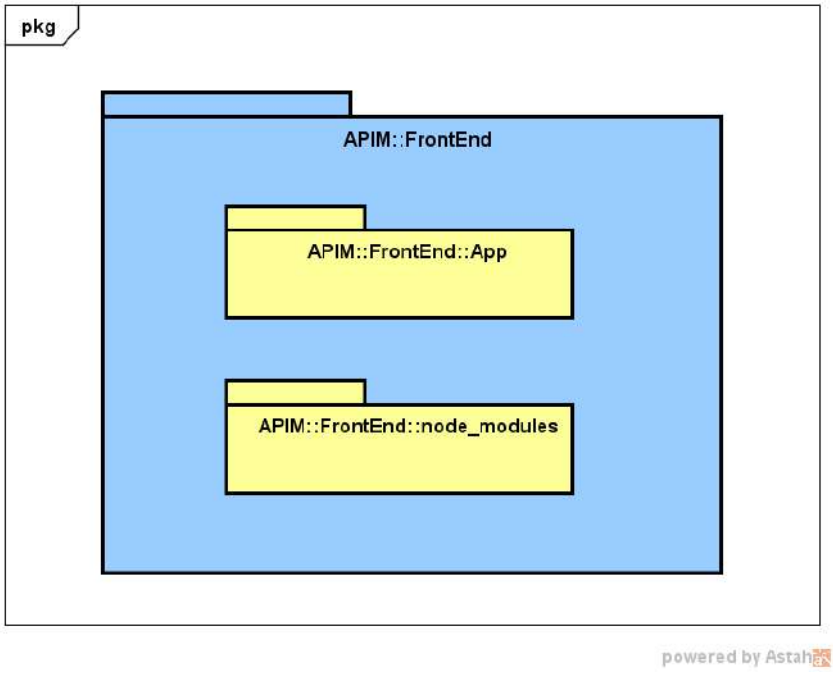

Figura 2: APIM::FrontEnd

- **Descrizione:** package contenente le componenti del lato front-end dell'applicazione web;
- **Packages contenuti:**
  - App;
  - node_modules.

3.2 APIM::FrontEnd::App

3.2.1 Informazioni generali

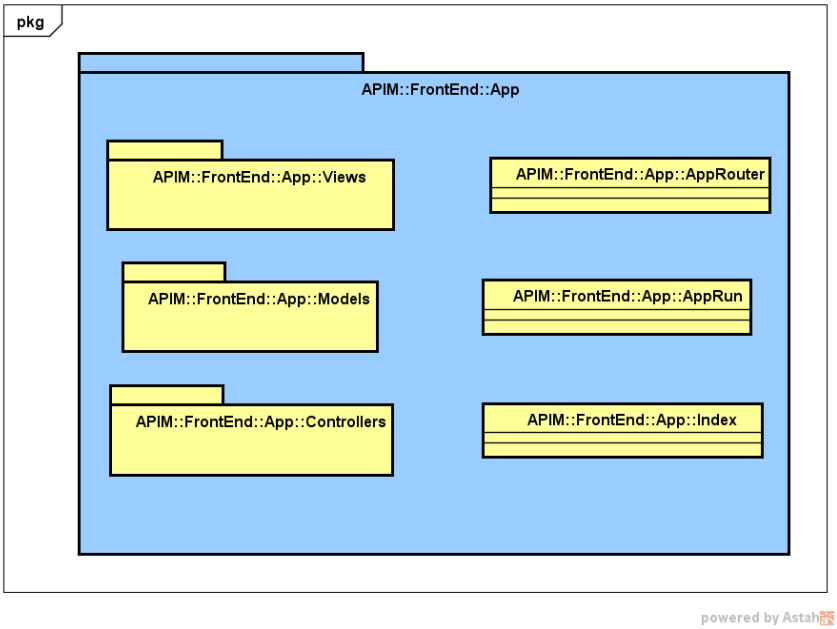

Figura 3: APIM::FrontEnd::App

- **Descrizione:** Il package App contiene tutto il necessario al funzionamento del front-end dell'applicazione web API Market;
- **Packages contenuti:**
  - Views;
  - Models;
  - Controllers.

3.2.2 Classi

3.2.2.1 APIM::FrontEnd::App::AppRouter

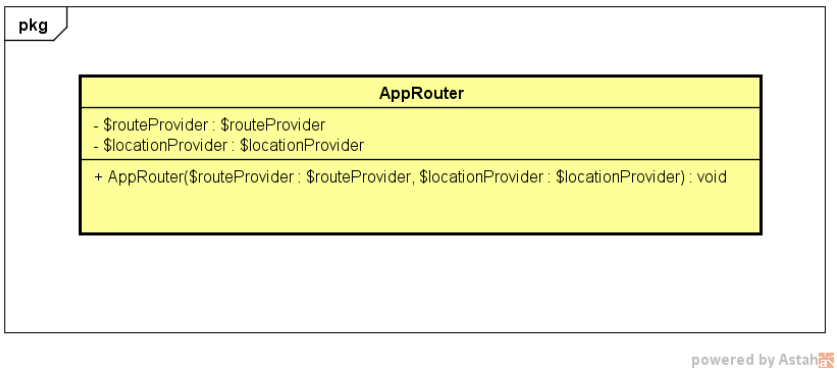

Figura 4: APIM::FrontEnd::App::AppRouter

- **Descrizione:** La classe AppRouter gestisce le routes dell'applicazione, collegando le views con i rispettivi controllers attraverso i servizi di $routeProvider e $locationProvider;
- **Attributi:**
  - **$routeProvider:** $routeProvider
    Campo dati con il riferimento al servizio di AngularJS dedicato al routing;
  - **$locationProvider:** $locationProvider
    Campo dati con il riferimento al servizio di AngularJS dedicato al pathing;
- **Metodi:**
  - **AppRouter($routeProvider: $routeProvider, $locationProvider: $location-Provider):** Metodo per la gestione di routing e pathing dell'applicazione.

3.2.2.2 APIM::FrontEnd::App::AppRun

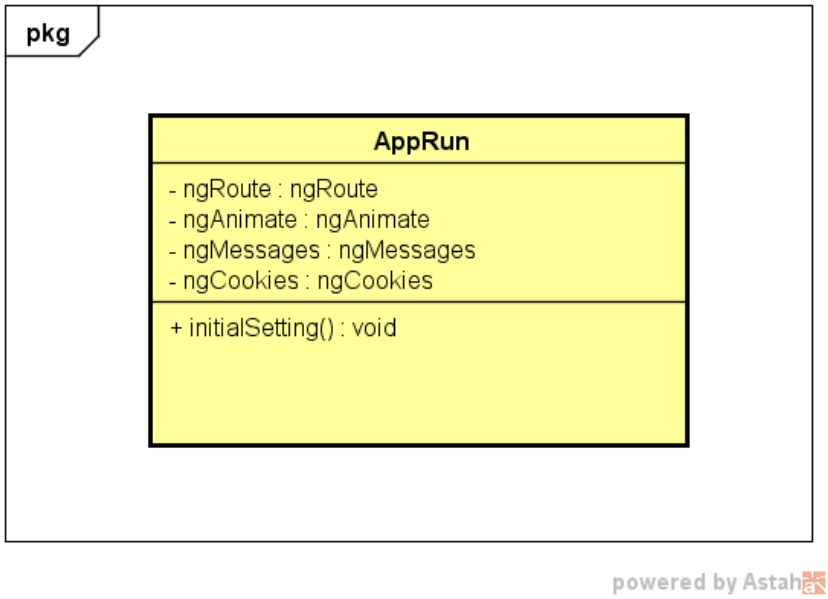

Figura 5: APIM::FrontEnd::App::AppRun

- **Descrizione:** La classe AppRun verifica la corretta autenticazione dell'utente e che le sue autorizzazioni gli consentano di visitare la pagina richiesta;
- **Attributi:**
  - **ngRoute**: ngRoute
    Campo dati con il riferimento al modulo ngRoute di AngularJS;
  - **ngAnimate**: ngAnimate
    Campo dati con il riferimento al modulo ngAnimate di AngularJS;
  - **ngMessages**: ngMessages
    Campo dati con il riferimento al modulo ngMessages di AngularJS;
  - **ngCookies**: ngCookies
    Campo dati con il riferimento al modulo ngCookies di AngularJS.
- **Metodi:**

- **initialSetting(): void** Metodo per l'inizializzazione dei campi dati ai valori di default.

- **Relazioni con altre classi:**
  - Utilizza il modulo ngRoute per effettuare il routing dell'applicazione;
  - Utilizza il modulo ngAnimate per impiegare transizioni ed animazioni nell'applicazione;
  - Utilizza il modulo ngMessages per mostrare messaggi d'aiuto durante le procedure e form dell'applicazione;
  - Utilizza il modulo ngCookies per effettuare il salvataggio dei cookies dell'applicazione.

3.2.2.3 APIM::FrontEnd::App::Index

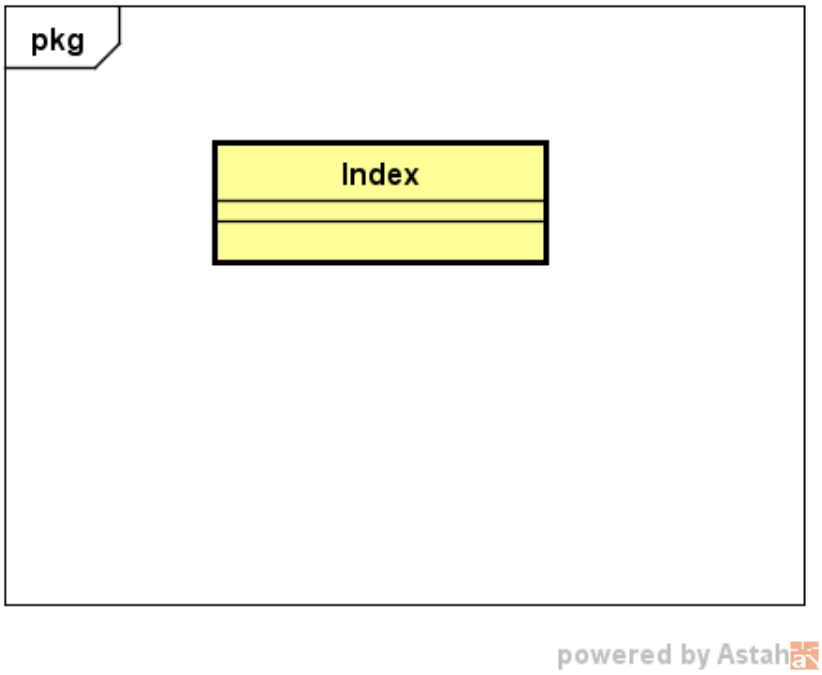

Figura 6: APIM::FrontEnd::App::Index

- **Descrizione:** La classe Index contiene la view principale dell'applicazione, dove l'utente visualizza dinamicamente le pagine che vuole visitare.

- **Relazioni con altre classi:**
  - Interagisce con il package Views, necessario alla corretta visualizzazione delle pagine.

3.3 APIM::FrontEnd::App::Views

3.3.1 Informazioni generali

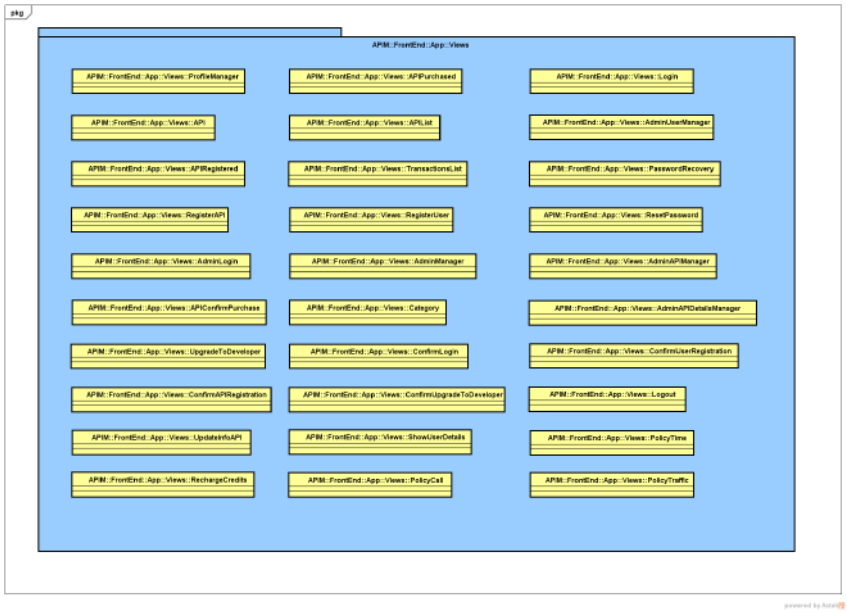

Figura 7: APIM::FrontEnd::App::Views

- **Descrizione:** Il package Views contiene tutte le view dell'applicazione.
- **Relazioni con altre classi:**
  - Il package **Controllers** collega le views ad un controller per gestirne la visualizzazione da parte dell'utente;
  - Il package **Models** definisce le strutture dati utilizzate da views e controllers.

3.3.2 Classi

3.3.2.1 APIM::FrontEnd::App::Views::ProfileManager

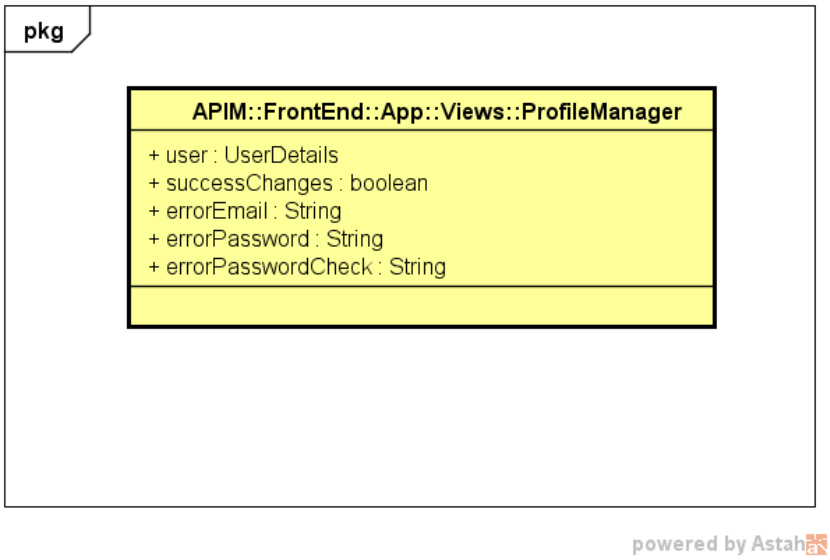

Figura 8: APIM::FrontEnd::App::Views::ProfileManager

- **Descrizione:** View contenente le informazioni del profilo personale di un utente registrato. Contiene, inoltre, l'informazione relativa al saldo del proprio conto virtuale;
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **successChanges : boolean**
    Campo dati contenente la conferma delle modifiche;
  - **errorEmail : string**
    Campo dati contenente l'eventuale errore di inserimento dell'email;
  - **errorPassword : string**
    Campo dati contenente l'eventuale errore di inserimento della password;
  - **errorPasswordCheck : string**
    Campo dati contenente l'eventuale errore della conferma della password.
- **Relazioni con altre classi:**
  - Interagisce con il controller **ProfileManagerController**;
  - Il model **UsersDetailModel** contiene le informazioni per rappresentare un utente.

3.3.2.2 APIM::FrontEnd::App::Views::API

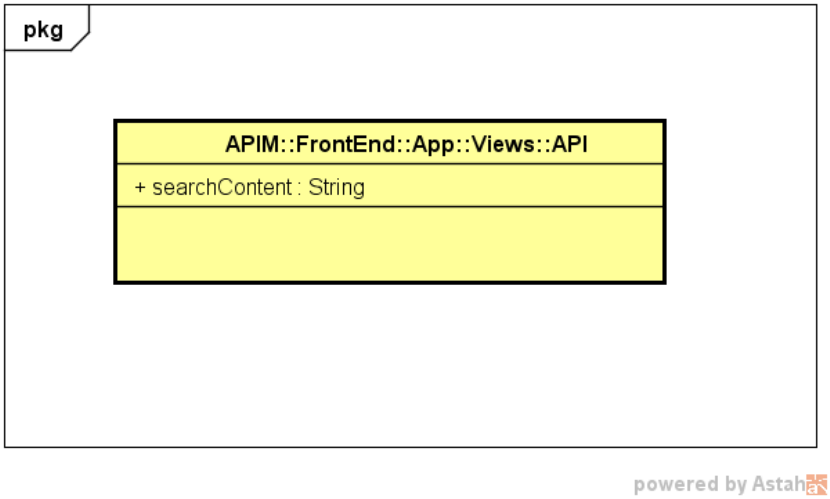

Figura 9: APIM::FrontEnd::App::Views::API

- **Descrizione:** View contenente i risultati della ricerca effettuata, che permette di selezionare un risultato presente al suo interno.
- **Attributi:**
  - **searchContent : string** Campo dati contenente le keywords di ricerca dell'API.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.3 APIM::FrontEnd::App::Views::APIRegistered

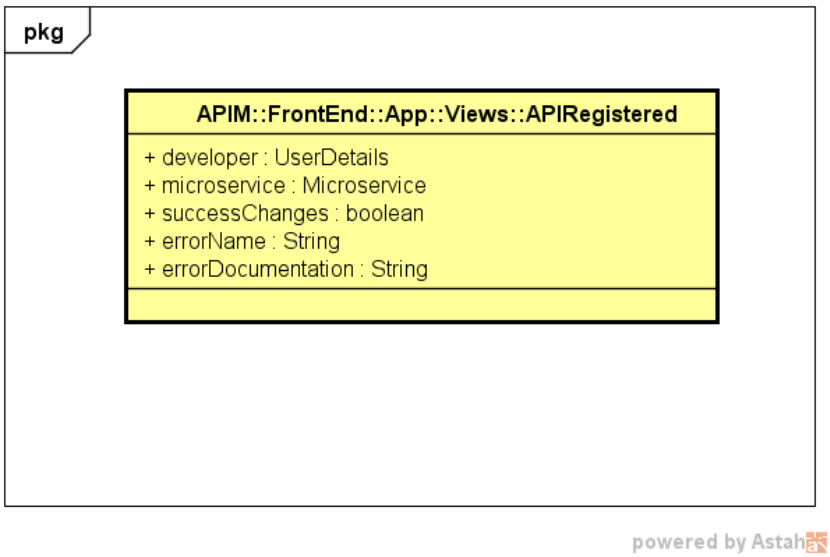

Figura 10: APIM::FrontEnd::App::Views::APIRegistered

- **Descrizione:** View contenente la lista delle API registrate dall'utente sulla piattaforma.
- **Attributi:**
  - **developer : UserDetails**
    Campo dati contenente le informazioni di un utente sviluppatore;
  - **microservice : Microservice**
    Campo dati contenente le informazioni di un microservizio;
  - **successChanges : boolean**
    Campo dati contenente la conferma delle modifiche;
  - **errorName : string**
    Campo dati contenente l'eventuale errore di inserimento del nome;
  - **errorDocumentation: string**
    Campo dati contenente l'eventuale errore di inserimento della documentazione.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIRegisteredController**;
  - Il model **UsersDetailModel** contiene le informazioni per rappresentare un utente;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.4 APIM::FrontEnd::App::Views::RegisterAPI

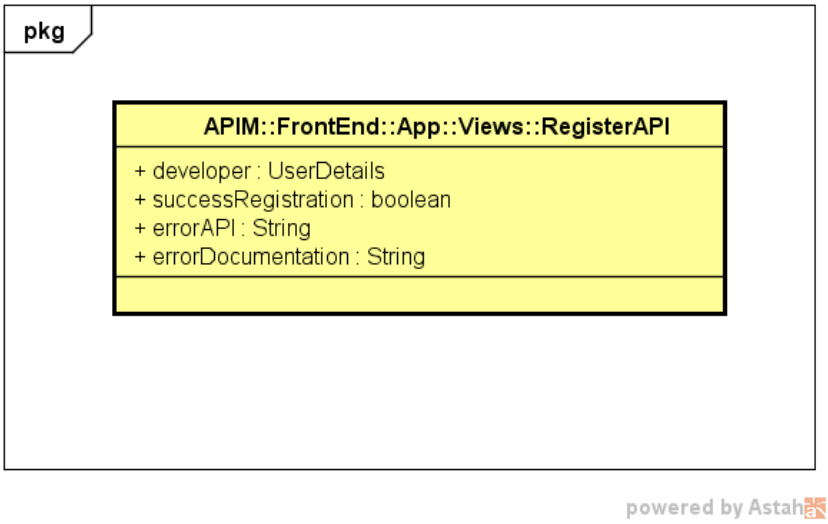

Figura 11: APIM::FrontEnd::App::Views::RegisterAPI

- **Descrizione:** View contenente il form per l'inserimento di una API da parte di un utente sviluppatore. Lo sviluppatore può inserire tutti i dati relativi al microservizio che intende esporre sul marketplace.
- **Attributi:**
  - **developer : UserDetails**
    Campo dati contenente le informazioni di un utente sviluppatore;
  - **successRegistration : boolean**
    Campo dati contenente la conferma della registrazione;

  - **errorAPI : string**
    Campo dati contenente l'eventuale errore di inserimento dell'API;
  - **errorDocumentation: string**
    Campo dati contenente l'eventuale errore di inserimento della documentazione.

- **Relazioni con altre classi:**
  - Interagisce con il controller **RegisterAPIController**;
  - Il model **UsersDetailModel** contiene le informazioni per rappresentare un utente;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.5 APIM::FrontEnd::App::Views::PolicyCall

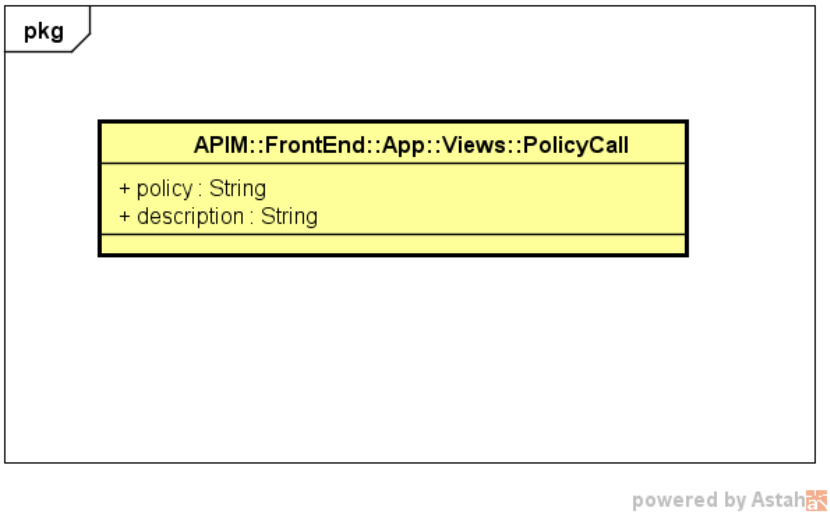

Figura 12: APIM::FrontEnd::App::Views::PolicyCall

- **Descrizione:** View contenente le informazioni relative alla policy per chiamate.
- **Attributi:**
  - **policy : string**
    Campo dati contenente la policy di una API;
  - **description : string**
    Campo dati contenente la descrizione di una policy a chiamate.
- **Relazioni con altre classi:**
  - Interagisce con il controller **PolicyCallController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.6 APIM::FrontEnd::App::Views::PolicyTime

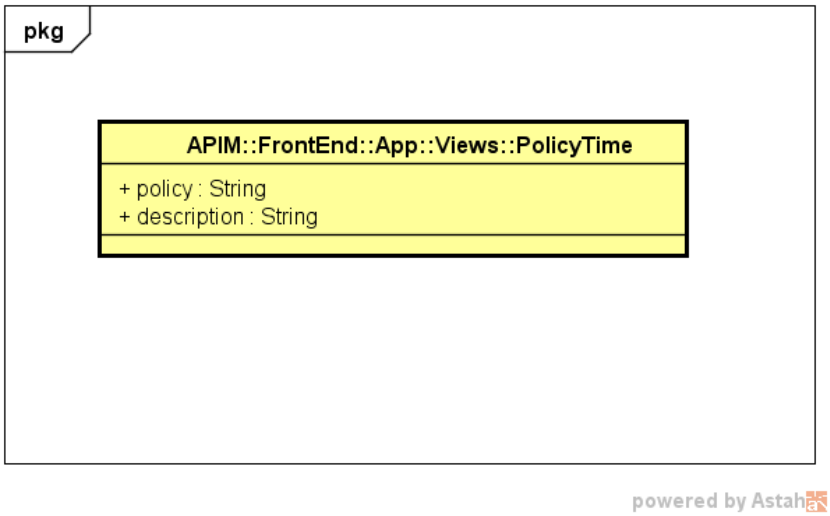

Figura 13: APIM::FrontEnd::App::Views::PolicyTime

- **Descrizione:** View contenente le informazioni relative alla policy per tempo.
- **Attributi:**
  - **policy : string**
    Campo dati contenente la policy di una API;
  - **description : string**
    Campo dati contenente la descrizione di una policy a tempo.
- **Relazioni con altre classi:**
  - Interagisce con il controller **PolicyTimeController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.7 APIM::FrontEnd::App::Views::PolicyTraffic

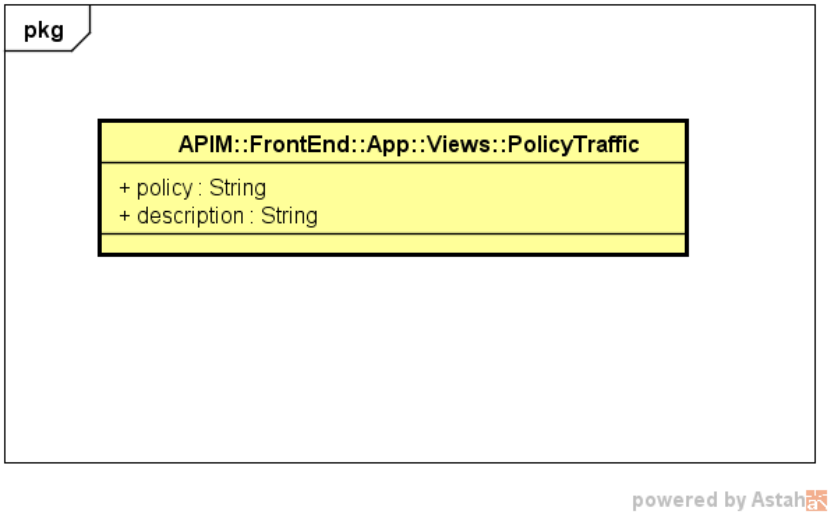

Figura 14: APIM::FrontEnd::App::Views::PolicyTraffic

- **Descrizione:** View contenente le informazioni relative alla policy per traffico dati.
- **Attributi:**
  - **policy : string**
    Campo dati contenente la policy di una API;
  - **description : string**
    Campo dati contenente la descrizione di una policy per traffico.
- **Relazioni con altre classi:**
  - Interagisce con il controller **PolicyTrafficController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.8 APIM::FrontEnd::App::Views::APIPurchased

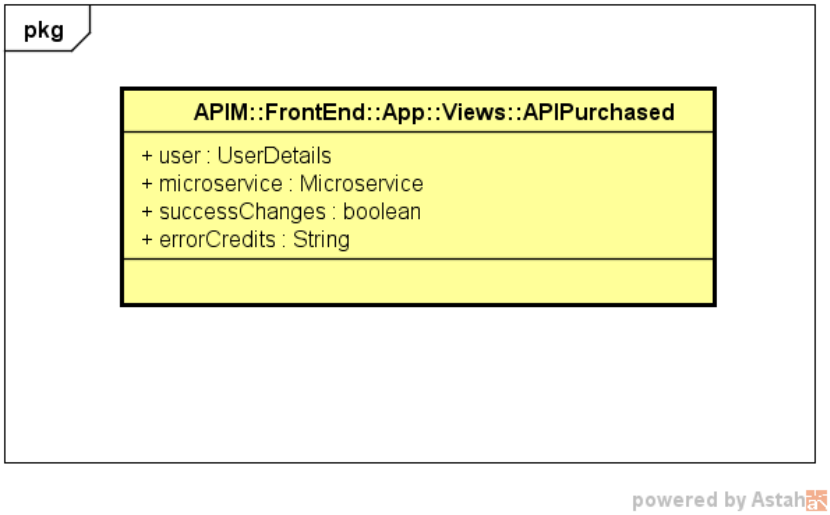

Figura 15: APIM::FrontEnd::App::Views::APIPurchased

- **Descrizione:** View contenente la lista delle API acquistate da un utente della piattaforma API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **microservice : Microservice**
    Campo dati contenente le informazioni di una API;
  - **successChanges : boolean**
    Campo dati contenente la conferma del rinnovo dell'API;
  - **errorCredits : string**
    Campo dati contenente l'eventuale errore di rinnovo dell'API.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIPurchasedController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente;

- Il model **TransactionModel** contiene le informazioni per rappresentare una transazione;
- Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.9 APIM::FrontEnd::App::Views::APIList

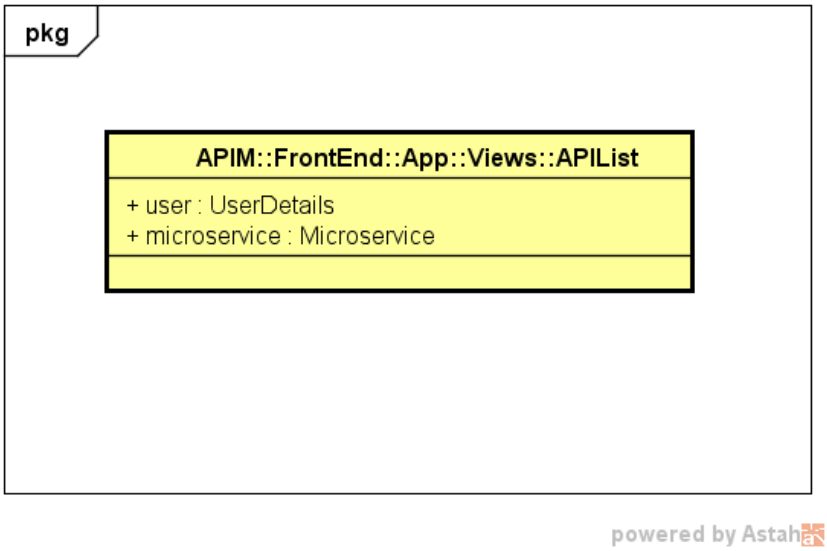

Figura 16: APIM::FrontEnd::App::Views::APIList

- **Descrizione:** View contenente l'elenco delle API disponibili sul marketplace API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **microservice : Microservice**
    Campo dati contenente le informazioni di una API.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIListController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare una API.

3.3.2.10 APIM::FrontEnd::App::Views::TransactionsList

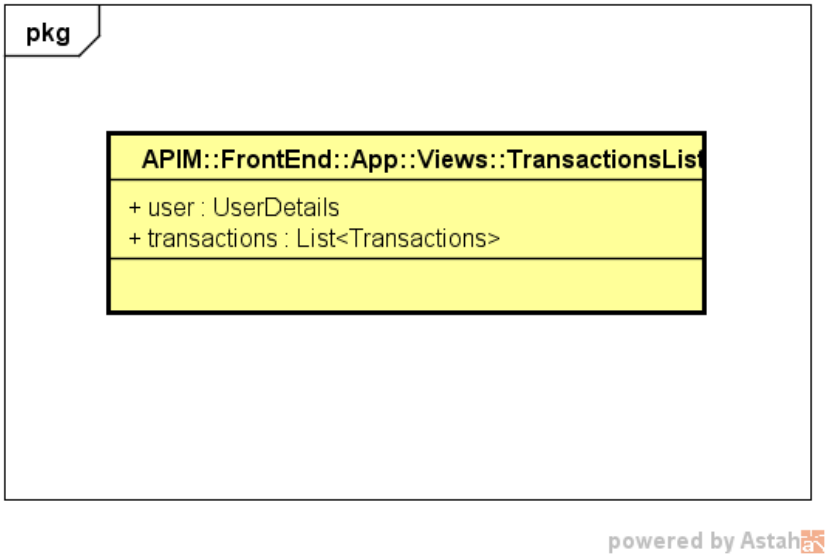

Figura 17: APIM::FrontEnd::App::Views::TransactionsList

- **Descrizione:** View contenente l'elenco delle transazioni effettuate da un utente su API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente.
  - **transactions : List<Transaction>**
    Campo dati contenente la lista delle transazioni di un utente.
- **Relazioni con altre classi:**
  - Interagisce con il controller **TransactionsListController**;
  - Il model **TransactionModel** contiene le informazioni per rappresentare una API.

3.3.2.11 APIM::FrontEnd::App::Views::RegisterUser

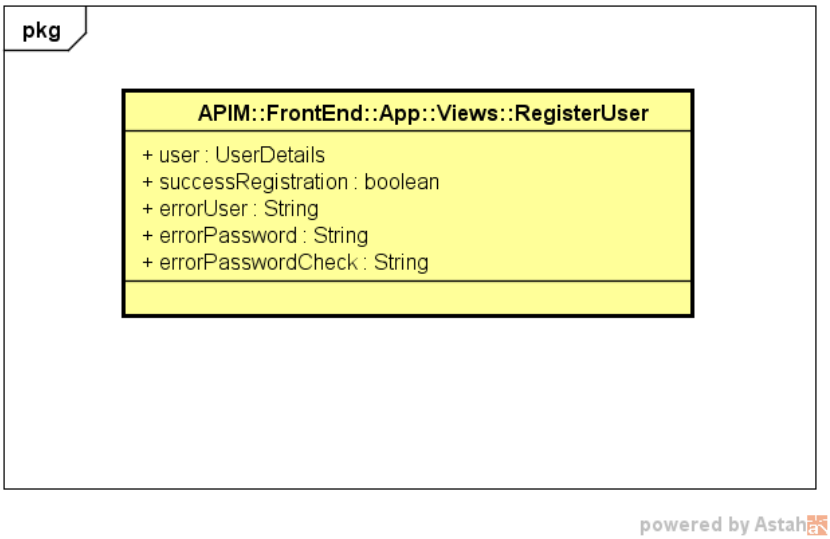

Figura 18: APIM::FrontEnd::App::Views::RegisterUser

- **Descrizione:** View contenente il form dedicato alla registrazione di un utente, il quale può inserire i campi necessari e registrarsi così alla piattaforma. Contiene, inoltre, un link alla pagina di login;
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **successRegistration : boolean**
    Campo dati contenente il flag di successo della registrazione;
  - **errorUser : string**
    Campo dati contenente l'eventuale errore di inserimento delle informazioni dell'utente;
  - **errorPassword : string**
    Campo dati contenente l'eventuale errore di inserimento della password;
  - **errorPasswordCheck : string**
    Campo dati contenente l'eventuale errore di reinserimento della password.
- **Relazioni con altre classi:**
  - Interagisce con il controller **RegisterUserController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.3.2.12 APIM::FrontEnd::App::Views::AdminManager

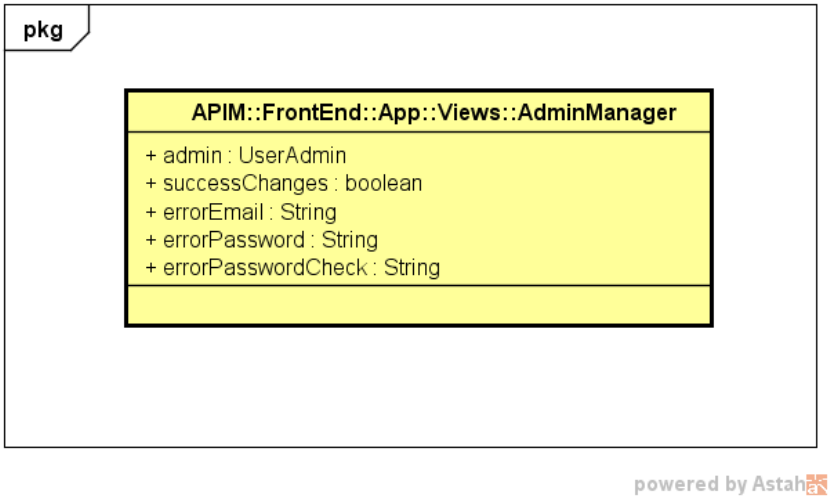

Figura 19: APIM::FrontEnd::App::Views::AdminManager

- **Descrizione:** View contenente le operazioni per la gestione del profilo amministratore API Market.
- **Attributi:**
  - **admin : UserAdmin**
    Campo dati contenente le informazioni di un admin;
  - **successChanges : boolean**
    Campo dati contenente il messaggio di successo delle modifiche;
  - **errorEmail : string**
    Campo dati contenente l'eventuale errore di inserimento delle informazioni dell'utente;
  - **errorPassword : string**
    Campo dati contenente l'eventuale errore di inserimento della password;
  - **errorPasswordCheck : string**
    Campo dati contenente l'eventuale errore di reinserimento della password.
- **Relazioni con altre classi:**
  - Interagisce con il controller **AdminManagerController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.3.2.13 APIM::FrontEnd::App::Views::Login

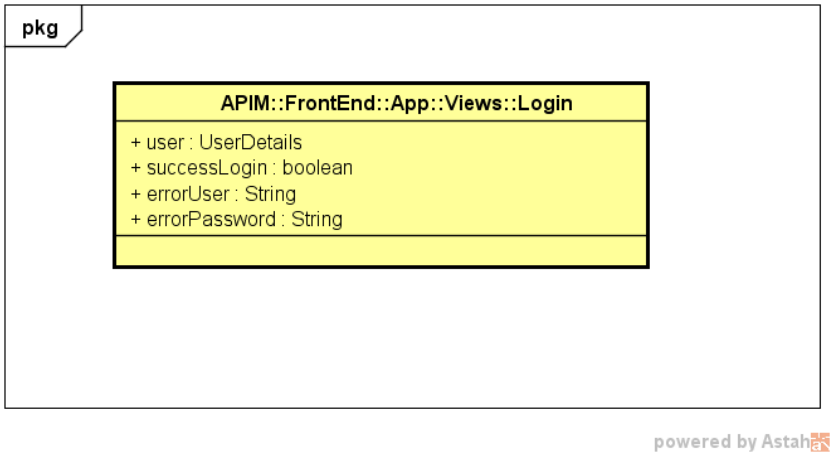

Figura 20: APIM::FrontEnd::App::Views::Login

- **Descrizione:** View contenente il form necessario affinchè l'utente possa effettuare il login ed autenticarsi al sistema. Contiene, inoltre, un link alla pagina di registrazione e uno alla pagina per il recupero della password
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **successLogin : boolean**
    Campo dati contenente il messaggio di successo del login;
  - **errorUser : string**
    Campo dati contenente l'eventuale errore di inserimento delle informazioni dell'utente;
  - **errorPassword : string**
    Campo dati contenente l'eventuale errore di inserimento della password.
- **Relazioni con altre classi:**
  - Interagisce con il controller **LoginController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.3.2.14 APIM::FrontEnd::App::Views::AdminAPIDetailsManager

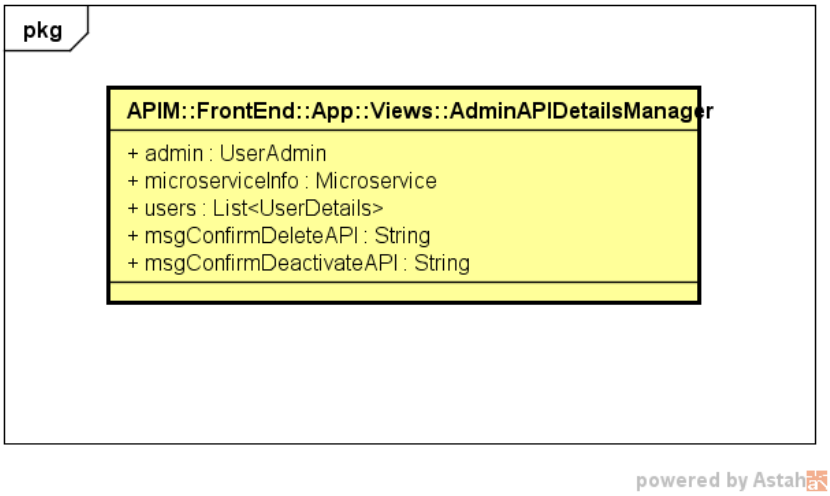

Figura 21: APIM::FrontEnd::App::Views::AdminAPIDetailsManager

- **Descrizione:** View contenente le statistiche di una API dell'API Market.

- **Attributi:**
  - **admin : UserAdmin**
    Campo dati contenente le informazioni di un admin;
  - **microserviceInfo : Microservice**
    Campo dati contenente le informazioni di un microservizio;
  - **users : List<UserDetails>**
    Campo dati contenente la lista degli utenti per una relativa API.
  - **msgConfirmDeleteAPI : string**
    Campo dati contenente il messaggio di conferma della cancellazione API.
  - **msgConfirmDeactivateAPI : string**
    Campo dati contenente il messaggio di conferma della disattivazione API.

- **Relazioni con altre classi:**
  - Interagisce con il controller **AdminAPIDetailsManagerController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio.

3.3.2.15 APIM::FrontEnd::App::Views::PasswordRecovery

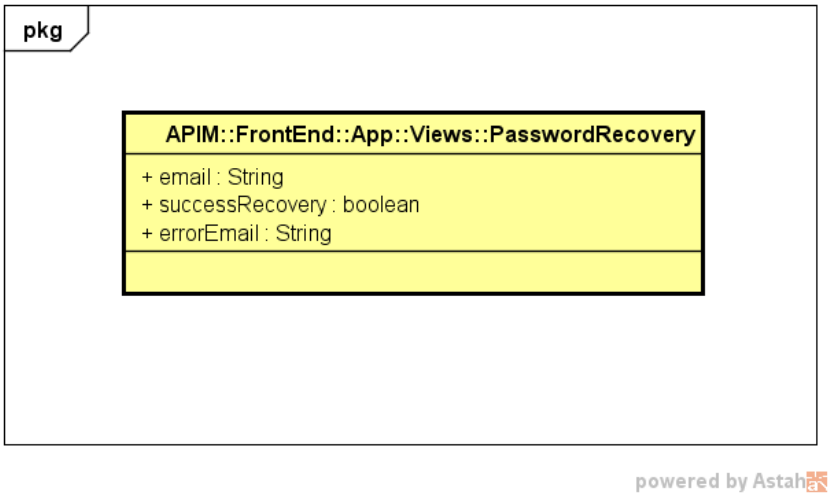

Figura 22: APIM::FrontEnd::App::Views::PasswordRecovery

- **Descrizione:** View contenente il form dedicato al recupero della password di un utente, il quale può inserire l'indirizzo email e ricevere una nuova password con la quale autenticarsi al sistema. Contiene, inoltre, un link alla pagina di login;
- **Attributi:**
  - **email : string**
    Campo dati contenente un indirizzo email;
  - **successRecovery : boolean**
    Campo dati contenente il flag di successo del recupero password;
  - **errorEmail : string**
    Campo dati contenente l'eventuale errore di inserimento dell'indirizzo email.
- **Relazioni con altre classi:**
  - Interagisce con il controller **PasswordRecoveryController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.3.2.16 APIM::FrontEnd::App::Views::ResetPassword

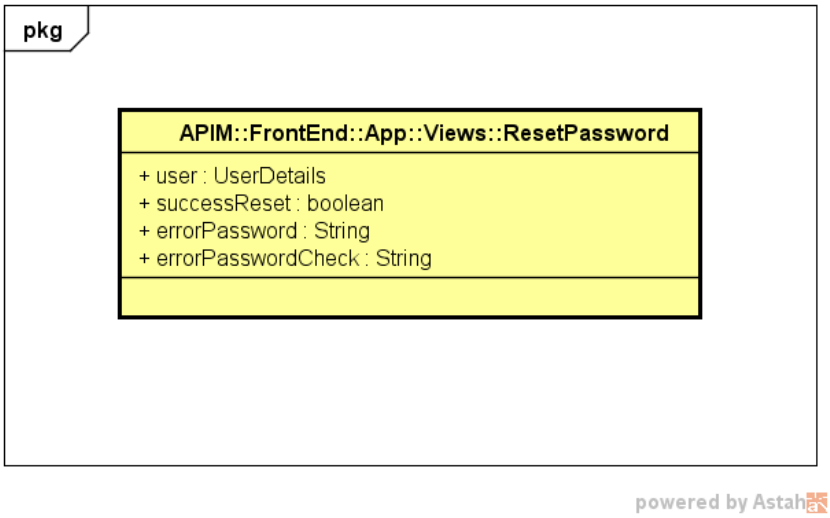

Figura 23: APIM::FrontEnd::App::Views::ResetPassword

- **Descrizione:** View contenente il form dedicato al cambio di password di un utente autenticato, il quale può inserire la nuova password che intende utilizzare per i futuri login al sistema.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **successReset : boolean**
    Campo dati contenente il flag di successo del reset password;
  - **errorPassword : string**
    Campo dati contenente l'eventuale errore di inserimento della password.
  - **errorPasswordCheck : string**
    Campo dati contenente l'eventuale errore di reinserimento della password.
- **Relazioni con altre classi:**
  - Interagisce con il controller **ResetPasswordController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.3.2.17 APIM::FrontEnd::App::Views::AdminUserManager

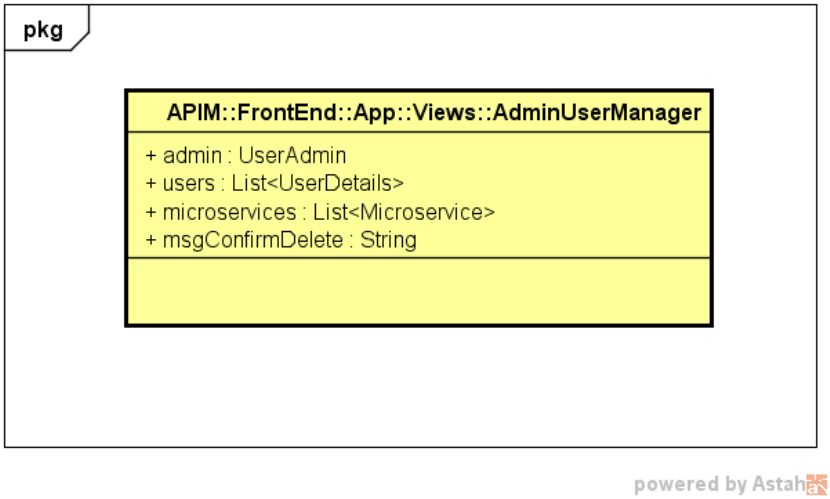

Figura 24: APIM::FrontEnd::App::Views::AdminUserManager

- **Descrizione:** View contenente le operazioni di un ammistratore sugli utenti dell'API Market.
- **Attributi:**
  - **admin : UserAdmin**
    Campo dati contenente le informazioni di un admin;
  - **users : List<UserDetails>**
    Campo dati contenente la lista di utenti registrati nella piattaforma.
  - **microservices : List<Microservice>**
    Campo dati contenente la lista dei microservizi registrati per ogni utente.
  - **msgConfirmDelete : string**
    Campo dati contenente il messaggio di conferma eliminazione
- **Relazioni con altre classi:**
  - Interagisce con il controller **AdminUserManagerController**;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio.

3.3.2.18 APIM::FrontEnd::App::Views::APIConfirmPurchase

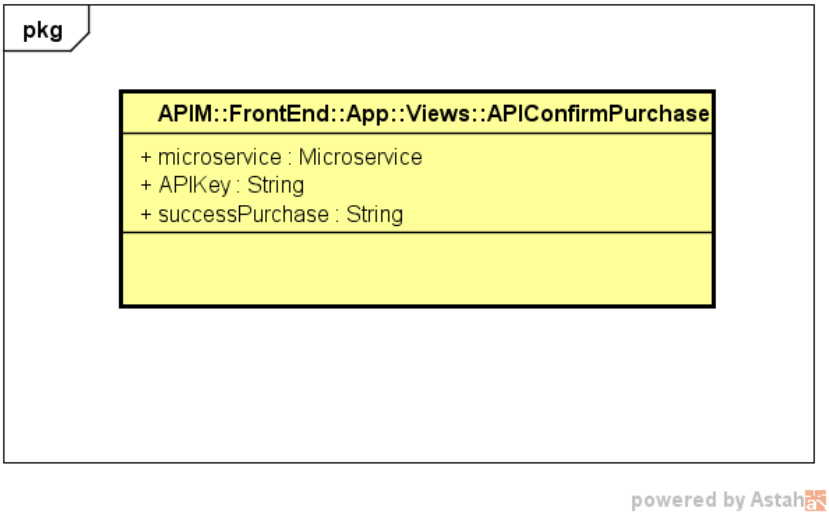

Figura 25: APIM::FrontEnd::App::Views::APIConfirmPurchase

- **Descrizione:** View contenente la conferma di un acquisto di una API.
- **Attributi:**
  - **microservice : Microservice**
    Campo dati contenente le informazioni di un microservizio;
  - **APIKey : string**
    Campo dati contenente l'API key per il microservizio acquistato.
  - **successPurchase : string**
    Campo dati contenente un messaggio di conferma per l'acquisto.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIConfirmPurchaseController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio.

3.3.2.19 APIM::FrontEnd::App::Views::Category

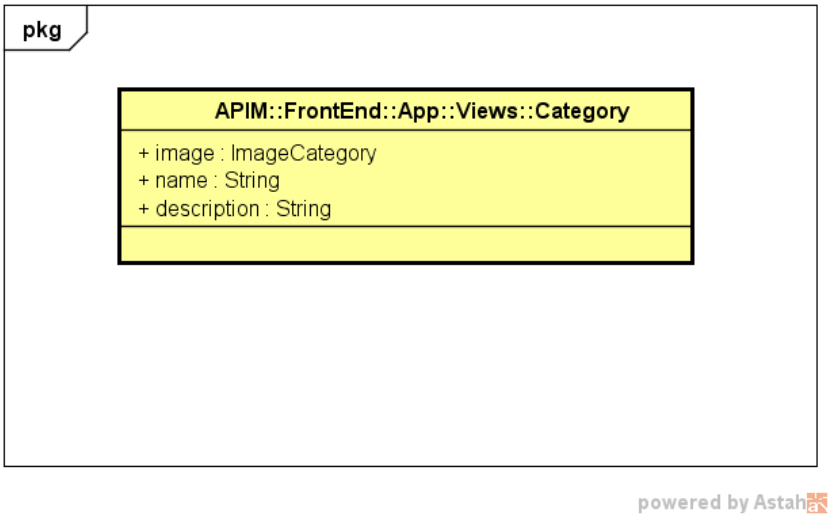

Figura 26: APIM::FrontEnd::App::Views::Category

- **Descrizione:** View contenente l'elenco delle categorie nell'API Market
- **Attributi:**
  - **image : ImageCategory**
    Campo dati contenente un immagine della categoria;
  - **name : string**
    Campo dati contenente il nome della categoria;
  - **description : string**
    Campo dati contenente la descrizione della categoria.
- **Relazioni con altre classi:**
  - Interagisce con il controller **CategoryController**;

3.3.2.20 APIM::FrontEnd::App::Views::ConfirmUpgradeToDeveloper

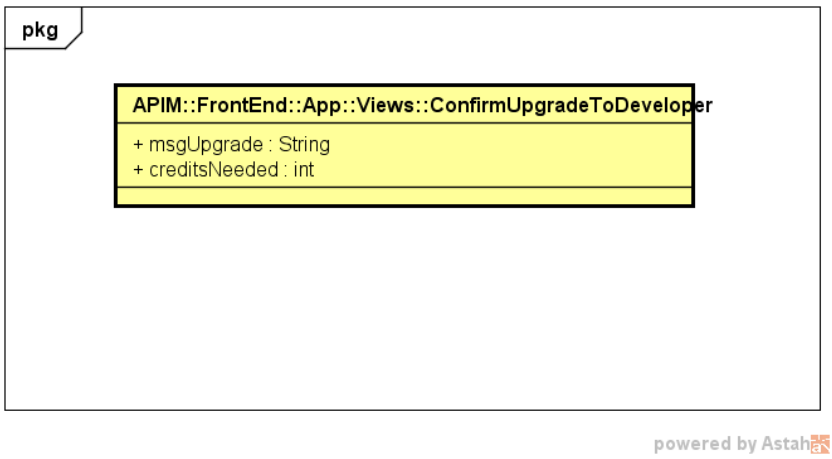

Figura 27: APIM::FrontEnd::App::Views::ConfirmUpgradeToDeveloper

- **Descrizione:** View contenente la conferma dell'upgrade di un utente a sviluppatore nell'API Market.
- **Attributi:**
  - **msgUpgrade : string**
    Campo dati contenente un messaggio per la procedura di upgrade;
  - **creditsNeeded : int**
    Campo dati contenente il numero di crediti necessari all'upgrade.
- **Relazioni con altre classi:**
  - Interagisce con il controller **ConfirmUpgradeToDeveloperController**;

3.3.2.21 APIM::FrontEnd::App::Views::ConfirmLogin

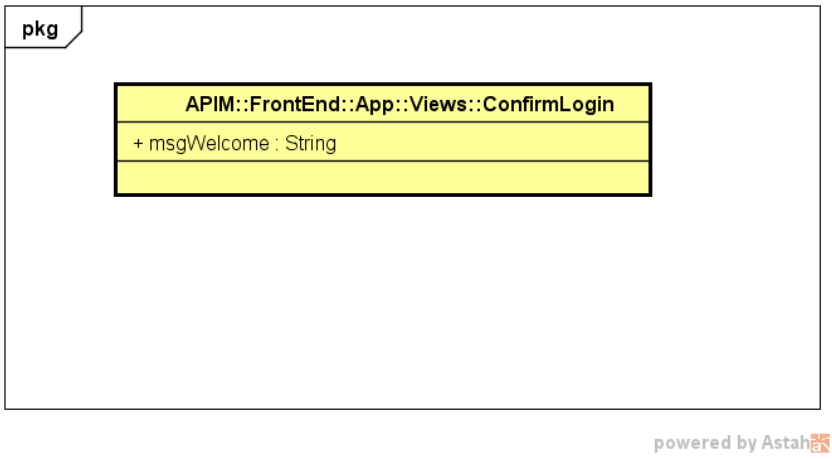

Figura 28: APIM::FrontEnd::App::Views::ConfirmLogin

- **Descrizione:** View contenente la conferma di login all'API Market.
- **Attributi:**
  - **msgWelcome : string**
    Campo dati contenente un messaggio di benvenuto post autenticazione;
- **Relazioni con altre classi:**
  - Interagisce con il controller **ConfirmLoginController**;

3.3.2.22 APIM::FrontEnd::App::Views::ConfirmUserRegistration

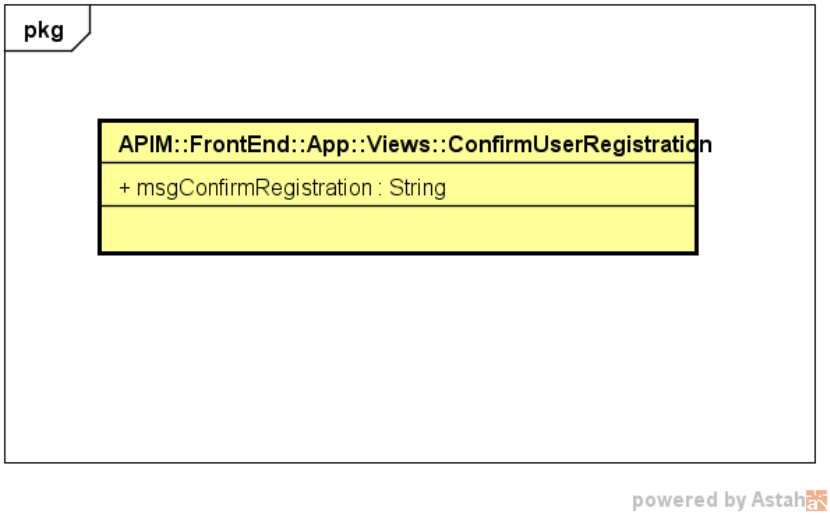

Figura 29: APIM::FrontEnd::App::Views::ConfirmUserRegistration

- **Descrizione:** View contenente la conferma di registrazione all'API Market..
- **Attributi:**
  - **msgConfirmRegistration : string**
    Campo dati contenente un messaggio di conferma registrazione;
- **Relazioni con altre classi:**
  - Interagisce con il controller **ConfirmUserRegistrationController**;

3.3.2.23 APIM::FrontEnd::App::Views::ConfirmAPIRegistration

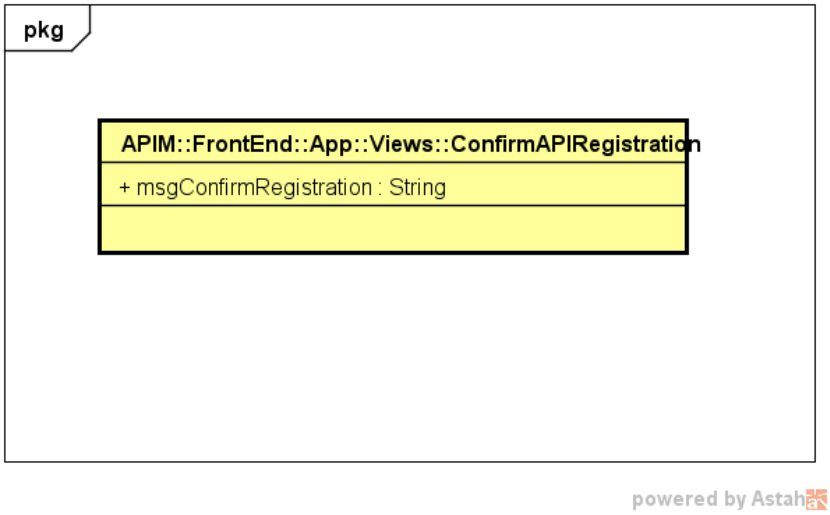

Figura 30: APIM::FrontEnd::App::Views::ConfirmAPIRegistration

- **Descrizione:** View contenente la conferma di registrazione di una API all'API Market.

- **Attributi:**
  - **msgConfirmRegistration : string**
    Campo dati contenente un messaggio di conferma registrazione API;
- **Relazioni con altre classi:**
  - Interagisce con il controller **ConfirmAPIRegistrationController**;

3.3.2.24 APIM::FrontEnd::App::Views::UpgradeToDeveloper

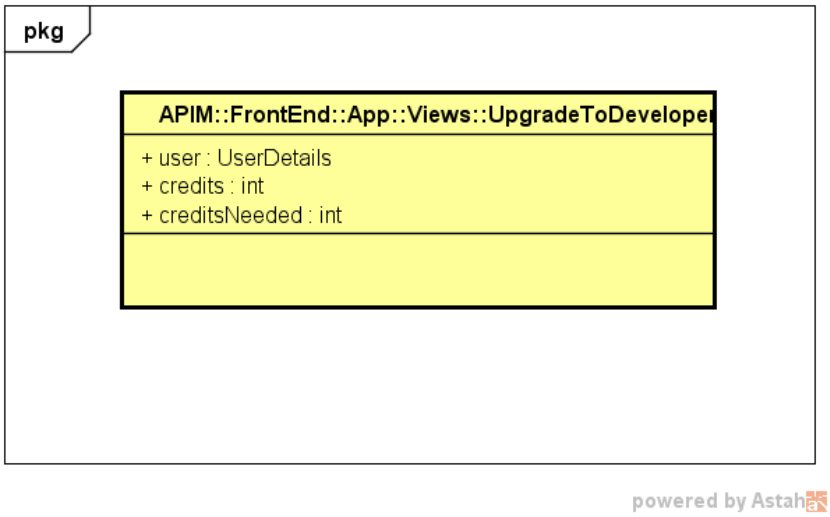

Figura 31: APIM::FrontEnd::App::Views::UpgradeToDeveloper

- **Descrizione:** View contenente il modulo per diventare sviluppatore all'interno dell'API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **credits : int**
    Campo dati contenente i crediti personali di un utente;
  - **creditsNeeded : int**
    Campo dati contenente i crediti necessari all'upgrade.
- **Relazioni con altre classi:**
  - Interagisce con il controller **UpgradeToDeveloperController**;

3.3.2.25 APIM::FrontEnd::App::Views::AdminAPIManager

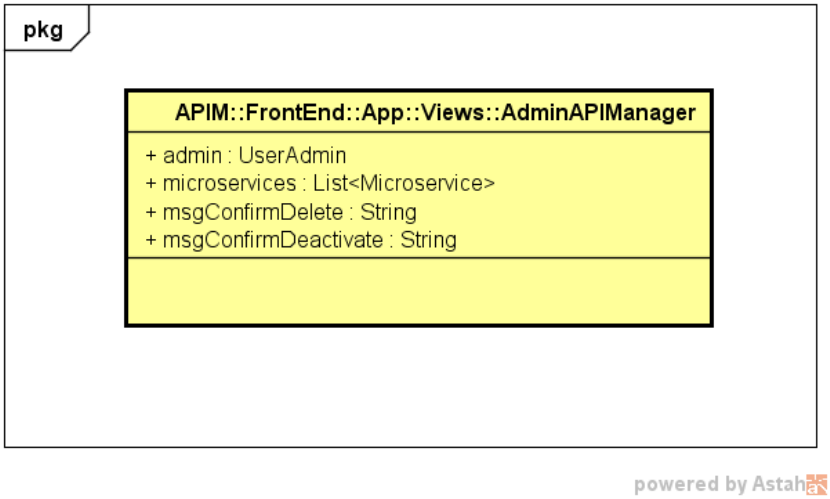

Figura 32: APIM::FrontEnd::App::Views::AdminAPIManager

- **Descrizione:** View contenente le operazioni dell'amministratore sulle API dell'API Market.
- **Attributi:**
  - **admin : UserAdmin**
    Campo dati contenente le informazioni di un admin;
  - **microservices : List<Microservice>**
    Campo dati contenente una lista di microservizi;
  - **msgConfirmDelete : string**
    Campo dati contenente un messaggio di conferma eliminazione di un API.
  - **msgConfirmDeactivate : string**
    Campo dati contenente un messaggio di conferma disattivazione di un API.
- **Relazioni con altre classi:**
  - Interagisce con il controller **AdminAPIManagerController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio.

3.3.2.26 APIM::FrontEnd::App::Views::AdminLogin

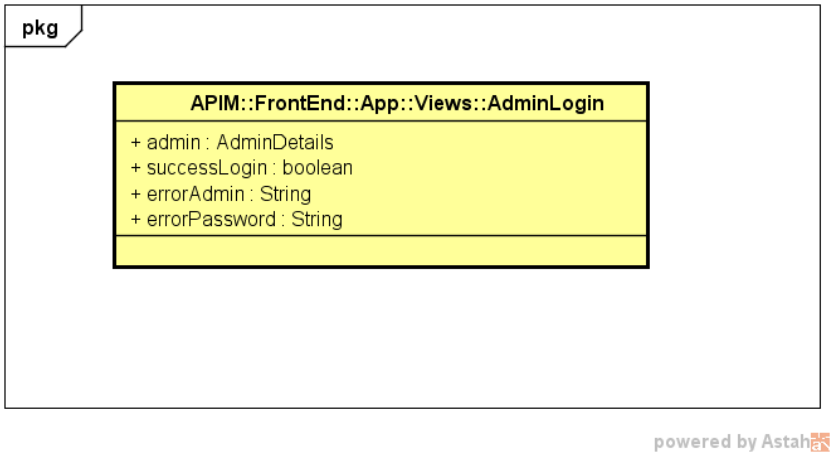

Figura 33: APIM::FrontEnd::App::Views::AdminLogin

- **Descrizione:** View contenente il form necessario affinchè l'amministratore possa effettuare il login ed autenticarsi al sistema.

- **Attributi:**
  - **admin : AdminDetails**
    Campo dati contenente le informazioni di un admin;
  - **successLogin : boolean**
    Campo dati contenente il flag per il login di un amministratore;
  - **errorAdmin : string**
    Campo dati contenente un messaggio di errore di autenticazione dell'admin;
  - **errorPassword : string**
    Campo dati contenente un messaggio di errore password errata.
- **Relazioni con altre classi:**
  - Interagisce con il controller **AdminLoginController**;

3.3.2.27 APIM::FrontEnd::App::Views::Logout

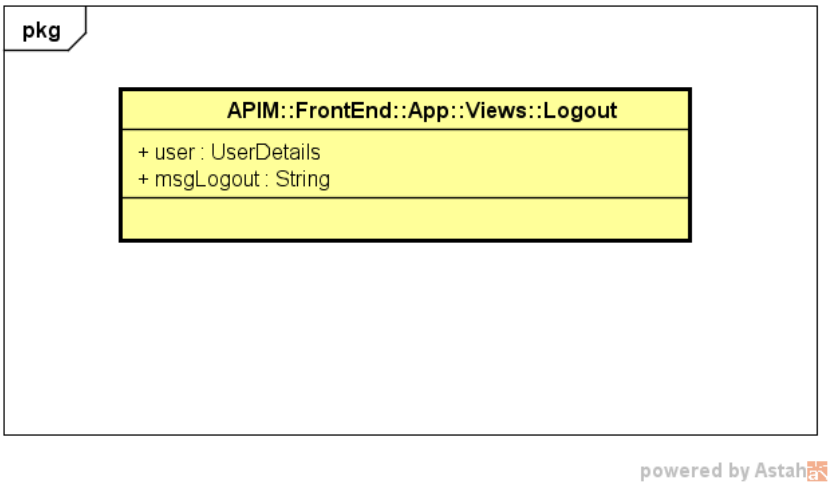

Figura 34: APIM::FrontEnd::App::Views::Logout

- **Descrizione:** View contenente la conferma di logout dall'API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente le informazioni di un utente;
  - **msgLogout : string**
    Campo dati contenente un messaggio di avvenuto logout dalla piattaforma;
- **Relazioni con altre classi:**
  - Interagisce con il controller **LogoutController**;

3.3.2.28 APIM::FrontEnd::App::Views::UpdateInfoAPI

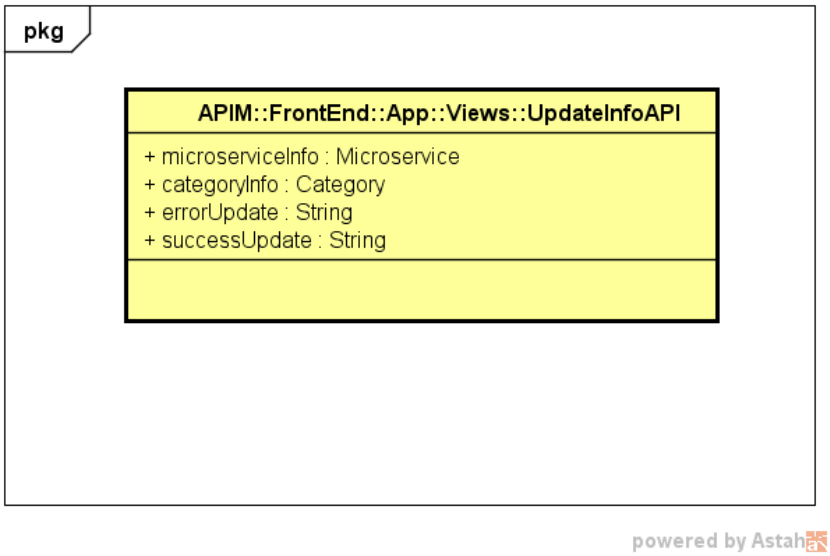

Figura 35: APIM::FrontEnd::App::Views::UpdateInfoAPI

- **Descrizione:** View contenente il form dedicato modifica delle informazioni di una API presente nel marketplace API Market.
- **Attributi:**
  - **microserviceInfo : Microservice**
    Campo dati contenente le informazioni di un microservizio;
  - **categoryInfo : category**
    Campo dati contenente le informazioni della categoria di un microservizio;
  - **errorUpdate : string**
    Campo dati contenente un messaggio di errore di modifica delle informazioni API;
  - **successUpdate : string**
    Campo dati contenente un messaggio di avvenuta modifica delle informazioni.
- **Relazioni con altre classi:**
  - Interagisce con il controller **UpdateInfoAPIController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio.

3.3.2.29 APIM::FrontEnd::App::Views::RechargeCredits

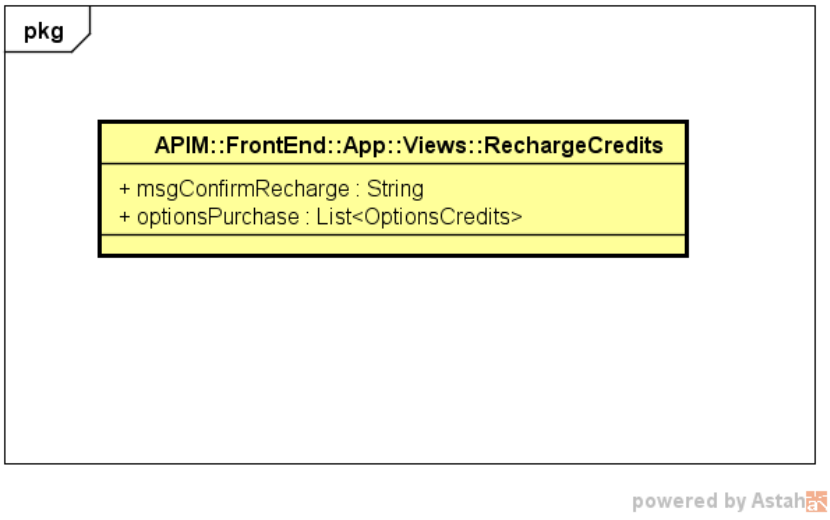

Figura 36: APIM::FrontEnd::App::Views::RechargeCredits

- **Descrizione:** View contenente la possibilità di scegliere quanti crediti ricaricare sul conto personale tra i tagli disponibili.
- **Attributi:**
  - **msgConfirmRecharge : string**
    Campo dati contenente un messaggio di conferma della ricarica crediti;
  - **optionsPurchase : List<OptionsCredits>**
    Campo dati contenente le opzioni di acquisto dei crediti previste.
- **Relazioni con altre classi:**
  - Interagisce con il controller **RechargeCreditsController**.

3.3.2.30 APIM::FrontEnd::App::Views::ShowUserDetails

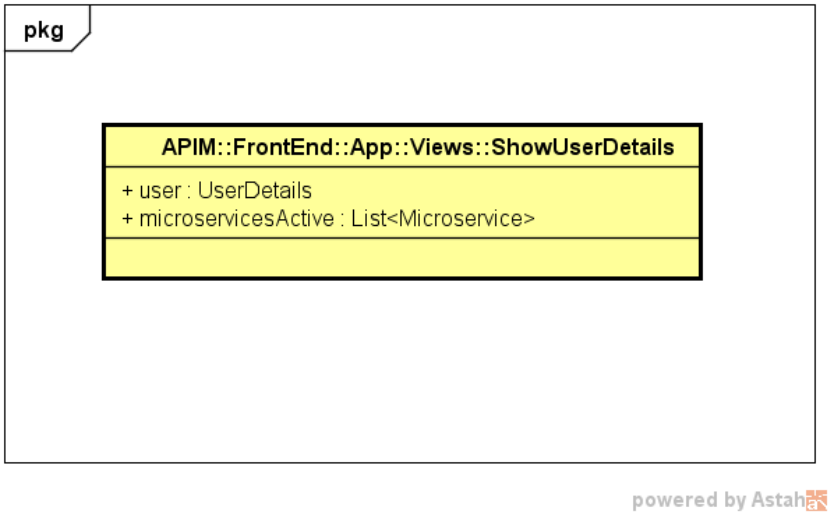

Figura 37: APIM::FrontEnd::App::Views::ShowUserDetails

- **Descrizione:** View contenente le informazioni personali di un utente visualizzabili da altri fruitori del marketplace API Market.
- **Attributi:**
  - **user : UserDetails**
    Campo dati contenente i dati di utente;
  - **microservicesActive : List<Microservice>**
    Campo dati contenente la lista dei microservizi attivi per il dato utente.
- **Relazioni con altre classi:**
  - Interagisce con il controller **ShowUserDetailsController**;
  - Il model **MicroserviceModel** contiene le informazioni per rappresentare un microservizio;
  - Il model **UserDetailsModel** contiene le informazioni per rappresentare un utente.

3.4 APIM::FrontEnd::App::Models

3.4.1 Informazioni generali

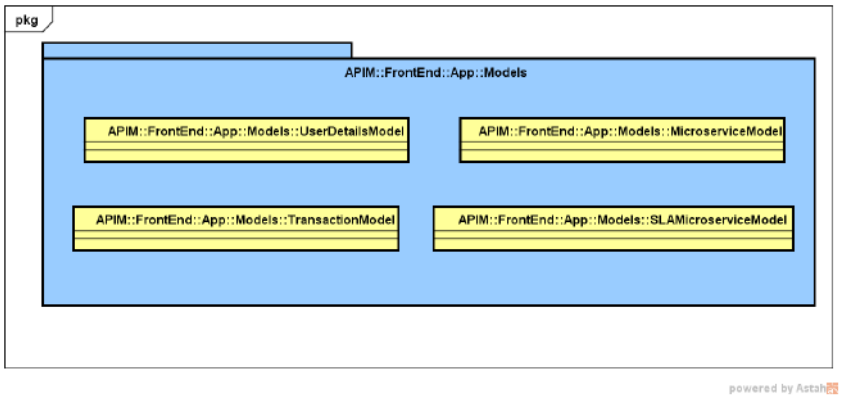

Figura 38: APIM::FrontEnd:App:Models

- **Descrizione:** Il package Models contiene le classi che definiscono le strutture dei dati di utenti, microservizi, transazioni e sondaggi SLA.
- **Relazioni con altre classi:**
  - Interagisce con i package **Controllers** e **Views** per garantire il dynamic binding di AngularJS.

3.4.2 Classi

3.4.2.1 APIM::FrontEnd::App::Models::UserDetailsModel

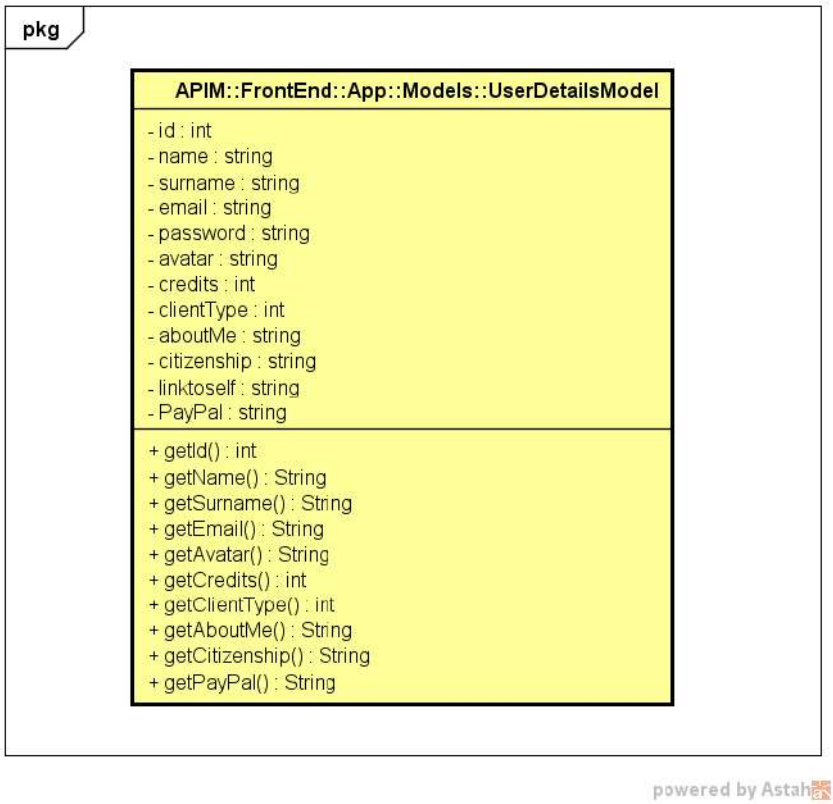

Figura 39: APIM::FrontEnd::App::Models:UserDetailsModel

- **Descrizione:** UserDetailsModel che rappresenta un utente e che contiene tutte le informazioni necessarie alla presentazione del contenuto di un utente, sia nella visualizzazione che nella gestione di un profilo.
- **Attributi:**
  - **id : int**
    Id dell'utente;
  - **name : string**
    Nome dell'utente;
  - **surname : string**
    Cognome dell'utente;
  - **email : string**
    Email dell'utente;
  - **password : string**
    Password dell'utente;
  - **avatar : string**
    Avatar dell'utente;
  - **registration : string**
    Data di registrazione dell'utente;

- **credits : int**
  Crediti dell'utente;
- **clientType : int**
  Tipo di account dell'utente;
- **aboutMe : string**
  AboutMe dell'utente sviluppatore;
- **citizenship : string**
  Cittadinanza dell'utente sviluppatore;
- **linkToSelf : string**
  Link al sito esterno dell'utente sviluppatore;
- **paypal : string**
  Email paypal dell'utente sviluppatore.

- **Relazioni con altre classi:**
  - Interagisce con il controller **LoginController**;
  - Interagisce con il controller **APIController**;
  - Interagisce con il controller **ProfileManagerController**;
  - Interagisce con il controller **ResetPasswordController**;
  - Interagisce con il controller **PasswordRecoveryController**;

3.4.2.2 APIM::FrontEnd::App::Models::TransactionModel

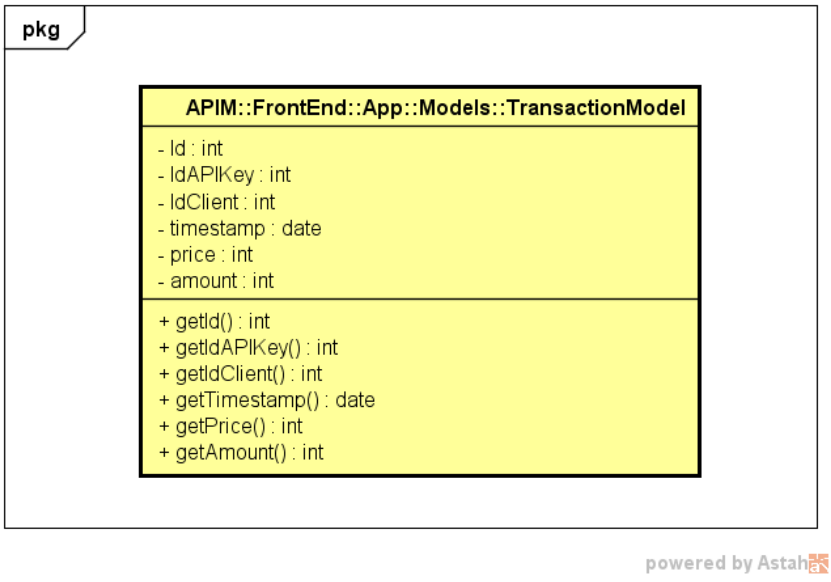

Figura 40: APIM::FrontEnd::App::Models::TransactionModel

- **Descrizione:** TransactionModel rappresenta una transazione avvenuta e che contiene tutte le informazioni necessarie alla presentazione del contenuto di una transazione, sia nella visualizzazione che nella gestione.

- **Attributi:**
  - **id : int**
    Id della transazione;

- **idAPIKey : int**
  Id dell'apikey;
- **idClient : int**
  Id del cliente;
- **timestamp : int**
  Data ed ora della transazione;
- **price : int**
  Prezzo della transazione;
- **amount : int**
  Ammontare quantitativo della transazione.

- **Relazioni con altre classi:**
  - Interagisce con il controller **TransactionsListController**;
  - Interagisce con il controller **APIPurchasedController**.

3.4.2.3 APIM::FrontEnd::App::Models::MicroserviceModel

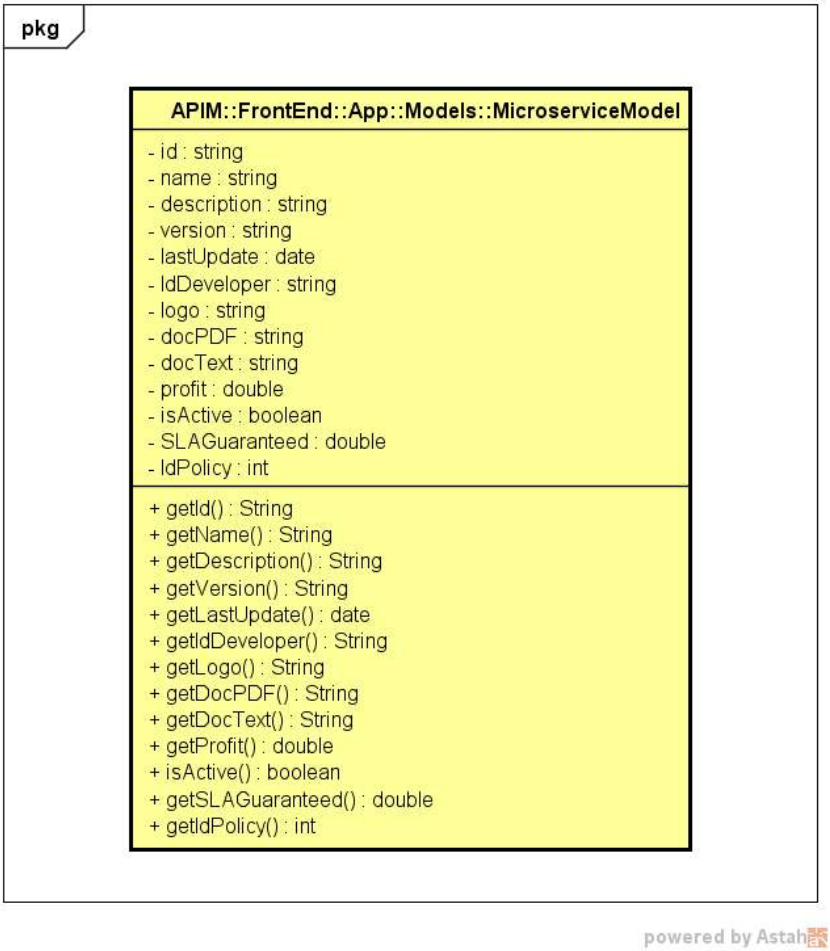

Figura 41: APIMarket::FrontEnd::App::Models::MicroserviceModel

- **Descrizione:** MicroserviceModel rappresenta un microservizio e che contiene tutte le informazioni necessarie alla presentazione del contenuto di un microservizio, sia nella visualizzazione che nella gestione.

- **Attributi:**
  - **id : int**
    Id del microservizio;
  - **name : string**
    Nome del microservizio;
  - **description : string**
    Descrizione del microservizio;
  - **version : string**
    Versione corrente del microservizio;
  - **lastUpdate : string**
    Data ultimo aggiornamento del microservizio;
  - **idDeveloper : string**
    Id dello sviluppatore del microservizio;
  - **logo : string**
    Link al logo del microservizio;
  - **docPDF : string**
    Link al file PDF del microservizio;
  - **docExt : string**
    Link alla documentazione esterna del microservizio;
  - **profit : int**
    Percentuale profitto dello sviluppatore del microservizio;
  - **isActive : bool**
    Funzionamento attuale del microservizio;
  - **SLAGuaranteed : double**
    SLA garantita dal microservizio;
  - **idPolicy : int**
    Id della policy di vendita del microservizio.
- **Relazioni con altre classi:**
  - Interagisce con il controller **APIRegisteredController**;
  - Interagisce con il controller **RegisterAPIController**.
  - Interagisce con il controller **PolicyCallController**;
  - Interagisce con il controller **PolicyTimeController**;
  - Interagisce con il controller **PolicyTrafficController**;
  - Interagisce con il controller **APIPurchasedController**;
  - Interagisce con il controller **APIListController**.

3.4.2.4 APIM::FrontEnd::App::Models::SLAMicroserviceModel

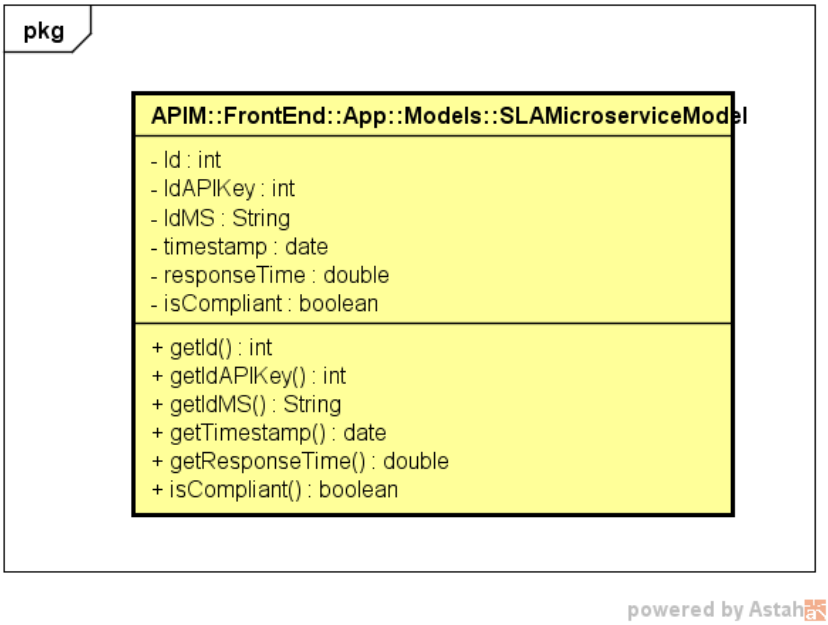

Figura 42: APIM::FrontEnd::App::Models::SLAMicroserviceModel

- **Descrizione:** SLAMicroserviceModel rappresenta la SLA di un microservizio e che contiene tutte le informazioni necessarie alla presentazione del contenuto di SLA di un microservizio, sia nella visualizzazione che nella gestione.
- **Attributi:**
  - **id : int**
    Id del sondaggio SLA;
  - **idapikey : int**
    Id dell'apike attribuita al sondaggio SLA;
  - **idMS : int**
    Id del microservizio attribuito al sondaggio SLA;
  - **timestamp : string**
    Data ed ora del sondaggio SLA;
  - **responseTime : int**
    Tempo di risposta del sondaggio SLA;
  - **isCompliant : boolean**
    Indicatore del rispetto del sondaggio SLA.
- **Relazioni con altre classi:**
  - Interagisce con il controller **PolicyCallController**;
  - Interagisce con il controller **PolicyTimeController**;
  - Interagisce con il controller **PolicyTrafficController**;
  - Interagisce con il controller **RegisterAPIController**.

3.5 APIM::FrontEnd::App::Controllers

3.5.1 Informazioni generali

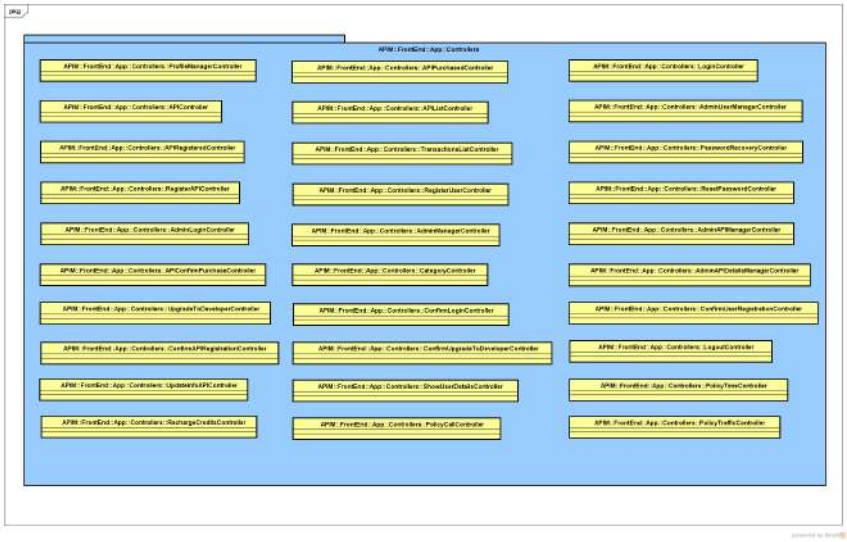

Figura 43: APIM::FrontEnd::App::Controllers

- **Descrizione:** Il package Controllers contiene tutti i controller dell'applicazione.
- **Relazioni con altre classi:**
  - Il package **View** è collegato ad un controller per gestirne la visualizzazione delle pagine ed il routing;
  - Il package **Models** contiene le strutture dati cui i controllers si riferiscono.

3.5.2 Classi

3.5.2.1 APIM::FrontEnd::App::Controllers::AdminAPIDetailsManagerController

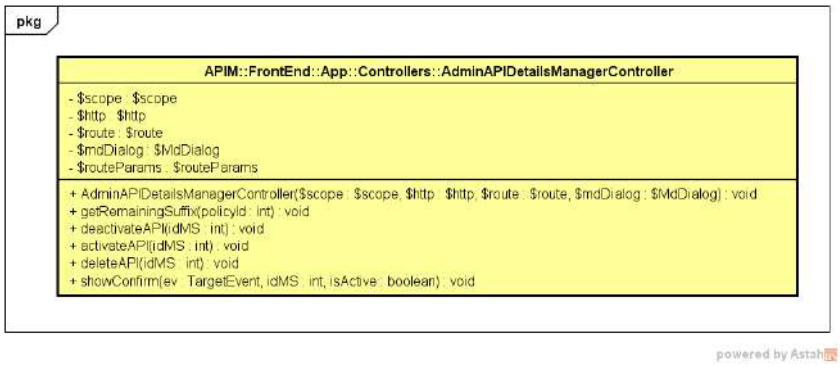

Figura 44: APIM::FrontEnd::App::Controllers::AdminAPIDetailsManagerController

- **Descrizione:** classe che permette di visualizzare le statistiche di un'API, attivarla, disattivarla o eliminarla.
- **Attributi:**

- **$scope : $scope**
  Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
- **$http : $http**
  Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **$route : $route**
  Campo dati contenente il riferimento all'oggetto globale $route creato da AngularJS. Viene utilizzato per le funzioni di routing;
- **$mdDialog : $mdDialog**
  Campo dati contenente il riferimento all'oggetto globale $mdDialog creato da AngularJS. Viene utilizzato per creare finestre di popup;
- **$routeParams : $routeParams**
  Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;

- **Metodi:**
  - **AdminAPIDetailsManagerController($scope : $scope, $http : $http, $route : $route, $mdDialog : $mdDialog, $routeParams : $routeParams)**
    Metodo costruttore della classe.

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    - **$route : $route**
      Parametro che contiene il riferimento all'oggetto globale $route di AngularJS. Viene utilizzato per le funzioni di routing.
    - **$mdDialog : $mdDialog**
      Parametro che contiene il riferimento all'oggetto globale $mdDialog di AngularJS. Viene utilizzato per creare finestre di popup.
    - **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.
  - **getRemainingSuffix(policyId : int) : void**
    Metodo per identificare la tipologia di policy dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:
    - **policyId : int**
      Identificativo del tipo di policy;
  - **deactivateAPI(IdMS : int) : void**
    Metodo per disattivare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **activateAPI(IdMS : int) : void**
    Metodo per attivare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **deleteAPI(IdMS : int) : void**
    Metodo per eliminare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **showConfirm(ev : TargetEvent, IdMS : int, isActive : boolean) : void**
    Metodo per mostrare una finestra di conferma di elimninazione dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **ev : TargetEvent**
      Identifica il target che ha fatto scatenare l'evento di apertura del popup;
    - **IdMS : int**
      Identificativo dell'API;
    - **boolean : isActive**
      Identifica lo stato dell'API;

- **Relazioni con altre classi:**

  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **AdminAPIDetails**;

3.5.2.2 APIM::FrontEnd::App::Controllers::AdminAPIManagerController

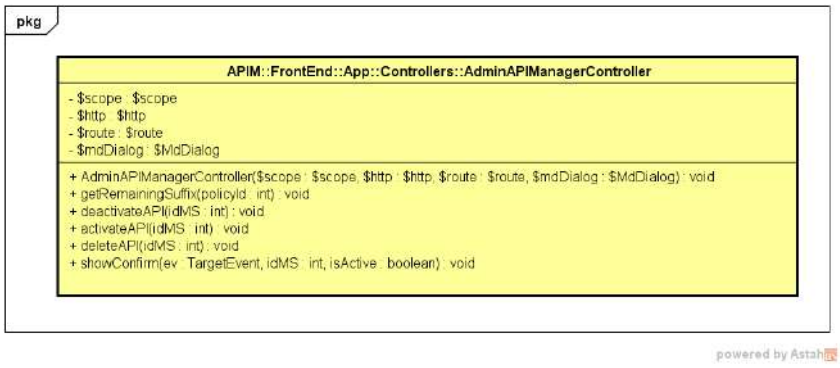

Figura 45: APIM::FrontEnd::App::Controllers::AdminAPIManagerController

- **Descrizione:** classe che permette di fornire la lista di tutte le API presenti e di effettuare alcune operazioni come attivazione, disattivazione ed eliminazione.

- **Attributi:**

- **$scope : $scope**
  Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
- **$http : $http**
  Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **$route : $route**
  Campo dati contenente il riferimento all'oggetto globale $route creato da AngularJS. Viene utilizzato per le funzioni di routing;
- **$mdDialog : $mdDialog**
  Campo dati contenente il riferimento all'oggetto globale $mdDialog creato da AngularJS. Viene utilizzato per creare finestre di popup;
- **$routeParams : $routeParams**
  Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;

- **Metodi:**
  - **AdminAPIManagerController($scope : $scope, $http : $http, $route : $route, $mdDialog : $mdDialog, $routeParams : $routeParams)**
    Metodo costruttore della classe;

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    - **$route : $route**
      Parametro che contiene il riferimento all'oggetto globale $route di AngularJS. Viene utilizzato per le funzioni di routing.
    - **$mdDialog : $mdDialog**
      Parametro che contiene il riferimento all'oggetto globale $mdDialog di AngularJS. Viene utilizzato per creare finestre di popup.
    - **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.
  - **getRemainingSuffix(policyId : int) : void**
    Metodo per identificare la tipologia di policy dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:
    - **policyId : int**
      Identificativo del tipo di policy;
  - **deactivateAPI(IdMS : int) : void**
    Metodo per disattivare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **activateAPI(IdMS : int) : void**
    Metodo per attivare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **deleteAPI(IdMS : int) : void**
    Metodo per eliminare l'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **IdMS : int**
      Identificativo dell'API;

  - **showConfirm(ev : TargetEvent, IdMS : int, isActive : boolean) : void**
    Metodo per mostrare una finestra di conferma di elimninazione dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **ev : TargetEvent**
      Identifica il target che ha fatto scatenare l'evento di apertura del popup;
    - **IdMS : int**
      Identificativo dell'API;
    - **boolean : isActive**
      Identifica lo stato dell'API;

- **Relazioni con altre classi:**

  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **AdminAPIDetails**;

3.5.2.3 APIM::FrontEnd::App::Controllers::AdminLoginController

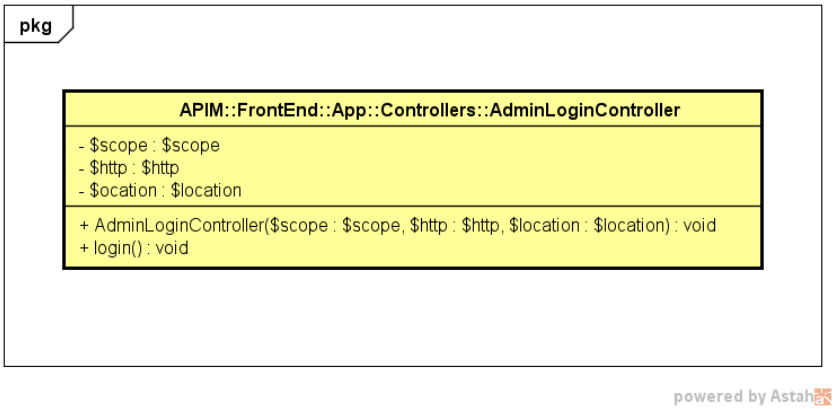

Figura 46: APIM::FrontEnd::App::Controllers::AdminLoginController

- **Descrizione:** AdminLoginController permette di effettuare il login all'area di amministrazione.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per le funzioni di routing;
- **Metodi:**
  - **AdminLoginController($scope : $scope, $http : $http, $location : $location)**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    * **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per le funzioni di routing.
  - **login() : void**
    Metodo per effettuare il login. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:
    * **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **AdminLogin**;

3.5.2.4 APIM::FrontEnd::App::Controllers::AdminUserManagerController

Figura 47: APIM::FrontEnd::App::Controllers::AdminUserManagerController

- **Descrizione:** classe che permette di gestire/moderare un utente (sostanzialmente eliminarlo).
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$route : $route**
    Campo dati contenente il riferimento all'oggetto globale $route creato da AngularJS. Viene utilizzato per le funzioni di routing;
  - **$mdDialog : $mdDialog**
    Campo dati contenente il riferimento all'oggetto globale $mdDialog creato da AngularJS. Viene utilizzato per creare finestre di popup;
- **Metodi:**
  - **AdminUserManagerController($scope : $scope, $http : $http, $route : $route, $mdDialog : $mdDialog)**
    Metodo costruttore della classe;

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    - **$route : $route**
      Parametro che contiene il riferimento all'oggetto globale $route di AngularJS. Viene utilizzato per le funzioni di routing.

    - **$mdDialog : $mdDialog**
      Parametro che contiene il riferimento all'oggetto globale $mdDialog di AngularJS. Viene utilizzato per creare finestre di popup.
  - **showConfirm(ev : TargetEvent, IdClient : int, isActivate : boolean) : void**
    Metodo per mostrare una finestra di conferma di elimninazione dell'utente. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **ev : TargetEvent**
      Identifica il target che ha fatto scatenare l'evento di apertura del popup;
    - **idClient : int**
      Identificativo dell'utente;
    - **numActive : int**
      Numero di API attive per quel utente;

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **AdminUserManager**;

3.5.2.5 APIM::FrontEnd::App::Controllers::APIConfirmPurchaseController

Figura 48: APIM::FrontEnd::App::Controllers::APIConfirmPurchaseController

- **Descrizione:** classe che permette di effettuare l'acquisto di una licenza per una API da parte di un cliente.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per le funzioni di routing;
  - **$routeParams : $routeParams**
    Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Metodi:**
  - **APIConfirmPurchaseController($scope : $scope, $http : $http, $route : $route, $location : $location, $routeParams : $routeParams)**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    * **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.
- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **APIConfirmPurchase**;

3.5.2.6 APIM::FrontEnd::App::Controllers::APIController

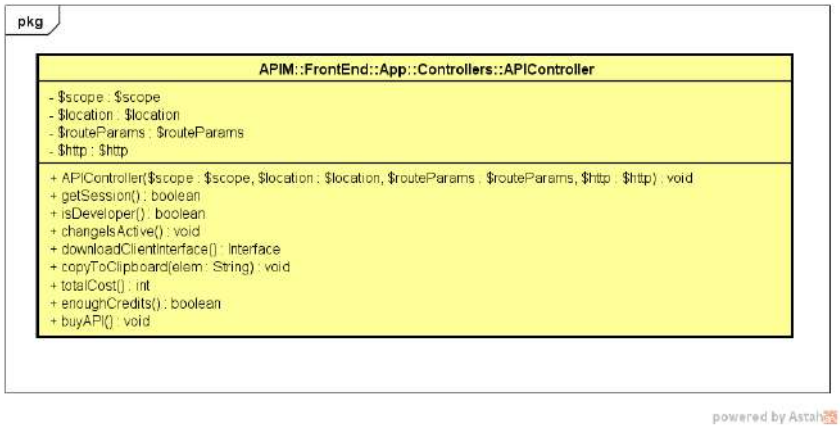

Figura 49: APIM::FrontEnd::App::Controllers::APIController

- **Descrizione:** APIController permette di ricavare tutte le informazioni riguardanti un'API e di effettuare l'acquisto di una licenza.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **$location : $location**
  Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per le funzioni di routing;
- **$routeParams : $routeParams**
  Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Metodi:**
  - **APIController($scope : $scope, $http : $http, $location : $location, $routeParams : $routeParams)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    * **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.
    * **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per le funzioni di routing.
  - **isDeveloper() : boolean**
    Metodo per verificare se l'utente che sta accedendo ai dati è il developer che la ha registrata;

    Parametri:

    * **void**

  - **changeIsActive() : void**
    Metodo per attivare lo stato attivo/non attivo dell'API;

    Parametri:

    * **void**

  - **downloadClientInterface() : interface**
    Metodo per scaricare il file contenente l'interfaccia dell'API;

    Parametri:

    * **void**

  - **copyToClipboard(elem : string) : void**
    Metodo per copiare le informazioni dell'API;

    Parametri:

    * **void**

- **totalCost() : int**
  Metodo per calcolare il costo totale della licenza;
  Parametri:
  * **void**

- **enoughCredits() : boolean**
  Metodo per verificare se l'utente che vuole acquistare la licenza ha abbastanza crediti;
  Parametri:
  * **void**

- **buyAPI() : void**
  Metodo per verificare effettuare la procedura di acquisto dell'API;
  Parametri:
  * **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **API**;

3.5.2.7 APIM::FrontEnd::App::Controllers::APIListController

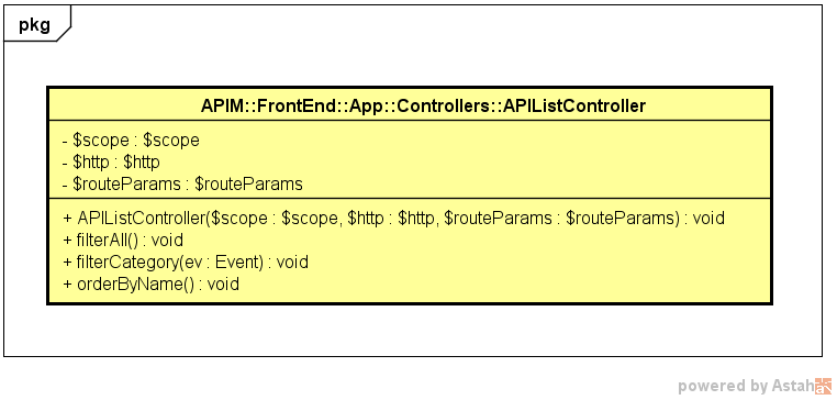

Figura 50: APIM::FrontEnd::App::Controllers::APIListController

- **Descrizione:** APIListController permette di ricavare la lista di tutte le API, di ordinare e di filtrare i risultati.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

  - **$routeParams : $routeParams**
    Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Metodi:**

  - **APIListController($scope : $scope, $http : $http, $routeParams : $routeParams)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    * **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

  - **filterAll() : void**
    Metodo per visualizzare tutt le categorie;

    Parametri:

    * **void**

  - **filterCategory(ev : Event) : void**
    Metodo per filtrare solo le categorie selezionate;

    Parametri:

    * **void**

  - **orderByName() : void**
    Metodo per ordinare le API in modo alfabetico;

    Parametri:

    * **void**

- **Relazioni con altre classi:**

  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **APIList**;

3.5.2.8 APIM::FrontEnd::App::Controllers::APIPurchasedController

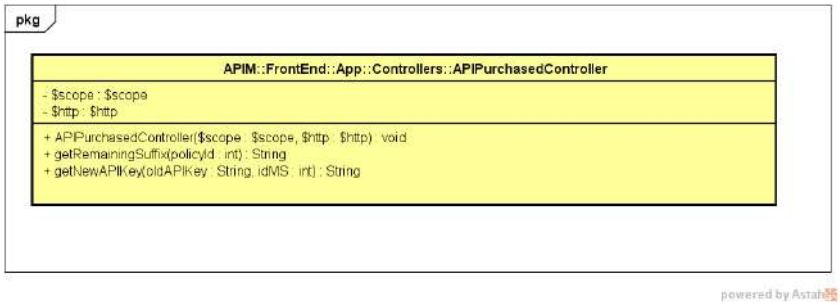

Figura 51: APIM::FrontEnd::App::Controllers::APIPurchasedController

- **Descrizione:** APIPurchasedController permette di vedere lo stato di un API di cui si possiede una licenza.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **APIPurchasedController($scope : $scope, $http : $http)**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
  - **getRemainingSuffix(policyId : int) : void**
    Metodo per ricavare la tipologia di policy acquistata;

    Parametri:
    * **policyId : int**
      Idetifica la tipologia di policy.
  - **getNewAPIKey(oldAPIKey : String, IdMS : int) : String**
    Metodo per generare una nuova API key;

    Parametri:
    * **oldAPIKey : String**
      Vecchia API key;

  * **IdMS : int**
    Identificativo dell'API;

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **APIPurchased**;

3.5.2.9 APIM::FrontEnd::App::Controllers::APIRegisteredController

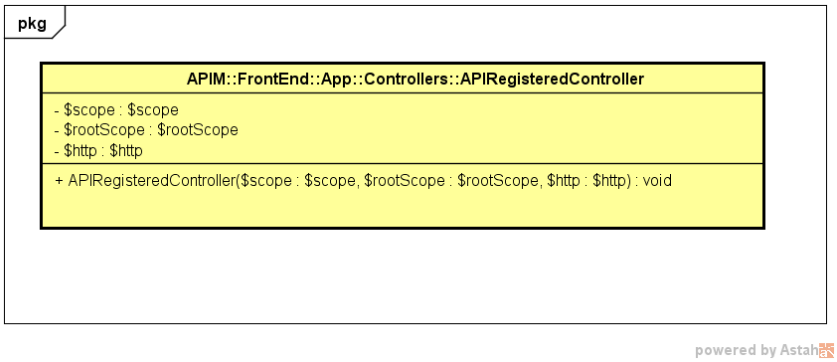

Figura 52: APIM::FrontEnd::App::Controllers::APIRegisteredController

- **Descrizione:** APIRegisteredController permette di ricavare lo stato di un API di cui si ha effettuato la registrazione.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$rootScope : $rootScope**
    Campo dati contenente un riferimento all'oggetto $rootScope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**
  - **APIRegisteredController($scope : $scope, $rootScope : $rootScope, $http : $http)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;

  - **$rootScope : $rootScope**
    Parametro che contiene un riferimento all'oggetto $rootScope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
  - **$http : $http**
    Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **APIRegistered**;

3.5.2.10 APIM::FrontEnd::App::Controllers::CategoryController

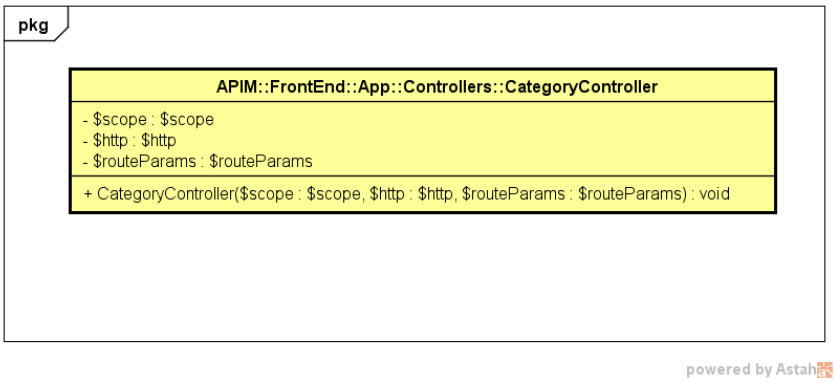

Figura 53: APIM::FrontEnd::App::Controllers::CategoryController

- **Descrizione:** CategoryController permette di ricavare le API di una singola categoria.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$routeParams : $routeParams**
    Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;
- **Metodi:**
  - **CategoryController($scope : $scope, $http : $http, $routeParams : $routeParams)**
    Metodo costruttore della classe;

    Parametri:

  - ∗ **$scope : $scope**
    Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
  - ∗ **$http : $http**
    Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
  - ∗ **$routeParams : $routeParams**
    Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **Category**;

3.5.2.11 APIM::FrontEnd::App::Controllers::ConfirmAPIRegistrationController

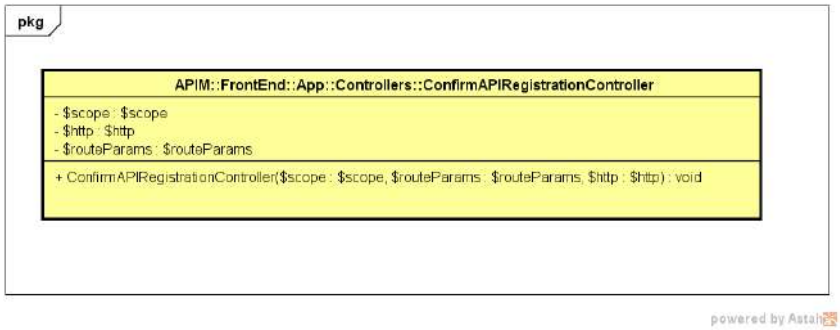

Figura 54: APIM::FrontEnd::App::Controllers::ConfirmAPIRegistrationController

- **Descrizione:** ConfirmAPIRegistrationController permette di effettuare la conferma di registrazione di un API da parte di un utente sviluppatore.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$routeParams : $routeParams**
    Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;

- **Metodi:**
  - **ConfirmAPIRegistrationController($scope : $scope, $http : $http, $routeParams : $routeParams)**
    Metodo costruttore della classe;

Parametri:

* **$scope : $scope**
  Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
* **$http : $http**
  Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
* **$routeParams : $routeParams**
  Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **ConfirmAPIRegistration**;

3.5.2.12 APIM::FrontEnd::App::Controllers::ConfirmUpgradeToDeveloperController

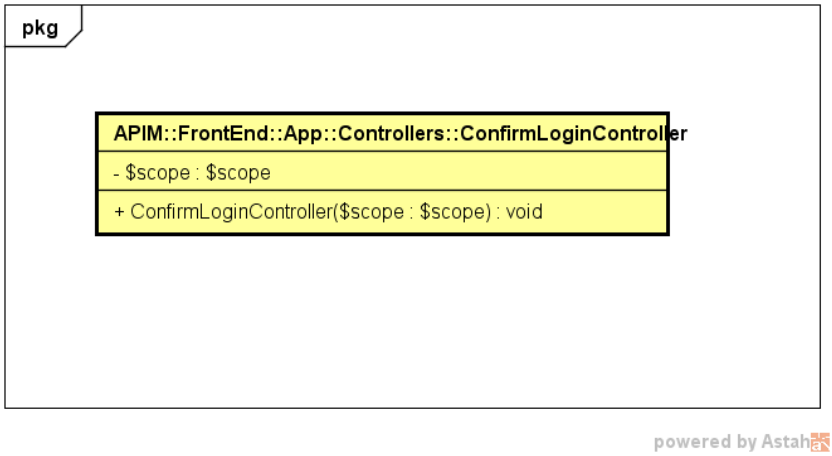

Figura 55: APIM::FrontEnd::App::Controllers::ConfirmLoginController

- **Descrizione:** ConfirmLoginController permette di confermare l'upgrade da utente cliente a utente sviluppatore.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **ConfirmLoginControllerl($scope : $scope, $http : $http)**
    Metodo costruttore della classe;

Parametri:

* **$scope : $scope**
  Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
* **$http : $http**
  Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **ConfirmUpgradeToDeveloper**;

3.5.2.13 APIM::FrontEnd::App::Controllers::ConfirmUserRegistrationController

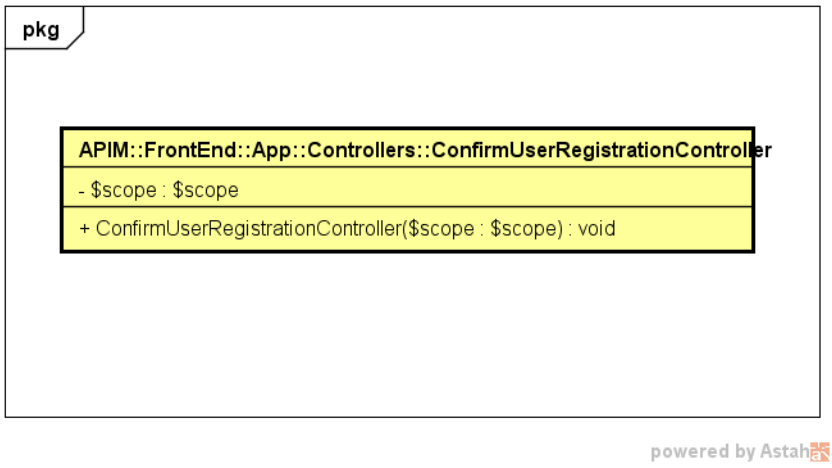

Figura 56: APIM::FrontEnd::App::Controllers::ConfirmUserRegistrationController

- **Descrizione:** ConfirmUserRegistrationController permette di confermare la registrazione di un utente.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
- **Metodi:**
  - **ConfirmUserRegistrationController($scope : $scope)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **ConfirmUserRegistration**;

3.5.2.14 APIM::FrontEnd::App::Controllers::LoginController

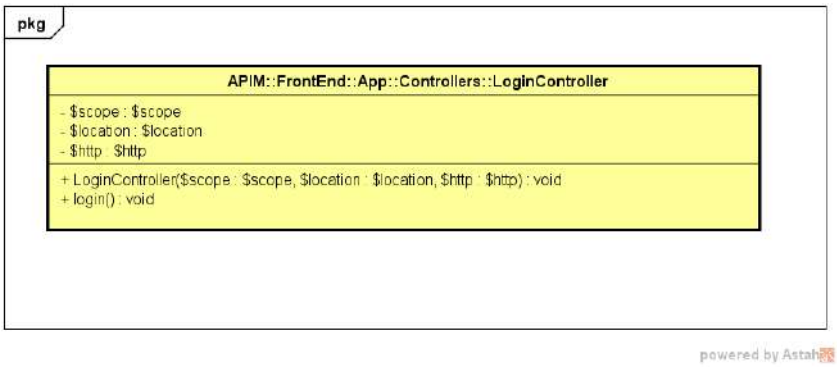

Figura 57: APIM::FrontEnd::App::Controllers::LoginController

- **Descrizione:** LoginController permette di effettuare il login di clienti e sviluppatori.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **LoginController($scope : $scope, $location : $location, $http : $http)**
    Metodo costruttore della classe;

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **login() : void**
  Metodo costruttore della classe;

  Parametri:

  * **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **Login**;

3.5.2.15 APIM::FrontEnd::App::Controllers::LogoutController

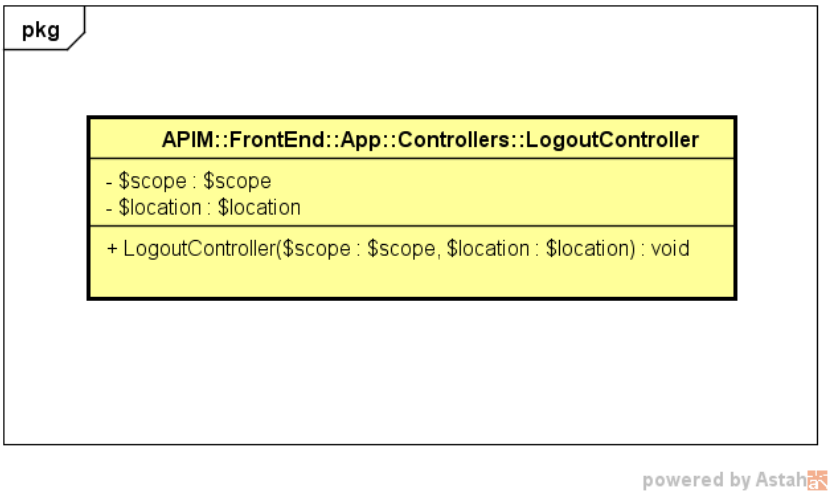

Figura 58: APIM::FrontEnd::App::Controllers::LogoutController

- **Descrizione:** LogoutController permette di effettuare il logout di utenti autenticati alla piattaforma.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
- **Metodi:**
  - **LogoutController($scope : $scope, $location : $location)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;

  - ∗ **$location : $location**
    Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **Logout**;

3.5.2.16 APIM::FrontEnd::App::Controllers::PasswordRecoveryController

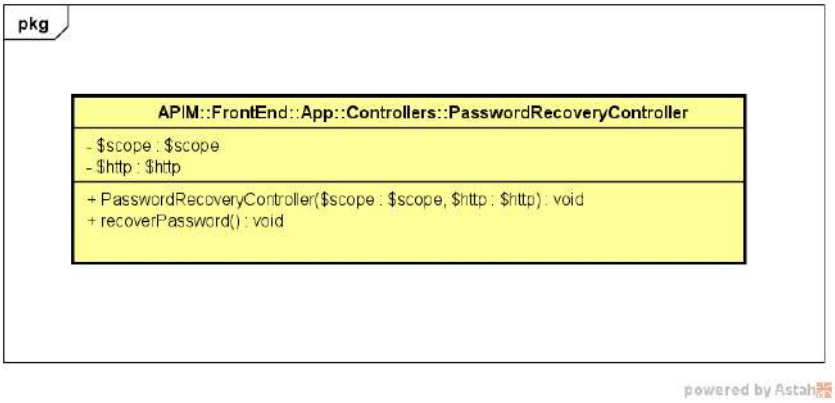

Figura 59: APIM::FrontEnd::App::Controllers::PasswordRecoveryController

- **Descrizione:** PasswordRecoveryController permette di effettuare il recupero password di clienti e sviluppatori.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**
  - **PasswordRecoveryController($scope : $scope, $http : $http)**
    Metodo costruttore della classe;

    Parametri:
    - ∗ **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - ∗ **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **recoverPassword() : void**
  Metodo che, attraverso la chiamata all'opportuno microservizio, avvia la procedura di generazione di una nuova password;

  Parametri:

  * **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **PasswordRecovery**;

3.5.2.17 APIM::FrontEnd::App::Controllers::ResetPasswordController

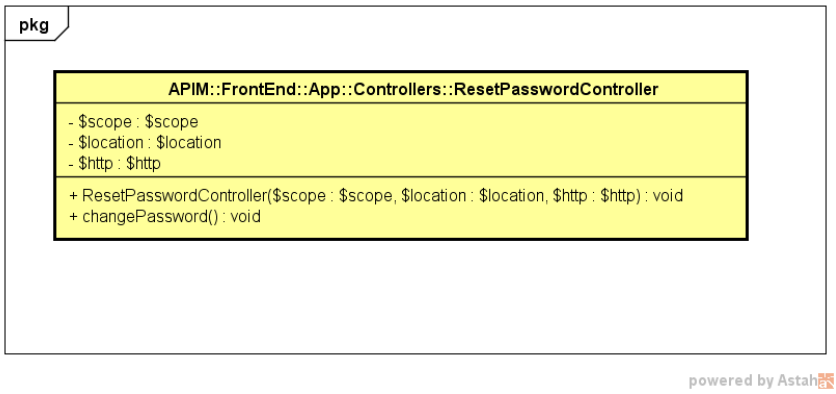

Figura 60: APIM::FrontEnd::App::Controllers::ResetPasswordController

- **Descrizione:** classe che permette di effettuare il cambio password di clienti e sviluppatori autenticati al sistema.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$location : $location**
    Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
- **Metodi:**
  - **ResetPasswordController($scope : $scope, $http : $http, $location : $location)**
    Metodo costruttore della classe;

    Parametri:

    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP;
    - **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
  - **changePassword() : void**
    Metodo che consente di impostare una nuova password per l'utente;

    Parametri:
    - **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **ResetPassword**;

3.5.2.18 APIM::FrontEnd::App::Controllers::PolicyCallController

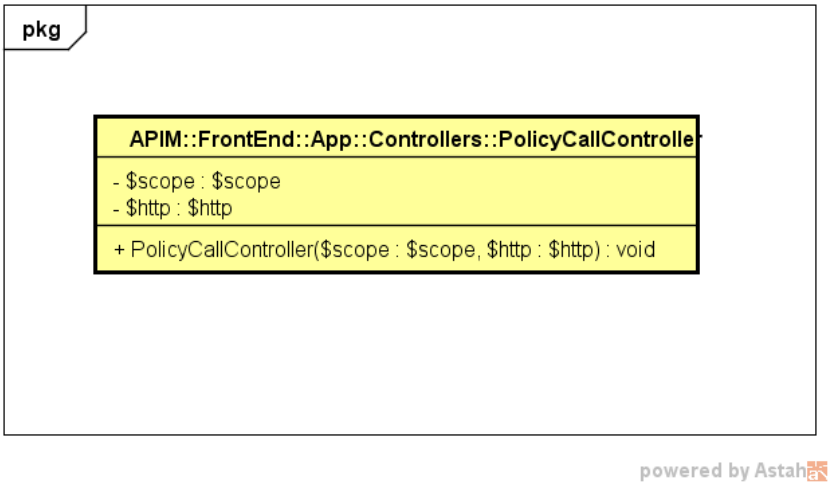

Figura 61: APIM::FrontEnd::App::Controllers::PolicyCallController

- **Descrizione:** PolicyCallController permette di ricavare le informazioni riguardanti la tipologia di policy a chiamata.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**
  - **PolicyCallController($scope : $scope, $http : $http)**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **PolicyCall**;

3.5.2.19 APIM::FrontEnd::App::Controllers::PolicyTimeController

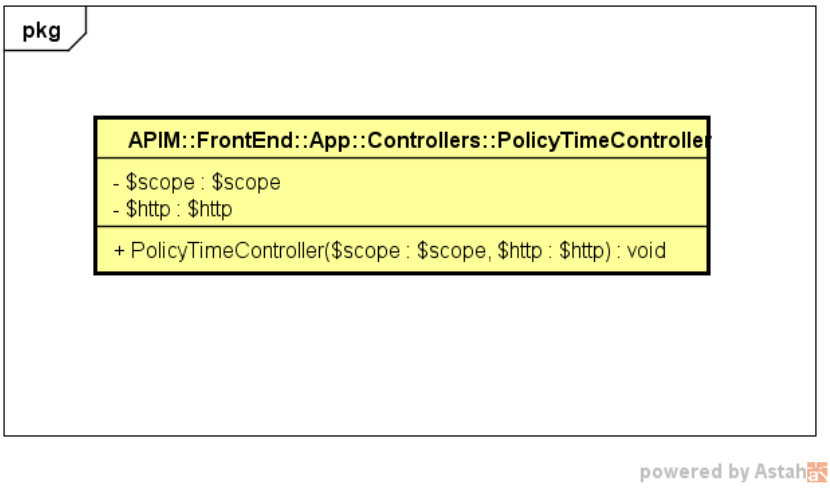

Figura 62: APIM::FrontEnd::App::Controllers::PolicyTimeController

- **Descrizione:** PolicyTimeController permette di ricavare le informazioni riguardanti la tipologia di policy a tempo.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**

- **PolicyTimeController($scope : $scope, $http : $http)**
  Metodo costruttore della classe;

  Parametri:

  * **$scope : $scope**
    Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
  * **$http : $http**
    Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **PolicyTime**;

3.5.2.20 APIM::FrontEnd::App::Controllers::PolicyTrafficController

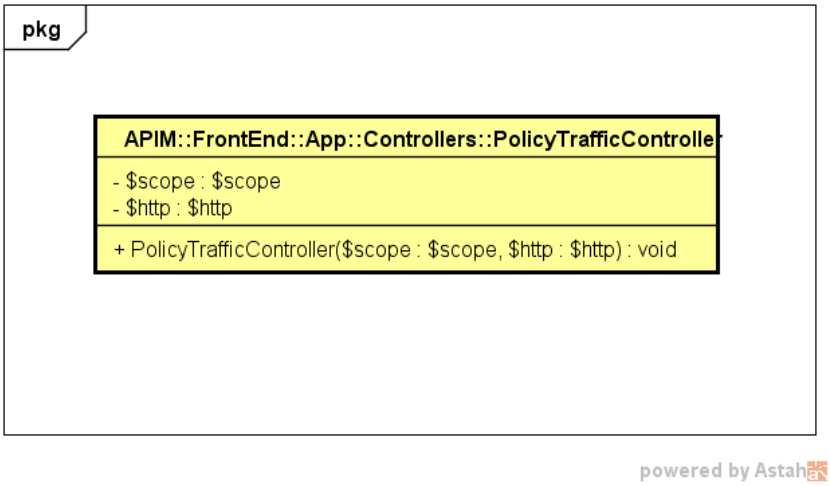

Figura 63: APIM::FrontEnd::App::Controllers::PolicyTrafficController

- **Descrizione:** PolicyTrafficController permette di ricavare le informazioni riguardanti la tipologia di policy a traffico dati.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**
  - **PolicyTrafficController($scope : $scope, $http : $http)**
    Metodo costruttore della classe;

Parametri:

* **$scope : $scope**
  Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
* **$http : $http**
  Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **PolicyTraffic**;

3.5.2.21 APIM::FrontEnd::App::Controllers::ProfileManagerController

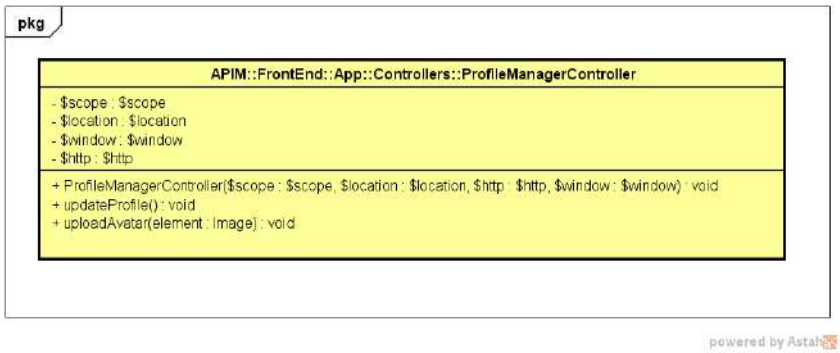

Figura 64: APIM::FrontEnd::App::Controllers::ProfileManagerController

- **Descrizione:** ProfileManagerController permette di gestire il profilo personale di un utente, fornendogli le funzionalità per poter modificare i propri dati.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$window : $window**
    Campo dati contenente il riferimento all'oggetto globale $window creato da AngularJS. Viene utilizzato per riferisi all'oggetto window del browser;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**
  - **ProfileManagerController($scope : $scope, $location : $location, $http : $http, $window : $window,**
    Metodo costruttore della classe;

Parametri:

* **$scope : $scope**
  Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
* **$location : $location**
  Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
* **$http : $http**
  Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
* **$window : $window**
  Parametro che contiene il riferimento all'oggetto globale $window di AngularJS. Viene utilizzato per interagire con l'oggeto window del browser.

– **updateProfile() : void**
Metodo per aggiornare i dati del profilo utente. Si serve di un'operazione di un servizio esposto dal package **Services**.

Parametri:

* **void**

– **updateAvatar(element : Image) : void**
Metodo per modificare l'avatar del profilo utente. Si serve di un'operazione di un servizio esposto dal package **Services**.

Parametri:

* **element : Image**
  Parametro che rappresenta l'immagine avatar dell'utente.

- **Relazioni con altre classi:**
  – Ricava i dati necessari dal package **Services**;
  – Gestisce il funzionamento della view **ProfileManager**;
  – Il model **UsersDetailModel** che contiene le informazioni per rappresentare un utente.

3.5.2.22 APIM::FrontEnd::App::Controllers::RechargedCreditsController

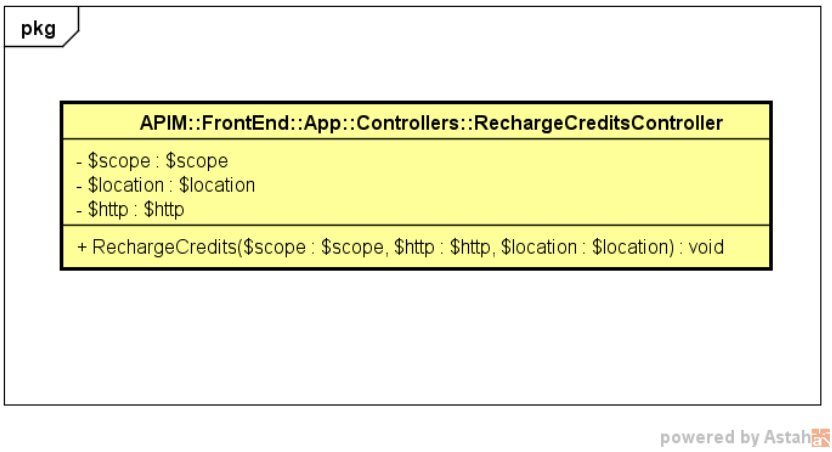

Figura 65: APIM::FrontEnd::App::Controllers::RechargeCreditsController

- **Descrizione:** RechargeCreditsController permette di effettuare la ricarica di crediti nel profilo di clienti e sviluppatori.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **RechargeCreditsController($scope : $scope, $location : $location, $http : $http)**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
- **Relazioni con altre classi:**

- Ricava i dati necessari dal package **Services**;
- Gestisce il funzionamento della view **RechargeCredits**;

3.5.2.23 APIM::FrontEnd::App::Controllers::RegisterAPIController

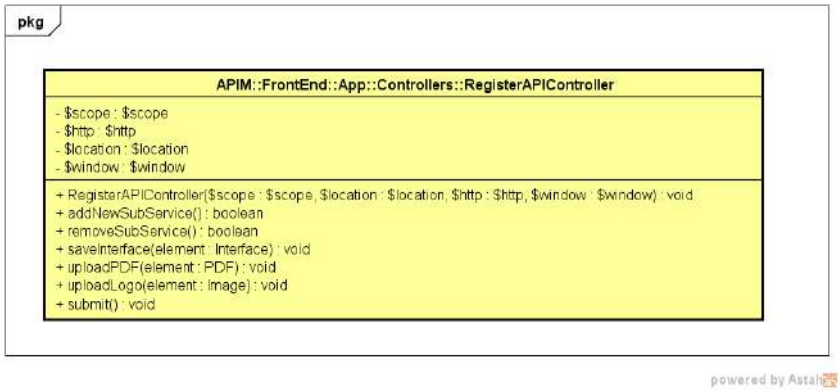

Figura 66: APIM::FrontEnd::App::Controllers::RegisterAPIController

- **Descrizione:** RegisterAPIController permette di effettuare tutte le operazioni per la registrazione di un API.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$window : $window**
    Campo dati contenente il riferimento all'oggetto globale $window creato da AngularJS. Viene utilizzato per riferisi all'oggetto window del browser;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **RegisterAPIController($scope : $scope, $location : $location, $http : $http, $window : $window,**
    Metodo costruttore della classe;

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.

    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    - **$window : $window**
      Parametro che contiene il riferimento all'oggetto globale $window di AngularJS. Viene utilizzato per interagire con l'oggeto window del browser.
  - **addNewSubService() : boolean**
    Metodo per aggiungere il riferimento ad un subservizio. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **void**

  - **removeSubService() : boolean**
    Metodo per eliminare il riferimento ad un subservizio. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **void**

  - **saveInterface(element : Interface) : void**
    Metodo per aggiungere il file di interfaccia del subservizio. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **element : Interface**
      File di interfaccia del servizio.
  - **uploadPDF(element : PDF) : void**
    Metodo per aggiungere il file di documentazione in formato PDF. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **element : PDF**
      File di documentazione del servizio.
  - **uploadLogo(element : Image) : void**
    Metodo per aggiungere il logo. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **element : Image**
      File imagine del logo del servizio.
  - **submit() : void**
    Metodo per effettuare la registrazione dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **RegisterAPI**;

3.5.2.24 APIM::FrontEnd::App::Controllers::RegisterUserController

Figura 67: APIM::FrontEnd::App::Controllers::RegisterUserController

- **Descrizione:** RegisterUserController permette di effettuare la registrazione di un utente.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$window : $window**
    Campo dati contenente il riferimento all'oggetto globale $window creato da AngularJS. Viene utilizzato per riferisi all'oggetto window del browser;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
- **Metodi:**
  - **RegisterUserController($scope : $scope, $location : $location, $http : $http, $window : $window,**
    Metodo costruttore della classe;

    Parametri:
    - **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    - **$location : $location**
      Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per riferirsi all'URL ed interagire con il routing.
    - **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

    - **$window : $window**
      Parametro che contiene il riferimento all'oggetto globale $window di AngularJS. Viene utilizzato per interagire con l'oggeto window del browser.

  - **submitForm() : void**
    Invia il form con i dati per la registrazione. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **void**

  - **updloadAvatar(element : Image) : void**
    Metodo per caricare l'avatar del profilo utente. Si serve di un'operazione di un servizio esposto dal package **Services**.

    Parametri:

    - **element : Image**
      Parametro che rappresenta un indirizzo email.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **RegisterUser**;

3.5.2.25 APIM::FrontEnd::App::Controllers::ShowUserDetailController

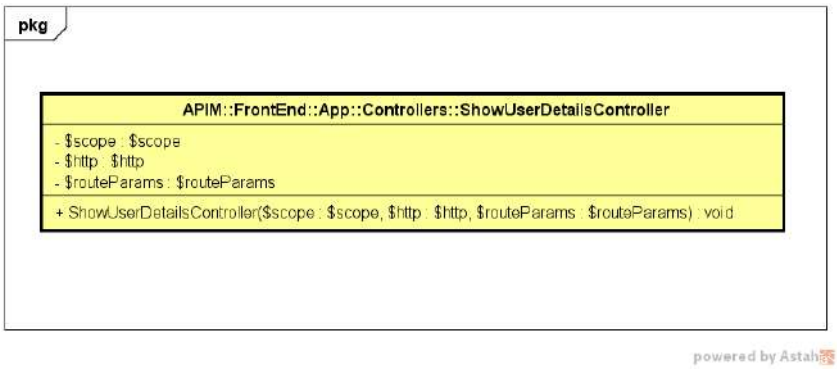

Figura 68: APIM::FrontEnd::App::Controllers::ShowUserDetailsController

- **Descrizione:** ShowUserDetailsController permette di ricavare le informazioni di un utente.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$routeParams : $routeParams**
    Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;

- **Metodi:**
  - **ShowUserDetailsController($scope : $scope, $http : $http, $routeParams : $routeParams,**
    Metodo costruttore della classe;

    Parametri:
    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
    * **$http : $http**
      Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
    * **$routeParams : $routeParams**
      Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **ShowUserDetails**;

3.5.2.26 APIM::FrontEnd::App::Controllers::TransactionsListController

Figura 69: APIM::FrontEnd::App::Controllers::TransactionsListController

- **Descrizione:** TransactionsListController permette di ricavare la lista delle transazioni per un utente.

- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;

- **Metodi:**

– **TransactionsListController($scope : $scope, $http : $http, $routeParams : $routeParams,**
Metodo costruttore della classe;

Parametri:

* **$scope : $scope**
Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
* **$http : $http**
Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.

- **Relazioni con altre classi:**
  – Ricava i dati necessari dal package **Services**;
  – Gestisce il funzionamento della view **TransactionsList**;

3.5.2.27 APIM::FrontEnd::App::Controllers::UpdateInfoAPIController

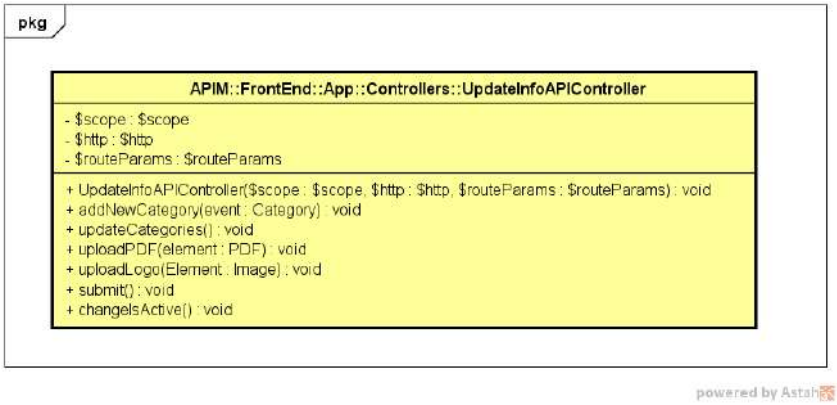

Figura 70: APIM::FrontEnd::App::Controllers::UpdateInfoAPIController

- **Descrizione:** UpdateInfoAPIController permette di effettuare tutte le operazioni per l'aggiornamento delle informazioni di una API.

- **Attributi:**
  – **$scope : $scope**
  Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  – **$http : $http**
  Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  – **$routeParams : $routeParams**
  Campo dati contenente il riferimento all'oggetto globale $routeParams creato da AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST;

- **Metodi:**

- **UpdateInfoAPIController($scope : $scope, $location : $location, $http : $http, $window : $window,**
  Metodo costruttore della classe;

  Parametri:

  * **$scope : $scope**
    Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;
  * **$http : $http**
    Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
  * **$routeParams : $routeParams**
    Parametro che contiene il riferimento all'oggetto globale $routeParams di AngularJS. Viene utilizzato per riferirsi alle variabili GET e POST.
- **addNewCategory(event : Category) : void**
  Metodo per aggiungere l'API alla categoria selezionata. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  * **event : Category**
    Categoria a cui aggiungere l'API.
- **updateCategories() : void**
  Metodo per aggiornare le categorie associate all'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  * **void**

- **uploadPDF(element : PDF) : void**
  Metodo per aggiungere il file di documentazione in formato PDF. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  * **element : PDF**
    File di documentazione del microservizio.
- **uploadLogo(element : Image) : void**
  Metodo per aggiungere il logo di un microservizio. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  * **element : Image**
    File immagine del logo del microservizio.
- **submit() : void**
  Metodo per effettuare l'invio e la modifica delle nuove informazioni per l'API selezionata. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  * **void**

- **changeIsActive() : void**
  Metodo per effettuare la commutazione dello stato (attivo/non attivo) dell'API. Si serve di un'operazione di un servizio esposto dal package **Services**.

Parametri:

* **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **UpdateInfoAPI**;

3.5.2.28 APIM::FrontEnd::App::Controllers::UpgradeToDeveloperController

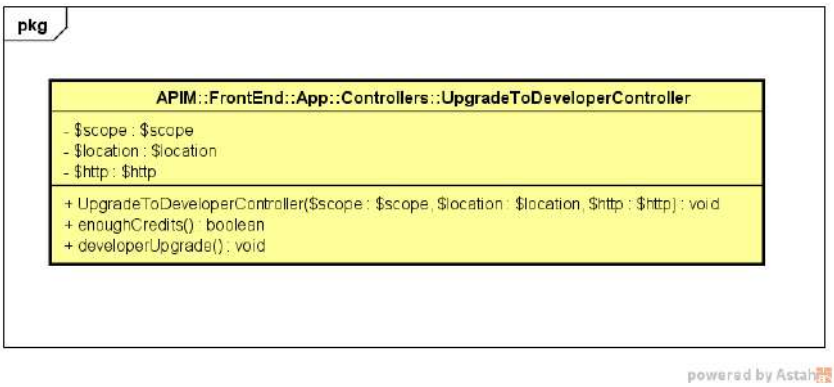

Figura 71: APIM::FrontEnd::App::Controllers::UpgradeToDeveloperController

- **Descrizione:** UpgradeToDeveloperController permette di effettuare l'aggiornamento di un cliente allo stato di sviluppatore.
- **Attributi:**
  - **$scope : $scope**
    Campo dati contenente un riferimento all'oggetto $scope creato da AngularJS, viene utilizzato come mezzo di comunicazione tra il controller e la view. Contiene gli oggetti che definiscono il model dell'applicazione;
  - **$http : $http**
    Campo dati che contiene un riferimento al servizio $http che permette la comunicazione con il protocollo HTTP;
  - **$location : $location**
    Campo dati contenente il riferimento all'oggetto globale $location creato da AngularJS. Viene utilizzato per le funzioni di routing;
- **Metodi:**
  - **UpgradeToDeveloperController($scope : $scope, $http : $http, $location : $location)**
    Metodo costruttore della classe;

    Parametri:

    * **$scope : $scope**
      Parametro che contiene un riferimento all'oggetto $scope di AngularJS, impiegato nella comunicazione tra i rispettivi view e controller. Contiene gli oggetti che definiscono i model dell'applicazione;

  - **$http : $http**
    Parametro che contiene il riferimento all'oggetto globale $http di AngularJS. Viene utilizzato per la comunicazione con il protocollo HTTP.
  - **$location : $location**
    Parametro che contiene il riferimento all'oggetto globale $location di AngularJS. Viene utilizzato per le funzioni di routing.

- **enoughCredits() : boolean**
  Metodo che verifica se il cliente possiede abbastanza crediti per effettuare l'upgrade. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  - **void**

- **developerUpgrade() : void**
  Metodo che effettua l'upgrade dell'utente. Si serve di un'operazione di un servizio esposto dal package **Services**.

  Parametri:

  - **void**

- **Relazioni con altre classi:**
  - Ricava i dati necessari dal package **Services**;
  - Gestisce il funzionamento della view **UpgradeToDeveloper**;

3.6 APIM::FrontEnd::App::node_modules

3.6.1 Informazioni generali

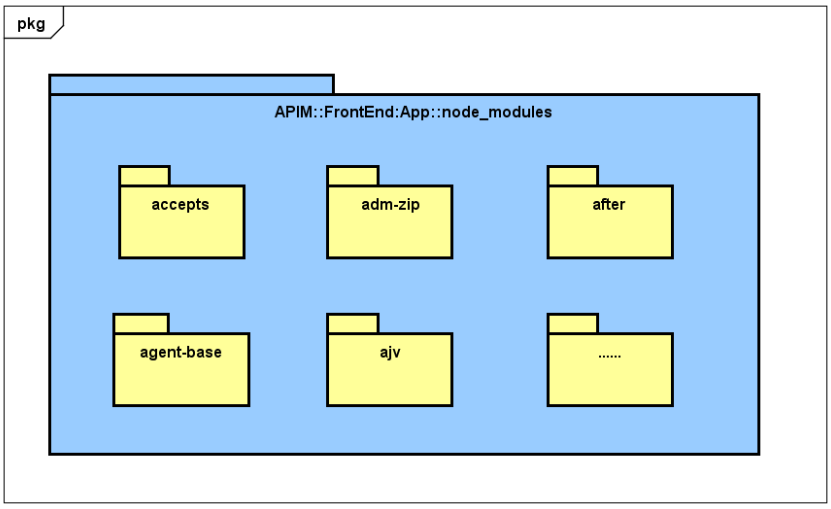

Figura 72: APIM::FrontEnd::App::node_modules

- **Descrizione:** Il package *node_modules* contiene tutti i moduli che Node.js mette a disposizione e che possono essere utilizzati dall'applicazione web.

4 Specifica Back-End

4.1 APIMarket::Back-End

4.1.1 Informazioni generali

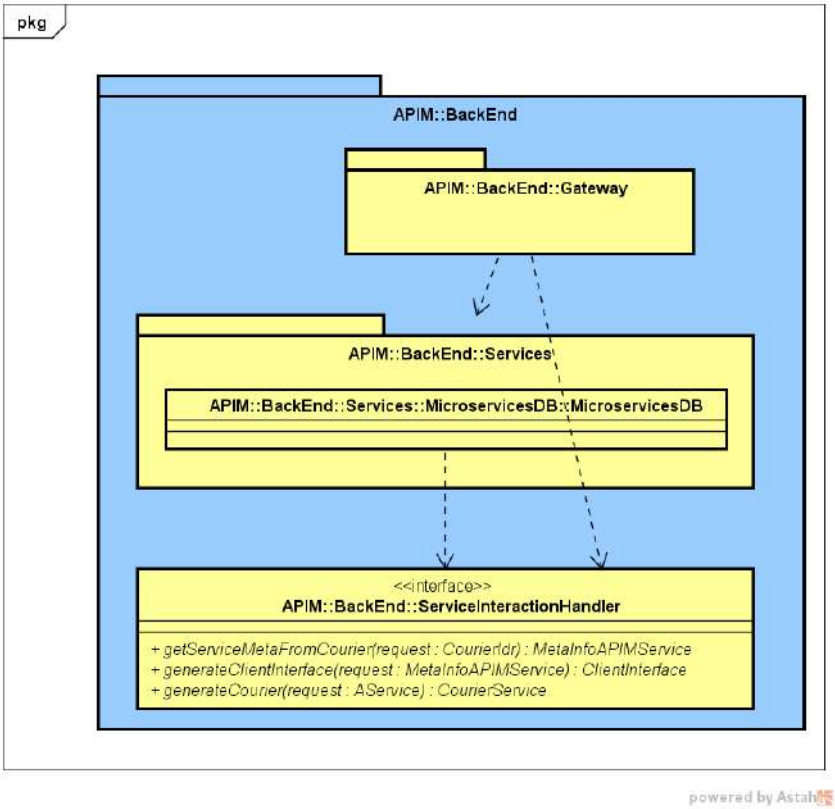

Figura 73: Package APIM::BackEnd

- **Descrizione:** Il package *BackEnd* contiene le componenti del lato back-end dell'applicazione web.
- **Packages contenuti:**
  - **Gateway:** package riguardante la gestione delle chiamate ai microservizi;
  - **Services:** package riguardante la comunicazione con i database di API Market.

4.1.2 Interfacce

4.1.2.1 ServiceInteractionHandler

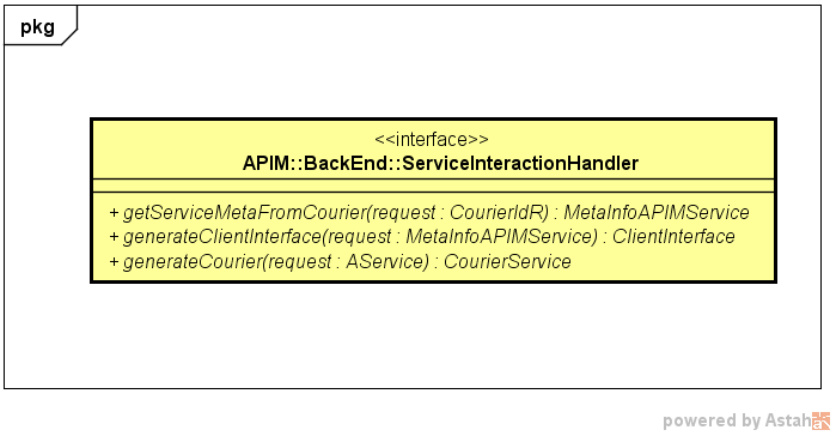

Figura 74: Package APIM::BackEnd::ServiceInteractionHandler

- **Descrizione:** L'interfaccia *ServiceInteractionHandler* contiene le operazioni riguardanti la gestione delle sessioni couriers e dei metadati dei microservizi. Viene utilizzata dal *Gateway* e da *MicroservicesDB*.

4.2 APIMarket::Back-End::Gateway

4.2.1 Informazioni generali

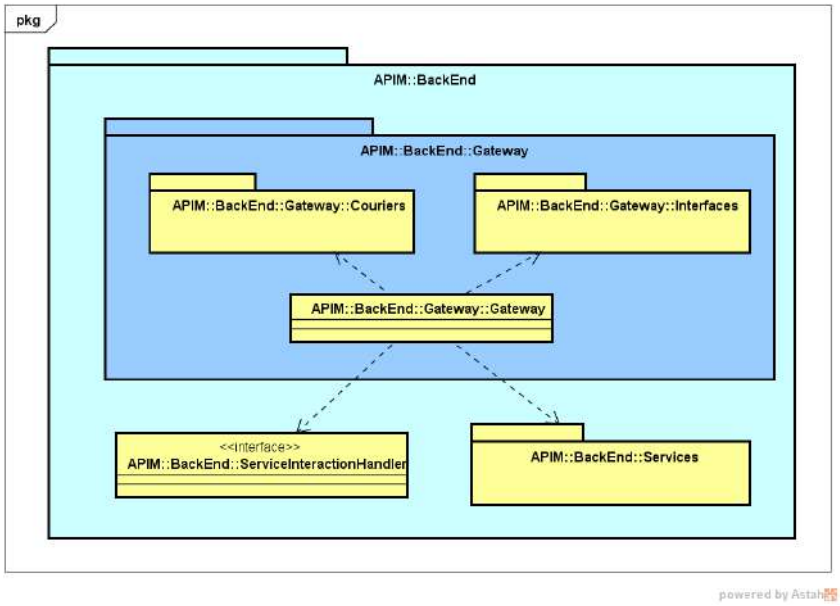

Figura 75: Package APIM::BackEnd::Gateway

- **Descrizione:** Il package *Gateway* contiene le componenti per l'API Gateway dell'applicazione web.
- **Packages contenuti:**

- **Couriers:** package riguardante l'archiviazione delle sessioni couriers;
- **Interfaces:** package riguardante le interfacce necessarie alla classe Gateway.

4.2.2 Classi

4.2.2.1 Gateway

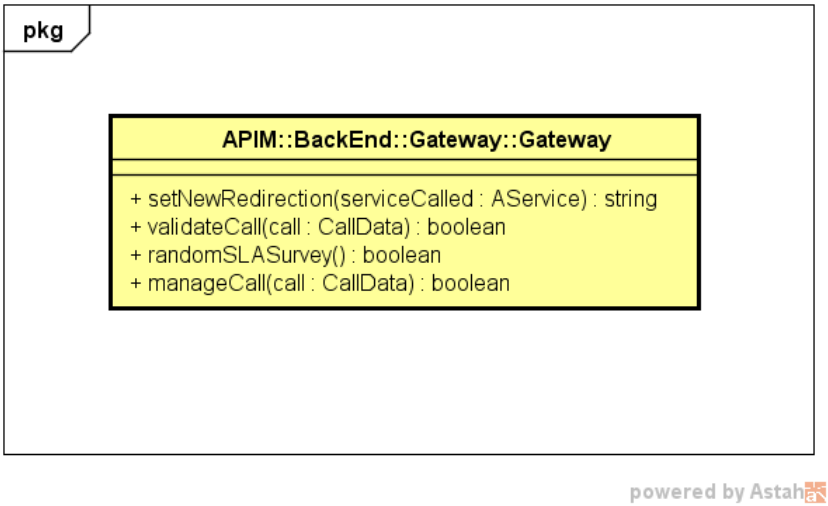

Figura 76: Package APIM::BackEnd::Gateway::Gateway

- **Descrizione:** La classe Gateway genera le sessioni couriers di ciascun microservizio presente in API Market al suo avvio, ed in seguito alla registrazione di un nuovo microservizio. Inoltre, si occupa della verifica della chiamata al microservizio rispetto ai dati utente ed API Key, dei rilevamenti casuali di rispetto della SLA e della loro archiviazione, della redirection verso il microservizio e l'operazione desiderati.
- **Relazioni:**
  - La classe Gateway implementa l'interfaccia RedirectorInterface;
  - La classe Gateway utilizza le operazioni esposte dall'interfaccia ServiceInteractionHandler;
  - La classe Gateway utilizza le operazione esposte dalle interfacce dei servizi di comunicazione con i database, presenti nel package Services;
  - La classe Gateway genera le sessioni courier che vengono archiviate nel package Couriers.
- **Operazioni:**
  - **setNewRedirection(AService)(string):** Genera il file di courier di un microservizio e ne imposta la redirection.

    Parametri:
    - **AService:** Dati del microservizio dal quale generare un file courier.
  - **validateCall(CallData)(boolean):** Valida la chiamata al microservizio, verificando utente e relativa apikey.

    Parametri:
    - **CallData:** Dati della chiamata al microservizio.

- **randomSLASurvey( )(boolean):** Algoritmo casuale per scegliere se effettuare un sondaggio della SLA.
- **manageCall(CallData)(boolean):** Gestisce la chiamata al microservizio.

  Parametri:

  * **CallData:** Dati della chiamata al microservizio.

4.2.2.2 ServiceInteractionHandler

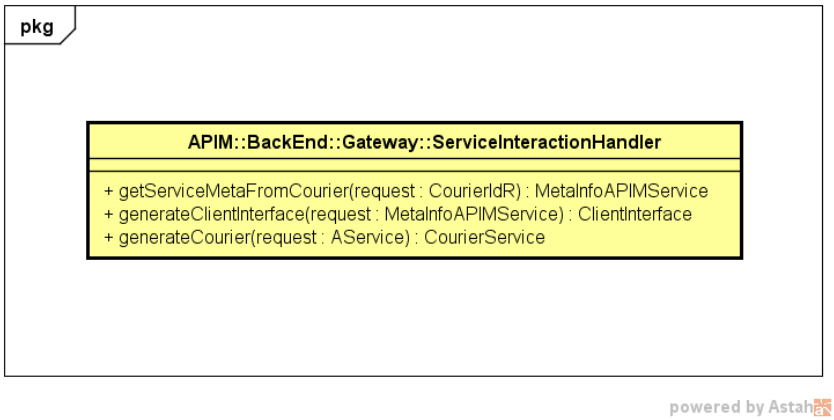

Figura 77: Package APIM::BackEnd::Gateway::ServiceInteractionHandler

- **Descrizione:** La classe ServiceInteractionHandler implementa l'interfaccia contenuta in APIM::BackEnd::Gateway::ServiceInteractionHandler.
- **Relazioni:**
  - La classe ServiceInteractionHandler espone le proprie operazioni alla classe Gateway;
  - La classe ServiceInteractionHandler espone le proprie operazioni alla classe MicroservicesDB.
- **Operazioni:**
  - **GetServiceMetaFromCourier(CourierIdR)(MetaInfoAPIMService):** Ricava da una courier i metadati del servizio, cioè tipi utilizzati ed operazioni fornite.

    Parametri:

    * **CourierIdR:** Directory del file courier da cui estrarre le metainfo;
    * **MetaInfoAPIMService:** Metadati di un microservizio registrato in API Market.
  - **generateClientInterface(MetaInfoAPIMService)(ClientInterface):** Genera l'interfaccia esposta al client del microservizio.

    Parametri:

    * **MetaInfoAPIMService:** Metadati di un microservizio registrato in API Market;
    * **ClientInterface:** Rappresentazione come stringa dell'interfaccia esposta al client del microservizio.
  - **generateCourier(AService)(CourierService):** Genera la rappresentazione come stringa del file courier del microservizio.

Parametri:

* **AService:** Dati del microservizio dal quale generare un file courier;
* **CourierService:** Rappresentazione come stringa del file courier del microservizio.

4.3 APIMarket::Back-End::Gateway::GatewayInterfaces

4.3.1 Informazioni generali

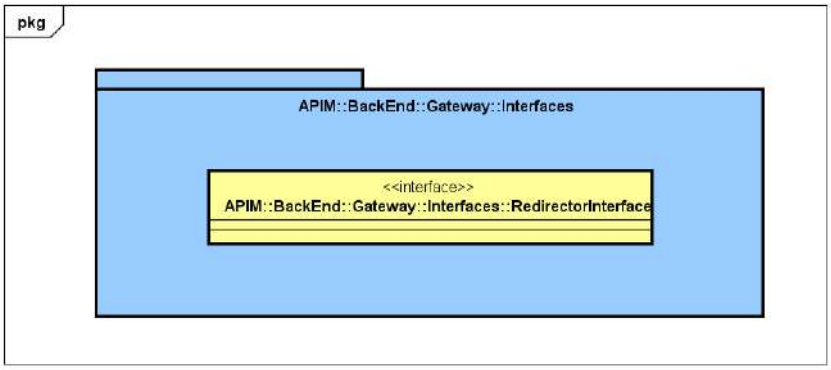

Figura 78: Package APIM::BackEnd::Gateway::GatewayInterfaces

- **Descrizione:** Il package GatewayInterfaces contiene l'interfaccia necessarie al funzionamento del Gateway. In particolare si occupa delle attività di verifica, di gestione SLA e di redirection, trattando i dati riguardanti interfaccia ed operazione della chiamata ad un microservizio.

4.3.2 Interfacce

4.3.2.1 RedirectorInterface

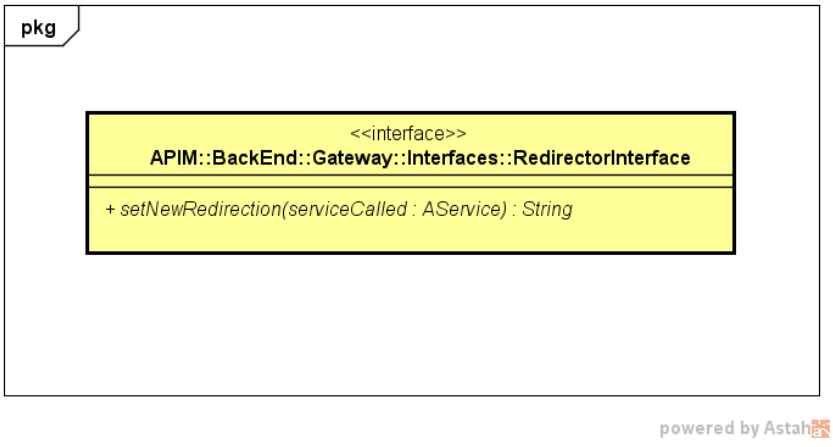

Figura 79: Package APIM::BackEnd::GatewayInterfaces::RedirectorInterface

- **Descrizione:** L'interfaccia RedirectorInterface si occupa delle attività di redirection di una chiamata ad un microservizio. Sfruttando la classe ServiceInteractionHandler per generare una sessione courier, ne crea la classe Courier e le indirizza la redirection. Viene implementata dalla classe Gateway.

4.4 APIMarket::Back-End::Services

4.4.1 Informazioni generali

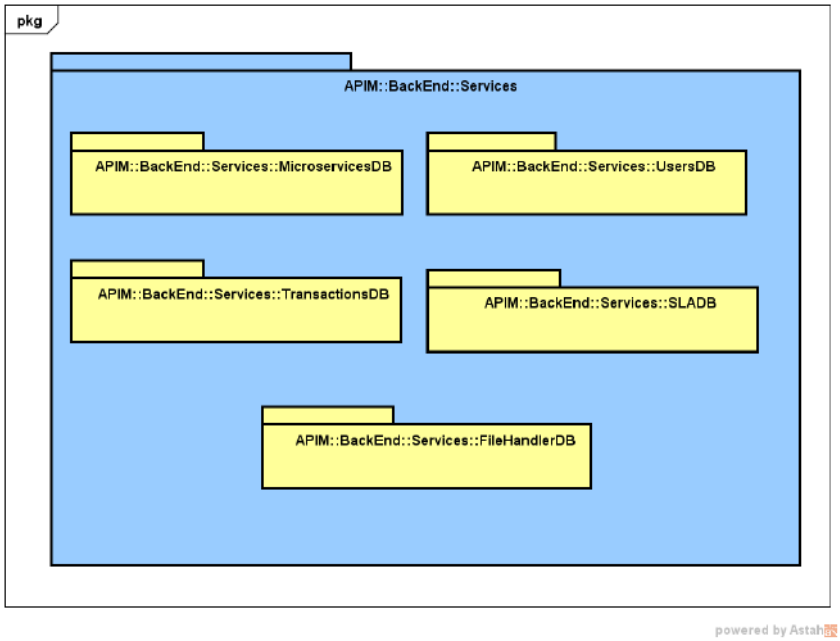

Figura 80: Package APIM::BackEnd::Services

- **Descrizione:** Il package Services contiene le componenti per la comunicazione con i database di API Market.
- **Packages contenuti:**
  - **MicroservicesDB:** package riguardante la comunicazione con il database dei microservizi registrati in API Market;
  - **UsersDB:** package riguardante la comunicazione con il database degli utenti in API Market;
  - **TransactionsDB:** package riguardante la comunicazione con il database delle transazioni in API Market;
  - **SLADB:** package riguardante la comunicazione con il database della SLA in API Market.
  - **FileHandlerDB:** package riguardante la comunicazione con il database dei file in API Market.

4.5 APIMarket::Back-End::Services::MicroservicesDB

4.5.1 Informazioni generali

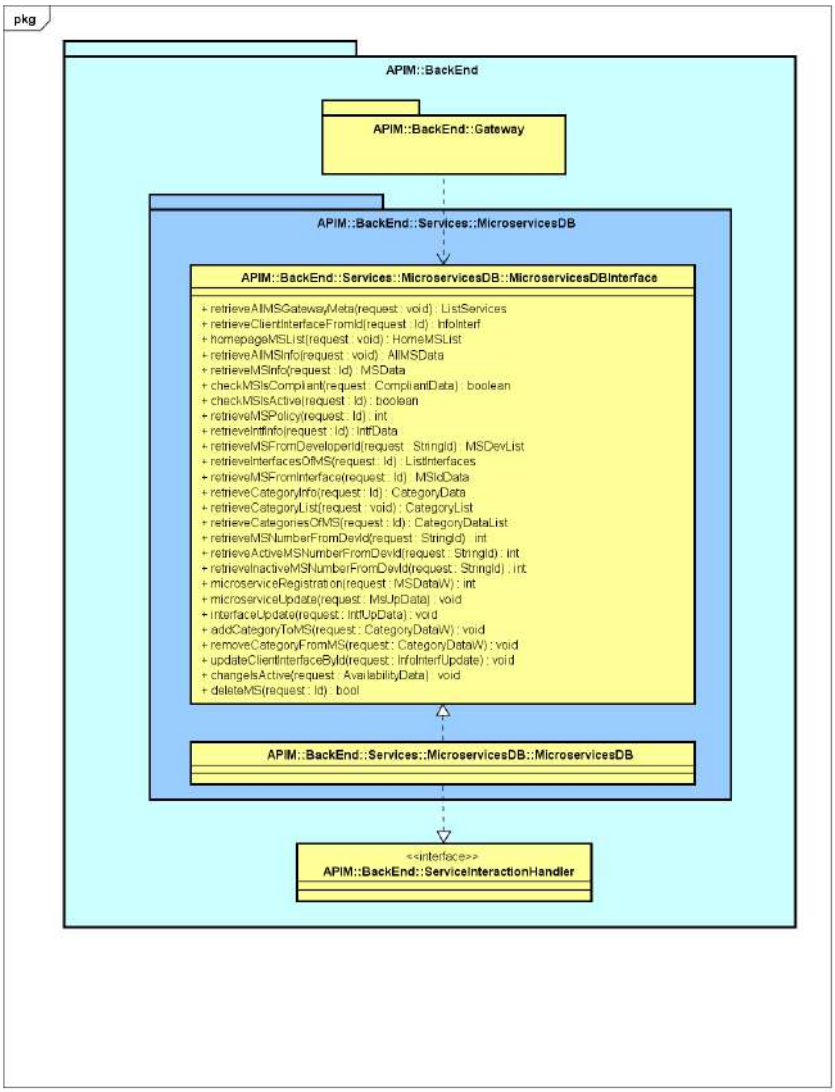

Figura 81: Package APIM::BackEnd::Services::MicroservicesDB

- **Descrizione:** Il package MicroservicesDB contiene le componenti per la comunicazione con i database dei microservizi di API Market.

4.5.2 Interfacce

4.5.2.1 MicroservicesDBInterface

Figura 82: Package APIM::BackEnd::Services::MicroservicesDBInterface

- **Descrizione:** L'interfaccia MicroservicesDB contiene le operazioni riguardanti lettura e scrittura dei dati dei microservizi, delle interfacce, delle categorie dei microservizi. Viene utilizzata dal Gateway.

4.5.3 Classi

4.5.3.1 MicroservicesDB

Figura 83: Package APIM::BackEnd::Services::MicroservicesDB

- **Descrizione:** La classe MicroservicesDB implementa l'interfaccia contenuta in APIM::BackEnd::Services::MicroservicesInterface.
- **Relazioni:**
  - La classe MicroservicesDB implementa l'interfaccia MicroservicesDBInterface;
  - La classe MicroservicesDB utilizza le operazioni esposte dall'interfaccia ServiceInteractionHandlerInterface;
  - La classe MicroservicesDB espone le proprie operazioni al FrontEnd.
- **Operazioni:**
  - **retrieveAllMSGatewayMeta(void)(ListServices):** Ricava le informazioni dei microservizi per il gateway in API Market.

    Parametri:
    - **ListServices:** Lista dei microservizi con relative informazioni.
  - **retrieveClientInterfaceFromID(Id)(InfoInterf):** Ricava le informazioni di ogni cliente tramite il suo ID in API Market.

    Parametri:
    - **Id:** Id del cliente di cui ricavare le informazioni;
    - **InfoInterf:** informazioni del cliente.
  - **homepageMSlist(void)(HomeMSList):** Ricava le informazioni di ogni microservizio registrato in API Market.

    Parametri:

    - **HomeMSList:** Lista dei microservizi per la homepage.

- **retreiveAllMSInfo(void)(MSData):** Ricava le informazioni di tutti microservizi.

  Parametri:

    - **MSData:** Informazioni di tutti i microservizi.

- **retreiveMSInfo(Id)(MSData):** Ricava le informazioni del microservizio a partire dal suo id.

  Parametri:

    - **Id:** Id del microservizio;
    - **MSData:** Informazioni del microservizio.

- **checkMSIsCompliant(CompliantData)(boolean):** controlla se un microsrvizio i soddisfa i rquisiti SLA.

  Parametri:

    - **CompliantData:** Id del microservizio da controllare;
    - **bool.**

- **checkMSIsActive(Id)(boolean):** controlla se un microservizio è attivo.

  Parametri:

    - **CompliantData:** Id del microservizio da controllare;
    - **bool.**

- **retrieveMSPolicy(Id)(int):** ritorna la policy del microservizio.

  Parametri:

    - **Id:** Id del microservizio da controllare;
    - **int:** interno che identifica la policy.

- **retrieveIntfInfo(Id)(IntfData):** recupera tutte le informazioni dell'interfaccia.

  Parametri:

    - **Id:** Id del microservizio da controllare;
    - **IntfData:** ritorna le informazioni dell'interfaccia.

- **retrieveMSFromDeveloperId(StringId)(MSDevList):** Ricava le informazioni di un microservizio a partire dall'Id di uno sviluppatore.

  Parametri:

    - **StringId:** Id del microservizio;
    - **MSDevList:** informazioni microservizio.

- **retrieveInterfacesOfMS(Id)(ListInterfaces):** Ricava l'interfaccia del microservizio a partire dall'id.

  Parametri:

    - **Id:** Id dell'interfaccia;
    - **ListInterfaces:** Interfaccia del microservizio.

- **retrieveMSFromInterface(Id)(MSIdData):** Ricava le informazioni di un microservizio a partire dalla sua interfaccia.

  Parametri:

    - **Id:** Id dell'interfaccia;
    - **MSIdData:** Lista degli id dei microservizi appartenenti alla categoria.

- **retrieveCategoryInfo(Id)(CategoryData):** Ricava le informazioni di una categoria a partire dal suo Id.

  Parametri:

  * **Id:** Id della categoria;
  * **CategoryData:** informazioni di una categoria.

- **retrieveCategoryList(void)(CategoryList):** Ricava le informazioni di una categoria a partire dal suo Id.

  Parametri:

  * **CategoryList:** lista API della categoria.

- **retrieveCategoryOfMS(Id)(CategoryDataList):** Ricava gli id delle categorie attribuite al microservizio, a partire dall'id del microservizio.

  Parametri:

  * **Id:** Id del microservizio;
  * **CategoryDataList:** Lista degli id delle categorie attribuite al microservizio.

- **retrieveMSNumberFromDevId(StringId)(int):** Ricava il numero di microservizi registrati da un developer.

  Parametri:

  * **StringId:** stringa Id associata a uno sviluppatore;
  * **int:** numero di microservizi registrati da un developer.

- **retrieveActiveMSNumberFromDevId(StringId)(int):** Ricava il numero di microservizi attivi di un developer.

  Parametri:

  * **StringId:** stringa Id associata a uno sviluppatore;
  * **int:** numero di microservizi attivi di un developer.

- **retrieveInactiveMSNumberFromDevId(StringId)(int):** Ricava il numero di microservizi inattivi di un developer.

  Parametri:

  * **StringId:** stringa Id associata a uno sviluppatore;
  * **int:** numero di microservizi inattivi di un developer.

- **microserviceRegistration(MSDataW)(int):** Registra un nuovo microservizio nel database dei microservizi.

  Parametri:

  * **MSDataW:** Informazioni del nuovo microservizio;
  * **int.**

- **microserviceUpdate(MSUpData)(void):** Aggiorna le informazioni di un microservizio esistente nel database dei microservizi.

  Parametri:

  * **MSUpData:** Informazioni aggiornate di un microservizio esistente.

- **interfaceUpdate(intfUpData)(void):** Aggiorna le informazioni di una interfaccia esistente nel database dei microservizi.

  Parametri:

  * **intfupdata:** Informazioni aggiornate di una interfaccia esistente.

- **addCategoryToMS(CategoryDataW)(void):** Attribuisce una categoria ad un microservizio.

  Parametri:

  * **CategoryDataW:** Id del microservizio e della categoria da attribuirgli.

- **removeCategoryFromMS(CategoryDataW)(void):** Rimuove una categoria da un microservizio.

  Parametri:

  * **CategoryDataW:** Id del microservizio e della categoria da rimuovergli.

- **updateClientInterfaceById(InfoInterfUpdate)(void):** Aggiorna le informazioni di una interfaccia esistente nel database dei microservizi.

  Parametri:

  * **InfoInterfUpdate:** Informazioni aggiornate di una interfaccia esistente.

- **changeIsActive(AvailabilityData)(void):** aggiorna lo stato di un microservizio.

  Parametri:

  * **AvailabilityData:** Informazioni di un microservizio.

- **deleteMS(Id)(bool):** Aggiorna le informazioni di una interfaccia esistente nel database dei microservizi.

  Parametri:

  * **Id:** Id del microservizio da eliminare;
  * **bool**.

4.6 APIMarket::Back-End::Services::UsersDB

4.6.1 Informazioni generali

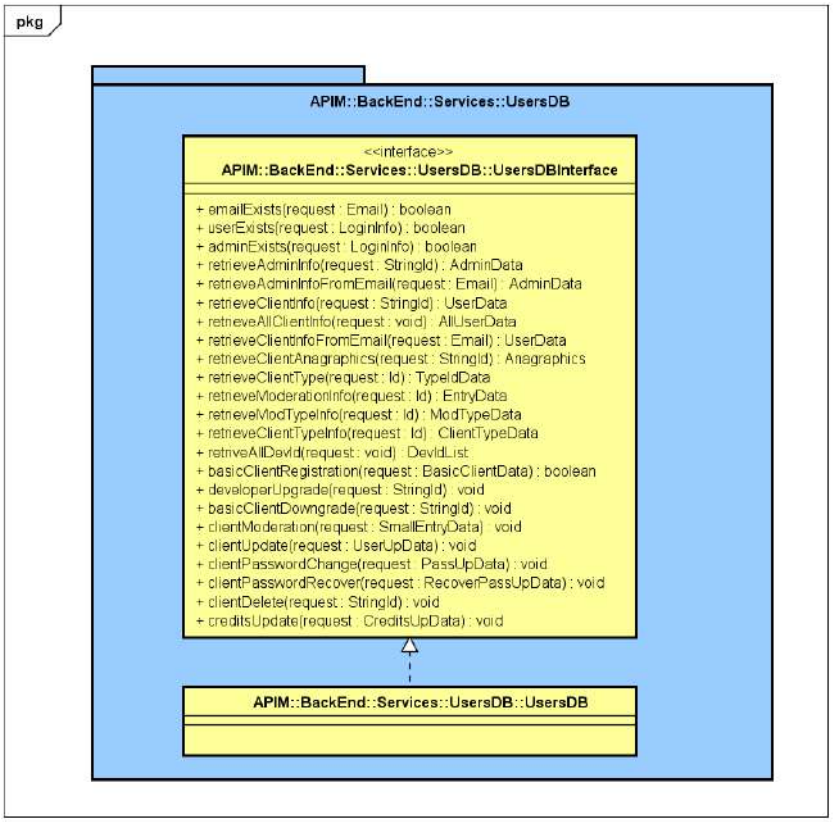

Figura 84: Package APIM::BackEnd::Services::UsersDB

- **Descrizione:** Il package *UsersDB* contiene le componenti per la comunicazione con il database degli utenti di API Market.

4.6.2 Interfacce

4.6.2.1 UsersDBInterface

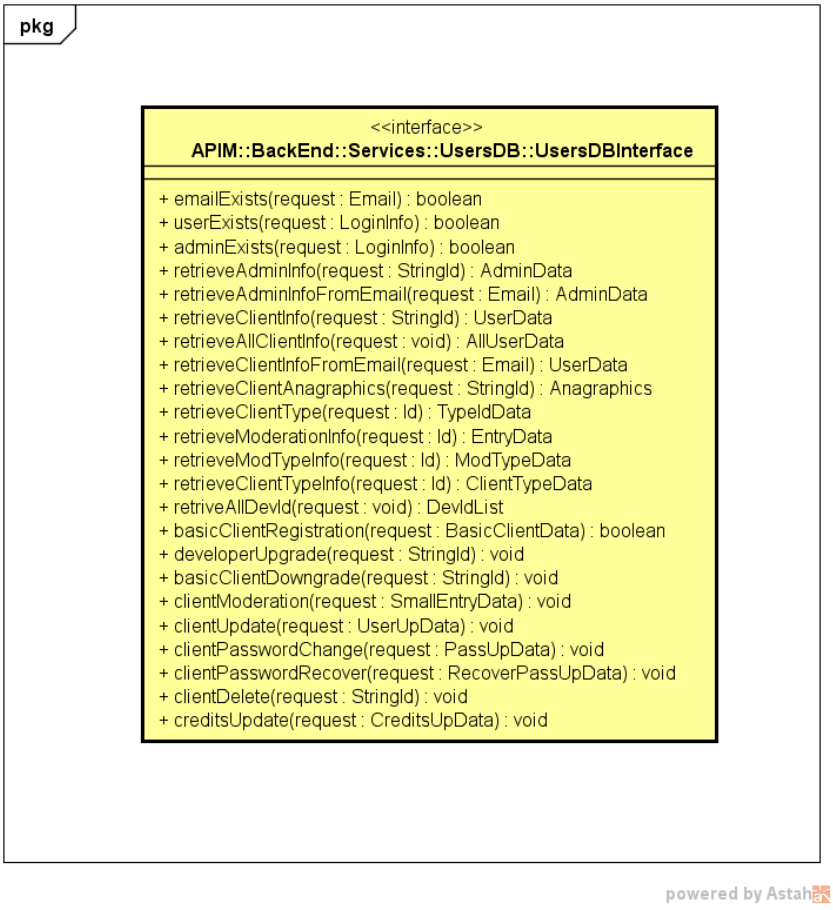

Figura 85: Package APIM::BackEnd::Services::UsersDBInterface

- **Descrizione:** L'interfaccia UsersDBInterface contiene le operazioni riguardanti lettura e scrittura dei dati degli utenti (sia admin che clienti), dei tipi di cliente, delle moderazioni attuate, dei tipi di moderazione. Viene utilizzata dal Gateway.

4.6.3 Classi

4.6.3.1 UsersDB

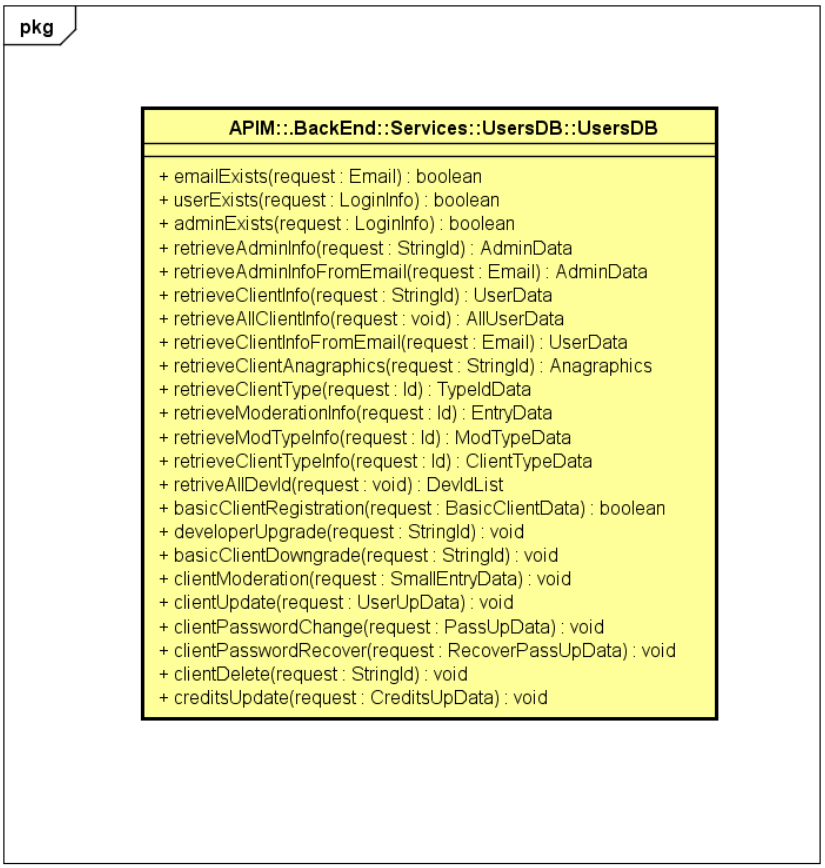

Figura 86: Package APIM::BackEnd::Services::UsersDB

- **Descrizione:** La classe UsersDB implementa l'interfaccia contenuta in APIM::BackEnd::Services::UsersDI
- **Relazioni:**
  - La classe UsersDB implementa l'interfaccia UsersDBInterface;
  - La classe UsersDB espone le proprie operazioni alla classe FrontendMS.
- **Operazioni:**
  - **emailExists(Email)(bool):** Controlla se l'email esista nel database utenti.
    Parametri:
    * **Email:** Email del cliente;
    * **bool**.
  - **userExists(LoginInfo)(bool):** Controlla se il cliente esista nel database utenti.
    Parametri:
    * **LoginInfo:** Email ed Password del cliente;
    * **bool**.
  - **adminExists(LoginInfo)(bool):** Controlla se l'aaministratore esista nel database admin.

Parametri:

* **LoginInfo:** Email ed Password dell'amministratore;
* **bool**.

– **retrieveAdminInfo(StringID)(AdminData):** Ricava le informazioni dell'admin, a partire dal suo id.

Parametri:

* **StringID:** Id dell'admin;
* **AdminData:** Informazioni dell'admin.

– **retrieveAdminInfoFromEmail(Email)(AdminData):** Ricava le informazioni dell'admin, a partire dalla sua email.

Parametri:

* **Email:** email dell'admin;
* **AdminData:** Informazioni dell'admin.

– **retrieveClientInfo(StringId)(UserData):** Ricava le informazioni del cliente, a partire dal suo id.

Parametri:

* **clientid:** Id del cliente.
* **clientdata:** Informazioni del cliente.

– **retrieveAllClientInfo(void)(AllUserData):** Ricava tutte le informazioni dei clienti.

Parametri:

* **AllUserData:** Informazioni dei clienti.

– **retrieveClientInfoFromEmail(Email)(UserData):** Ricava le informazioni del cliente, a partire dalla sua email.

Parametri:

* **Email:** email del cliente;
* **AdminData:** Informazioni del cliente.

– **retrieveClientAnagraphics (StringId)(Anagraphics):** Ricava Name e Surname del client, a partire dal suo id.

Parametri:

* **StringId:** Id del cliente;
* **Anagraphics:** anagrafica completa del cliente.

– **retrieveClientType(Id)(TypeIdData):** Ricava il tipo di account del cliente, a partire dal suo id.

Parametri:

* **Id:** Id del cliente;
* **TypeIdData:** Id del tipo di account del cliente.

– **retrieveModerationInfo(Id)(EntryData):** Ricava le informazioni della moderazione, a partire dal suo id.

Parametri:

* **Id:** Id della moderazione;
* **EntryData:** Informazioni della moderazione.

- **retrieveModTypeInfo(Id)(ModTypeData):** Ricava le informazioni del tipo di moderazione, a partire dal suo id.

  Parametri:

  * **Id:** Id del tipo di moderazione;
  * **ModTypeData:** Informazioni del tipo di moderazione.

- **retrieveClientTypeInfo(Id)(ClientTypeData):** Ricava le informazioni del tipo di account del cliente, a partire dal suo id.

  Parametri:

  * **Id:** Id del tipo di account del cliente;
  * **ClientTypeData:** Informazioni del tipo di account del cliente.

- **RetrieveAllDevId(void)(DevIdList):** Ricava le informazioni di tutti gli sviluppatori.

  Parametri:

  * **DevIdList:** informazioni di tutti gli sviluppatori.

- **basicClientRegistration(BasicClientData)(void):** Registra un nuovo account cliente di tipo base.

  Parametri:

  * **BasicClientData:** Informazioni del nuovo cliente.

- **developerUpgrade(StringId)(void):** Effettua l'upgrade ad account sviluppatore di un cliente.

  Parametri:

  * **StringId:** Informazioni dei campi aggiuntivi da sviluppatore.

- **basicClientDowngrade(StringId)(void):** Effettua il downgrade ad account base di un cliente sviluppatore.

  Parametri:

  * **StringId:** Id del cliente.

- **clientModeration(SmallEntryData)(void):** Inserisce una nuova moderazione nel database degli utenti.

  Parametri:

  * **SmallEntryData:** Informazioni della nuova moderazione.

- **clientUpdate(UserUpData)(void):** Aggiorna le informazioni di un cliente.

  Parametri:

  * **userdata:** Informazioni aggiornate del cliente.

- **clientePasswordChange(PassUpData)(void):** Cambia la password di un utente.

  Parametri:

  * **PassUpData:** Cambia la password di un utente.

- **clientePasswordRecovery(RecoverPassUpData)(void):** avvia il recupero password di un cliente.

  Parametri:

  * **RecoverPassUpData:** nuova password del cliente del cliente.

- **clientDelete(StringId)(void):** Elimina un cliente da API Market.

  Parametri:

  * **clientDelete:** Id del cliente.

4.7 APIMarket::Back-End::Services::TransactionsDB

4.7.1 Informazioni generali

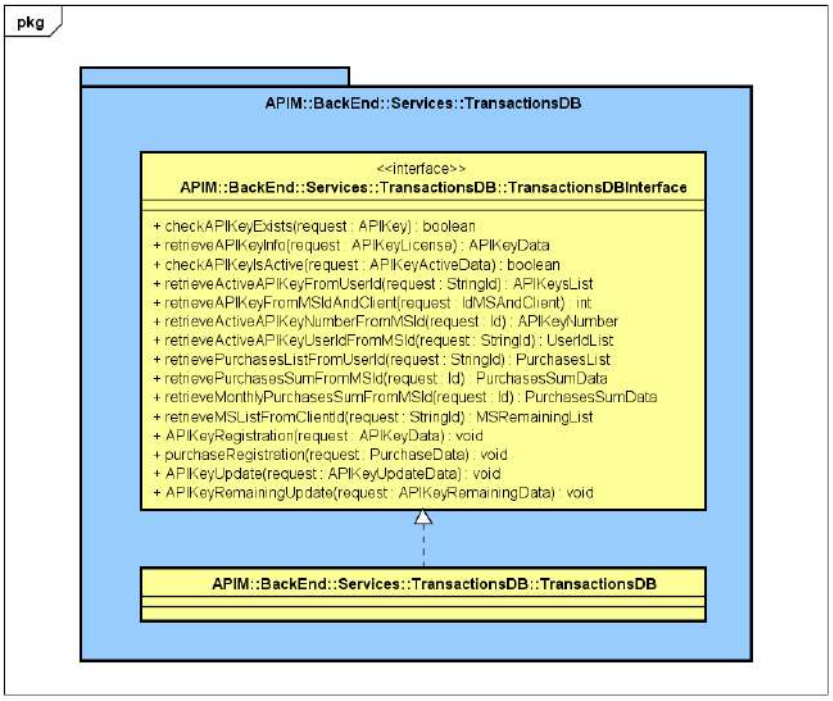

Figura 87: Package APIM::BackEnd::Services::TransactionsDB

- **Descrizione:** Il package *TransactionsDB* contiene le componenti per la comunicazione con il database delle transazioni di API Market.

4.7.2 Interfacce

4.7.2.1 TransactionsDBInterface

Figura 88: Package APIM::BackEnd::Services::TransactionsDBInterface

- **Descrizione:** L'interfaccia TransactionsDBInterface contiene le operazioni riguardanti lettura e scrittura dei dati delle apikey, degli acquisti. Viene utilizzata dal Gateway.

4.7.3 Classi

4.7.3.1 TransactionsDB

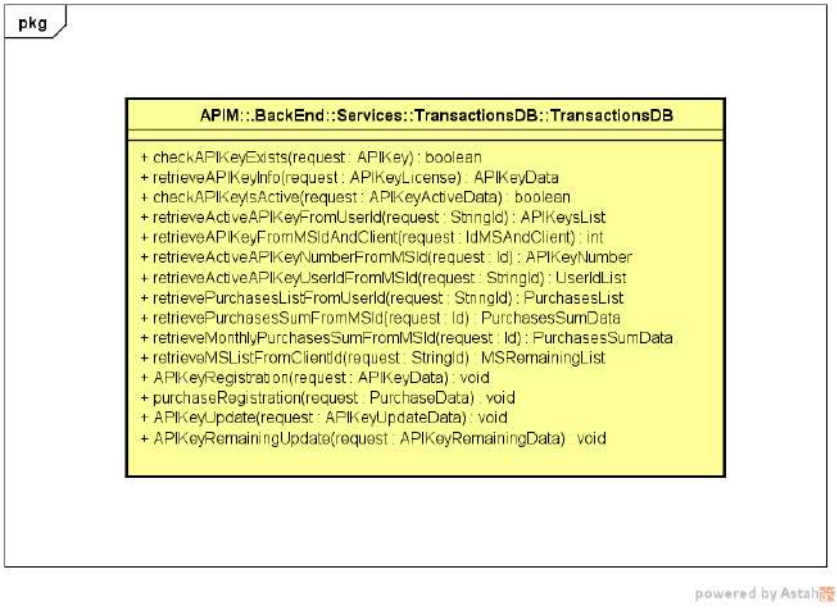

Figura 89: Package APIM::BackEnd::Services::TransactionsDB

- **Descrizione:** La classe TransactionsDB implementa l'interfaccia contenuta in APIM::BackEnd::Services::T

- **Relazioni:**
  - La classe TransactionsDB implementa l'interfaccia TransactionsDBInterface;
- **Operazioni:**
  - **checkAPIKeyExist(APIKey)(boolean):** verifica che l'API Key esista.

    Parametri:
    * **APIKey:** Informazioni dell'apikey;
    * **bool.**
  - **retrieveAPIKeyInfo(APIKeyLicense)(APIkeyData):** Ricava le informazioni dell'apikey, a partire dal suo id.

    Parametri:
    * **APIKeyLicense:** Id dell'apikey;
    * **APIkeyData:** Informazioni dell'apikey.

    item **checkAPIKeyIsActive(APIKeyActiveData)(boolean):** verifica che l'API Key sia attiva.

    Parametri:
    * **APIKey:** Informazioni dell'apikey;
    * **bool.**
  - **retrieveActiveAPIKeyFromUserId(StringId)(APIKeyList):** Ricava la lista con le api attive del cliente, a partire dal suo id.

    Parametri:
    * **StringId:** Id del cliente;
    * **APIKeyList:** lista con le api attive del cliente.
  - **retrieveAPIKeyFromMSIdAndClient(IdMSAndClient)(int):** Ricava la lista delle API da un Id di un microservizio e del cliente.

    Parametri:
    * **IdMSAndClient:** Id del microservizio e cliente.
    * **int.**
  - **retrieveActiveAPIKeyNumberFromMSId(Id)(APIKeyNumber):** Ricava il numero di APIKey attive di un microservizio, a partire dal suo Id.

    Parametri:
    * **Id:** Id del microservizio;
    * **APIKeyNumber:** numero di APIKey attive.
  - **retrieveActiveAPIKeyUserFromMSId(StringId)(UserIdList):** Ricavala lista di clienti con una API Key attiva di un microservizio, a partire dal suo Id.

    Parametri:
    * **Id:** Id del microservizio;
    * **UserIdList:** utenti con una APIKey attive.
  - **retrievePurchasesListFromUserId(StringId)(PurchasesList):** Ricava la lista di microservizi acquistati da un cliente, a partire dal suo Id.

    Parametri:
    * **StringId:** Id del cliente;

    - **UserIdList:** lista API acquistate.

- **retrievePurchasesSumFromMSId(StringId)(PurchasesSumData):** recupera la somma dell'ammontare degli acquisti di un microservizio, a partire dal suo Id.

  Parametri:

  - **StringId:** Id del microservizio;
  - **PurchasesSumData:** somma dell'ammontare degli acquisti.

- **retrieveMonthlyPurchasesSumFromMSId(StringId)(PurchasesSumData):** recupera la somma dell'ammontare degli acquisti di un microservizio in un mese, a partire dal suo Id.

  Parametri:

  - **StringId:** Id del microservizio;
  - **PurchasesSumData:** somma dell'ammontare degli acquisti mensile.

- **retrieveMSListFromClientId(StringId)(MSremainingList):** recupera la lista dei microservizi con apikey attiva, a partire dall'id del cliente.

  Parametri:

  - **StringId:** Id del cliente;
  - **MSremainingList:** sommalista microservizi con APIKey attiva.

- **APIKeyRegistration(apikeydataw)(bool):** Registra una nuova apikey nel database delle transazioni.

  Parametri:

  - **apikeydataw:** Informazioni della nuova apikey.
  - **bool**.

- **purchaseRegistration(purchasedata)(bool):** Inserisce una nuova transazione nel database delle transazioni.

  Parametri:

  - **purchasedata:** Informazioni della nuova transazione.
  - **bool**.

- **APIKeyUpdate(APIKeyUpdateData)(void):** aggiorna l'apikey con una nuova apikey.

  Parametri:

  - **APIKeyUpdateData:** nuova apikey registrata.

- **APIKeyRemainingUpdate(apikeyremainingdata)(bool):** Aggiorna il Remaining della apikey, a partire dal suo id e dal valore del cambiamento.

  Parametri:

  - **apikeyremainingdata:** Id dell'apikey e valore del cambiamento.
  - **bool**.

4.8 APIMarket::Back-End::Services::SLADB

4.8.1 Informazioni generali

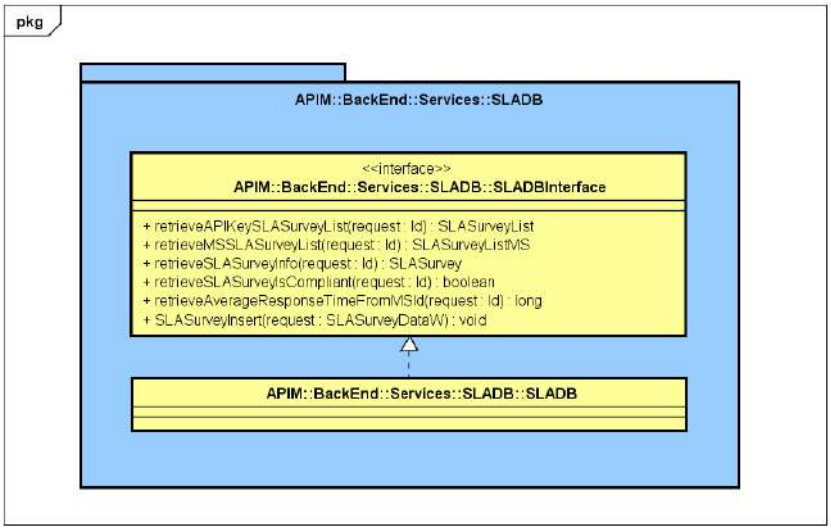

Figura 90: Package APIM::BackEnd::Services::SLADB

- **Descrizione:** Il package *SLADB* contiene le componenti per la comunicazione con il database della SLA di API Market.

4.8.2 Interfacce

4.8.2.1 SlaDBInterface

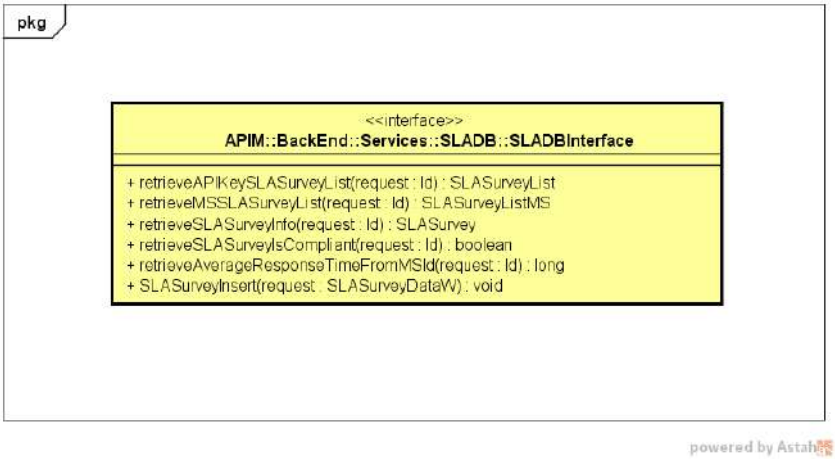

Figura 91: Package APIM::BackEnd::Services::SlaDBInterface

- **Descrizione:** L'interfaccia SlaDBInterface contiene le operazioni riguardanti lettura e scrittura dei dati dei sondaggi SLA. Viene utilizzata dal Gateway.

4.8.3 Classi

4.8.3.1 SlaDB

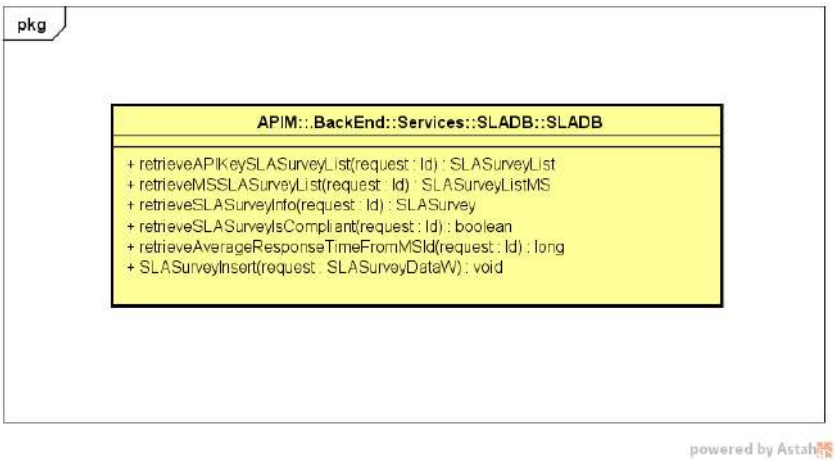

Figura 92: Package APIM::BackEnd::Services::SlaDB

- **Descrizione:** La classe SlaDB implementa l'interfaccia contenuta in // APIM::BackEnd::Services::SlaDBI
- **Relazioni:**
  - La classe SlaDB implementa l'interfaccia SlaDBInterface;
- **Operazioni:**
  - **retrieveAPIKeySLASurveyList(Id)(SLASurveyList):** Ricava la lista dei sondaggi SLA dell'apikey con relative informazioni, a partire dall'id dell'apikey.

    Parametri:
    - **apikeyid:** Id dell'apikey;
    - **SLASurveyList:** Lista dei sondaggi SLA con relative informazioni.
  - **retrieveMSSLASurveyList(Id)(SLASurveyListMS):** Ricava la lista dei sondaggi SLA del microservizio con relative informazioni, a partire dall'id del microservizio.

    Parametri:
    - **Id:** Id del microservizio;
    - **SLASurveyListMS:** Lista dei sondaggi SLA con relative informazioni.
  - **retrieveSLASurveyInfo(Id)(SLASurvey):** Ricava le informazioni del sondaggio SLA, a partire dal suo id.

    Parametri:
    - **Id:** Id del sondaggio SLA;
    - **SLASurvey:** Informazioni del sondaggio SLA.
  - **retrieveSLASurveyIsCompliant(Id)(boolean):** Ricava IsCompliant del sondaggio SLA, a partire dal suo id.

    Parametri:
    - **Id:** Id del sondaggio SLA;
    - **bool**.

– **retrieveAverageResponseTimeFromMSId(Id)(long):** Ricava il tempo medio di risposta di un microservizio, a partire dal suo id.

Parametri:

* **Id:** Id del microservizio;
* **long**.

– **SLASurveyInsert(SLASurveyDataW)(void):** Inserisce un nuovo sondaggio SLA nel database della SLA.

Parametri:

* **SLASurveyDataW:** Informazioni del nuovo sondaggio SLA.

4.9 APIMarket::Back-End::Services::FilehandlerDB

4.9.1 Informazioni generali

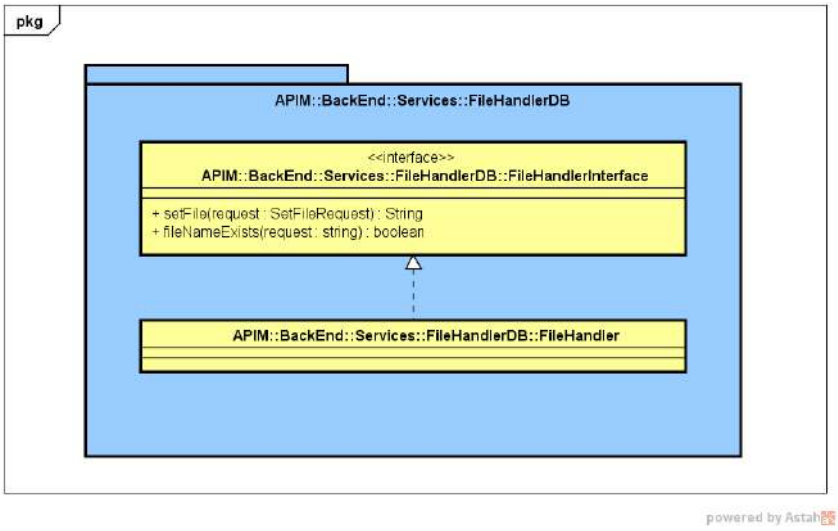

Figura 93: Package APIM::BackEnd::Services::FilehandlerDB

- **Descrizione:** Il package FilehandlerDB contiene le componenti per la comunicazione con il database dei file di API Market.

4.9.2 Interfacce

4.9.2.1 FilehandlerInterface

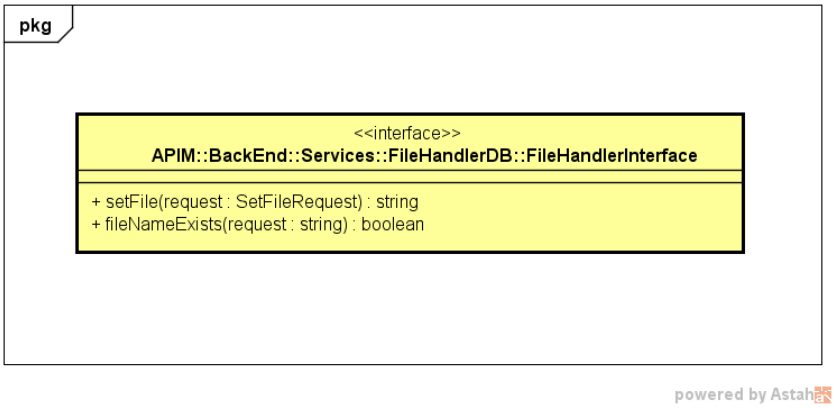

Figura 94: Package APIM::BackEnd::Services::FilehandlerDB::FilehandlerInterface

- **Descrizione:** L'interfaccia FilehandlerDBInterface contiene le operazioni riguardanti lettura e scrittura dei dati dei file. Viene utilizzata dal Gateway.

4.9.3 Classi

4.9.3.1 FileHandlerDB

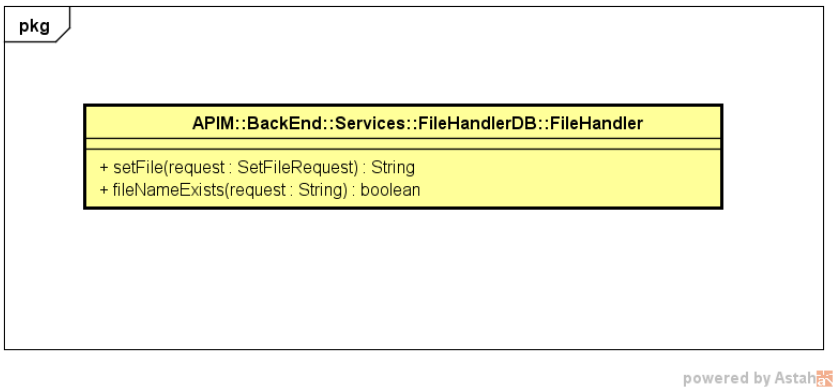

Figura 95: Package APIM::BackEnd::Services::FileHandlerDB::FileHandler

- **Descrizione:** La classe Filehandler implementa l'interfaccia contenuta in APIM::BackEnd::Services::FileH
- **Relazioni:**
  - La classe FileHandlerDB implementa l'interfaccia FileHandlerDBInterface.
- **Operazioni:**
  - **setFile(SetFileRequest)(string):** Archivia il file, se non è già presente nell'archivio, salvandolo con un nome univoco e ricava l'uri corrispondente.

    Parametri:
    - **SetFileRequest:** Informazioni del file;

    * **string**.

– **fileNameExists(string)(boolean):** Controlla se nel database dei file esista un file con il nome specificato.

  Parametri:

    * **string**;
    * **bool**.

5 Diagrammi di sequenza

5.1 Front-End

I seguenti diagrammi di sequenza prendono in considerazione le principali operazioni del front-end e vanno ad illustrarne le interazioni tra le classi.

5.1.1 Registrazione utente

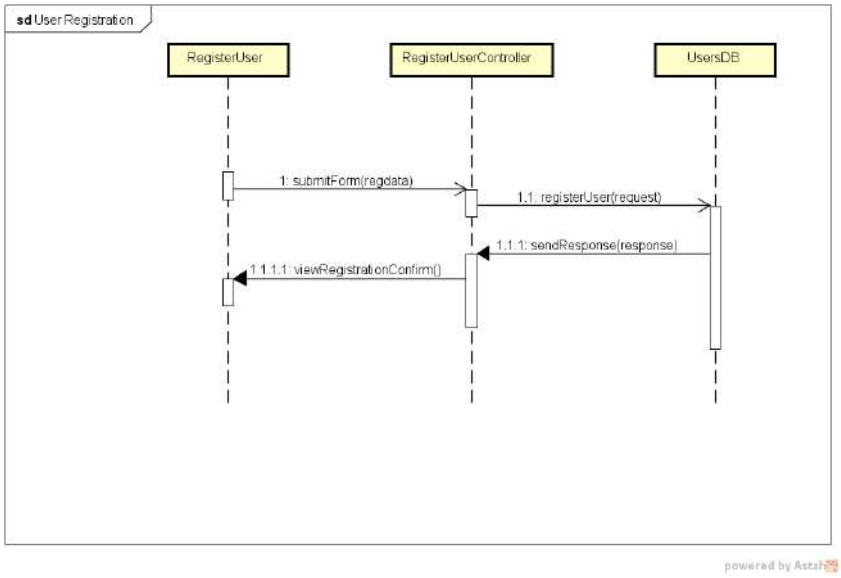

Figura 96: Diagramma di sequenza: Registrazione utente

- **Pre-condizioni**: l'utente si trova nella schermata di registrazione utente;
- **Post-condizioni**: l'utente ha compilato il form per la registrazione ed ora possiede le credenziali per l'autenticazione alla piattaforma API Market;
- **Descrizione**: l'utente compila il form per la registrazione di un account, provvedendo ad inserire tutte le informazioni obbligatorie richieste. Confermando la registrazione, i services di API Market provvedono ad inserire nel database il nuovo utente. A registrazione avvenuta, l'utente riceve un messaggio di successo e viene reindirizzato alla pagina di login.

5.1.2 Autenticazione

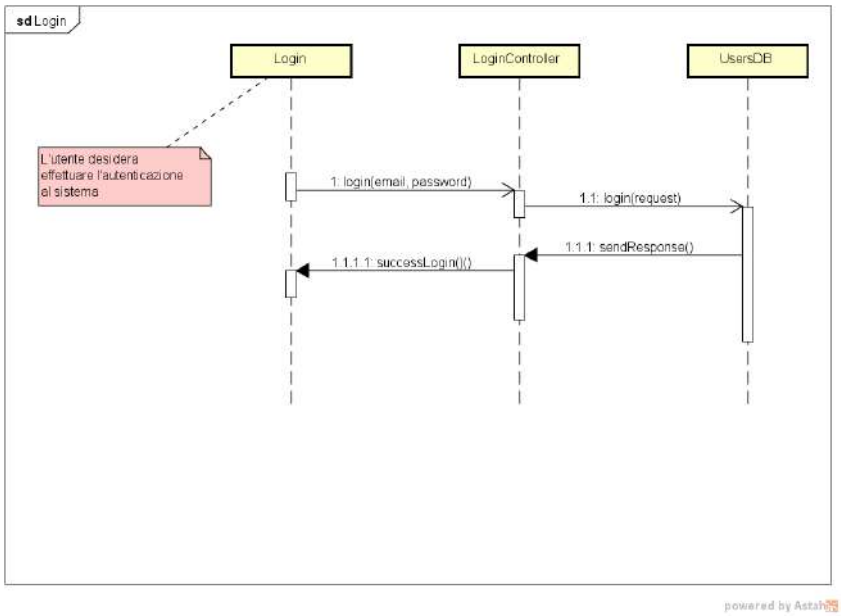

Figura 97: Diagramma di sequenza: Autenticazione

- **Pre-condizioni**: l'utente si trova nella schermata di autenticazione;
- **Post-condizioni**: l'utente ha compilato il form per il login ed ora risulta un utente autenticato al sistema API Market;
- **Descrizione**: l'utente compila il form per l'autenticazione, provvedendo ad inserire la propria email e password con le quali si era registrato. Confermando l'autenticazione, la piattaforma API Market provvede a creare una sessione per l'utente autenticato, il quale, nel caso di autenticazione avvenuta con successo, viene reindirizzato alla home dell'applicazione.

5.1.3 Recupero password

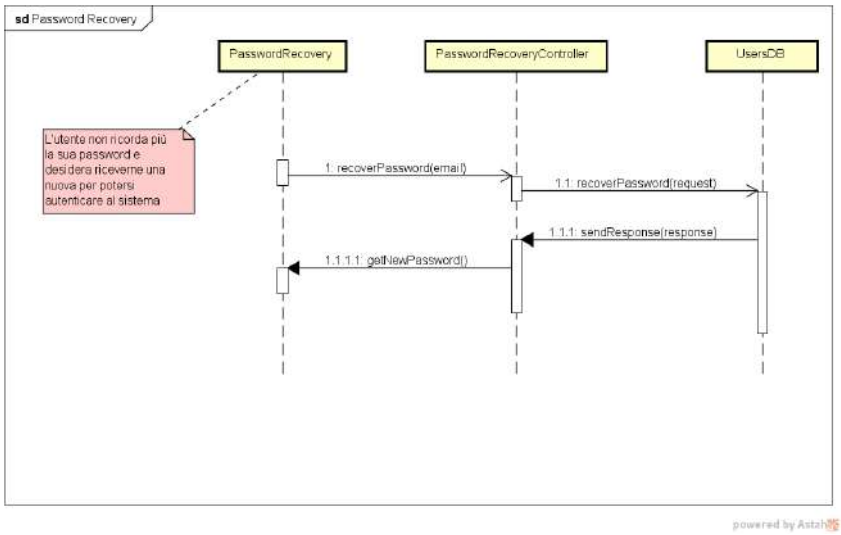

Figura 98: Diagramma di sequenza: Recupero password

- **Pre-condizioni**: l'utente si trova nella schermata di recupero password;
- **Post-condizioni**: l'utente ha compilato il form per il recupero della propria password tramite email ed ora può ri-effettuare l'autenticazione alla piattaforma API Market;
- **Descrizione**: l'utente compila il form per il recupero della password relativa al proprio account, inserendo l'email dove il sistema inoltrerà la nuova password generata. Confermando l'operazione, i services di API Market provvedono ad aggiornare il campo password nel database degli utenti. A recupero avvenuto, l'utente riceve un messaggio di successo e viene reindirizzato alla pagina di login.

5.1.4 Ricerca API

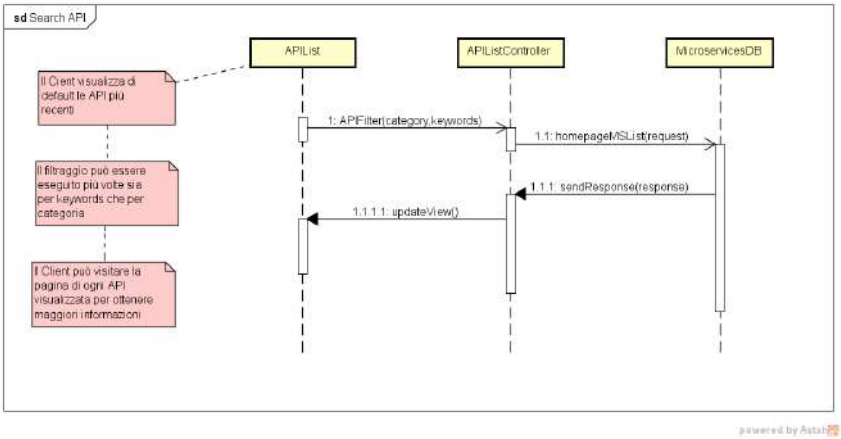

Figura 99: Diagramma di sequenza: Ricerca API

- **Pre-condizioni**: l'utente si trova nella homepage dell'applicazione;
- **Post-condizioni**: l'utente ha visualizzato le API secondo la categoria e le keywords inserite;
- **Descrizione**: di default, la homepage di API Market mostra le API registrate più recentemente. L'utente può immettere delle keywords nell'apposito riquadro per alterare i risultati, e scegliere di restringere la ricerca ad una singola categoria. I risultati visualizzati cambiano dinamicamente non appena l'applicazione web riceve risposta dai servizi che la collegano ai database.

5.1.5 Acquisto API

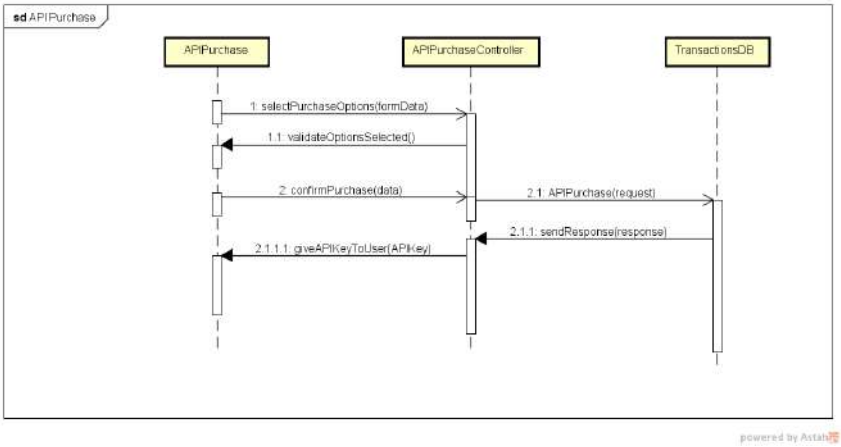

Figura 100: Diagramma di sequenza: Acquisto API

- **Pre-condizioni**: il cliente si trova nella schermata di acquisto di una specifica API;
- **Post-condizioni**: il cliente ha ricevuto l'API Key per l'API acquistata, secondo le modalità da lui scelte, e gli sono stati sottratti i corrispondenti crediti;
- **Descrizione**: il cliente compila il form per l'acquisto dell'API desiderata, visualizzando il preventivo di crediti spesi in base ai parametri scelti. Confermando l'acquisto (che può essere anche un rinnovo), i services di API Market provvedono a generare una apikey ed a scalare i crediti dall'account utente. L'utente potrà visualizzare l'API Key ed un messaggio di ringraziamento.

5.1.6 Inserimento API

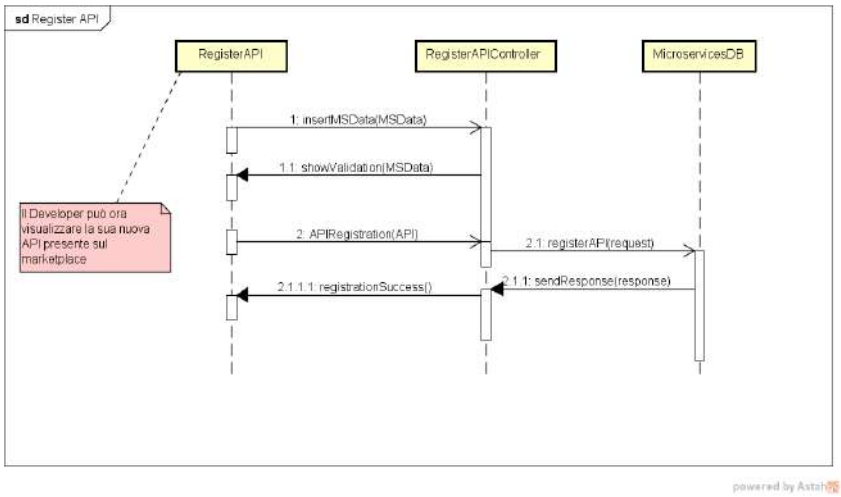

Figura 101: Diagramma di sequenza: Inserimento API

- **Pre-condizioni**: lo sviluppatore si trova nella schermata di registrazione di una nuova API;
- **Post-condizioni**: lo sviluppatore ha registrato la sua nuova API ed essa è ora disponibile in API Market;
- **Descrizione**: lo sviluppatore compila il form per la registrazione di una nuova API, provvedendo anche a caricare sul server di API Market i file del logo e della documentazione PDF. Confermando la registrazione, i services di API Market provvedono ad inserire nel database la nuova API e renderla accessibile attraverso il Gateway. A registrazione avvenuta, lo sviluppatore riceve un messaggio di successo.

5.1.7 Ricarica saldo conto virtuale

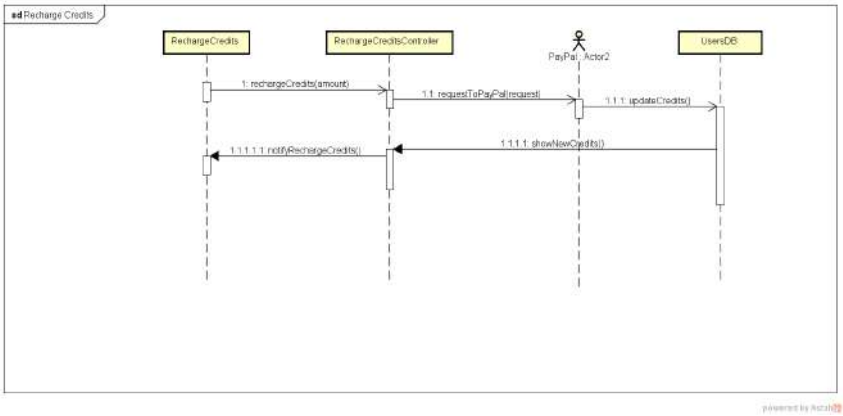

Figura 102: Diagramma di sequenza: Ricarica saldo conto virtuale

- **Pre-condizioni**: il cliente si trova nella schermata di ricarica del proprio conto virtuale;
- **Post-condizioni**: il cliente ha effettuato la ricarica dei propri crediti;
- **Descrizione**: il cliente stabilisce l'ammontare dei crediti da ricaricare nel proprio conto virtuale. A decisione avvenuta, verrà reindirizzato alla procedura di acquisto di PayPal, che lo guiderà nell'acquisto. Portata a termine la transazione, PayPal avvertirà i services di API Market che si occuperanno di incrementare i crediti del conto virtuale del cliente.

6 Tracciamento classi

6.1 Tracciamento Classi - Requisiti

Nome Classe	Codice Requisito
APIM::FrontEnd::App::Index	RFO1
	RFO2
	RFO4
	RFO4.1
	RFO4.2
	RFO4.3
	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO11
APIM::FrontEnd::App::Views::RegisterUser	RFO1
	RFO1.1
	RFO1.2
	RFO1.3
	RFO1.4
	RFO1.5
	RFO1.6
	RFO1.7
	RFO1.8
	RFO1.9
	RFO1.10
APIM::FrontEnd::App::Views::Login	RFO2
	RFO2.1
	RFO2.1.1
	RFO2.1.2
	RFO2.1.3

	RFO2.1.4
APIM::FrontEnd::App::Views::PasswordRecovery	RFD3
	RFD3.1
	RFD3.2
	RFD3.3
	RFD3.4
APIM::FrontEnd::App::Views::API	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO5
	RFO5.1
	RFO5.2
	RFO5.3
	RFO5.4
	RFO5.5
	RFO5.5.1
	RFO5.5.2
	RFO5.6
	RFO5.6.1
	RFO5.6.2
	RFO5.7
	RFD5.7.1
	RFO5.7.2
	RFO5.8
	RFO5.9
	RFO5.10
	RFO5.11
	RFO5.12
	RFO5.13
	RFO7

	RFO7.1
	RFO7.1.1
	RFO7.1.2
	RFO7.1.3
	RFO7.2
	RFO7.3
	RFO7.4
	RFO7.5
	RFD7.5.1
	RFO7.5.2
	RFO7.5.3
	RFO7.6
APIM::FrontEnd::App::Views::ProfileManager	RFO6
	RFO10
	RFO10.1
	RFO10.1.1
	RFO10.1.1.1
	RFO10.1.1.2
	RFO10.1.1.3
	RFO10.1.1.4
	RFO10.1.1.5
	RFO10.1.1.6
	RFO10.1.2
	RFO10.1.2.1
	RFO10.1.2.2
	RFO10.1.2.3
	RFO10.1.2.4
	RFO10.1.2.5
	RFD10.1.2.6
	RFO10.1.2.7
	RFO10.1.2.8
	RFD10.1.2.8

	RFO10.2
	RFO11
APIM::FrontEnd::App::Views::ResetPassword	RFD3
	RFD3.1
	RFD3.2
	RFD3.3
	RFD3.4
APIM::FrontEnd::App::Views::RegisterAPI	RFO9
	RFO9.1
	RFO9.2
	RFO9.3
	RFO9.4
	RFO9.5
	RFD9.6
	RFD9.7
	RFO9.8
	RFO9.8.1
	RFO9.8.2
	RFO9.8.3
	RFO9.9
	RFD9.10
	RFD9.11
	RFO9.12
	RFD9.13
APIM::FrontEnd::App::Views::SellingPolicy	RFD5.7.1
	RFO5.8
	RFO5.12
APIM::FrontEnd::App::Views::APIRegistered	RFO8
	RFO8.1
	RFO8.2
	RFO8.2.1
	RFO8.2.2

RFO8.2.3
RFO8.2.4
RFO8.2.4.1
RFO8.2.4.2
RFD8.2.4.3
RFO8.2.4.4
RFO8.2.4.5
RFO8.2.4.6
RFO8.2.4.7
RFO8.2.4.8
RFO8.2.4.9
RFO8.2.4.10
RFO8.2.4.11
RFO8.2.5
RFO8.2.6
RFO8.2.7
RFO8.2.8
RFO8.2.9
RFO8.2.9.1
RFO8.2.9.2
RFO9
RFO9.1
RFO9.2
RFD9.3
RFO9.4
RFO9.5
RFD9.6
RFD9.7
RFO9.8
RFO9.8.1
RFO9.8.2
RFO9.8.3

	RFO9.9
	RFD9.10
	RFD9.11
	RFO9.12
	RFD9.13
APIM::FrontEnd::App::Views::APIPurchased	RFO6
	RFO6.1
	RFO6.2
	RFO6.2.1
	RFO6.2.2
	RFO6.2.3
	RFO6.2.4
	RFO6.2.5
	RFD6.2.6
	RFD6.2.7
	RFO7.5.2
	RFO7.5.3
APIM::FrontEnd::App::Views::APIList	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO5
	RFO5.1
	RFO5.2
	RFO5.3
	RFO5.4
	RFO5.5
	RFO5.5.1
	RFO5.5.2
	RFO5.6
	RFO5.6.1

	RFO5.6.2
	RFO5.7
	RFD5.7.1
	RFO5.7.2
	RFO5.8
	RFO5.9
	RFO5.10
	RFO5.11
	RFO5.12
	RFO5.13
	RFO7
APIM::FrontEnd::App::Views::TransactionsList	RF010.3
	RF010.3.1
	RF010.3.1.1
	RF010.3.1.2
	RF010.3.2
	RF010.3.2.1
	RF010.3.2.2
	RF010.3.2.3
	RF010.3.2.4
	RF010.3.2.5
APIM::FrontEnd::App::Views::VirtualAccount	RFO10.2
	RFO10.2.1
	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
APIM::FrontEnd::App::Views::AdminManager	RFO12

	RFO12.1
	RFO12.1.1
	RFO12.1.1.1
	RFO12.1.1.1.1
	RFO12.1.1.1.2
	RFD12.1.1.1.3
	RFD12.1.1.1.4
	RFO12.1.1.1.5
	RFO12.1.1.1.5.1
	RFF12.1.1.1.5.2
	RFD12.1.1.1.6
	RFO12.1.1.2
	RFO12.1.1.2.1
	RFO12.1.1.2.2
	RFD12.1.1.3
	RFO12.1.1.3.1
	RFD12.1.1.3.2
	RFO12.1.1.3.3
APIM::FrontEnd::App::Views::AdminModeration	RFO12.2
	RFO12.2.1
	RFO12.2.1.1
	RFO12.2.1.1.1
	RFO12.2.1.1.2
	RFO12.2.1.2
	RFO12.2.1.2.1
	RFO12.2.1.2.2
	RFO12.2.1.3
	RFO12.2.1.4
	RFO12.2.1.5
	RFO12.2.1.5.1
	RFO12.2.1.5.2
APIM::FrontEnd::App::Models::UserDetailsModel	RFO1

RFO1.1
RFO1.2
RFO1.3
RFO1.4
RFO1.5
RFO1.6
RFO1.7
RFO1.8
RFO1.9
RFO1.10
RFO6
RFO10
RFO10.1
RFO10.1.1
RFO10.1.1.1
RFO10.1.1.2
RFO10.1.1.3
RFO10.1.1.4
RFO10.1.1.5
RFO10.1.1.6
RFO10.1.2
RFO10.1.2.1
RFO10.1.2.2
RFO10.1.2.3
RFO10.1.2.4
RFO10.1.2.5
RFD10.1.2.6
RFO10.1.2.7
RFO10.1.2.8
RFD10.1.2.8
RFO10.2
RFO10.2.1

	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
	RFO11
APIM::FrontEnd::App::Models::MicroserviceModel	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO5
	RFO5.1
	RFO5.2
	RFO5.3
	RFO5.4
	RFO5.5
	RFO5.5.1
	RFO5.5.2
	RFO5.6
	RFO5.6.1
	RFO5.6.2
	RFO5.7
	RFD5.7.1
	RFO5.7.2
	RFO5.8
	RFO5.9
	RFO5.10
	RFO5.11

RFO5.12

RFO5.13

RFO6

RFO6.1

RFO6.2

RFO6.2.1

RFO6.2.2

RFO6.2.3

RFO6.2.4

RFO6.2.5

RFD6.2.6

RFD6.2.7

RFO7

RFO7.1

RFO7.1.1

RFO7.1.2

RFO7.1.3

RFO7.2

RFO7.3

RFO7.4

RFO7.5

RFD7.5.1

RFO7.5.2

RFO7.5.3

RFO7.6

RFO8

RFO8.1

RFO8.2

RFO8.2.1

RFO8.2.2

RFO8.2.3

RFO8.2.4

RFO8.2.4.1
RFO8.2.4.2
RFD8.2.4.3
RFO8.2.4.4
RFO8.2.4.5
RFO8.2.4.6
RFO8.2.4.7
RFO8.2.4.8
RFO8.2.4.9
RFO8.2.4.10
RFO8.2.4.11
RFO8.2.5
RFO8.2.6
RFO8.2.7
RFO8.2.8
RFO8.2.9
RFO8.2.9.1
RFO8.2.9.2
RFO9
RFO9.1
RFO9.2
RFD9.3
RFO9.4
RFO9.5
RFD9.6
RFD9.7
RFO9.8
RFO9.8.1
RFO9.8.2
RFO9.8.3
RFO9.9
RFD9.10

	RFD9.11
	RFO9.12
	RFD9.13
APIM::FrontEnd::App::Models::TransactionModel	RFO10.2
	RFO10.2.1
	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
	RF010.3
	RF010.3.1
	RF010.3.1.1
	RF010.3.1.2
	RF010.3.2
	RF010.3.2.1
	RF010.3.2.2
	RF010.3.2.3
	RF010.3.2.4
	RF010.3.2.5
APIM::FrontEnd::App::Models::SLAMicroserviceModel	RFO5.7
	RFD5.7.1
	RFO5.7.2
APIM::FrontEnd::App::Controllers::UserRegistrationController	RFO1
	RFO1.1
	RFO1.2
	RFO1.3
	RFO1.4
	RFO1.5

	RFO1.6
	RFO1.7
	RFO1.8
	RFO1.9
	RFO1.10
APIM::FrontEnd::App::Controllers::LoginController	RFO2
	RFO2.1
	RFO2.1.1
	RFO2.1.2
	RFO2.1.3
	RFO2.1.4
APIM::FrontEnd::App::Controllers::PasswordRecoveryController	RFD3
	RFD3.1
	RFD3.2
	RFD3.3
	RFD3.4
APIM::FrontEnd::App::Controllers::SearchController	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
APIM::FrontEnd::App::Controllers::ProfileManagerController	RFO6
	RFO10
	RFO10.1
	RFO10.1.1
	RFO10.1.1.1
	RFO10.1.1.2
	RFO10.1.1.3
	RFO10.1.1.4
	RFO10.1.1.5
	RFO10.1.1.6
	RFO10.1.2

	RFO10.1.2.1
	RFO10.1.2.2
	RFO10.1.2.3
	RFO10.1.2.4
	RFO10.1.2.5
	RFD10.1.2.6
	RFO10.1.2.7
	RFO10.1.2.8
	RFD10.1.2.8
	RFO10.2
	RFO11
APIM::FrontEnd::App::Controllers::ResetPasswordController	RFD3
	RFD3.1
	RFD3.2
	RFD3.3
	RFD3.4
APIM::FrontEnd::App::Controllers::APIRegistrationController	RFO9
	RFO9.1
	RFO9.2
	RFO9.3
	RFO9.4
	RFO9.5
	RFD9.6
	RFD9.7
	RFO9.8
	RFO9.8.1
	RFO9.8.2
	RFO9.8.3
	RFO9.9
	RFD9.10
	RFD9.11
	RFO9.12

	RFD9.13
APIM::FrontEnd::App::Controllers::SellingPolicyController	RFD5.7.1
	RFO5.8
	RFO5.12
APIM::FrontEnd::App::Controllers::APIRegisteredController	RFO8
	RFO8.1
	RFO8.2
	RFO8.2.1
	RFO8.2.2
	RFO8.2.3
	RFO8.2.4
	RFO8.2.4.1
	RFO8.2.4.2
	RFD8.2.4.3
	RFO8.2.4.4
	RFO8.2.4.5
	RFO8.2.4.6
	RFO8.2.4.7
	RFO8.2.4.8
	RFO8.2.4.9
	RFO8.2.4.10
	RFO8.2.4.11
	RFO8.2.5
	RFO8.2.6
	RFO8.2.7
	RFO8.2.8
	RFO8.2.9
	RFO8.2.9.1
	RFO8.2.9.2
	RFO9
	RFO9.1
	RFO9.2

	RFD9.3
	RFO9.4
	RFO9.5
	RFD9.6
	RFD9.7
	RFO9.8
	RFO9.8.1
	RFO9.8.2
	RFO9.8.3
	RFO9.9
	RFD9.10
	RFD9.11
	RFO9.12
	RFD9.13
APIM::FrontEnd::App::Controllers::APIPurchasedController	RFO6
	RFO6.1
	RFO6.2
	RFO6.2.1
	RFO6.2.2
	RFO6.2.3
	RFO6.2.4
	RFO6.2.5
	RFD6.2.6
	RFD6.2.7
	RFO7.5.2
	RFO7.5.3
APIM::FrontEnd::App::Controllers::APIListController	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO5

	RFO5.1
	RFO5.2
	RFO5.3
	RFO5.4
	RFO5.5
	RFO5.5.1
	RFO5.5.2
	RFO5.6
	RFO5.6.1
	RFO5.6.2
	RFO5.7
	RFD5.7.1
	RFO5.7.2
	RFO5.8
	RFO5.9
	RFO5.10
	RFO5.11
	RFO5.12
	RFO5.13
	RFO7
APIM::FrontEnd::App::Controllers::TransactionsListController	RF010.3
	RF010.3.1
	RF010.3.1.1
	RF010.3.1.2
	RF010.3.2
	RF010.3.2.1
	RF010.3.2.2
	RF010.3.2.3
	RF010.3.2.4
	RF010.3.2.5
APIM::FrontEnd::App::Controllers::VirtualAccountController	RFO10.2
	RFO10.2.1

	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
APIM::FrontEnd::App::Controllers::AdminManagerController	RFO12
	RFO12.1
	RFO12.1.1
	RFO12.1.1.1
	RFO12.1.1.1.1
	RFO12.1.1.1.2
	RFD12.1.1.1.3
	RFD12.1.1.1.4
	RFO12.1.1.1.5
	RFO12.1.1.1.5.1
	RFF12.1.1.1.5.2
	RFD12.1.1.1.6
	RFO12.1.1.2
	RFO12.1.1.2.1
	RFO12.1.1.2.2
	RFD12.1.1.3
	RFO12.1.1.3.1
	RFD12.1.1.3.2
	RFO12.1.1.3.3
APIM::FrontEnd::App::Controllers::AdminModerationController	RFO12.2
	RFO12.2.1
	RFO12.2.1.1
	RFO12.2.1.1.1
	RFO12.2.1.1.2

	RFO12.2.1.2
	RFO12.2.1.2.1
	RFO12.2.1.2.2
	RFO12.2.1.3
	RFO12.2.1.4
	RFO12.2.1.5
	RFO12.2.1.5.1
	RFO12.2.1.5.2
APIM::BackEnd::Services::UsersDB	RFO1
	RFO1.1
	RFO1.2
	RFO1.3
	RFO1.4
	RFO1.5
	RFO1.6
	RFO1.7
	RFO1.8
	RFO1.9
	RFO1.10
	RFO6
	RFO10
	RFO10.1
	RFO10.1.1
	RFO10.1.1.1
	RFO10.1.1.2
	RFO10.1.1.3
	RFO10.1.1.4
	RFO10.1.1.5
	RFO10.1.1.6
	RFO10.1.2
	RFO10.1.2.1
	RFO10.1.2.2

	RFO10.1.2.3
	RFO10.1.2.4
	RFO10.1.2.5
	RFD10.1.2.6
	RFO10.1.2.7
	RFO10.1.2.8
	RFD10.1.2.8
	RFO10.2
	RFO10.2.1
	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
	RFO11
APIM::BackEnd::Services::MicroservicesDB	RFO4.3.1
	RFO4.3.2
	RFO4.3.3
	RFO4.3.4
	RFO4.3.5
	RFO5
	RFO5.1
	RFO5.2
	RFO5.3
	RFO5.4
	RFO5.5
	RFO5.5.1
	RFO5.5.2
	RFO5.6

	RFO5.6.1
	RFO5.6.2
	RFO5.7
	RFD5.7.1
	RFO5.7.2
	RFO5.8
	RFO5.9
	RFO5.10
	RFO5.11
	RFO5.12
	RFO5.13
	RFO6
	RFO6.1
	RFO6.2
	RFO6.2.1
	RFO6.2.2
	RFO6.2.3
	RFO6.2.4
	RFO6.2.5
	RFD6.2.6
	RFD6.2.7
	RFO7
	RFO7.1
	RFO7.1.1
	RFO7.1.2
	RFO7.1.3
	RFO7.2
	RFO7.3
	RFO7.4
	RFO7.5
	RFD7.5.1
	RFO7.5.2

RFO7.5.3

RFO7.6

RFO8

RFO8.1

RFO8.2

RFO8.2.1

RFO8.2.2

RFO8.2.3

RFO8.2.4

RFO8.2.4.1

RFO8.2.4.2

RFD8.2.4.3

RFO8.2.4.4

RFO8.2.4.5

RFO8.2.4.6

RFO8.2.4.7

RFO8.2.4.8

RFO8.2.4.9

RFO8.2.4.10

RFO8.2.4.11

RFO8.2.5

RFO8.2.6

RFO8.2.7

RFO8.2.8

RFO8.2.9

RFO8.2.9.1

RFO8.2.9.2

RFO9

RFO9.1

RFO9.2

RFD9.3

RFO9.4

	RFO9.5
	RFD9.6
	RFD9.7
	RFO9.8
	RFO9.8.1
	RFO9.8.2
	RFO9.8.3
	RFO9.9
	RFD9.10
	RFD9.11
	RFO9.12
	RFD9.13
APIM::BackEnd::Services::TransactionsDB	RFO10.2
	RFO10.2.1
	RFO10.2.2
	RFO10.2.2.1
	RFO10.2.2.2
	RFO10.3.3
	RFO10.3.3.1
	RFO10.3.3.2
	RFO10.3.3.3
	RFO10.3.3.4
	RF010.3
	RF010.3.1
	RF010.3.1.1
	RF010.3.1.2
	RF010.3.2
	RF010.3.2.1
	RF010.3.2.2
	RF010.3.2.3
	RF010.3.2.4
	RF010.3.2.5

APIM::BackEnd::Services::SLADB	RFO5.7
	RFD5.7.1
	RFO5.7.2
APIM::BackEnd::Services::FileHandlerDB	RFD4.3.4
	RFO5.5.2
	RFO5.6
	RFO5.6.1
	RFD6.2.3
	RFD6.2.7
	RFO8.2.4.5
	RFO8.2.4.7
	RFD9.10
	RFD9.11
	RFD10.1.1.5
	RFD10.1.2.6

Tabella 1: Tracciamento Classi-Requisiti

6.2 Tracciamento Requisiti - Classi

Codice Requisito	Nome Classe
RF01	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
RFO1.1	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.2	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.3	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.3	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.4	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
RFO1.5	APIM::FrontEnd::App::Views::RegisterUser

	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
RFO1.6	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.7	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO1.8	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
RFO1.9	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
RFO1.10	APIM::FrontEnd::App::Views::RegisterUser
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::UserRegistrationController
	APIM::BackEnd::Services::UsersDB
	APIM::FrontEnd::App::Views::RegisterUser
RFO2	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::Login
	APIM::FrontEnd::App::Controllers::LoginController
RFO2.1	APIM::FrontEnd::App::Views::Login
	APIM::FrontEnd::App::Views::Login
	APIM::FrontEnd::App::Controllers::LoginController
RFO2.1.1	APIM::FrontEnd::App::Views::Login

	APIM::FrontEnd::App::Controllers::LoginController
RFO2.1.2	APIM::FrontEnd::App::Views::Login
	APIM::FrontEnd::App::Controllers::LoginController
RFO2.1.3	APIM::FrontEnd::App::Views::Logi
	APIM::FrontEnd::App::Controllers::LoginController
RFO2.1.4	APIM::FrontEnd::App::Views::Login
	APIM::FrontEnd::App::Controllers::LoginController
RFD3	APIM::FrontEnd::App::Views::PasswordRecovery
	APIM::FrontEnd::App::Views::ResetPassword
	APIM::FrontEnd::App::Controllers::PasswordRecoveryController
	APIM::FrontEnd::App::Controllers::ResetPasswordController
RFD3.1	APIM::FrontEnd::App::Views::PasswordRecovery
	APIM::FrontEnd::App::Views::ResetPassword
	APIM::FrontEnd::App::Controllers::PasswordRecoveryController
	APIM::FrontEnd::App::Controllers::ResetPasswordController
RFD3.2	APIM::FrontEnd::App::Views::PasswordRecovery
	APIM::FrontEnd::App::Views::ResetPassword
	APIM::FrontEnd::App::Controllers::PasswordRecoveryController
	APIM::FrontEnd::App::Controllers::ResetPasswordController
RFD3.3	APIM::FrontEnd::App::Views::PasswordRecovery
	APIM::FrontEnd::App::Views::ResetPassword
	APIM::FrontEnd::App::Controllers::PasswordRecoveryController
	APIM::FrontEnd::App::Controllers::ResetPasswordController
RFD3.4	APIM::FrontEnd::App::Views::PasswordRecovery
	APIM::FrontEnd::App::Views::ResetPassword
	APIM::FrontEnd::App::Controllers::PasswordRecoveryController
	APIM::FrontEnd::App::Controllers::ResetPasswordController
RFO4	APIM::FrontEnd::App::Index
RFO4.1	APIM::FrontEnd::App::Index
RFO4.2	APIM::FrontEnd::App::Index
RFO4.3	APIM::FrontEnd::App::Index
RFO4.3.1	APIM::FrontEnd::App::Index

	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SearchController
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO4.3.2	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SearchController
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO4.3.3	APIM::FrontEnd::App::Index RFO1
	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SearchController
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO4.3.4	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SearchController
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO4.3.5	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SearchController

	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
RFO5.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList RFO4.3.1
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.3	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.4	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.5	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.5.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel

	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
R	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFO5.5.6	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFO5.6.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFO5.6.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.7	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Models::SLAMicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::SLADB

RFD5.7.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::SellingPolicy
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Models::SLAMicroserviceModel
	APIM::FrontEnd::App::Controllers::SellingPolicyController
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::SLADB
RFO5.7.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Models::SLAMicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::SLADB
RFO5.8	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::SellingPolicy
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SellingPolicyController
	APIM::BackEnd::Services::MicroservicesDB
RFO5.9	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
RFO5.10	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB

RFO5.11	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO5.12	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::SellingPolicy
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::SellingPolicyController
	APIM::FrontEnd::App::Controllers::APIListController
RFO5.13	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO6	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::MicroservicesDB
RFO6.1	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB

	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Controllers::APIPurchasedController
RFO6.2	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::FrontEnd::App::Views::APIPurchased
RFO6.2.1	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO6.2.2	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO6.2.3	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO6.2.4	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO6.2.5	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFD6.2.6	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFD6.2.7	APIM::FrontEnd::App::Views::APIPurchased

	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFO7	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIList
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIListController
	APIM::BackEnd::Services::MicroservicesDB
RFO7.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.1.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.1.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.1.3	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.3	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.4	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.5	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel

	APIM::BackEnd::Services::MicroservicesDB
RF7.5.1	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO7.5.2	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO7.5.3	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Views::APIPurchased
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIPurchasedController
	APIM::BackEnd::Services::MicroservicesDB
RFO7.6	APIM::FrontEnd::App::Views::API
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::BackEnd::Services::MicroservicesDB
RFO8	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.1	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.1	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController

	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.2	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.3	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.1	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.2	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFD8.2.4.3	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.4	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.5	APIM::FrontEnd::App::Views::APIRegistered APIM::FrontEnd::App::Models::MicroserviceModel APIM::FrontEnd::App::Controllers::APIRegisteredController

	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFO8.2.4.6	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.7	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.8	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.9	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.10	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.4.11	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.5	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO8.2.6	APIM::FrontEnd::App::Views::APIRegistered

	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
FO8.2.7	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
FO8.2.8	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
FO8.2.9	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
FO8.2.9.1	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
FO8.2.9.2	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.1	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel

	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.2	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
RFO9.3	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Controllers::APIRegistrationController
RFO9.4	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.5	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFD9.6	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFD9.7	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::BackEnd::Services::MicroservicesDB

RFO9.8	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.8.1	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.8.2	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.8.3	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.9	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFD9.10	APIM::FrontEnd::App::Views::RegisterAPI

	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::BackEnd::Services::MicroservicesDB
	APIM::BackEnd::Services::FileHandlerDB
RFD9.11	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::BackEnd::Services::MicroservicesDB
RFO9.12	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFD9.13	APIM::FrontEnd::App::Views::RegisterAPI
	APIM::FrontEnd::App::Views::APIRegistered
	APIM::FrontEnd::App::Models::MicroserviceModel
	APIM::FrontEnd::App::Controllers::APIRegistrationController
	APIM::FrontEnd::App::Controllers::APIRegisteredController
	APIM::BackEnd::Services::MicroservicesDB
RFO10	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.1	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsModel

	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.1	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.2	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.3	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.4	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.5	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.6	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.7	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode

	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.1.2.8	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsMode
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO10.2	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
RFO10.2.1	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.2.2	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.2.2.1	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.2.2.2	APIM::FrontEnd::App::Views::VirtualAccount

	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.3.3	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.3.3.1	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.3.3.2	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.3.3.3	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel
	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO10.3.3.4	APIM::FrontEnd::App::Views::VirtualAccount
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Models::TransactionModel

	APIM::FrontEnd::App::Controllers::VirtualAccountController
	APIM::BackEnd::Services::UsersDB
	APIM::BackEnd::Services::TransactionsDB
RFO11	APIM::FrontEnd::App::Index
	APIM::FrontEnd::App::Views::ProfileManager
	APIM::FrontEnd::App::Models::UserDetailsModel
	APIM::FrontEnd::App::Controllers::ProfileManagerController
	APIM::BackEnd::Services::UsersDB
RFO12	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.1	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.2	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.3	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.4	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.5	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.5.1	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.1.5.2	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController
RFD12.1.1.1.6	APIM::FrontEnd::App::Views::AdminManager
	APIM::FrontEnd::App::Controllers::AdminManagerController

RFO12.1.1.2	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.2.1	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.2.2	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.3	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.3.1	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.3.2	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.1.1.3.3	APIM::FrontEnd::App::Views::AdminManager APIM::FrontEnd::App::Controllers::AdminManagerController
RFO12.2	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.1	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.1.1	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.1.2	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.2	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.2.1	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.2.2	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.3	APIM::FrontEnd::App::Views::AdminModeration

	APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.4	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.5	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.5.1	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController
RFO12.2.1.5.2	APIM::FrontEnd::App::Views::AdminModeration APIM::FrontEnd::App::Controllers::AdminModerationController

Tabella 2: Tracciamento Requisiti-Classi