SWEG GROUP

Definizione di Prodotto

Versione 1.0.0
Data di Rilascio 08/05/2017
Redazione Piergiorgio Danieli
Sebastiano Marchesini
Validazione Pietro Lonardi
Responsabile Alberto Gelmi
Uso Esterno
Destinato ItalianaSoftware S.r.l
Prof. Vardanega Tullio
Prof. Cardin Riccardo

Sommario
Questo documento descrive in dettaglio l'architettura dell' APIMarket, le sue componenti e funzioni.

Registro Modifiche

Ver.	Modifica	Nome	Data
1.0.10	Capitolo 3 Aggiunta descrizione diagrammi di sequenza	Alberto Gelmi	26/06/2017
1.0.9	Capitolo 3 Aggiunte Immagini	Alberto Gelmi	21/06/2017
1.0.8	Capitolo 3 Rettificati sottocap.2 , 7, 8	Piergiorgio Danieli	21/06/2017
1.0.7	Capitolo 4 Rettificati sottocap.2.12 , 2.13, 2.25, 2.26 Aggiunti sottocap. 2.14, 2.15, 2.27, 2.28	Sebastiano Marchesini	21/06/2017
1.0.6	Capitolo 4 Aggiunti sottocap. 2.13, 2.25, 2.26	Sebastiano Marchesini	20/06/2017
1.0.5	Tracciamento	Piergiorgio Danieli	20/06/2017
1.0.4	Diagrammi di sequenza aggiunti	Alberto Gelmi	20/06/2017
1.0.3	Capitolo 3 Modificati ed aggiornati metodi	Piergiorgio Danieli	07/06/2017
1.0.2	Capitolo 4 Aggiunti sottocapitoli 1.14, 1.15, 1.16, 1.17 Modificare sottocapitoli 1.18	Sebastiano Marchesini	06/06/2017
1.0.2	Capitolo 6 Modificate tabelle 1, 2, 3 Modificare intestazioni tab. 1, 2, 3	Sebastiano Marchesini	06/06/2017
1.0.1	Capitolo 5 Modificate figure 34, 35, 36 Modificare intestazioni fig. 34, 35, 36	Sebastiano Marchesini	06/06/2017
1.0.1	Aggiunta introduzione	Sebastiano Marchesini	08/05/2017
1.0.0	Verifica capitoli	Lonardi Pietro	08/05/2017
0.0.10	Realizzati diagrammi di sequenza	Sebastiano Marchesini	03/05/2017
0.0.9	Scritto Capitolo 4 e 5	Gianluca Crivellaro	03/05/2017
0.0.8	Realizzati diagrammi di sequenza	Sebastiano Marchesini	03/05/2017
0.0.7	Disegnati schemi front end	Sebastiano Marchesini	03/05/2017
0.0.6	Schemi Back End	Alberto Gelmi	1/05/2017
0.0.5	Tracciamento	Piergiorgio Danieli	30/04/2017
0.0.4	Componenti Back End	Piergiorgio Danieli	26/04/2017
0.0.3	Standard di progetto	Piergiorgio Danieli	23/04/2017
0.0.2	Introduzione	Piergiorgio Danieli	22/04/2017
0.0.1	Impostazione documento	Piergiorgio Danieli	20/04/2017

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire nel dettaglio la struttura dell' APIMarket, approfondendo quanto gia' riportato nel documento *Specifica Tecnica v2.0.0*. Tale documeto fornisce una struttura dettagliata e completa che viene utilizzata dai Programmatori per le attivita' di codifica.

1.2 Scopo del prodotto

L'obiettivo e' costruire una piattaforma che permetta di acquistare o mettere a disposizione dei microservizi.

1.3 Glossario

Al fine di evitare ambiguità e ottimizzare la comprensione dei documenti, viene incluso un Glossario, nel quale saranno inseriti i termini tecnici, acronimi e parole che necessitano di essere chiarite.
Un glossario è una raccolta di termini di un ambito specifico e circoscritto. In questo caso per raccogliere termini desueti e specialistici inerenti al progetto.

1.4 Riferimenti

1.4.1 Informativi

- **Specifica Tecnica**:
  "Specifica Tecnica v2.0.0".
- **Analisi dei Requisiti**:
  "Specifica Tecnica v3.0.0".
- **Norme di Progetto**:
  "Specifica Tecnica v3.0.0".

1.4.2 Normativi

- **Documentazione Jolie**:
  http://docs.jolie-lang.org/#!documentation/.
- **Documentazione Angular2**:
  https://angular.io/docs/ts/latest/.
- **http://valor-software.com/ng2-charts/**:
  http://getbootstrap.com/.
- **Documentazione ng2 charts**:
  http://valor-software.com/ng2-charts/.
- **Documentazione ngx rating**:
  https://github.com/pleerock/ngx-rating.

1.4.3 Repository

Il codice è stato copiato in una repository pubblica utilizzando il sito di hosting GitHub.com al link:

- **Repository**:
  https://github.com/SwegGroup/apimarket.

2 Standard di progetto

2.1 Progettazione architetturale

Gli standard di progettazione architetturale sono definiti nel documento *SPecifica Tecnica v2.0.0*.

2.2 Documentazione codice

Gli standard per la scrittura della documentazione del codice sono definiti nel documento *Norme di progetto v3.0.0*.

2.3 Denominazione di entita' e relazioni

Tutti gli elementi definiti, siano essi package, classi, metodi od attributi devono avere una denominazione chiara ed autoesplicativa. Nel caso in cui il nome risulti essere lungo, e' preferibile anteporre la chiarezza alla lunghezza.
Sono ammesse abbreviazioni se:

- Immediatamente comprensibili;
- Non ambigue;
- Sufficientemente contestualizzate.

2.4 Codifica

Gli standard di programmazione sono definiti e descritti nel documento *Norme di Progetto v3.0.0*.

2.5 Strumenti di lavoro

Gli strumenti da adottare e le procedure da seguire per utilizzarli correttamente durante la realizzazione del prodotto software sono definiti nel documento *Norme di Progetto v3.0.0*.

3 Specifica componenti Back end

3.1 Server

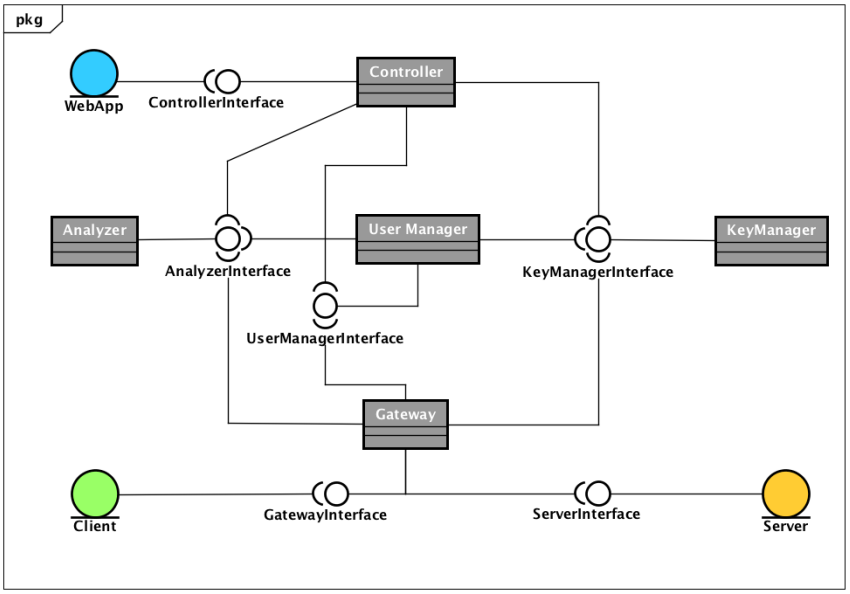

Figura 1: GeneralBackEnd

Contiene l'intera sezione back end. Non viene specificato se i metodi (che nel linguaggio Jolie si chiamano operazioni) sono pubblici o privati in quanto nel linguaggio da noi utilizzato sono tutti pubblici.

3.2 Server::Gateway

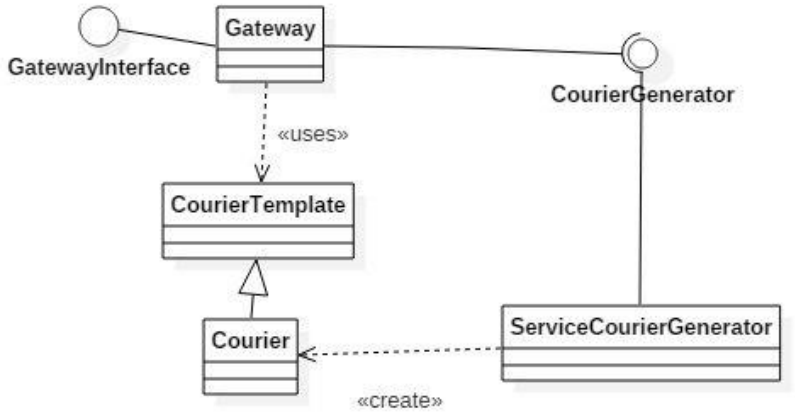

Figura 2: gateway

3.2.1 GatewayInterface

- **Descrizione :** La classe GatewayInterface e' l'interfaccia del Gateway, cioè quel passaggio che permette l'interazione tra gli utenti ed i microservizi.

- **Utilizzo:**Questa classe e' l'interfaccia in Jolie del Gateway, ogni volta che un utente fa una chiamata ad un microservizio, deve passare da qui.
- **Metodi**
  - setnewredirection Permette di creare un collegamento con i microservizi.

3.2.2 Gateway

- **Descrizione:** Questa classe implementa l'interfaccia del Gateway.
- **Utilizzo:** Viene utilizzata per gestire le chiamate che vengono effettuate dagli utenti che vogliono avere accesso ad un microservizio.
- **Attributi:**
  - Location : APIManagerLocation;
  - Protocol : sodep;
  - Interfaces : APIManagerInterface;
  - Interfaces : CourierGeneratorInterface;
- **Metodi:**
  - setnewredirection
    Crea un collegamento tra gli utenti che vogliono accedere ad una API e la API stessa.

3.2.3 CourierGeneratorInterface

- **Descrizione:** E' l'interfaccia della courier.
- **Utilizzo:** Viene utilizzata per creare una nuova courier.
- **Metodi:**
  - generate
    Crea una nuova courier.

3.2.4 CourierGenerator

- **Descrizione:** Implemntazione della courier.
- **Utilizzo:** Usata per creare una nuova funzione courier.
- **Attributi:**
- **Metodi:**
  - generate
    Crea una nuova funzione courier.

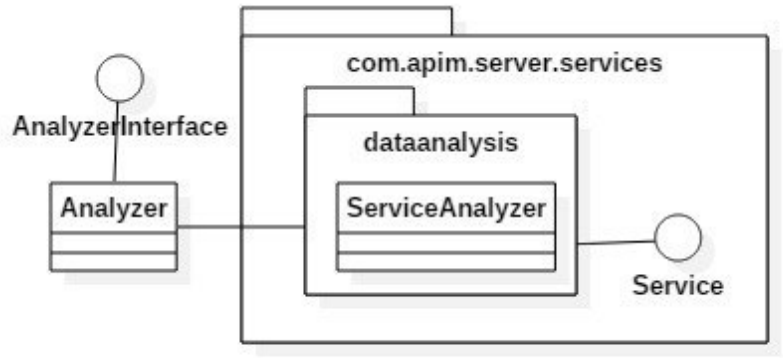

Figura 3: Analyzer

3.3 Server::Analyzer

3.3.1 AnalyzerInterface

- **Descrizione:** Questa e' l'interfaccia in Jolie del servizio che mette a disposizione le statistiche delle API.
- **Utilizzo:** Viene utilizzata per interrogare il database ed ottenere le statistiche delle API.
- **Metodi:**
  - getTime
    Ritorna il timestamp di adesso.
  - timeDiff
    Si occupa di calcolare la differenza tra due timestamp.
  - getValueSize
    Calcola in byte il peso dell'oggetto Jolie che viene creato.

3.3.2 Analyzer

- **Descrizione:** Questa classe permette di ottenere le statistiche di ogni microservizio.
- **Utilizzo:** Viene utilizzata per ottenere le staistiche.
- **Attributi:**
  - Interfaces : AnalyzerInterface
- **Metodi:**
  - getTime
    Ritorna il timestamp attuale.
  - timeDiff
    Calcola in millisecondi la differenza tra due timestamp.
  - getValueSize
    Ritorna la grandezza in byte.

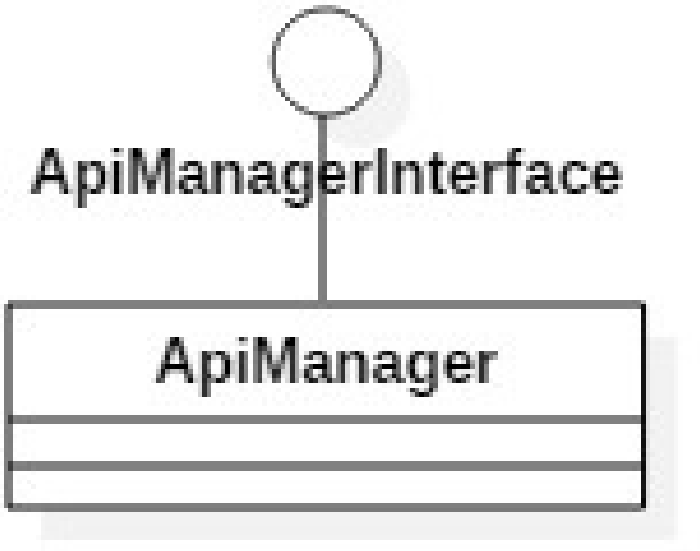

Figura 4: APIManager

3.4 Server::APIManager

3.4.1 APIManagerInterface

- **Descrizione:** Interfaccia in Jolie che consente ad un admin di modificare i dati relativi alle API.
- **Utilizzo:** Viene utilizzata da un admin per modificare i dati relativi ad una API.
- **Metodi:**
  - editAPIAdmin
    Un admin modifica i dati di una API;
  - deleteAPIAdmin
    Un admin elimina una API;
  - viewAPI
    Visualizza una API;
  - insertAPI
    Inserisce una nuova API;
  - searchAPI
    Cerca una API per nome;
  - deleteAPI
    Un utente normale elimina una propria API;
  - editAPI
    Un utente normale cambia i dati di una sua API;
  - viewStats
    Visualizza le statistiche di una API;

- buyAPI
  Acquista una API;
- incrementInterval
  Aumenta il tempo d'uso nelle statistiche.
- incrementSize
  Aumenta la dimensione in byte dell'utilizzo nelle statistiche.
- sendDocumentation
  Invia la documentazione
- sendLogo
  Invia il logo
- sendInterface
  Invia l'interfaccia
- getFrontPageApis
  Invia la pagina iniziale dell'API
- getFrontPageApisCategories
  Carica la pagina per catogoria
- getBoughtApisLIst
  Carica la pagina delle Api comprate
- getUploadedApisList
  Carica la pagina delle Api caricate
- addPolicyToApi
  Aggiunge una policy ad una Api
- removePolicyToApi
  Toglie una policy da una Api
- getPolicyFromId
  Visualizza una policy in base ad un ID
- getPolicyFromApiName
  Visualizza la policy di una Api dato il suo nome

3.4.2 APIManager

- **Descrizione:** Implementazione dell'interfaccia APIManagerInterface.
- **Utilizzo:** Usata per gestire le API da parte di un admin.
- **Attributi:**
  - Location : UserManagerLocation;
  - Protocol : sodep;
  - Interfaces : UserManagerInterface;
  - Location : GatewayLocation;
  - Protocol : sodep;
  - Interfaces : GatewayInterfaces;
  - Location : DAOApiLocation;
  - Protocol : sodep;

  - Interfaces : DAOInterface, DAOApiInterface;
  - Location : DAOUserLocation;
  - Protocol : sodep;
  - Interfaces : DaoInterface, DAOUSerInterface;
  - Location : DAOKeyLocation;
  - Protocol : sodep;
  - Interfaces : DAOKeyInterface;

- **Metodi:**
  - editAPIAdmin
    Concente ad un admin di modificare i dati di una API;
  - deleteAPIAdmin
    Consente ad un admin di eliminare una API;
  - viewAPI
    Visualizza una API;
  - insertAPI
    Inserisce una nuova API;
  - searchAPI
    Cerca una API per nome;
  - deleteAPI
    Un utente normale elimina una propria API;
  - editAPI
    Un utente normale cambia i dati di una sua API;
  - viewStats
    Visualizza le statistiche di una API;
  - buyAPI
    Acquista una API;
  - incrementInterval
    Aumenta il tempo d'uso nelle statistiche.
  - incrementSize
    Aumenta la dimensione in byte dell'utilizzo nelle statistiche.
  - sendDocumentation
    Carica la documentazione
  - sendLogo
    Carica il logo
  - sendInterface
    Carica l'interfaccia
  - getFrontPageApis
    Carica la pagina inziale dell'Api
  - getFrontPageApisCategories
    Carica la pagina inziale di una categoria
  - getBoughtApisLIst
    Visualizza la pagina delle Api comprate

- getuUploadedApisList
  Visualizza la pagina delle Api caricate
- addPolicyToApi
  Carica una policy ad una Api
- removePolicyFromApi
  Elimina una policy da una Api
- getPolicyFromId
  Visualizza le policy di una Api dato il suo Id
- getPoliciesFromApiName
  Visualizza la policy di una Api dato il suo nome

3.5 Server::CommentManager

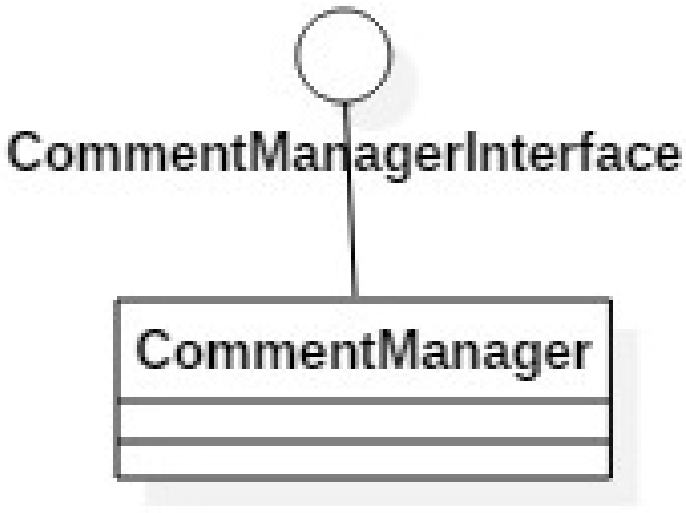

Figura 5: CommentManager

Questa classe permette di gestire i commenti.

3.5.1 CommentManagerInterface

- **Descrizione:** La classe CommentManagerInterface permette di inserire, eliminare, aggiornare un commento.
- **Utilizzo:** Viene utilizzata sia dagli utenti normali per inserire un commento, sia dagli admin per poterli cancellare.
- **Metodi:**
  - getComments
    Ritorna la lista di commenti associati ad una API.
  - putComment
    Inserisce un nuovo commento;

- updateComment
  Modifica un commento;
- deleteComment
  Un admin elimina un commento;

3.5.2 CommentManager

- **Descrizione:** Questa classe e' l'implementazione de CommentManagerInterface.
- **Utilizzo:** Viene utilizzata per inserire, modificare ed eliminare i commenti.
- **Attributi:**
  - Location : DAOCommentLocation;
  - Protocol : sodep:
  - Interfaces : DAOInterface, DAOCommentInterface;
- **Metodi:**
  - getComments
    Ritorna la lista di commenti relativi ad una certa API data in input.
  - putComment
    Permentte ad ogni utente di inserire un nuovo commento;
  - updateComment
    Permette ad un utente di modificare un commento;
  - deleteComment
    Permette ad un admin di eliminare un commento;

3.6 Server::Daos

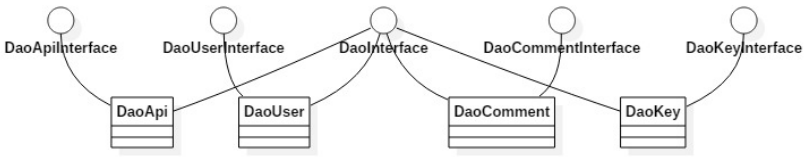

Figura 6: Daos

3.6.1 DaoApiInterface

- **Descrizione:** Questa classe serve per visualizzare le API piu' utilizzate, la loro pagina e le loro statistiche.
- **Utilizzo:** Viene utilizzata da tutti gli utenti per visualizzare i dettagli delle API.
- **Metodi:**
  - getMostDownloadedAPIsId
    Visualizza le 10 API piu' acquistate;

- getMostDownloadedAPIsIdFromcategory
  Visualizza le 10 API piu' acquistate per categoria;
- getAPIPreview
  Visualizza una anteprima di una API;
- getAPIPage
  Visualizza la pagina di una API;
- getAPIStats
  Visualizza le statistiche di una API;
- saveAcquisto
  Salva l'acquisto di una Api;
- getApiFromName
  Visualizza una Api;
- getApiFromDescription
  Visualizza una Api dalla descrizione;
- getBoughtApiFromUsername
  Visualizza la Api dato un username;
- addPolicyToApi
  Aggiunge una policy ad una Api;
- removePolicyFromApi
  Rimuove una policy da una Api;
- getPolicyFromId
  Visualizza le policy di una Api dato un id;
- getPoliciesFromApiName
  Visualizza la policy di una Api dato il suo nome.

3.6.2 DaoApi

- **Descrizione:** Questa classe e' l'implentazione di DaoApiInterface.
- **Utilizzo:** Viene utilizzata per interfacciarsi con le entita' del database.
- **Attributi:**
  - Location : DAOApiLocation;
  - Protocol : sodep;
  - Interfaces : DAOInterface, DAOApiInterface;
- **Metodi:**
  - getMostDownloadedAPIsId
    Visualizza l'elenco delle 10 API piu' acquistate;
  - getMostDownloadedAPIsIdFromcategory
    Visualizza l'elennco delle 10 API piu' acquistate per categoria;
  - getAPIPreview
    Visualizza una anteprima di una API;
  - getAPIPage
    Visualizza la pagina di una API;

- getAPIStats
  Visualizza le statistiche di una API;
- saveAcquisto
  Salva l'acquisto di una Api;
- getApiFromName
  Visualizza una Api;
- getApiFromDescription
  Visualizza una Api dalla descrizione;
- getBoughtApiFromUsername
  Visualizza la Api dato un username;
- addPolicyToApi
  Aggiunge una policy ad una Api;
- removePolicyFromApi
  Rimuove una policy da una Api;
- getPolicyFromId
  Visualizza le policy di una Api dato un id;
- getPoliciesFromApiName
  Visualizza la policy di una Api dato il suo nome.

3.6.3 DaoCommentInterface

- **Descrizione:** Permette di interfacciari con la tabella commenti del database.
- **Utilizzo:** Viene utilizzata da tutti gli utenti registrati che vogliono inserire, modificare o eliminare un commento.
- **Metodi:**
  - getCommentsFromApiId
    Permette di inserire un commento.

3.6.4 DaoComment

- **Descrzione:** Questa classe e' l'implementazione di DaoCommentInterface.
- **Utilizzo:** Usata da tutti gli utenti autenticati.
- **Attributi:**
  - Location : DAOCommentLocation;
  - Protocol : sodep;
  - Interfaces : DAOInterface, DAOCommentInterface;
- **Metodi:**
  - getCommentsFromApiId
    Permette di inserire un commento relativo ad una API.

3.6.5 DaoInterface

- **Descrizione:** Permette di interfacciarsi con tutte le tabelle del database.
- **Utilizzo:** Viene usata ogni qual volta c'e' necessita' di accedere al database.
- **Metodi:**
  - create
    Crea un nuovo oggetto;
  - delete
    Elimina un oggetto;
  - find
    Cerca un oggetto;
  - update
    Modifica un oggetto;
  - findAll
    Cerca tutti gli oggetti di uno stesso tipo indicato;

3.6.6 DaoKeyInterface

- **Descrizione:** Gestisce le chiavi associate alle API;
- **Utilizzo:** Viene utilizzato per creare chiavi;
- **Metodi:**
  - create
    Crea una nuova chiave;
  - delete
    Elimina una chiave;
  - find
    Effettua una ricerca di una chiave;
  - update
    Rinnova una chiave;

3.6.7 DaoKey

- **Descrizione:** Implementazione di DaoKeyInterface.
- **Utilizzo:** Serve per accedere alla tabella Chiavi del database.
- **Attributi:**
  - Location : DAOKeyLocation;
  - Protocol : sodep;
  - Interfaces : DAOInterface;
- **Metodi:**
  - create
    Crea una nuova chiave;

- delete
  Elimina una chiave;
- find
  Effettua una ricerca di una chiave;
- update
  Rinnova una chiave;

3.6.8 DaoUserInterface

- **Descrizione:** Interfaccia per la tabella User.
- **Utilizzo:** Accede alla tabella User.
- **Metodi:**
  - seacrhByUsername
    Ricerca un utente per username;
  - createSession
    Crea una sessione;
  - verifySession
    Verifica la validità di una sessione;
  - deleteSession
    Cancella la sessione;
  - generateRevoeryCode
    Crea il recupero della password;
  - verifiyRecovery
    Controlla il recupero della password.

3.6.9 DaoUser

- **Descrizione:** Implemetazione di DaoUserInterface.
- **Utilizzo:** Cerca un utente dato un username.
- **Attributi:**
  - Location : DAOUserLocation;
  - Protocol : sodep;
  - Interfaces : DAOInterface, DAOUserInterface;
- **Metodi:**
  - seacrhByUsername
    Ricerca un utente per username;
  - createSession
    Crea una sessione;
  - verifySession
    Verifica la validità di una sessione;
  - deleteSession
    Cancella la sessione.

3.7 Server::KeyManager

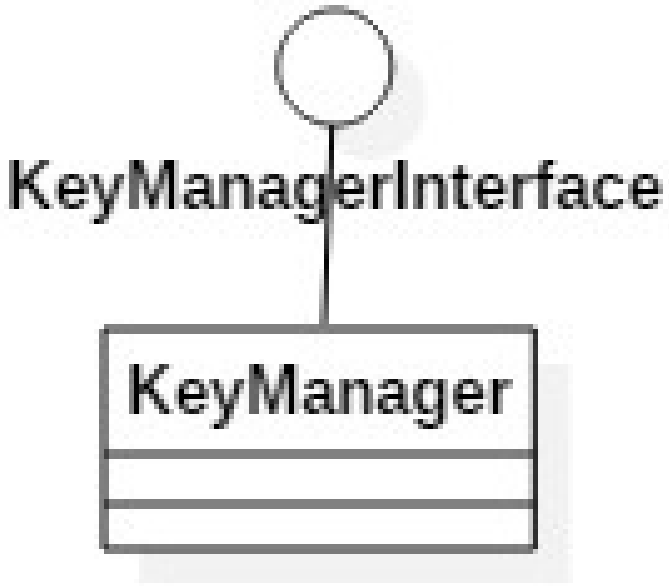

Figura 7: KeyManager

3.7.1 KeyManagerInterface

- **Descrizione:** Permette di ottenere delle Chiavi e di giudicare se e' utilizzabile oppure no.
- **Utilizzo:** Permette di ottenere una chiave.
- **Metodi:**
  - checkKey
    Controlla che una chiave non sia scaduta;
  - keyGen
    Genera una nuova chiave;
  - incrementSize
    Incrementa la chiave nel caso vi sia raggiunto un numero cospicuo di attivazioni.

3.7.2 KeyManager

- **Descrizione:** Implementazione di KeyManagerInterface.
- **Utilizzo:** Accede alla tabella Chiavi del database.
- **Attributi:**
  - Location : DAOKeyLocation;

  - Protocol : sodep;
  - Interfaces : DAOInterface, DAOKeyInterface;

- *Metodi:*
  - checkKey
    Controlla che una chiave non sia scaduta;
  - keyGen
    Genera una nuova chiave;
  - incrementSize
    (cosa fa questo metodo?)

3.8 Server::UserManager

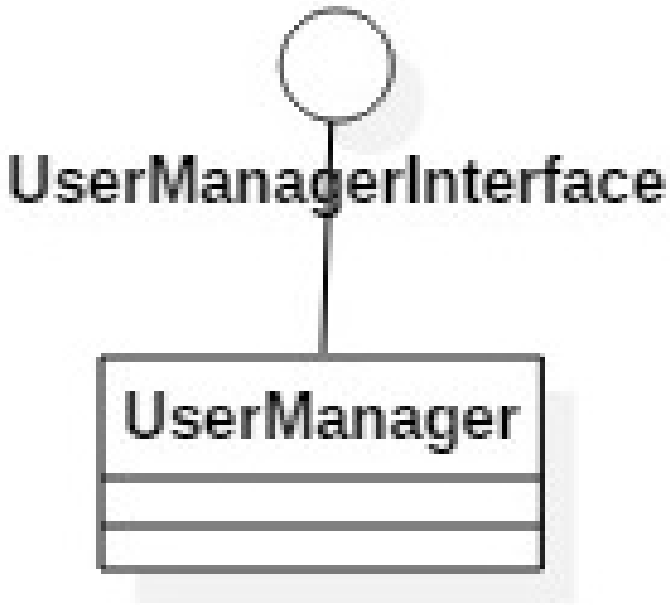

Figura 8: UserManager

3.8.1 UserManagerInterface

- **Descrzione:** Permette all'utente di effettuare il login, logout, agli admin di eliminare un utente, modificare e visualizzare il proprio profilo e di cercare un utente.
- **Utilizzo:** Accede alla tabella User e permette di visualizzare e modificarne i dati.
- **Metodi:**
  - login
    Permette di fare il login;

- editProfile
  Modifica il proflio;
- editProfileAdmin
  Modifica il profilo di un admin;
- deleteUser
  Elimina un utente;
- viewUser
  Visualizza il profilo utente;
- viewUserAdmin
  Visualizza il profilo di un admin;
- signIn
  Effettua la registrazione;
- searchUser
  Ricerca un utente;
- isAdmin
  Verifica se un utente e' admin;
- logout
  Effettua il logout;
- getUserFromSid
  Visualizza un utente dato il suo ID;
- addCredits
  Permette di aggiungere dei crediti;
- passwordRecover
  Permette il recupero della password;
- passwordRecoverConfirm
  Conferma il recupero della password;
- addCreditsToUser
  Aggiunge dei crediti ad un utente.

3.8.2 UserManager

- **Descrizione:** Permette la modifica e la visualizzazione dei profili utente, oltre ad il login ed il logout.
- **Utilizzo:** Accede alla tabella User.
- **Attributi:**
  - Location : DAOUserLocation;
  - Protocol : sodep;
  - Interfaces : DAOInterface, DAOUserInterface;
- *Metodi:*
  - login
    Permette di fare il login;

- editProfile
  Modifica il proflio;
- editProfileAdmin
  Modifica il profilo di un admin;
- deleteUser
  Elimina un utente;
- viewUser
  Visualizza il profilo utente;
- viewUserAdmin
  Visualizza il profilo di un admin;
- signIn
  Effettua la registrazione;
- searchUser
  Ricerca un utente;
- isAdmin
  Verifica se un utente e' admin;
- logout
  Effettua il logout;
- getUserFromSid
  Visualizza un utente dato il suo ID;
- addCredits
  Permette di aggiungere dei crediti.

4 Specifica componenti Front End

4.1 view

4.1.1 view::menu

- **menu.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo al menu di navigazione.
  - **Utilizzo:** Viene usato per mostrare il menu di navigazione all'utente e per contenere la barra di ricerca.
- **menu.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo al menu di navigazione
  - **Utilizzo:** Viene usato per definire la grafica del template HTML menu.html.

4.1.2 view::home

- **home.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla home page.
  - **Utilizzo:** Viene usato per visualizzare la home page dell'appicazione.
- **home.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla home page.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML home.html.

4.1.3 view::apiHome

- **apiHome.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla lista di API da visualizzare nella home page.
  - **Utilizzo:** Viene usato per mostrare all'interno di home.html la lista delle ultime API aggiunte al sistema.
- **apiHome.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla lista di API della home page.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML apiHome.html.

4.1.4 view::user-api

- **user-api.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla lista di API registrate dall'utente autenticato.
  - **Utilizzo:** Viene usato per visualizzare la lista delle API registrate dall'utente autenticato.
- **user-api.css**

  - **Descrizione:** La classe rappresenza un template CSS relativo alla lista di API registrate dall'utente autenticato.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML user-api.html.

4.1.5 view::user-page

- **user-page.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla visualizzazione della pagina utente.
  - **Utilizzo:** Viene usato per visualizzare la pagina utente dell'utente autenticato.
- **user-page.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla visualizzazione della pagina utente.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML user-page.html.

4.1.6 view::registration

- **registration.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla registrazione di un nuovo utente.
  - **Utilizzo:** Viene usato per visualizzare la pagina di registrazione di un nuovo utente.
- **registration.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla pagina di registrazione di un nuovo utente.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML registration.html.

4.1.7 view::user-api

- **user-comments.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla lista di commenti scritti da un utente.
  - **Utilizzo:** Viene usato per visualizzare la lista di commenti scritti da un utente.
- **user-comments.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla lista di commenti scritti da un utente.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML user-comments.html.

4.1.8 view::login-page

- **login-page.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla pagina di autenticazione.
  - **Utilizzo:** Viene usato per visualizzare il form per l'autenticazione dell'utente.
- **login-page.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla pagina di autenticazione.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML login-page.html.

4.1.9 view::category

- **category.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla lista di API appartenenti ad una certa categoria.
  - **Utilizzo:** Viene usato per visualizzare la lista delle API appartenenti ad una categoria scelta dall'utente.
- **category.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla lista di API appartenenti ad una certa categoria.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML category.html.

4.1.10 view::viewApi

- **viewApi.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla visualizzazione della pagina pubblica di una api.
  - **Utilizzo:** Viene usato per visualizzare la pagina di una api selezionata.
- **viewApi.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla visualizzazione della pagina pubblica di una api.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML viewApi.html.

4.1.11 view::user-api

- **search-page.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla pagina dei risultati di una ricerca.
  - **Utilizzo:** Viene usato per visualizzare la lista delle API che corrispondono ad una particolare ricerca fatta dall'utente sulla barra di ricerca.
- **search-page.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla pagina dei risultati di una ricerca.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML search-page.html.

4.1.12 view::user-api

- **search-bar.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla barra di ricerca.
  - **Utilizzo:** Viene usato per visualizzare la barra di ricerca che permette all'utente di ricercare le api nel sito.
- **search-bar.css**
  - **Descrizione:** La classe rappresenza un template CSS relativo alla barra di ricerca.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML search-bar.html.

4.1.13 view::administrator

- **administrator.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla pagina amministrativa.
  - **Utilizzo:** Viene usato per visualizzare la pagina amministrativa dell'utente amministrativo autenticato.
- **administrator.css**
  - **Descrizione:** La classe rappresenta un template CSS relativo alla pagina amministrativa.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML administrator.html.

4.1.14 view::buyApi

- **buyApi.html**
  - **Descrizione:** La classe rappresenta un template HTML relativo alla pagina di acquisto di una nuova Api.
  - **Utilizzo:** Viene usato per visualizzare la pagina di conferma acquisto di una nuova Api.
- **buyApi.css**
  - **Descrizione:** La classe rappresenta un template CSS relativo alla di acquisto di una nuova Api.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML buyApi.html.

4.1.15 view::passRecovery

- **passRecovery.html**
  - **Descrizione:** La classe rappresenta un template HTML relativo alla pagina di recupero della password di un account.
  - **Utilizzo:** Viene usato per visualizzare la pagina con le form relative al recupero password e conferma di processo.
- **passRecovery.css**
  - **Descrizione:** La classe rappresenta un template CSS relativo al recupero della password.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML passRecovery.html.

4.1.16 view::buyCredits

- **buyCredits.html**
  - **Descrizione:** La classe rappresenta un template HTML relativo alla pagina di acquisto di nuovi credits.
  - **Utilizzo:** Viene usato per visualizzare la pagina relativa all'acquisto di nuovi crediti.
- **buyCredits.css**
  - **Descrizione:** La classe rappresenta un template CSS relativo all'acquisto i nuovi crediti.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML buyCredits.html.

4.1.17 view::compare

- **compare.html**
  - **Descrizione:** La classe rappresenta un template HTML relativo alla pagina di comparazione tra Api di specifiche tecniche e sociali.
  - **Utilizzo:** Viene usato per visualizzare la pagina relativa al confronto tra Api.
- **compare.css**
  - **Descrizione:** La classe rappresenta un template CSS relativo al confronto tra due Api.
  - **Utilizzo:** Viene usato per definire la grafica del template HTML compare.html.

4.1.18 view::

- **base.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo all'impostazione di base della pagina, dentro cui si muoveranno le altre pagine html.
  - **Utilizzo:** Viene usato per visualizzare il container di base di tutte le altre pagine html.
- **index.html**
  - **Descrizione:** La classe rappresenza un template HTML relativo alla scheletro della pagina che contiene gli headers.
  - **Utilizzo:** Viene usato per inserire gli headers all'interno del sito e viene usato come punto d'ingresso sull'applicazione.

4.2 viewModel

4.2.1 viewModel::AppModule

- **Descrizione:** modulo che contiene l'applicazione
- **Utilizzo:** si occupa di risolvere le dependency injections, caricare i componenti, i servizi, i routing e di fornire il punto d'ingresso per l'utente.

4.2.2 viewModel::Component::BaseComponent

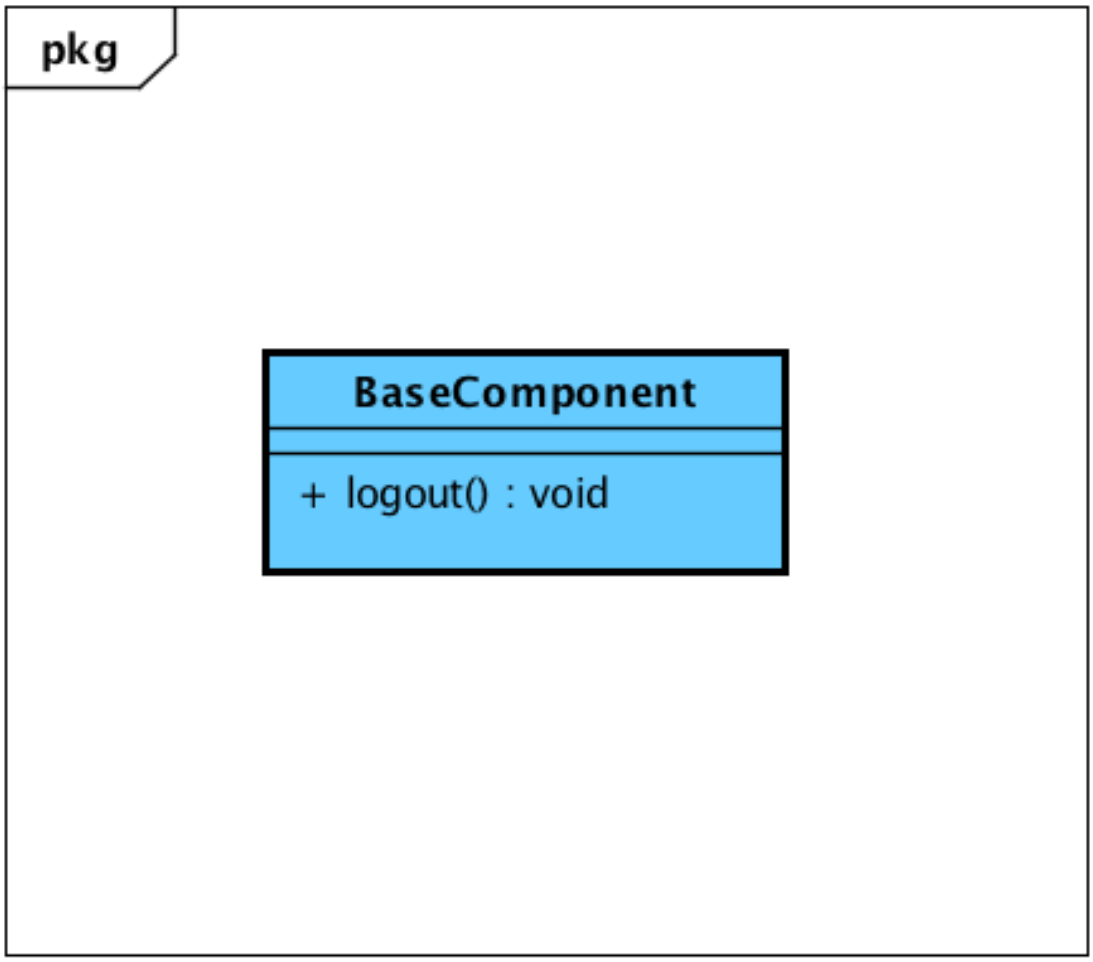

Figura 9: BaseComponent

- **Descrizione:** componente che gestisce il container di base dell'applicazione.
- **Utilizzo:** si occupa di unire il menu di navigazione con i vari contenuti dell'applicazione.
- **Metodi:**
  - + void logout()
    richiama il SessionService ed effettua logout in seguito alla richiesta dell'utente, se l'utente è autenticato.

4.2.3 viewModel::Component::MenuComponent

- **Descrizione:** componente che gestisce il menu di navigazione che contiene anche la barra di ricerca.
- **Utilizzo:** viene usato per navigare all'interno dell'applicazione.

4.2.4 viewModel::Component::UserCommentsComponent

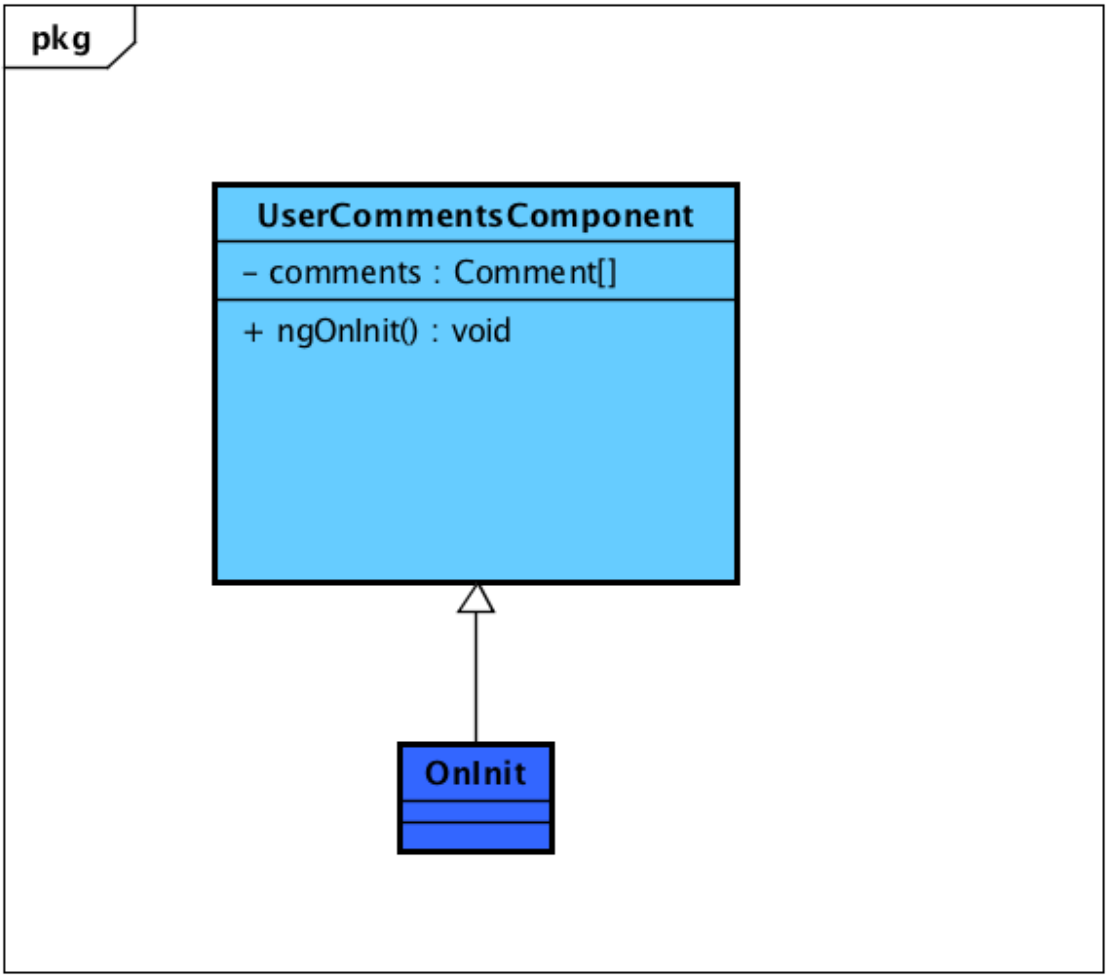

Figura 10: UserCommentsComponent

- **Descrizione:** componente che gestisce la visualizzazione dei commenti scritti dall'utente autenticato.
- **Utilizzo:** viene utilizzato per gestire le sezioni in cui viene richiesto di visualizzare i commenti dell'utente autenticato.
- **Classi ereditate:**
  - OnInit
    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.;
- **Attributi:**
  - + Comment[] comments
    contiene l'elenco dei commenti dell'utente
- **Metodi:**
  - + void ngOnInit()
    definisce cosa eseguire alla creazione del componente, ovvero connettersi al service UserCommentsService e richiedere i commenti dell'utente per salvarli in comments

4.2.5 viewModel::Component::RegistrationComponent

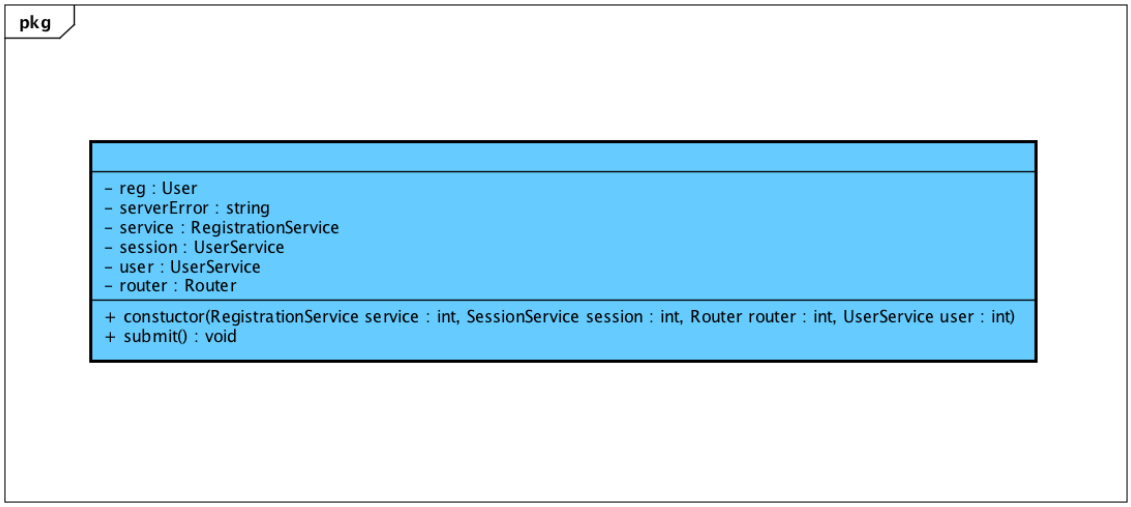

Figura 11: RegistrationComponent

- **Descrizione:** componente che gestisce la registrazione di un nuovo utente.
- **Utilizzo:** viene utilizzato per gestire la pagina di registrazione, raccogliere i dati e mostrare eventuali errori all'utente in fase di registrazione.
- **Attributi:**
  - + User reg
    contiene l'insieme di informazioni inserite dall'utente sull'apposito form;
  - + string serverError
    contiene gli errori riscontrati dopo la connessione al server.
  - - RegistrationService service
    effettua Dependency Injection del service RegistrationService
  - - SessionService session
    dependency injection del service SessionService
  - - UserService user
    dependency injection del service UserService
  - - Router router
    dependency injection del service Router
- **Metodi:**
  - + constuctor(RegistrationService service, SessionService session, Router router, UserService user)
    inizializza reg come utente vuoto istanzia le varie dependency injections.
  - + void submit()
    Invia i dati recuperati dalla form al service RegistrationService in seguito alla conferma dell'utente. Se la registrazione va a buon fine effettua un redirect sulla pagina utente altrimenti visualizza un errore tramite serverError.

4.2.6 viewModel::Component::LoginPageComponent

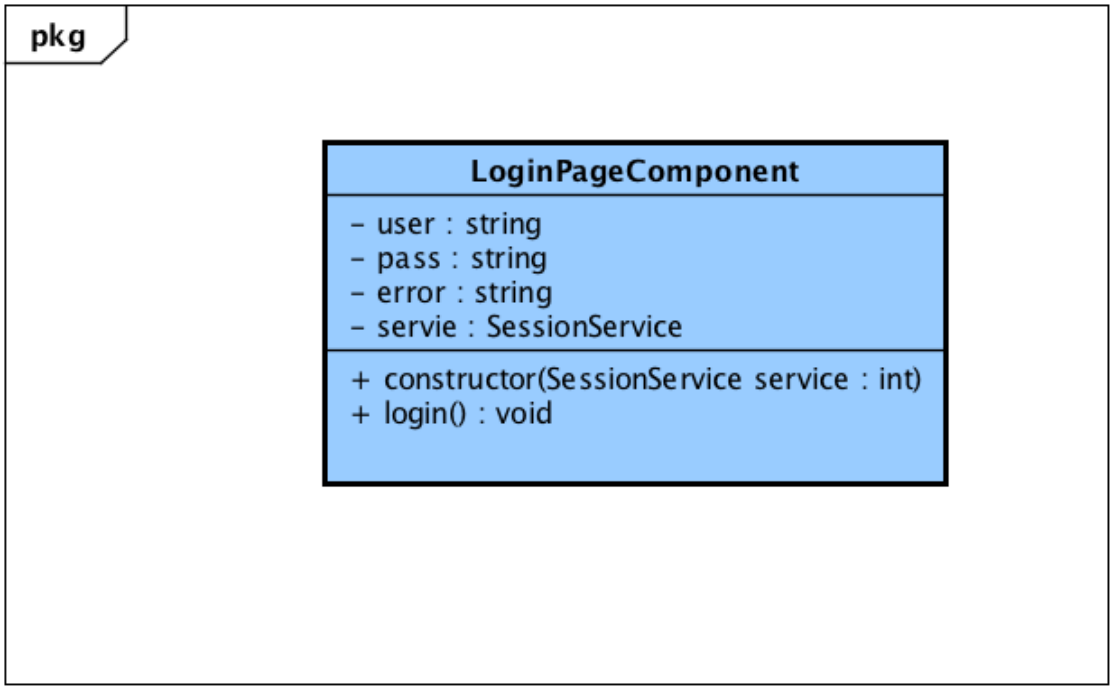

Figura 12: RegistrationComponent

- **Descrizione:** componente che si occupa della gestione della pagina di login.
- **Utilizzo:** viene utilizzato per gestire la pagina di login, raccogliere i dati e mostrare eventuali errori all'utente in fase di autenticazione.
- **Attributi:**
  - + string user
    contiene lo username inserito dall'utente
  - + string pass
    contiene la password inserita dall'utente
  - + string error
    contiene gli errori da visualizzare che provengono dalla richiesta di autenticazione
  - - SessionService service dependency injection del service SessionService
- **Metodi:**
  - + constructor(SessionService service) inizializza service per la dependency injection.
  - + void login()
    in seguito alla conferma dell'utente invia i dati raccolti dalla form ed effettua una richista a SessionService di login.

4.2.7 viewModel::Component::UserPageComponent

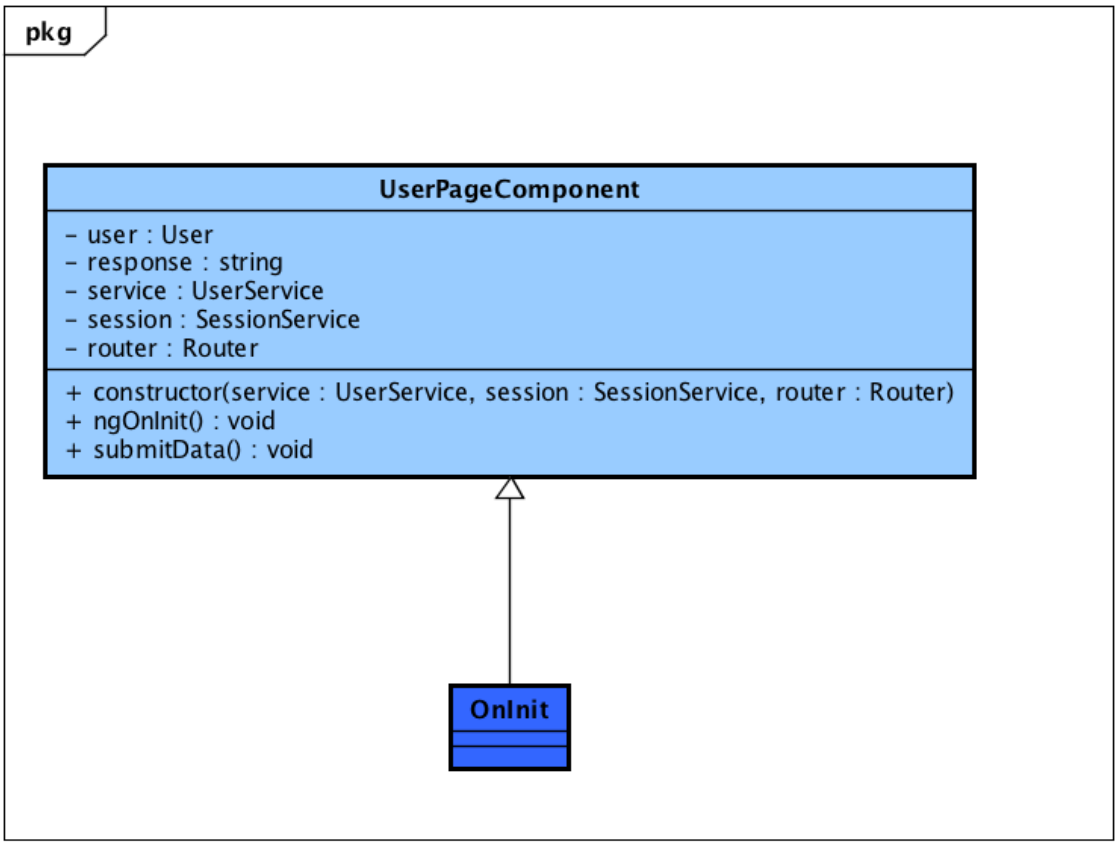

Figura 13: UserPageComponent

- **Descrizione:** componente che si occupa della gestione della pagina utente dell'utente autenticato.
- **Utilizzo:** viene utilizzata per mostrare all'utente le sue informazioni personali e per permetterne la modifica.
- **Classi ereditate:**
  - OnInit

    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.
- **Attributi:**
  - + User user
    contiene i dati utente dell'utente autenticato
  - - string response
    contiene la risposta del servere alla richiesta di modifica dei dati
  - - UserService service dependency injection del service UserService
  - - SessionService session dependency injection del service SessionService
  - - router Router dependency injection del service Router
- **Metodi:**

- constructor(UserService service, SessionService session, Router router)
  inizializza le proprietà per la dependency injection.
- + void ngOnInit()
  alla creazione del componente esegue la richiesta al service SessionService per il recupero delle informazioni utente e le assegna a user.
- + void submitData()
  richiama la funzione di modifica dei dati di UserService ed esegue una richiesta di modifica con i dati inseriti dall'utente.

4.2.8 viewModel::Component::ApiHomeComponent

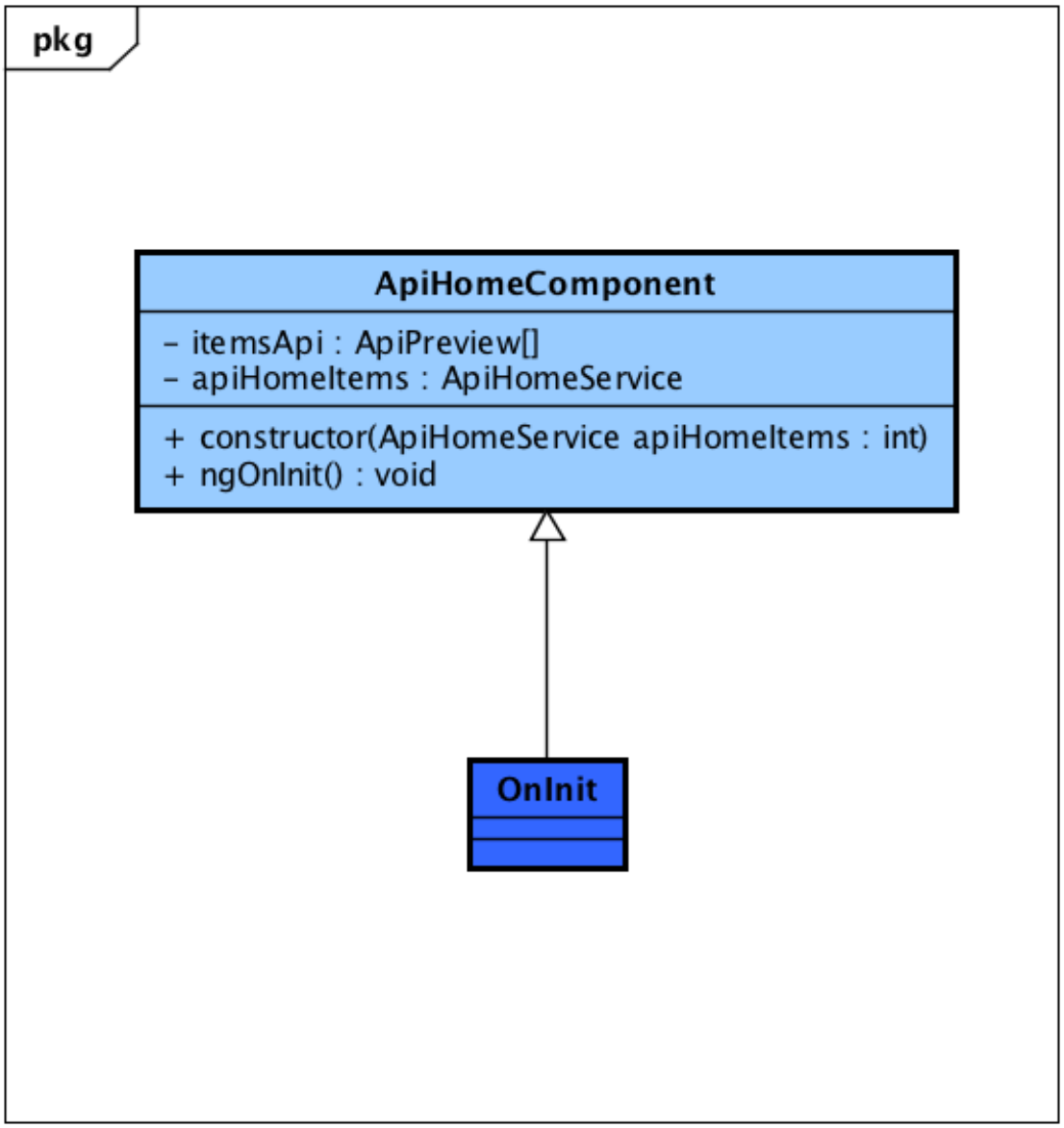

Figura 14: ApiHomeComponent

- **Descrizione:** componente che gestisce la pagina di home.
- **Utilizzo:** viene usato per recuperare e visualizzare le ultime API inserite nel sistema.
- **Classi ereditate:**

  - OnInit

    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.

- **Attributi:**
  - + ApiPreview[] itemsApi
    contiene l'insieme di Api da visualizzare in anteprima sulla home
  - - ApiHomeService apiHomeItems
    dependency injection del service ApiHomeService
- **Metodi:**
  - + constructor(ApiHomeService apiHomeItems)
    effettua la dependency injection di ApiHomeService.
  - + void ngOnInit()
    durante l'inizializzazione del component effettua la richiesta al service ApiHome Service delle api da visualizzare e le memorizza in itemsApi.

4.2.9 viewModel::Component::SearchBarComponent

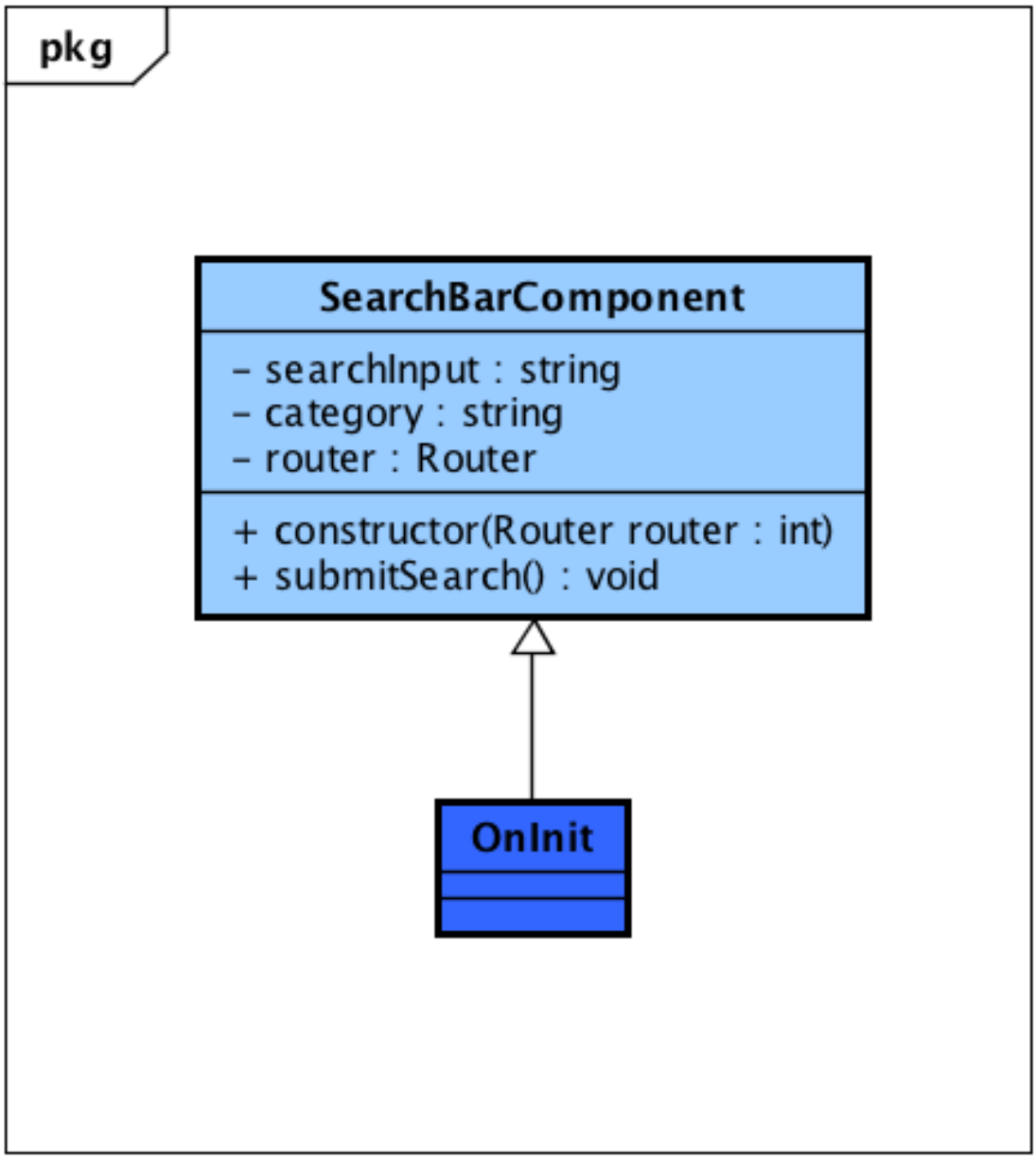

Figura 15: SearchBarComponent

- **Descrizione:** componente che gestisce la barra di ricerca nel menu di navigazione.
- **Utilizzo:** viene utilizzato per effettuare le ricerche nel sistema.
- **Attributi:**
  - + string searchInput
    contiene la stringa inserita dall'utente.
  - + string category
    contiene la categoria selezionata dall'utente.
  - - Router router
    dependency injection del service Router.

- **Metodi:**
  - + constructor(Router router)
    effettua le dependency injection.
  - + void submitSearch()
    redirect alla pagina di ricerca passando sul routing le variabili inserite dall'utente.

4.2.10 viewModel::Component::SearchPageComponent

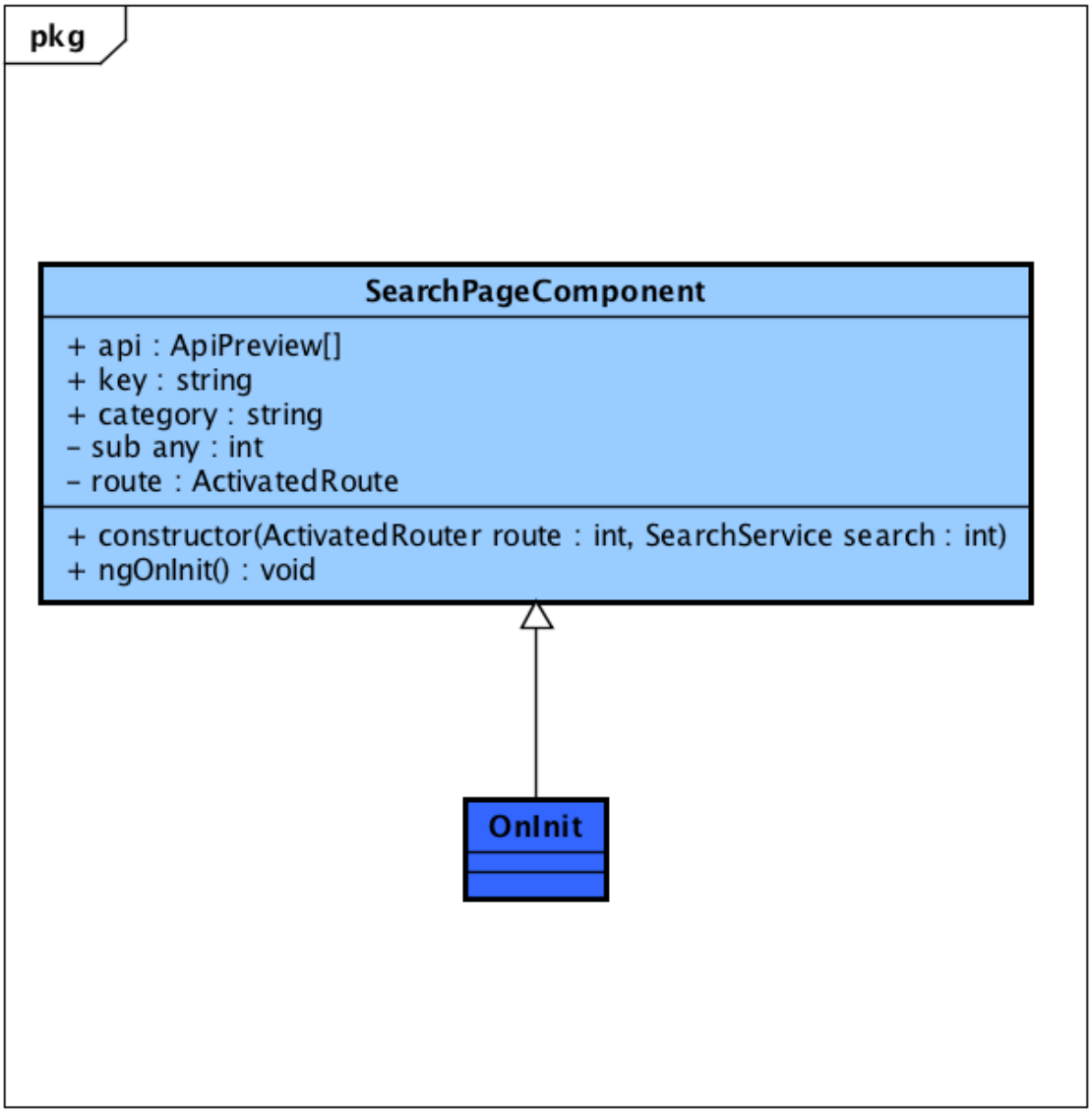

Figura 16: SearchPageComponent

- **Descrizione:** componente che gestisce la pagina di ricerca.
- **Utilizzo:** viene utilizzato per visualizzare le API ricercate dall'utente per mezzo della barra di ricerca.
- **Classi ereditate:**
  - OnInit

Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.

- **Attributi:**
  - + ApiPreview[] api
    contiene l'elenco di api risultate dalla ricerca.
  - + string key
    contiene i termini di ricerca inseriti dall'utente.
  - + string category
    contiene la categoria selezionata dall'utente in fase di ricerca.
  - - any sub
    variabile utilizzata per effettuare redirect.
  - - ActivatedRoute route
    dependency injection del service ActivatedRoute.
  - - SearchService search
    dependency injection del service SearchService.
- **Metodi:**
  - + constructor (ActivatedRouter route, SearchService search)
    imposta le varie dependency injections.
  - + void ngOnInit()
    esegue la ricerca secondo i termini key e category tramite il service SearchService e salva i risultati sulla proprietà api.

4.2.11 viewModel::Component::CategoryComponent

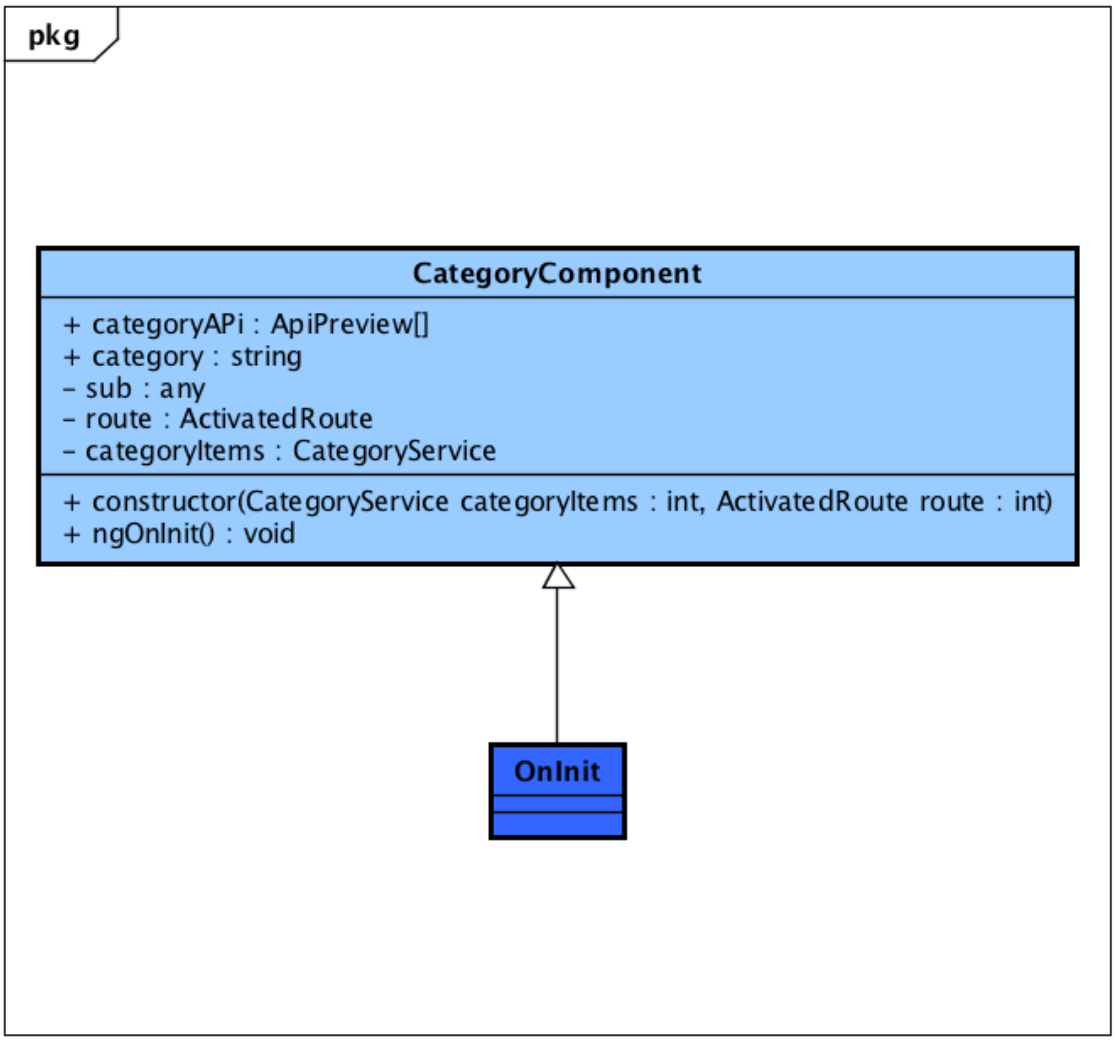

Figura 17: CategoryComponent

- **Descrizione:** componente che si occupa di visualizzare le API secondo categorie.
- **Utilizzo:** viene utilizzato per visualizzare una preview delle ultime API inserite di una categoria specifica.
- **Classi ereditate:**
  - OnInit

    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.
- **Attributi:**
  - + ApiPreview[] categoryApi
    contiene l'elenco di API che appartengono alla categoria
  - + string category
    contiene la categoria selezionata

  - - any sub
    propietà usata per il redirect
  - - CategoryService categoryItems dependency injection del service CategoryService
  - - ActivatedRoute route
    dependency injection del service route

- **Metodi:**
  - + constructor(CategoryService categoryItems, ActivatedRoute route)
    imposta le dependency injections.
  - + void ngOnInit()
    alla creazione del component recupera la categoria dal routing ed esegue la richiesta delle api tramite il service CategoryService. Salva il risultato nella propietà categoryApi.

4.2.12 viewModel::Component::UploadApiComponent

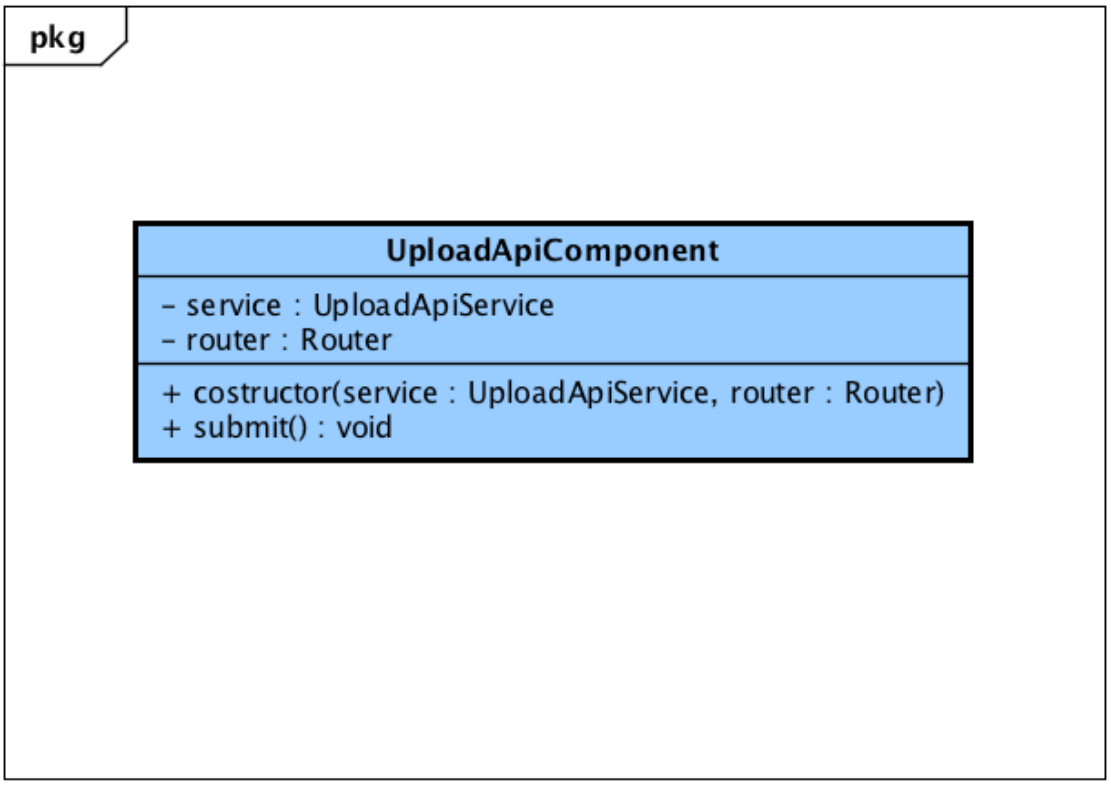

Figura 18: UploadApiComponent

- **Descrizione:** component che si occupa dell'inserimento di una nuova API da parte di un utente autenticato.
- **Utilizzo:** recupera i dati dalla form ed esegue l'inserimento.
- **Classi ereditate:**
  - UploadApiService servizi che danno funzionalità di caricamento di una nuova API
  - UserApi modello di una API utente
  - SessionService servizi di sessione

- Router servizi di routing
- Policy modello dove ricavare le policy

- **Attributi:**
  - - UploadApiService service
    dependency injection del serviziouploadapiservice
  - - Router router
    dependency injection del service Router
  - + UserApi api
    Microservizio da caricare
  - + boolean success
    Valore di conferma di caricamento dell'API
  - + Policy[] policy
    Collezione di policy di una data applicazioni.

- **Metodi:**
  - + costructor(UploadApiService service, Router router, SessionService session)
  - + void submit() importa un microservizio nella base di dati
  - + void addPolicy() aggiunge una policy al microservizio
  - + void fileChange(event) cambia il file a seconda di un evento di errore.

4.2.13 viewModel::Component::administrator

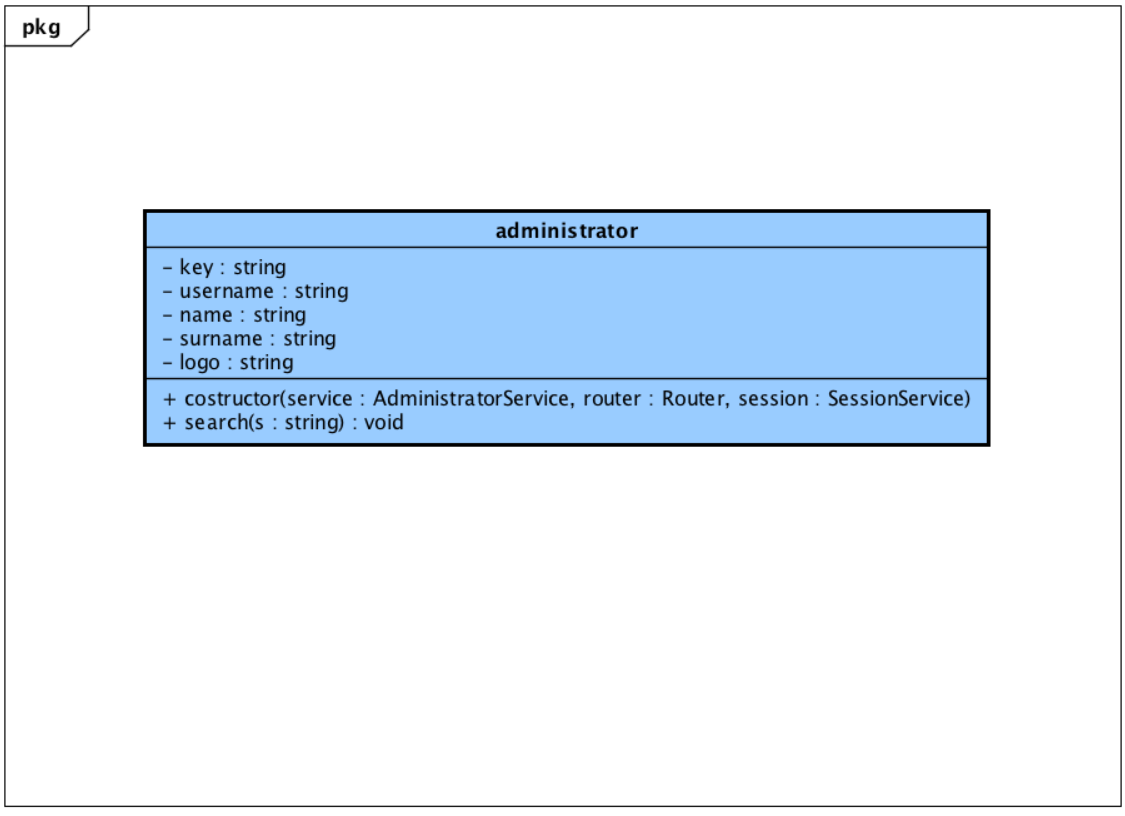

Figura 19: AdministratorComponent

- **Descrizione:** component che si occupa della gestione delle funzioni amministrative.
- **Utilizzo:** recupera e crea sessioni amministrative.
- **Classi ereditate:**
  - AdministratorService servizi che danno funzionalità di gestire la parte amministrativa
  - OnInit

    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.
  - SessionService servizi di sessione
  - Router servizi di routing
  - User modello dove ricavare l'utente
- **Attributi:**
  - - string key
    chiave univoca dell'amministratore
  - - string username
    username dell'amministratore
  - - string name
    nome proprio dell'amministratore
  - - string surname
    cognome proprio dell'amministratore
  - - string logo
    immagini profilo dell'amministratore
- **Metodi:**
  - + constructor(AdministratorService service, Router router, SessionService session)
  - + void search(string s) ricerca un utente

4.2.14 viewModel::Component::buy-api

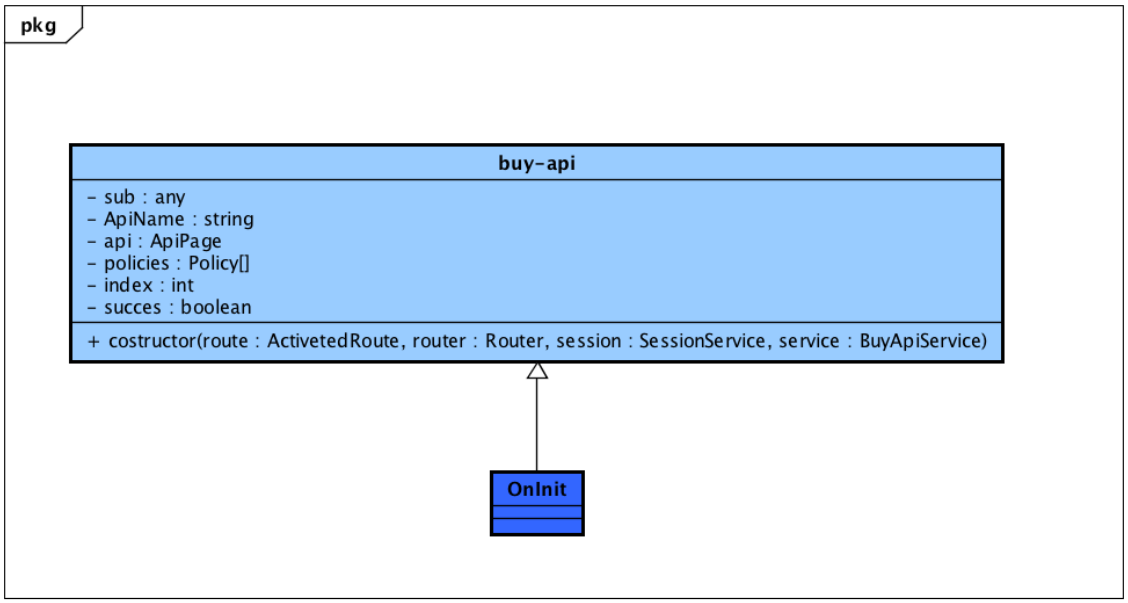

Figura 20: BuyApiComponent

- **Descrizione:** component che si dell'acquisto di una nuova API.
- **Utilizzo:** è acquistare un microservizio.
- **Classi ereditate:**
  - OnInit
    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.
  - BuyApiService servizi che danno funzionalità di acquisto di una nuova API
  - ApiPage modello di un anteprima di API
  - SessionService servizi di sessione
  - Router servizi di routing
  - Policy modello dove ricavare le policy
- **Attributi:**
  - - any sub
  - - string ApiName
    nome dell'API
  - - ApiPage api
    indirizzo alla pagina dell'API
  - - Policy[] policies
    insieme di policies dell'API
  - - number index
    indicizzazione dell'API

  - - boolean success
    conferma avvenuto acquisto.

- **Metodi:**
  - constructor(ActivatedRoute route, Router router, SessionService session, BuyApiService service)
  - + void buyApi() funzione di acquisto della API selezionata.

4.2.15 viewModel::Component::compare

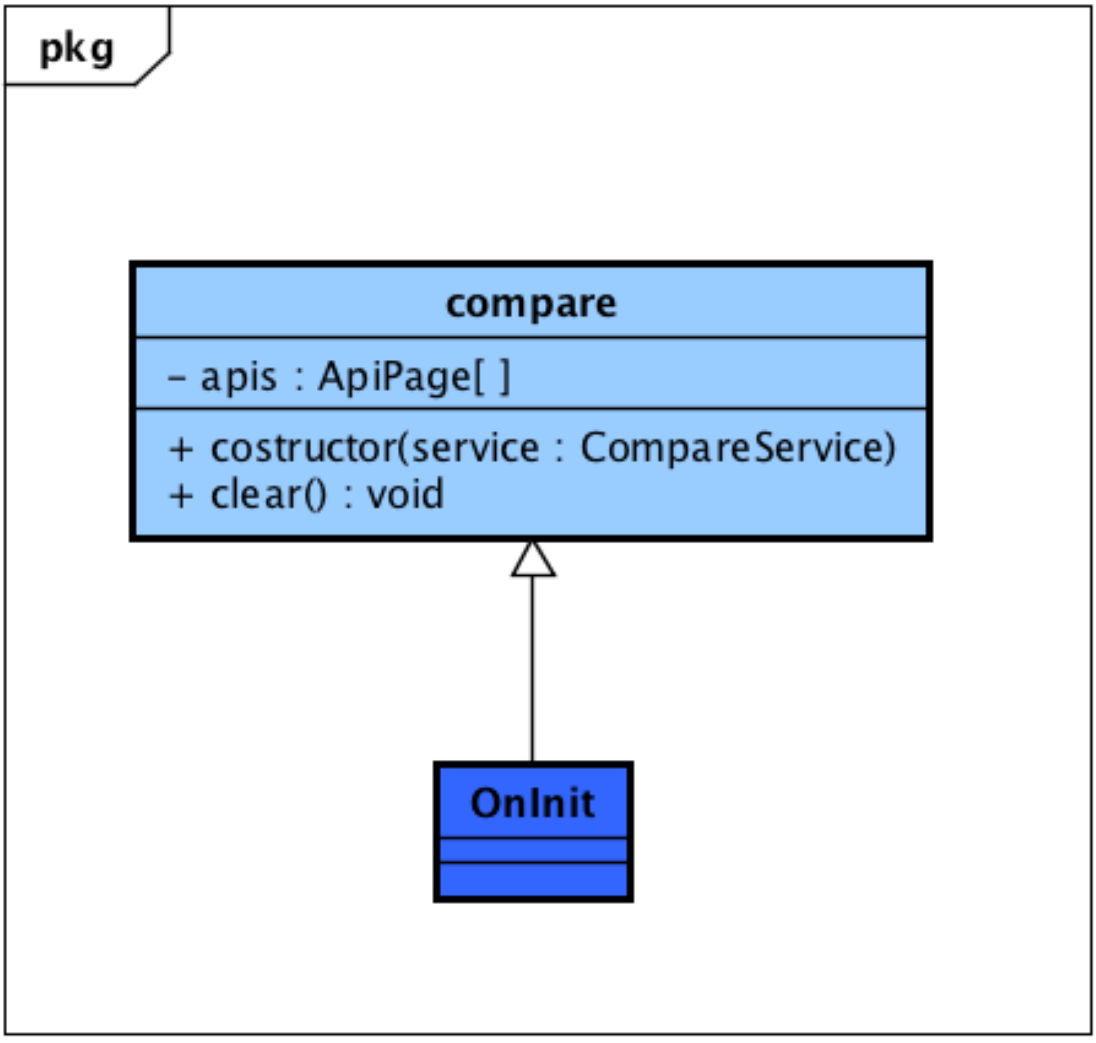

Figura 21: CompareComponent

- **Descrizione:** component che si dedica al confronto tra applicazioni.
- **Utilizzo:** è possibile confrontare più microservizi.
- **Classi ereditate:**
  - OnInit
    Rappresenta la fase di inizializzazione del componente e si verifica dopo il primo evento OnChanges. Questa fase viene eseguita una sola volta durante il ciclo di vita del componente.

- CompareService servizi che danno funzionalità di confrontare due o più API
- ApiPage modello di un anteprima di API

- **Attributi:**
  - - ApiPage[] apis
    collezione di API da confrontare
- **Metodi:**
  - constructor(- CompareService service)
  - + void clear() rimuove tutte le API dalla collezione permettendo così di fare un nuovo confronto.

4.2.16 viewModel::Service::UserCommentsService

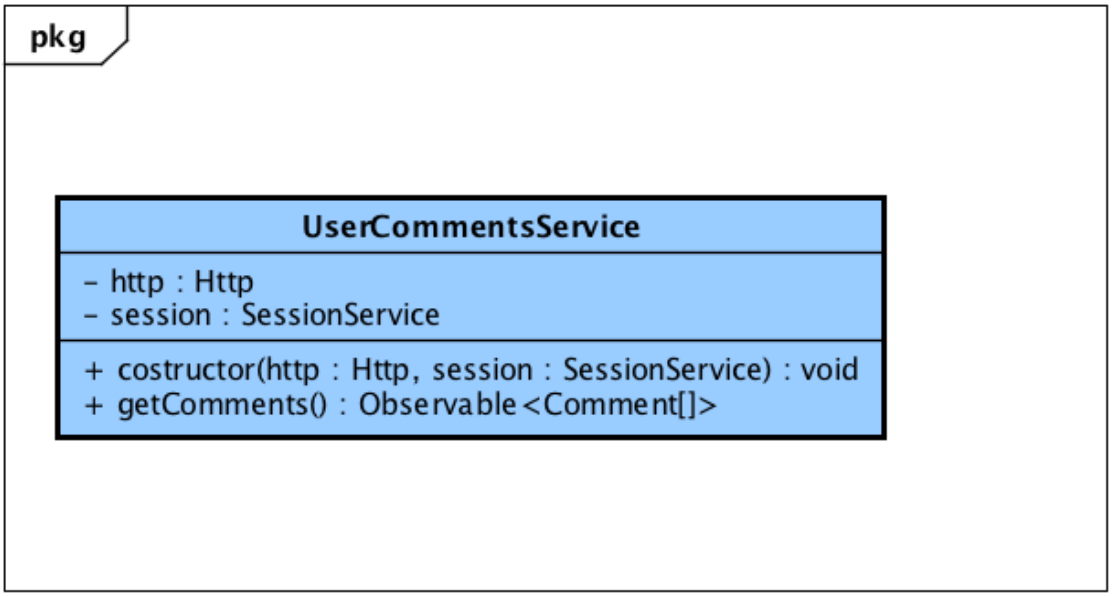

Figura 22: UserCommentsService

- **Descrizione:** Service che si occupa del recupero dal server dei commenti di un utente autenticato.
- **Utilizzo:** viene usato nella parte utente per recuperare i commenti fatti da quell'utente.
- **Attributi:**
  - - Http:http
    dependency injection del service Http
  - - SessionService session
    dependency injection del service session.
- **Metodi:**
  - + constructor(Http http, SessionService session)
    costruttore che esegue le dependency injections.
  - + Observable<Comment[]> getComments()
    esegue la chiamata al server e restituisce il risultato sotto forma di Observable.

4.2.17 viewModel::Service::ApiPageService

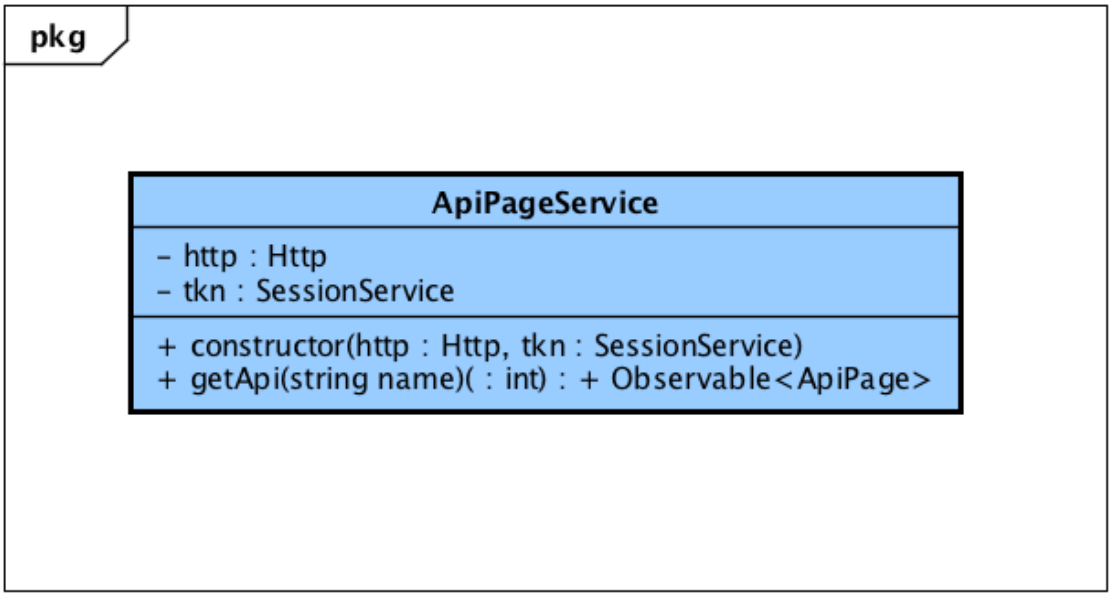

Figura 23: ApiPageService

- **Descrizione:** service che recupera una API dal server per essere visualizzata nella sua pagina dedicata.
- **Utilizzo:** viene utilizzato per recuperare API dal server.
- **Attributi:**
  - - Http http
    dependency injection del service Http
  - - SessionService tkn
    dependency injection del service SessionService
- **Metodi:**
  - + constructor(Http http, SessionService tkn)
    costruttore che esegue le dependency injections.
  - + Observable<ApiPage> getApi(string name)
    esegue la richiesta al server per l'API di nome name e ritorna un Observable con il risultato.

4.2.18 viewModel::Service::RegistrationService

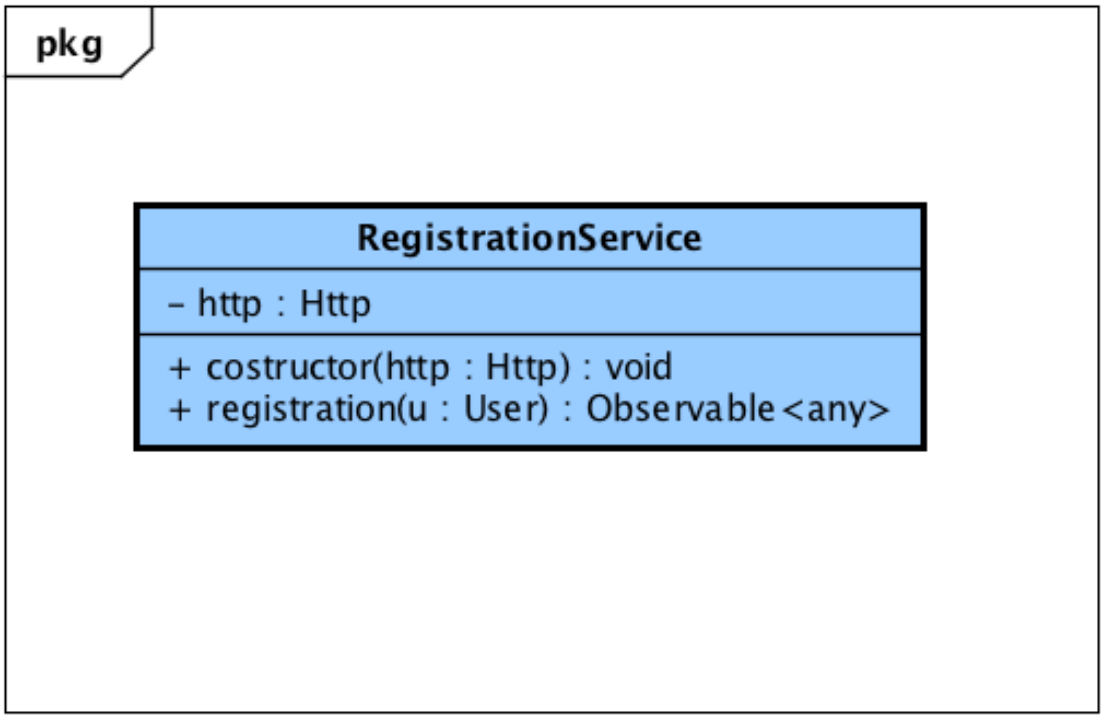

Figura 24: RegistrationService

- **Descrizione:** service che si occupa della richiesta di registrazione al server.
- **Utilizzo:** viene usato dal component RegistrationComponent per effettuare la registrazione.
- **Attributi:**
  - - Http http
    dependency injection del service Http
- **Metodi:**
  - + constructor (Http:http)
    costruttore che imposta le dependency injections.
  - + Observable<any> registration(User u)
    invia al server la richiesta di registrazione dell'utente u e ritorna la risposta tramite Observable<any>.

4.2.19 viewModel::Service::SessionService

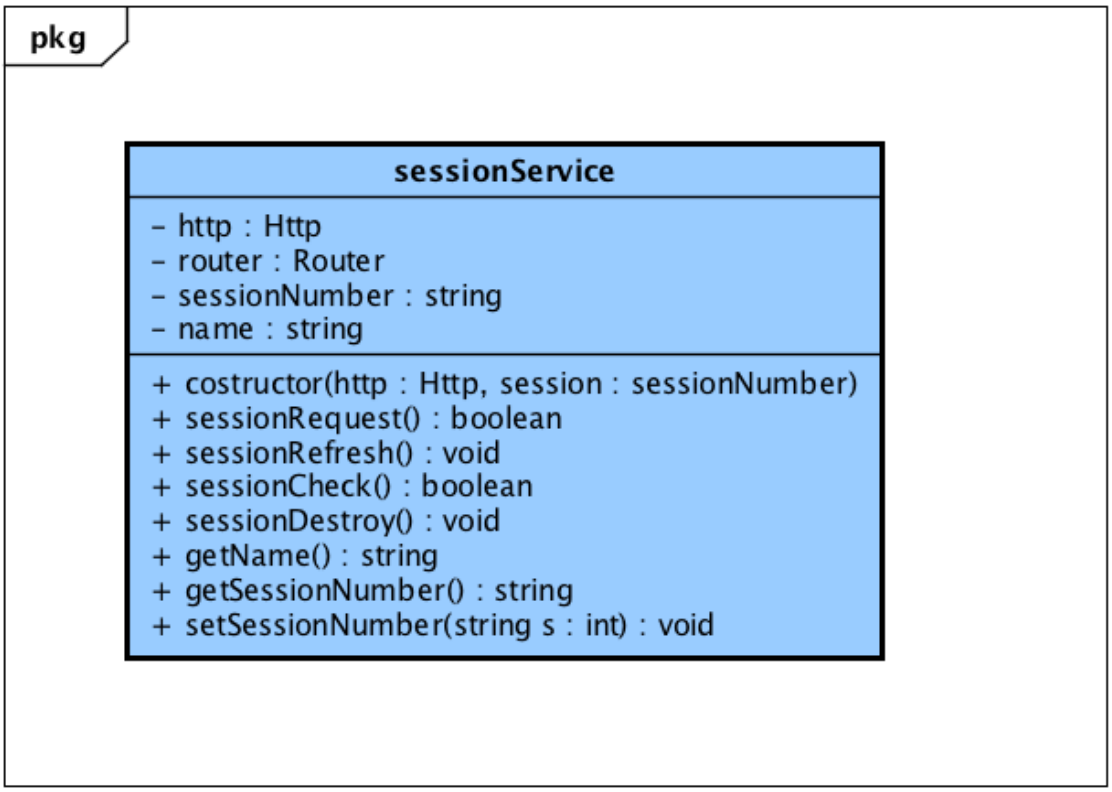

Figura 25: SessionService

- **Descrizione:** service che si occupa di richiedere, creare e distruggere sessioni.
- **Utilizzo:** viene utilizzato per eseguire login, logout, e per fare il check sullo stato della sessione.
- **Attributi:**
  - - Http http
    dependency injection del service Http
  - - Router router
    dependency injection del service Router
  - - string sessionNumber
    numero di sessione
  - - string name
    username dell'utente associato alla sessione corrente.
- **Metodi:**
  - + constructor(Http http, Router router)
    costruttore che imposta le dependency injection, e pone sessionNumber e name = "".
  - + bool sessionRequest(string username, string password)
    chiamata al server per effettuare il login, ritorna true se il login ha avuto successo, false altrimenti.
  - + void sessionRefresh()
    refresha la session resettando il tempo.

- + bool sessionCheck()
  controlla se esiste la sessione e se è ancora valida con una chiamata al server. Ritorna true se è valida, false altrimenti.
- + void sessionDestroy()
  esegue logout e distrugge la sessione.
- + string getName()
  metodo get per l'attributo name.
- + string getSessionNumber()
  metodo get per l'attributo sessionNumber.
- + void setSessionNumber(string s)
  modifica sessionNumber con s.

4.2.20 viewModel::Service::UserService

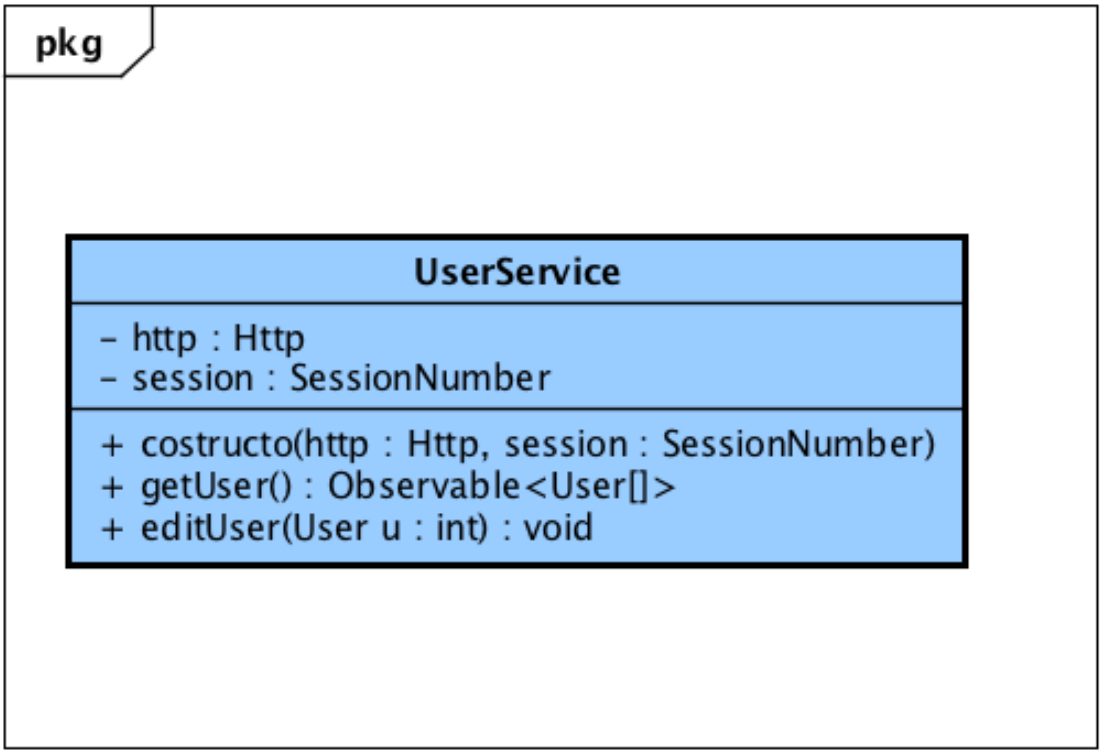

Figura 26: UserService

- **Descrizione:** Gestisce le richieste dell'utente e recupera i dati dell'utente loggato dal server.
- **Utilizzo:** viene utilizzato per recuperare i dati della pagina utente dal server.
- **Attributi:**
  - - Http http
    dependency injection del service Http
  - - SessionNumber session
    dependency injection del service SessionNumber
- **Metodi:**

- + constructor(Http http, SessionService session)
  costruttore che imposta le dependency injection.
- + Observable<User>getUser()
  funzione che ritorna esegue una richiesta al server per ottenere i dati dell'utente autenticato e li ritorna sotto forma di observable.
- + void editUser(User u)
  funzione che invia esegue una richiesta di modifica dei dati dell'utente con u.

4.2.21 viewModel::Service::UserApiService

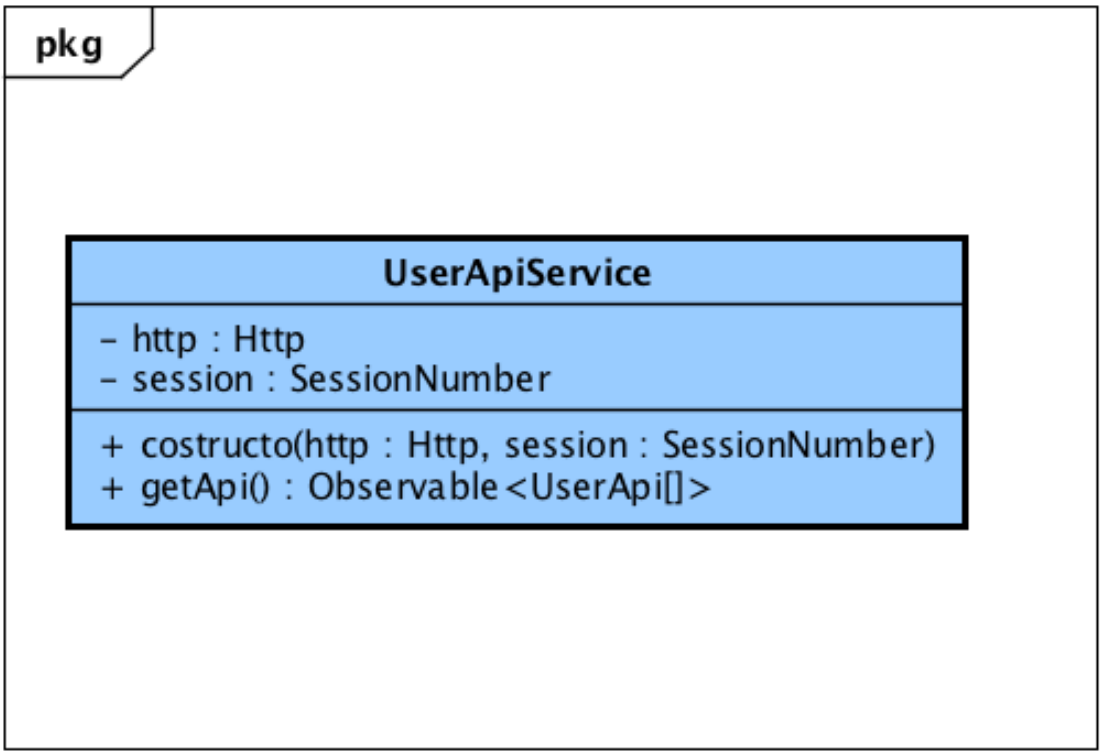

Figura 27: UserApiService

- **Descrizione:** service che si occupa di recuperare le api caricate dall'utente autenticato.
- **Utilizzo:** viene utilizzato per recuperare le api dell'utente nella parte dedicata alla pagina utente.
- **Attributi:**
  - - Http http
    dependency injection del service Http
  - - SessionNumber session
    dependency injection del service SessionNumber
- **Metodi:**
  - + constructor(Http http, SessionService session)
    costruttore che imposta le dependency injection.
  - + Observable<UserApi[]> getApi()
    funzione che e segue la richiesta al server per le api registrate dall'utente autenticato e ritorna un Observable.

4.2.22 viewModel::Service::ApiHomeService

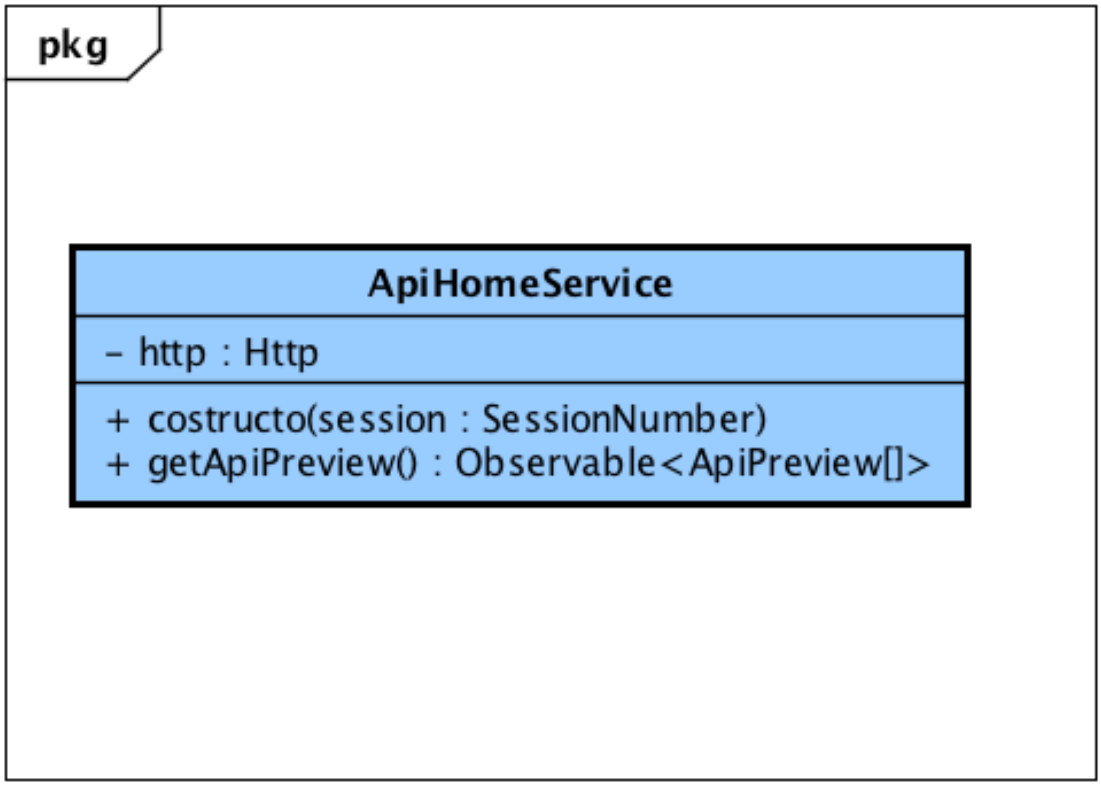

Figura 28: ApiHomeService

- **Descrizione:** service che si occupa di recuperare la lista di API da visualizzare nella home page.
- **Utilizzo:** viene utilizzato nella home page per recuperare le API dal server.
- **Attributi:**
  - \- Http http
    dependency injection del service Http
- **Metodi:**
  - \+ constructor(Http http)
    costruttore che imposta la dependency injection di Http.
  - \+ Observable<ApiPreview[]> getApiPreview()
    funzione che richiede al server l'elenco di API da visualizzare sulla home e le restituisce sotto forma di Observable.

4.2.23 viewModel::Service::SearchService

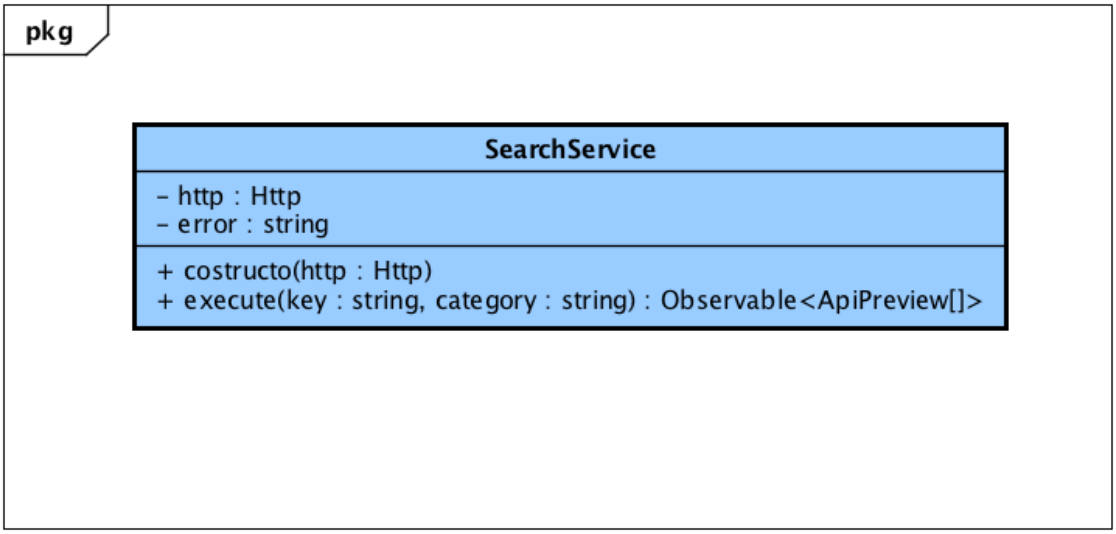

Figura 29: SearchService

- **Descrizione:** service che esegue la ricerca impostata dall'utente.
- **Utilizzo:** viene usato per recuperare dal server la lista s API che corrispondono alla ricerca richiesta dall'utente.
- **Attributi:**
  - + string error
    contiene gli errori del server in seguito alla ricerca.
  - - Http http
    dependency injection del service Http
- **Metodi:**
  - + constructor(Http http)
    costruttore che imposta la dependency injection di Http
  - + Observable<ApiPreview[]> execute(string key, string category)
    funzione che esegue la richiesta al server dell'elenco di APIs che corrispondono alla ricerca effettuata con key e category e ritorna le API restituite sotto forma di Observable.

4.2.24 viewModel::Service::CategoryService

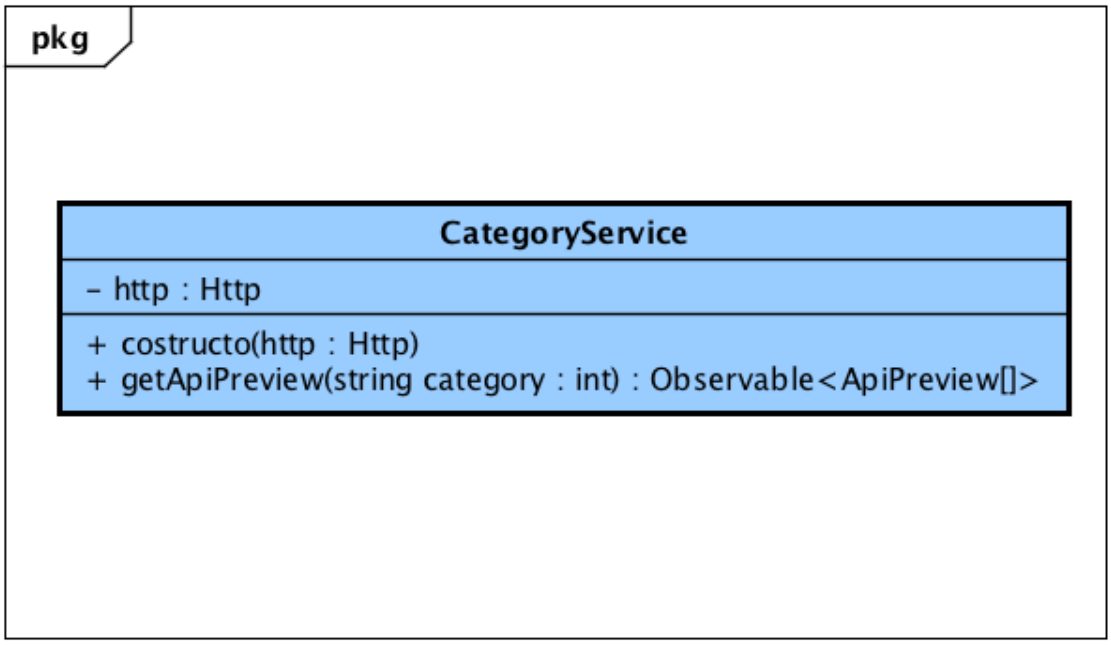

Figura 30: CategoryService

- **Descrizione:** service che recupera le API appartenenti alla categoria scelta dall'utente
- **Utilizzo:** permette di richiedere al server l'elenco di API appartenenti ad una certa categoria
- **Attributi:**
  - - Http http
    dependency injection del service Http
- **Metodi:**
  - + constructor(Http http)
    costruttore che imposta la dependency injection di Http
  - + Observable<ApiPreview[]> getApiPreview(string category)
    funzione che richiede al server l'elenco delle API appartenenti alla categoria category e le ritorna sotto forma di Observable.

4.2.25 viewModel::Service::UploadApiService

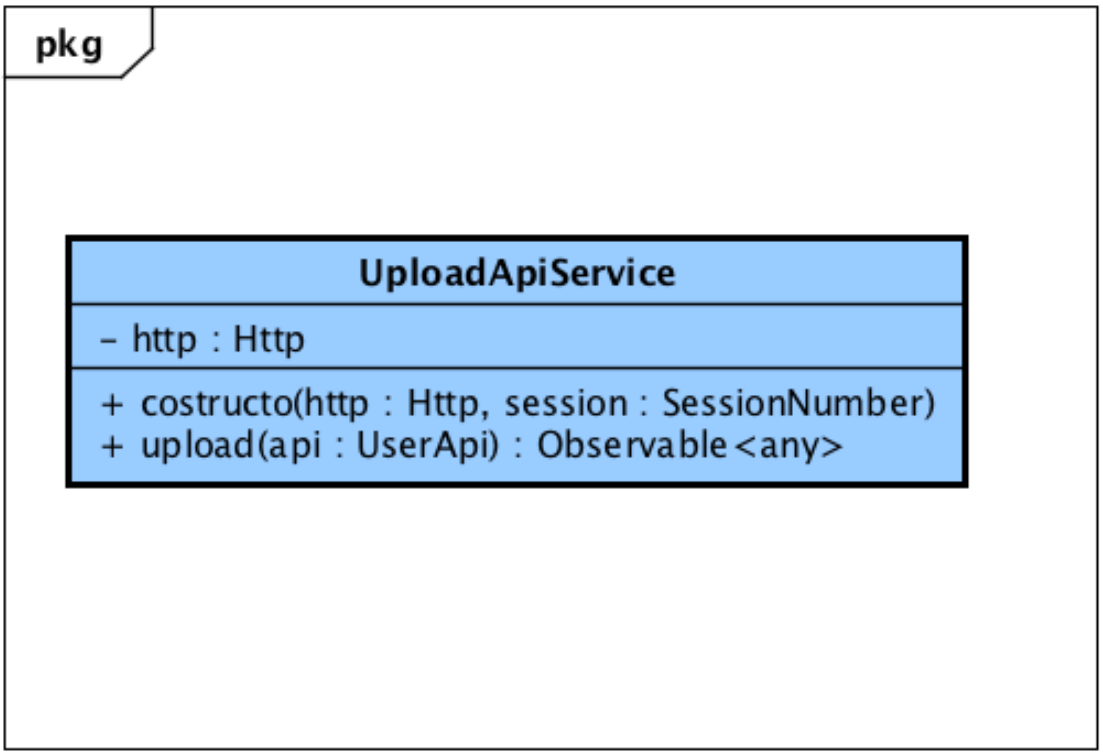

Figura 31: UploadApiService

- **Descrizione:** funzionalità per la registrazione di una nuova API.
- **Utilizzo:** permette all'utente autenticato di registrare una nuova API.
- **Classi ereditate:**
  - - Http http
    dependency injection del service Http
- **Attributi:**
  - - Http http
    dependency injection del service Http
- **Metodi:**
  - + constructor(Http http)
    costruttore che imposta la dependency injection di Http
  - + Observable<any> upload(UserApi api)
    funzione che esegue la richiesta al server di inserimento di una API e restituisce il risultato dell'operazione come observable.

4.2.26 viewModel::Service::AdministratorService

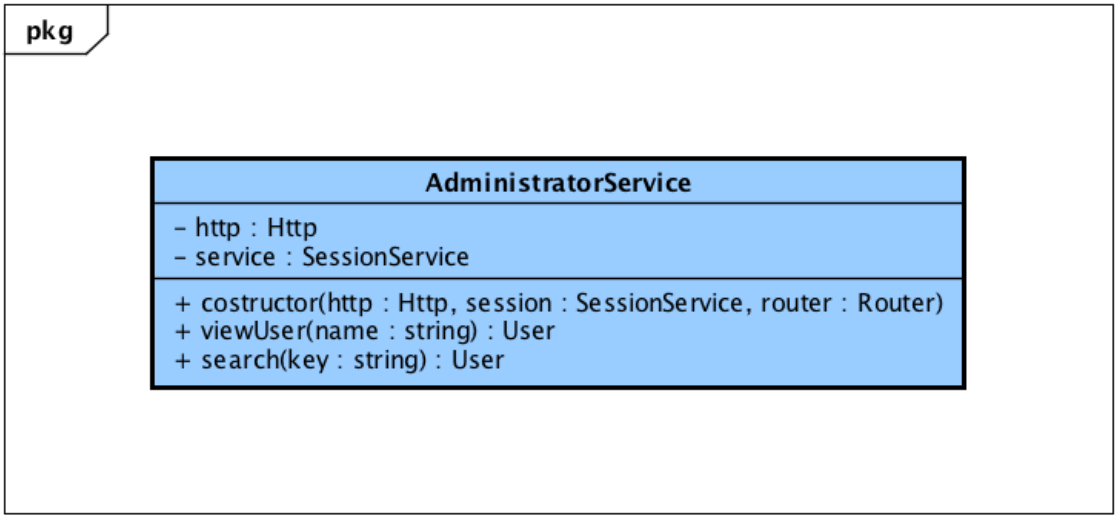

Figura 32: AdministratorService

- **Descrizione:** funzionalità per la gestione dell'amministratore.
- **Utilizzo:** visualizzazione e ricerca di un utente.
- **Classi ereditate:**
  - - Http http
    dependency injection del service Http
  - - SessionService Service
    dependency injection per il il servizio di sessione
- **Metodi:**
  - + constructor(Http http, SessionService session, Router router)
    costruttore che imposta la dependency injection di Http
  - + viewUser(string name)
    Funzione che ritorna un utente data una stringa, tale stringa è il suo username.
  - + search(string key)
    Ricerca utente tramite chiave identificativa.

4.2.27 viewModel::Service::BuyApiService

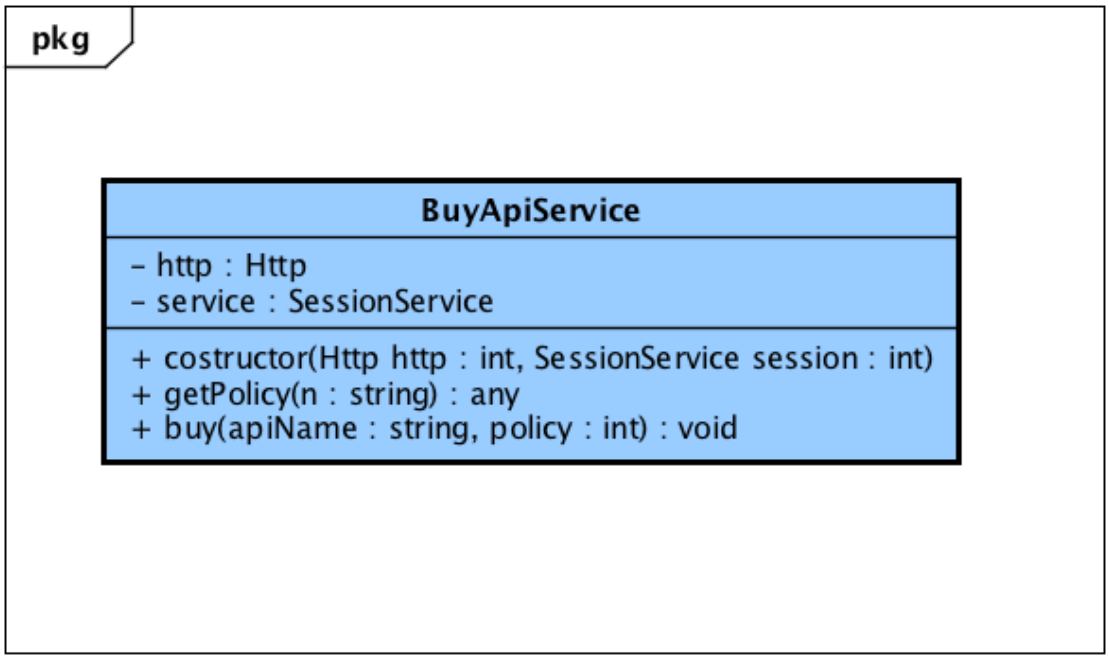

Figura 33: BuyApiService

- **Descrizione:** funzionalità per la l'acquisto di una nuova API.
- **Utilizzo:** assegnazione di un microservizio all'utente e scelta della policy.
- **Classi ereditate:**
  - - Http http
    dependency injection del service Http
  - - SessionService Service
    dependency injection per il il servizio di sessione
- **Metodi:**
  - + constructor(Http http, SessionService session)
    costruttore che imposta la dependency injection di Http e SessionService
  - + any getPolicy(string n)
    Funzione che ritorna un Policy di un API data la sua stringa identificativa
  - + buy(string apiName, number policy)
    Procede all'acquisto di un API da parte dell'utente in sessione.

4.2.28 viewModel::Service::CompareService

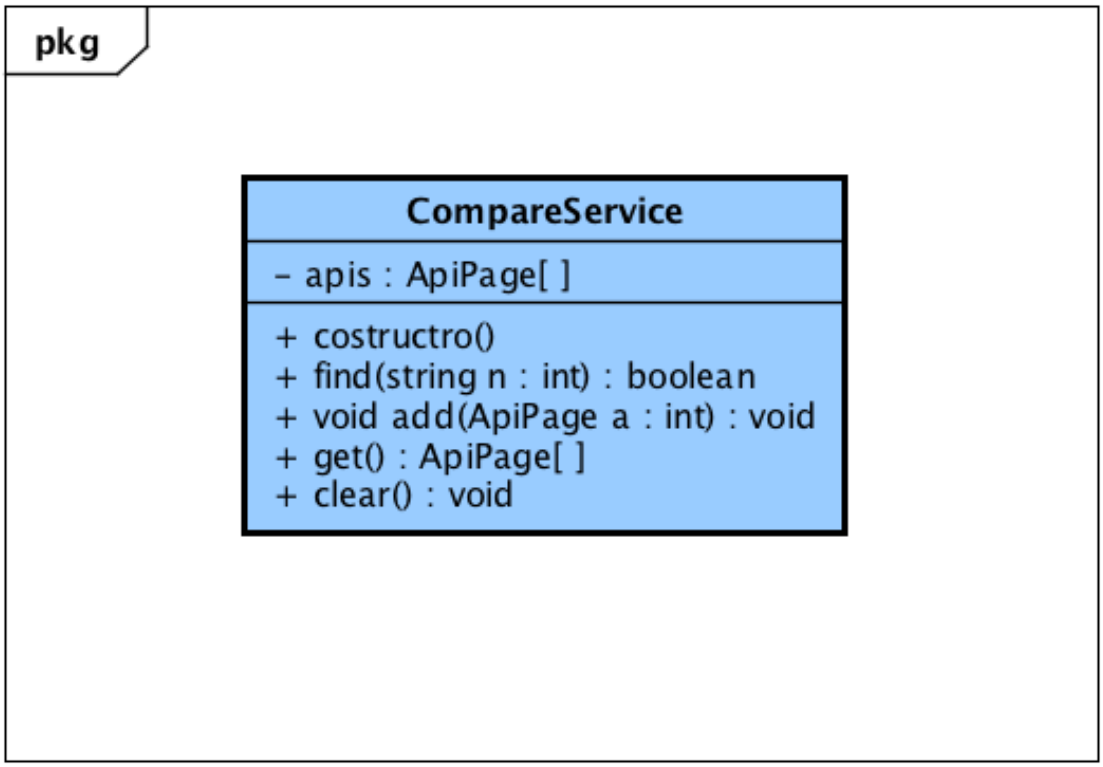

Figura 34: CompareService

- **Descrizione:** funzionalità per il confronto di microservizi.
- **Utilizzo:** confronto di microservizi e svuotamento della tabella di confronto.
- **Classi ereditate:**
  - - ApiPage Api
    ricavo del modello API
- **Attributi:**
  - - ApiPage[] apis
    contenitore dei microservizi da confrontare
- **Metodi:**
  - + constructor()
    costruttore di default
  - + boolean find(string n)
    Ritorna positivo se tramite stringa è stata trovata un API da confrontare
  - + void add(ApiPage a)
    Aggiungere un microservizio per il confronto
  - + ApiPage[] get()
    Ritorna il contenitore di applicazioni da confrontare
  - + void clear()
    Cancella la collezione di API per un nuovo confronto

4.3 Model

4.3.1 model::Data

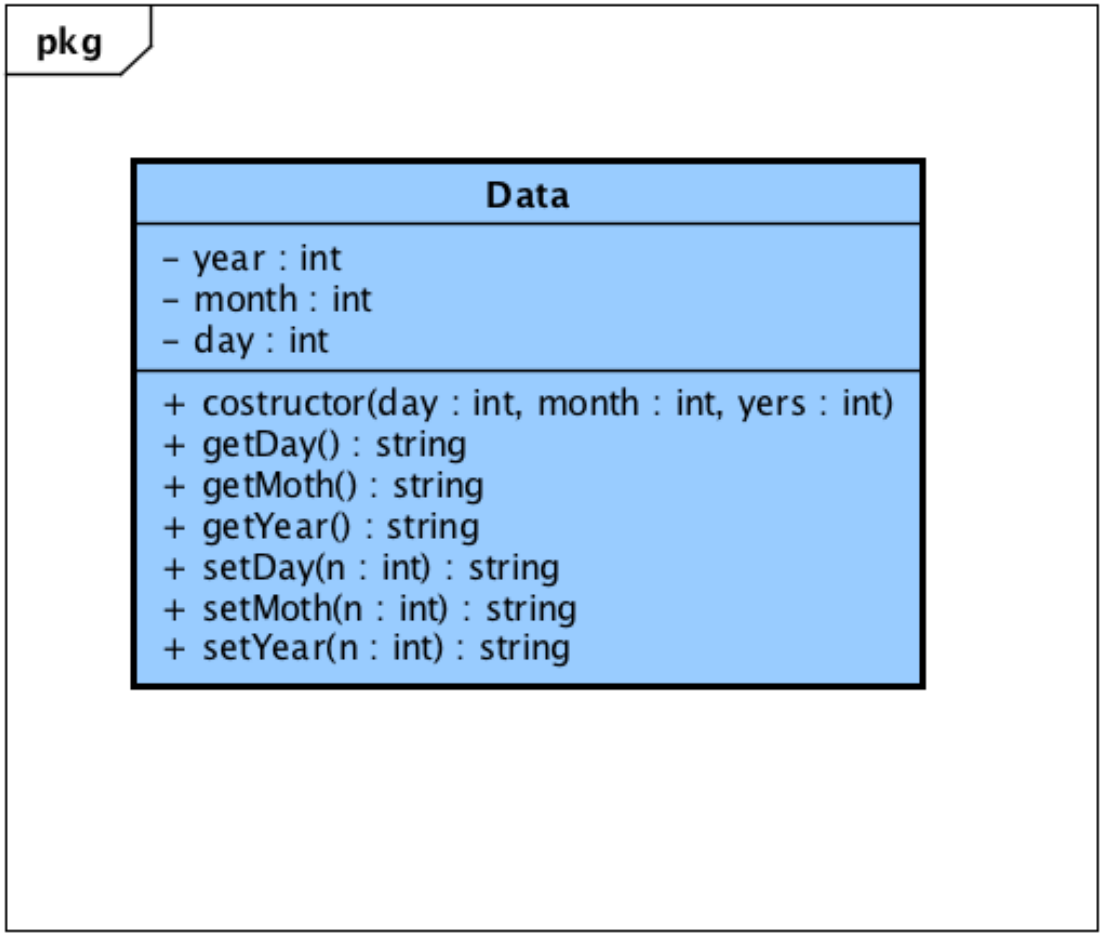

Figura 35: Data

- **Descrizione:**classe che descrive una data.
- **Utilizzo:** viene utilizzata per definire il tipo data.
- **Attributi:**
  - - number year
    definisce l'anno
  - - number month
    definisce il mese
  - - number day
    definisce il giorno
- **Metodi:**
  - + constructor (number day, number month, number year)
    costruttore di data.
  - + string getPlainDate()
    restituisce la data sottoforma di stringa

- + string getDay()
  restituisce giorno
- + string getMonth()
  restituisce mese
- + string getYear()
  restituisce anno
- + void setDay(number n)
  modifica giorno e restituisce stringa di conferma
- + void setMonth(number n)
  modifica mese e restituisce stringa di conferma
- + void setYear(number n)
  modifica anno e restituisce stringa di conferma

4.3.2 model::Comment

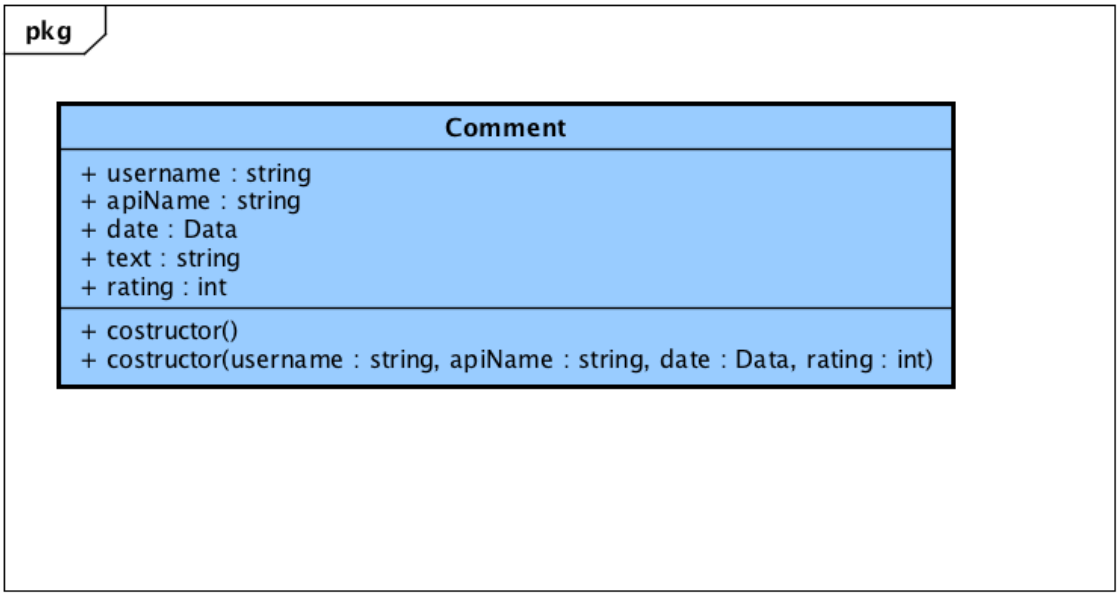

Figura 36: Comment

- **Descrizione:** definisce la forma di un commento
- **Utilizzo:** viene utilizzato per contenere un commento di un utente
- **Attributi:**
  - + string userName
    contiene lo username dell'autore del commento
  - + string apiName
    contiene il nome dell'API a cui si rifierisce
  - + Data date
    contiene la data di creazione del commento
  - + string text
    contiene il testo del commento

  - + int rating
    contiene il voto all'API commentata

- **Metodi:**
  - + constructor()
    crea un commento vuoto
  - + constructor(string userName, string apiName, Data date, string text, number rating)
    costruttore di un commento con i parametri.
  - + string getUserName()
    restituisce la user dell'utente in formato stringa
  - + string getApiName()
    restituisce il nome dell'API in formato stringa
  - + string getData()
    restituisce data dell'invio del commento in formato *Data*
  - + string getText()
    restituisce il testo del commento in formato strigna
  - + string getRating()
    restituisce il voto dato con tale commento
  - + string setUserName()
    inserisce e modifica la user dell'utente nel commento
  - + string setApiName()
    inserisce e modifica il nome dell'API del nel commento
  - + Data setData()
    inserisce e modifica data dell'invio del commento in formato *Data*
  - + string setText()
    inserisce e modifica il testo del commento
  - + number setRating()
    inserisce e modifica il voto dato con tale commento

4.3.3 model::User

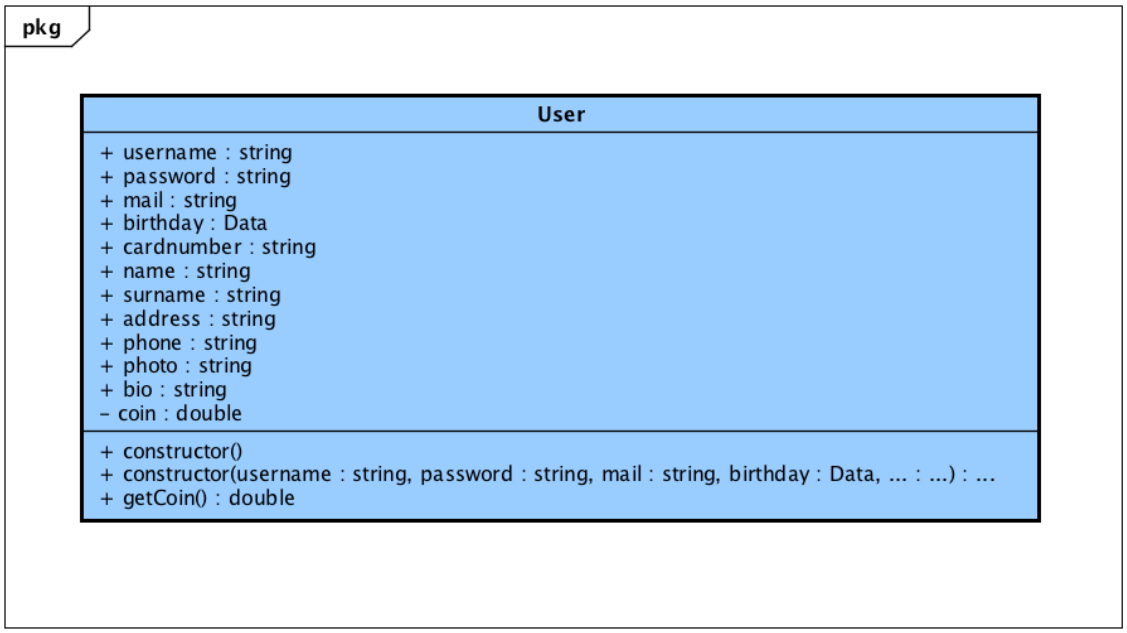

Figura 37: User

- **Descrizione:** classe che definisce un utente
- **Utilizzo:** viene usato per contiene tutte le informazioni di un utente
- **Attributi:**
  - + string username
    contiene lo username dell'utente
  - + string password
    contiene la password dell'utente
  - + string mail
    contiene l'indirizzo email dell'utente
  - + Data birthday
    contiene la data di nascita dell'utente
  - + string cardnumber
    contiene la carta di credito dell'utente
  - + string name
    contiene il nome dell'utente
  - + string surname
    contiene il cognome dell'utente
  - + string address
    contiene l'indirizzo dell'utente
  - + string city
    contiene la città dell'utente
  - + string province
    contiene la provincia dove risiede l'utente

  - + string zip
    contiene il codice postale dell'utente
  - + string country
    contiene lo stato di residenza dell'utente
  - + string phone
    contiene il numero di telefono dell'utente
  - + string photo
    contiene il link alla foto profilo dell'utente
  - + string bio
    contiene le informazioni generali dell'utente
  - + number coin
    contiene il credito dell'utente
  - + boolean isAdmin
    definisce se l'utente è amministratore o meno
  - + apiBuy[] apiBuyed
    raccoglie le API acquistate

- **Metodi:**
  - constructor()
    costruttore per definire un utente vuoto.
  - constructor(string username, string password, string mail, Data birthday, string cardnumber, string name, string surname, string address, string phone, string photo, string bio, number coin)
    cosruttore per definire un utente.
  - number getCoin()
    restituisce il saldo utente.
  - string getUsername()
    restituisce l'username dell'utente.
  - string getPassword()
    restituisce la password dell'utente.
  - string getMail()
    restituisce la mail dell'utente.
  - string getBirthday()
    restituisce il compleanno dell'utente.
  - string getCardnumber()
    restituisce il numero di carta dell'utente.
  - string getName()
    restituisce il nome proprio di carta dell'utente.
  - string getSurname()
    restituisce il cognome proprio dell'utente.
  - string getAddress()
    restituisce il cognome proprio dell'utente.
  - number getPhone()
    restituisce il telefono dell'utente.

- string getPhoto()
  restituisce la foto profilo dell'utente.
- string getBio()
  restituisce la descrizione biografica dell'utente.
- boolean getIsAdmin()
  restituisce il telefono dell'utente.
- ApiBuy[] getApiBuyed()
  restituisce le Api acquistate.
- void setUsername(string)
  inserisce l'username
- void setPassword(string)
  inserisce la password
- void setMail(string)
  inserisce la email
- void setBirthday(string)
  inserisce la data di nascita
- void setCardNumber(number)
  inserisce il numero della carta di credito
- void setName(string)
  inserisce il nome proprio dell'utente item void setSurname(string)
  inserisce il cognome proprio dell'utente item void setAddress(string)
  inserisce l'indirizzo dell'utente item void setName(number)
  inserisce il telefono dell'utente item void setPhoto(string)
  inserisce la foto profilo all'utente item void seBio(string)
  inserisce la biografia dell'utente item void setCoin(number)
  inserisce e modifica il saldo dell'utente item void setIsAdmin(boolean)
  definisce se l'utente è amministratore item void setApiBuy(ApiBuy[])
  definisce una lista di API acquistate dall'utente item void pushApiBuyed(ApiBuy)
  aggiunge alla lista di API acquistate una nuova API

4.3.4 model::ApiPreview

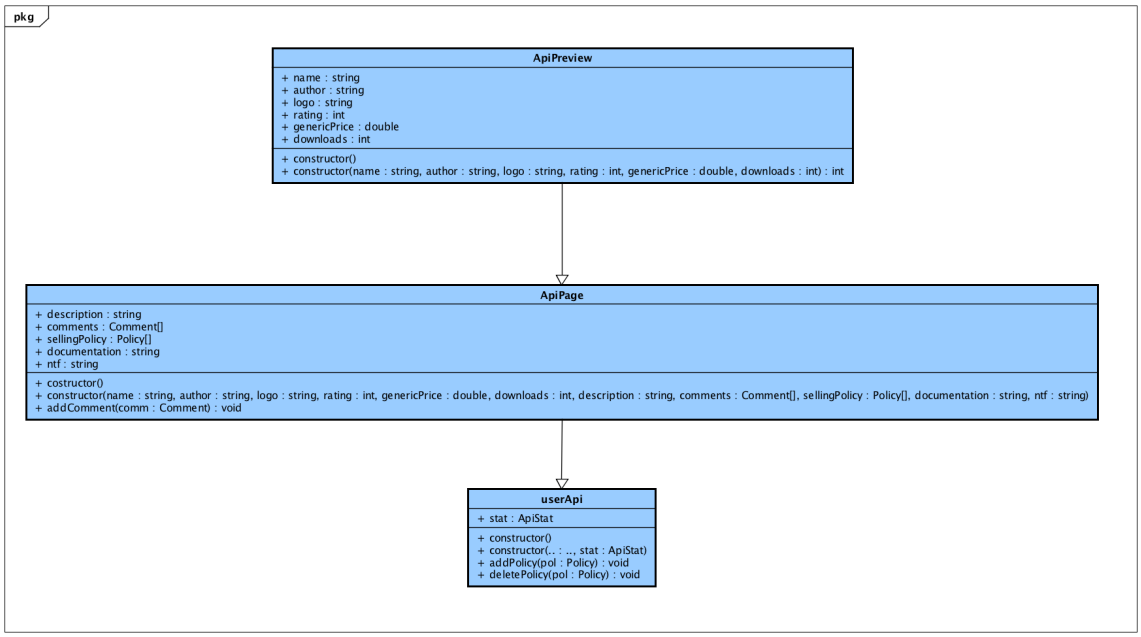

Figura 38: ApiPreview

- **Descrizione:** classe che definisce la preview di una API
- **Utilizzo:** viene utilizzata per salvare i dati di una API da mostrare come preview sulla home page o sulla pagina di ricerca
- **Attributi:**
  - + string name
    contiene il nome dell'API
  - + string author
    contine lo username dell'autore dell'API
  - + string logo
    contiene il link al logo dell'API
  - + number rating
    contiene il rating medio dell'API
  - + number genericPrice
    contiene il prezzo più basso impostato sull'API
  - + number downloads
    contiene il numero di downloads totali dell'API
- **Metodi:**
  - + constructor()
    costruttore di default che costruisce un oggetto vuoto.
  - + constructor(string name, string author, string logo, number rating, number genericPrice, number downloads)
    costruttore dell'oggetto con parametri.

4.3.5 model::ApiPage

- **Descrizione:** classe che definisce una API
- **Utilizzo:** viene utilizzata per salvare i dati di una API da mostrare sulla pagina dell'API.
- **Classi ereditate:**
  - ApiPreview
- **Attributi:**
  - + string description
    contiene la descrizione dell'API
  - + Comment[] comments
    contine l'elenco dei commenti collegato all'API
  - + Policy[] sellingPolicy
    contiene l'elenco delle policy applicate a questa API
  - + string documentation
    contiene il link alla documentazione
  - + string ntf
    contiene l'interfaccia dell'API
- **Metodi:**
  - + constructor()
    costruttore di default che costruisce un oggetto vuoto.
  - + constructor(string name, string author, string logo, number rating, number genericPrice, number downloads,string description, Comment[] comments,Policy[] sellingPolicy,string documentation,string ntf)
    costruttore dell'oggetto con parametri.
  - + void addComment(Comment comm)
    aggiunge un nuovo commento all'API.

4.3.6 model::UserApi

- **Descrizione:** classe che definisce un'API completa
- **Utilizzo:** viene utilizzata per salvare le API viste dall'utente propietario e autenticato.
- **Classi ereditate:**
  - ApiPage
- **Attributi:**
  - + ApiStat stat
    contiene le statistiche dell'API
- **Metodi:**
  - + constructor()
    costruttore di default che costruisce un oggetto vuoto.

- + constructor(string name, string author, string logo, number rating, number genericPrice, number downloads,string description, Comment[] comments,Policy[] sellingPolicy,string documentation,string ntf, ApiStat stat)
  costruttore dell'oggetto con parametri.
- + void addPolicy(Policy pol)
  aggiunge una nuova policy di vendita all'API.
- + void deletePolicy(Policy pol)
  rimuove una policy di vendita all'API.

4.3.7 model::Categories

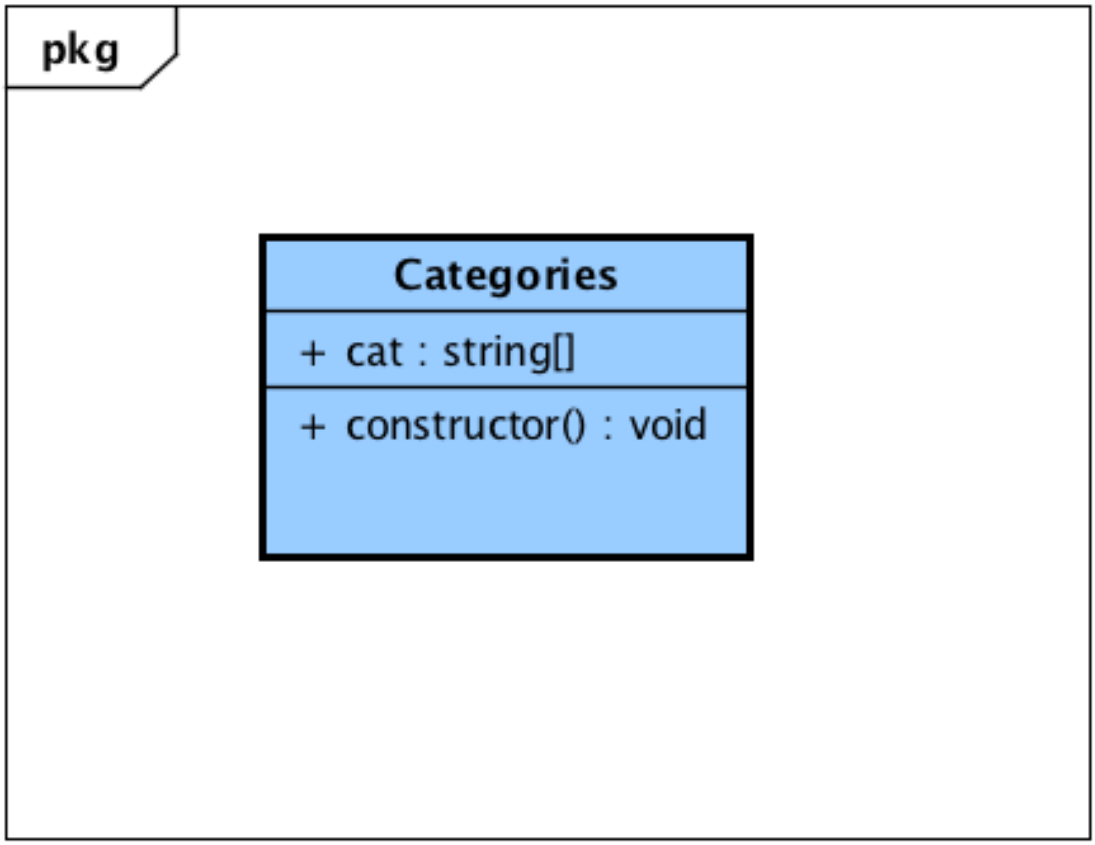

Figura 39: Categories

- **Descrizione:** contiene l'elenco delle categorie
- **Utilizzo:** viene utilizzato nella selezione delle cateogrie
- **Attributi:**
  - +string[] cat
    contiene l'elenco delle categorie
- **Metodi:**
  - + constructor()
    imposta l'elenco delle categorie

4.3.8 model::Policy

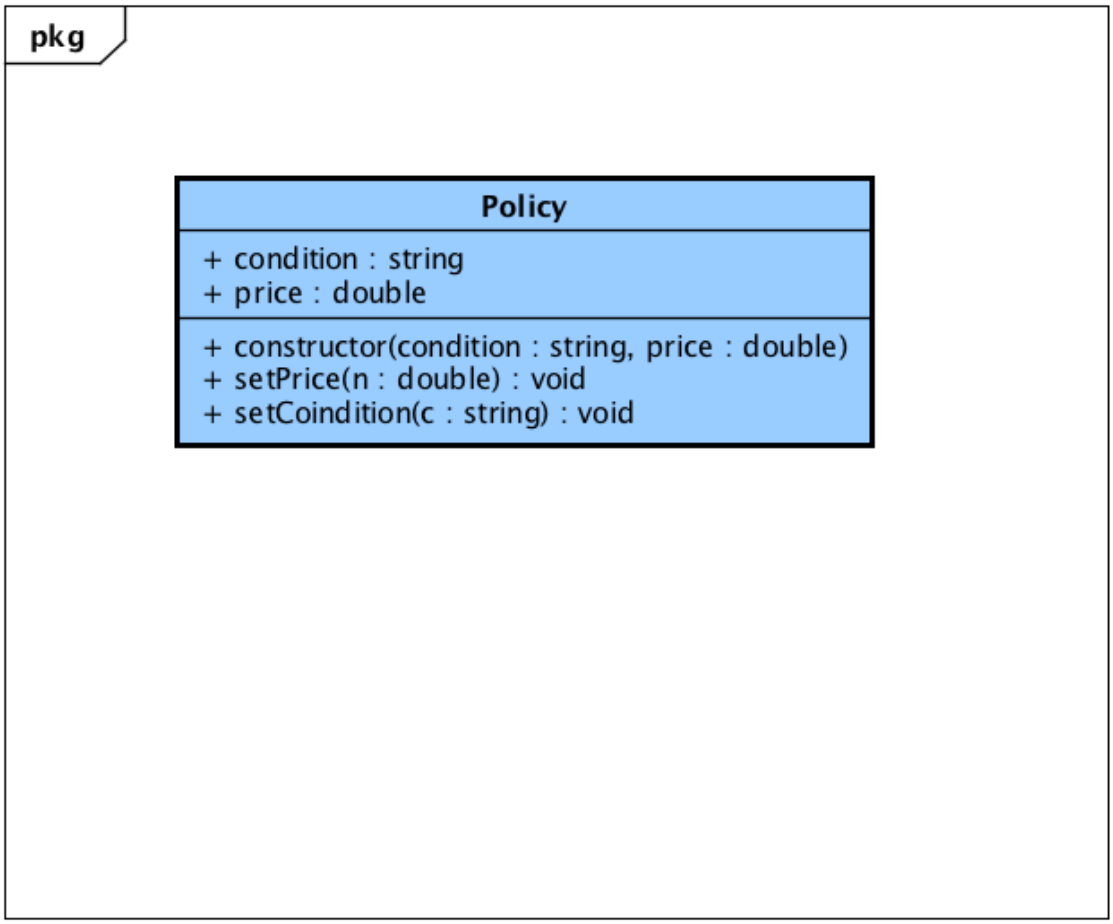

Figura 40: Policy

- **Descrizione:** contiene una policy di vendita
- **Utilizzo:** viene usata per descrivere una policy di vendita
- **Attributi:**
  - + string name
    definisce il nome della policy
  - + string condition
    definisce la condizione della policy
  - + number price
    definisce il prezzo per la policy
  - + number bytes
    definisce il numero di bytes per la policy
  - + number days
    definisce il numero di giorni che è attiva la policy
  - + number id
    definisce un numero identificativo per la policy
- **Metodi:**

- + constructor(string condition, number price, number price, number bytes, number days, number id)
  costruttore della policy

- + string getName()
  ritorna la stringa del nome della Policy
- + string getCondition()
  ritorna la condizione in formato stringa della Policy
- + number getPrice()
  ritorna il prezzo standard della Policy
- + number getBytes()
  ritorna il numero di bytes impostati nella Policy
- + number getDays()
  ritorna il giorni di vita della Policy
- + number getId()
  ritorna il numero identificativo della Policy
- + void setName(string c)
  modifica il nome della condizione
- + void setCondition(string c)
  modifica la condizione
- + void setPrice(number n)
  modifica il prezzo
- + void setBytes(number n)
  modifica i numero di bytes
- + void setDay(number n)
  modifica il numero di giorni di vita della Policy

5 Diagrammi di sequenza

5.1 Sequenza di Acquisto API

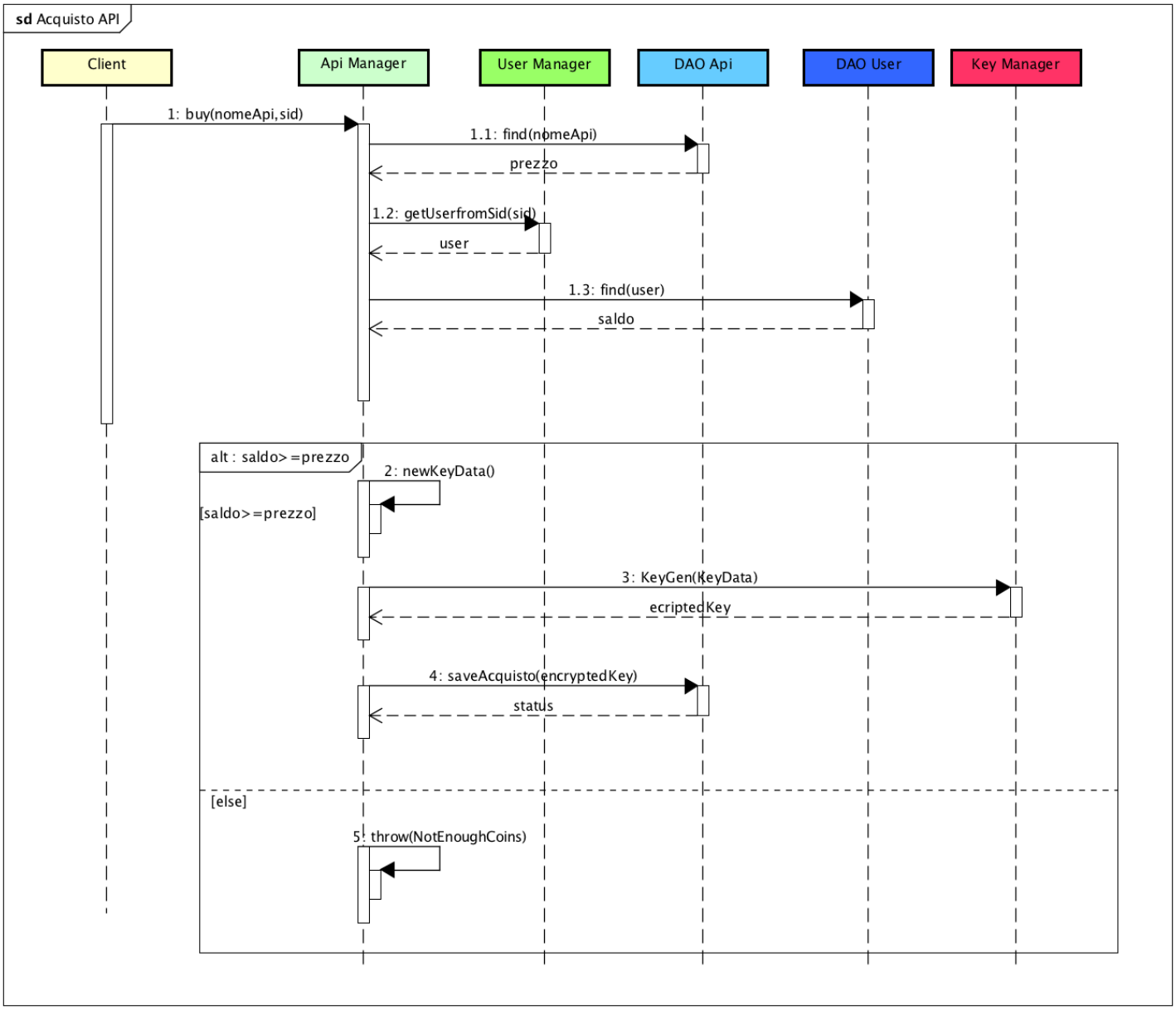

Figura 41: Sequenza di Acquisto API

Questo diagramma descrive le interazioni tra gli oggetti necessari del backend per eseguire l'acquisto di una API.

APImanager: Questo micro servizio contiene i metodi per eseguire e gestire le operazioni che riguardano la compravendita delle interfacce. Espone il metodo buy a tale scopo. Per prima cosa recupera le informazioni della API attraverso una chiamata find verso Il microservizio DAOApi e recupera il prezzo della api, in parallelo esegue una ricerca del nome dell'utente a partire dal session ID, per poi procedere al recupero delle informazioni dell'utente attraverso DAOUser e ne recupera il saldo. In caso il saldo sia maggiore o uguale al prezzo si procede alla generazione di una nuova chiave attraverso il micro servizio Keymanager,

il quale ritorna una chiave criptata, infine si procede all'inserimento di tali informazioni nel database. In caso il saldo non sia sufficiente all'acquisto viene sollevata un'eccezione di tipo NotEnoughCoins.

UserManager: Questo micro servizio si occupa di gestire le informazioni dell'utente e delle sessioni. Espone il metodo getUserFromSid(), Che recupera username a partire dall'id della sessione.

DAOApi: Questo micro servizio si occupa di eseguire le interrogazioni al database che riguardano le API. Espone il metodo find() per recuperare le informazioni di una certa API ed il metodo saveAcquisto() il quale memorizza nel database l'acquisto appena effettuato.

DAOUser: Questo micro servizio si occupa di eseguire le interrogazioni al database che guardano gli utenti. Espone il metodo find() per recuperare le informazioni di un certo user.

KeyManager: Questo micro servizio si occupa di generare e salvare le chiavi dei microservizi all'interno del database. Espone il metodo keygen() il quale genera una nuova chiave e la memorizza nel database.

5.2 Sequenza Gateway Redirect

Client: Questo oggetto indica un generico utilizzatori del microservizio. Invoca un'operazione definita nell'interfaccia che lo sviluppatore ha caricato nel market verso il gateway il quale si occuperà di reindirizzarla.

Gateway: Questo micro servizio si occupa di reindirizzare dinamicamente le chiamate verso ogni micro servizio del market. Una volta che c'è voluta una richiesta da un client per prima cosa valida la chiave nel servizio Keymanager. Se la chiave è valida allora genera un timestamp attraverso il servizio Analyzer, prima di forwardare la richiesta verso il server del microservizio. Una volta ricevuta la risposta dal server del microservizio calcola un ulteriore timestamp. A questo punto calcola parallelamente sia in 1000 secondi impiegati che la dimensione in byte della risposta. Infine incrementa il numero di byte utilizzati della Key attraverso keyManager e incrementa parallelamente la dimensione e l'intervallo di tempo in APIManager.

Analyzer: Questo micro servizio si occupa di calcolare il tempo impiegato da un determinato micro servizio ad eseguire le operazioni richieste e a calcolare la dimensione in byte del valore ritornato. Espone i metodi getTime(), il quale si occupa di generare il timestamp corrispondente al momento della chiamata, timeDiff() il quale dati due timestamp genera il tempo impiegato in millisecondi, getValueSize()il quale si occupa di calcolare la dimensione in byte di un determinato oggetto.

KeyManager: Questo micro servizio si occupa di gestire le chiavi utilizzate per verificare l'accesso a microservizi. Espone il metodo validateKey(), il quale data una chiave controlla la presenza nel Database e la validità, Ossia se la data di scadenza e il numero massimo di byte non sono stati superati. Inoltre dispone un metodo incrementSize() il quale incrementa la dimensione dei byte utilizzati associata a tale chiave.

ApiManager: Questo micro servizio si occupa di gestire le informazioni delle API. Espone i metodi incrementSize() e incrementInterval(). Il primo si occupa di incrementare i byte inviati da quella determinata API, mentre il secondo si occupa di incrementare il tempo totale di utilizzo.

Server: Questo micro servizio indica un generico server sul quale è eseguita l'implementazione dell'interfaccia fornita dallo sviluppatore.

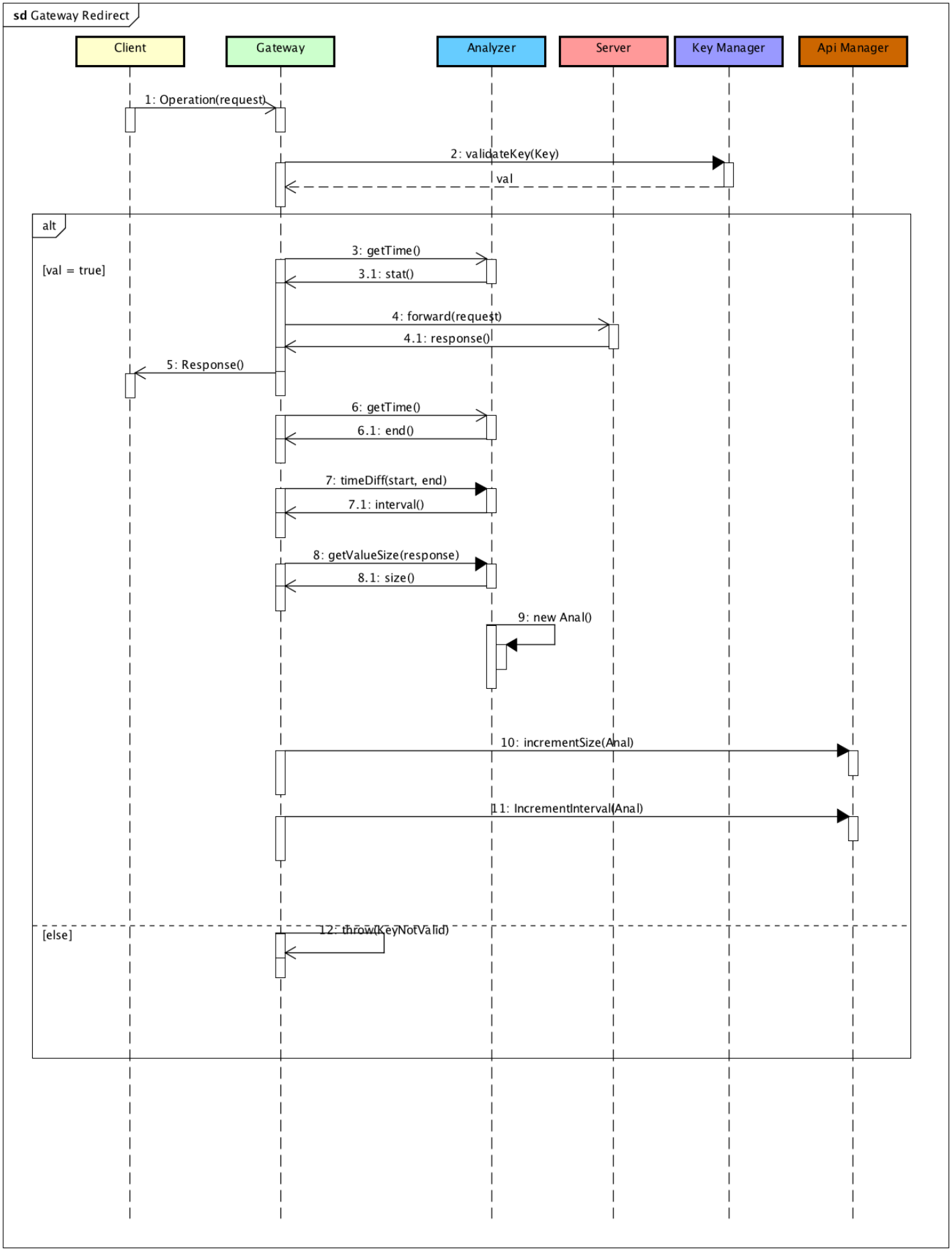

Figura 42: Sequenza Gateway Redirect

5.3 Sequenza Autenticazione

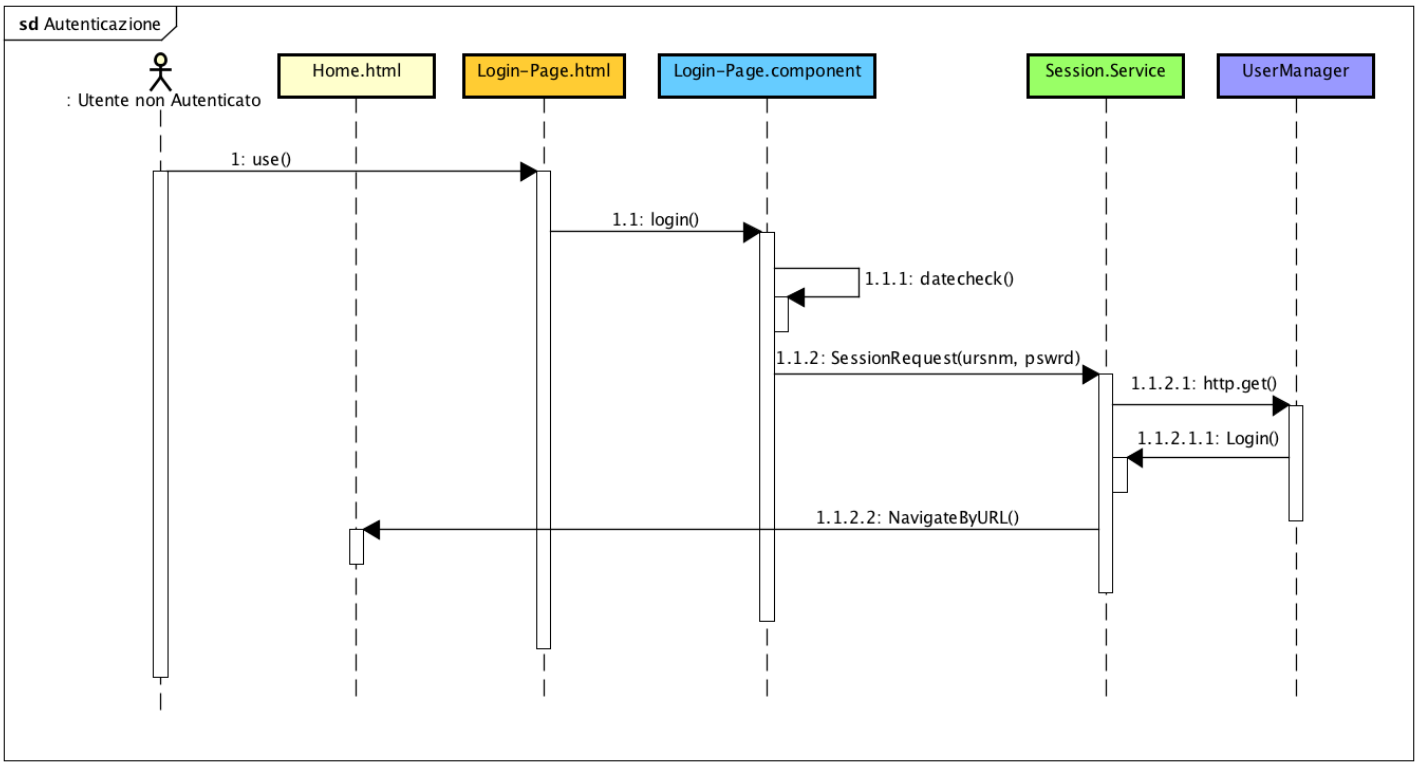

Figura 43: Sequenza Autenticazione

Il diagramma descrive le interazioni degli oggetti Front End che vengono coinvolte durante la procedura di login.

- **Utente non autenticato:** Rappresenta un qualunque utente che accede all'applicazione e vuole effettuare login. Deve riempire il form adibito della pagina di autenticazione;
- **login-page.html:** Rappresenta la pagina di login che raccoglie i dati dell'utente e li invia al componente tramite il bottone "accedi" che chiama la funzione login();
- **login-page.component:** Il componente adibito al login esegue un controllo sui dati, e se i dati sono corretti esegue la funzione sessionRequest del service SessionService iniettato nel componente;
- **session.service:** Il service controlla che non esista già una sessione ed esegue la richiesta al server di un nuovo token tramite chiamata http.get();
- **userManager:** Nel Back End il modulo userManager esegue il login, genera un token di sessione e lo rimanda al service. Il service una volta ricevuto il token di sessione dal backend lo salva ed esegue un redirect dell'utente verso la pagina home.
- **home.html:** L'utente autenticato viene mandato nella pagina di home da cui può accedere a tutte le funzionalità del sito.

5.4 Inserimento nuova Api

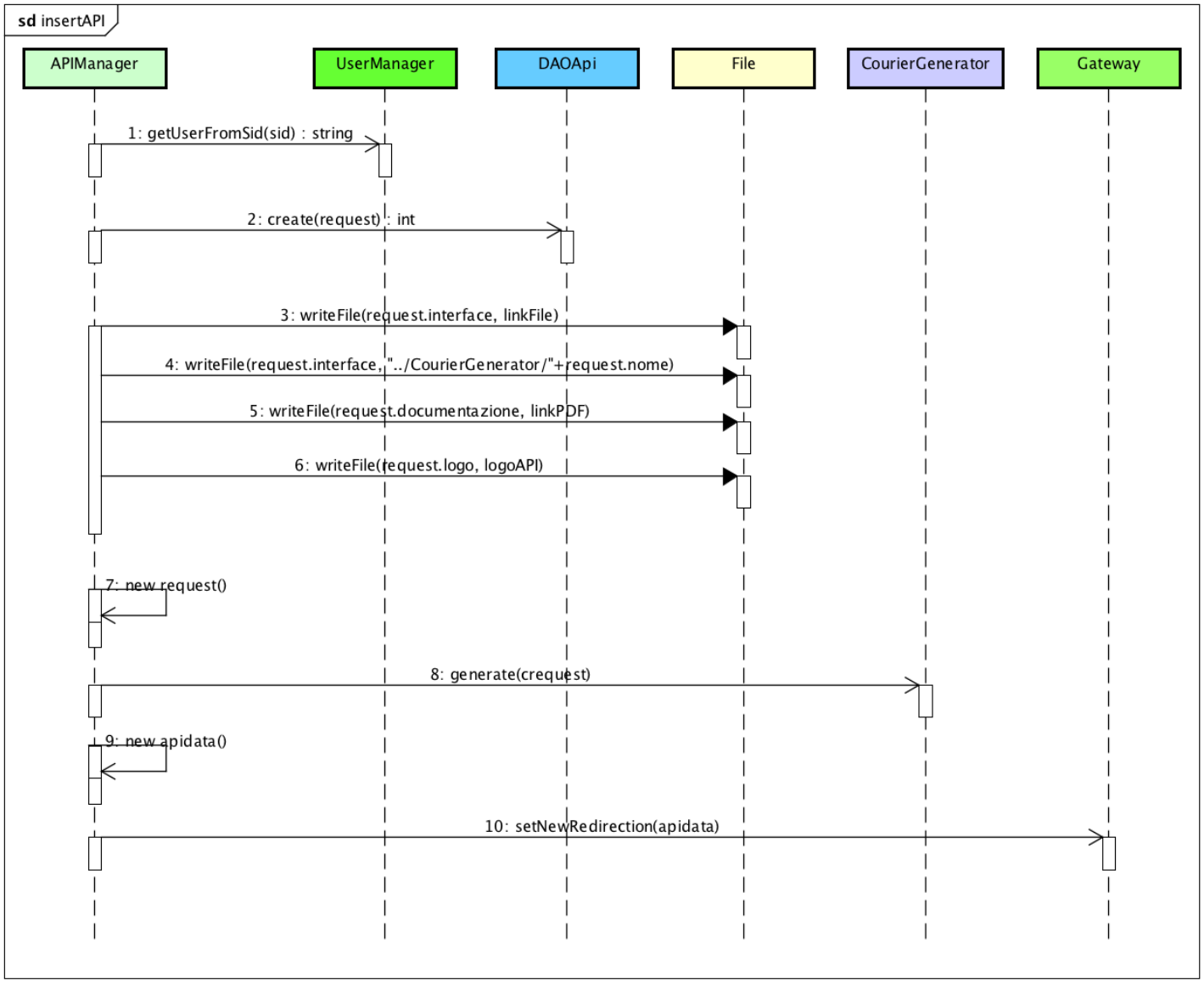

Figura 44: Sequenza Inserimento Nuova Api

Il diagramma descrive la procedura attraverso la quale viene inserita una nuova API nell'API Market.

- **ApiManager:** Questo micro servizio si occupa di gestire le informazioni delle API. Espone insertAPI(), il quale si occupa di inserire i dati dell'api nel database e scrivere i files dell'interfaccia e della documentazione nella cartella del server.
- **DAOApi:** Questo micro servizio si occupa di eseguire le interrogazioni al database che riguardano le API. Espone il metodo create() per creare le informazioni di una nuova API ed il metodo addPolicyToAPI il quale associa le specifiche della policy definita dall'utente all'API stessa.
- **File:** Servizio offerto da Jolie per scrivere i files su disco;
- **CourierGenerator:** Espone il metodo generate() il quale si occupa di creare il file da embeddare nel Gateway e il file dell'interfaccia estesa scaricabile dall'utente che acquista l'API;
- **Gateweay:** Microservizio che raccoglie i dati relativi alle chiamate alle API e contolla le relative chiavi. Espone setnewredirection() con la quale viene embeddato un nuovo servizio nel Gateway.

- **home.html:** L'utente autenticato viene mandato nella pagina di home da cui può accedere a tutte le funzionalità del sito.

5.5 Visualizzazione Specifica Api

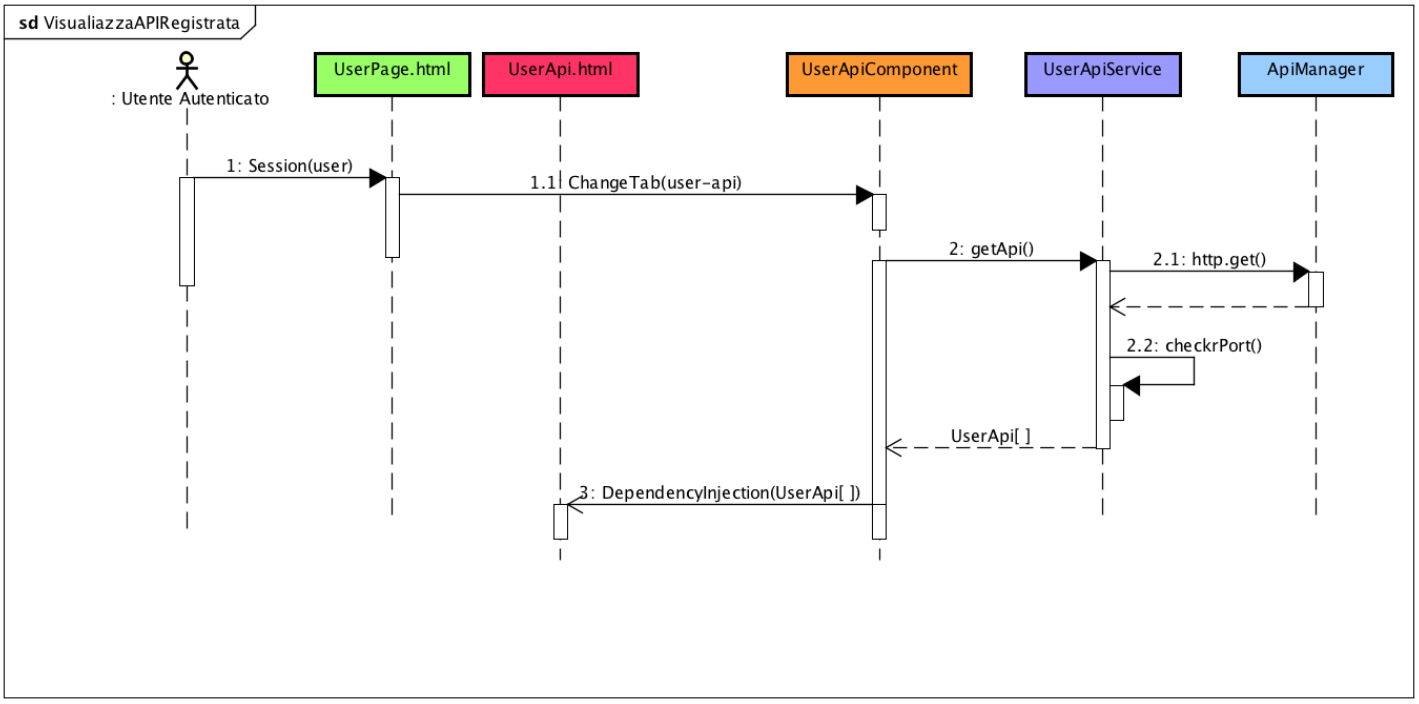

Figura 45: Sequenza Inserimento Nuova Api

Utente Autenticato: Un utente autenticato ha la possibilità di visualizzare le proprio *API* registrate nel sistema. In particolare visualizza nello specifico i suoi dettagli, come ad esempio le sue statistiche.

UserPage.html: Nella pagina principale dell'utente presenterà una sezione dedicata alla visualizzazione delle proprie API personali (acquistate e caricate).

UserApi.component: Il componente che gestisce le funzionalità si appoggerà al suo servizio di riferimento per utilizzare in praticità elementi della base di dati.

UserApi.service: Se il component coordina e gestisce tutte le parti è il service che lavora e utilizza funzionalità REST. Queste permettono la richiesta al database o l'inserimento.
Nel caso specifico della visualizzazione api http.get() , richiamata anticipatamente da getAPI() nel component. Richiederà al server una lista di API in formato array.
In seguito si è scelto di eseguire un controllo grazie dei campi in modo che siano leggibili dall'utente.

UserApi.html: Tramite quindi questo semplice processo avremo visualizzate nella nostra pagina utente dedicata alle API una lista di anteprima di microservizi.

1. UserApi : getApi();
2. UserService : http.get();
3. ritorno http.get();
4. CheckPort();
5. ritorno di getApi().

6 Tracciamento

6.1 Tracciamento requisiti - classi

Requisito	Classi
R0F1	session.service.spec.ts session.service.ts menu.html menu.css home.html menu.css registration.html registration.css base.html index.html appModule BaseComponent MenuComponent RegistrationComponent RegistrationService User DAOInterface DAOKeyInterface DAOKey DAOUserInterface DAOUser KeyManagerInterface KeyManager UserManagerInterface UserManager
R0F2	session.service.spec.ts session.service.ts logged-in.guard.spec.ts logged-in.ts menu.html menu.css home.html home.css login-page.html login-page.css base.html index.html appModule BaseComponent MenuComponent LoginPageComponent DAOInterface DAOUserInterface DAOUser UserManagerInterface UserManager
R0F2.4	new-api.component.css new-api.component.html new-api.component.spec.ts new-api.component.ts
R0F2.4.1	new-api.component.css new-api.component.html new-api.component.spec.ts new-api.component.ts

R0F2.5	logged-out.guard.spec.ts logged-out.guard.ts DAOUserInterface DAOUser UserManagerInterface UserManager
R0F3	session.service.spec.ts session.service.ts ap.ts new-api.service.spec.ts new-api.service.ts api-user.service.spec.ts api-user.service.ts new-api.component.css new-api.component.html new-api.component.spec.ts new-api.component.ts menu.html menu.css user-api.html user-api.css base.html index.html appModule BaseComponent MenuComponent UploadApiComponent UserApiService UploadApiService APIManagerInterface APIManager DAOApiInterface DAOApi DAOInterface
R0F3.1	APIManagerInterface
R0F3.4	policy.ts
R0F4	session.service.spec.ts session.service.ts menu.html menu.css user-page.html user-page.css base.html index.html appModule BaseComponent MenuComponent UserPageComponent UserService User Policy DAOInterface DAOUserInterface DAOUser UserManagerInterface UserManager
R1F4.1	user-page.html user-page.css UserComponent UserService

R1F4.2	user-page.html user-page.css UserComponent UserService
R0F4.3	user-page.html user-page.css UserComponent UserService
R0F4.3	APIManagerInterface APIManager
R0F4.3.2	AnalyzerInterface Analyzer
R1F4.4.3	buy-credit.service.spec.ts buy-credit.service.ts buy-api.component.css buy-api.component.html buy-api.component.spec.ts buy-api.component.ts
R0F5	session.service.spec.ts session.service.ts menu.html menu.css search-bar.html search-bar.css search-page.html search-page.html base.html index.html appModule BaseComponent MenuComponent SearchBarComponent SearchPageComponent SearchService APIManagerInterface APIManager DAOApiInterface DAOApi DAOInterface
R0F5.1	search-bar.html search-bar.css SearchBarComponent SearchService
R1F5.3	serach-page.html search-page.css SearchPageComponent SearchService

R0F6	session.service.spec.ts session.service.ts view-api-user.service.spec.ts view-api-user.ts data.ts menu.html menu.css apiHome.html apiHome.css viewApi.html viewApi.css base.html index.html search-page.html search-page.css BaseComponent MenuComponent ApiHomeComponent SearchPageComponent ApiPageService ApiHomeService ApiPrevew ApiPage UserApi DAOApiInterface DAOApi DAOInterface APIManagerInterface APIManager
R0F6.5	stat.ts AnalyzerInterface Analyzer
R0F6.5	KeyManagerInterface KeyManager DAOKeyInterface DAOKey
R0R6.6	buy-api.service.spec.ts buy-api.service.ts buy-api.component.css buy-api.component.html buy-api.component.spec.ts buy-api.component.ts apiBuy.ts
R0F6.6.1	polyfills.ts
ROF6.6.3	buy-credit,component.css buy-credit,component.html buy-credit,component.spec.ts buy-credit,component.ts
R0F6.8	compare.service.spec.ts compare.service.ts compare.component.css compare.component.html compare.component.spec.ts compare.component.ts AnalyzerInterface Analyzer

R1F6.9	comment.ts user-comments.spec.ts user-comments.ts user-comment.html user-comments.css UserCommentsComponent UserCommentService Comment CommentManagerInterface CommentManager DAOCommnentInterface DAOComment
R0F7	session.service.spec.ts session.service.ts DAOKeyInterface DAOKey KeyManagerInterface KeyManager
R0F7.1.1	KeyManager
R0F7.2	KeyManager
R1F9	session.service.spec.ts session.service.ts UserManagerInterface UserManager
R1F9.1	administrator.html administrator.css
R2F9.2	admministrator.html administrator.css UserManagerInterface UserManager
R2F9.3	admministrator.html administrator.css CommentManagerInterface CommentManager DAOCommentInterface DAOComment
R1F9.4	admministrator.html administrator.css DAOApiInterface APIManagerInterface APIManager
RF9.5	administrator.html administrator.css SearchBarComponent

Tabella 1: Requisiti Requisiti Classi

6.2 Tracciamento classi - requisiti

Classe	**Requisiti**
menu.component.html	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6

menu.component.css	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6
home.component.html	R0F1 R0F2
home.component.css	R0F1 R0F2
home.component.spec.ts	R0F1 R0F2
home.component.ts	R0F1 R0F2
apiHome.component.html	R0F6
apiHome.component.css	R0F6
apiHome.component.spec.ts	R0F6
apiHome.component.ts	R0F6
user-api.component.html	R0F3
user-api.component.css	R0F3
user-api.component.spec.ts	R0F3
user-api.component.ts	R0F3
user-page.component.html	R0F4 R1F4.1 R1F4.2 R0F4.3
user-page.component.css	R0F4 R1F4.1 R1F4.2 R0F4.3
user-page.component.spec.ts	R0F4 R1F4.1 R1F4.2 R0F4.3
user-page.component.ts	R0F4 R1F4.1 R1F4.2 R0F4.3
registration.component.html	R0F1
registration.component.css	R0F1
registration.modUser.html	R0F1
user-comments.component.html	R1F6.9
user-comments.component.css	R1F6.9
user-comments.component.spec.ts	R1F6.9
user-comments.component.ts	R1F6.9
login-page.component.html	R0F2
login-page.component.css	R0F2
login-page.component.spec.ts	R0F2

login-page.component.ts	R0F2
viewApi.component.html	R0F6
viewApi.component.css	R0F6
viewApi.component.spec.ts	R0F6
viewApi.component.ts	R0F6
search-bar.component.html	R0F5 R0F5.1 R1F9.5
search-bar.component.css	R0F5 R0F5.1 R1F9.5
search-bar.component.spec.ts	R0F5 R0F5.1 R1F9.5
search-bar.component.ts	R0F5 R0F5.1 R1F9.5
search-page.component.html	R0F5 R1F5.3 R0F6
search-page.component.css	R0F5 R1F5.3 R0F6
search-page.component.spec.ts	R0F5 R1F5.3 R0F6
search-page.component.ts	R0F5 R1F5.3 R0F6
search-page.component2.html	R0F5 R1F5.3 R0F6
base.component.html	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6
base.component.css	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6
base.component.spec.tsl	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6

base.component.ts	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6
index.html	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6
appModule.ts	R0F1 R0F2 R0F3 R0F4 R0F5
api-page.service.ts	R0F6
api-page.service.spec.ts	R0F6
registration.service.spec.ts	R0F1
registration.service.ts	R0F1
user.service.spec.ts	R0F4 R1F4.1 R1F4.2 R0F4.3
user.service.ts	R0F4 R1F4.1 R1F4.2 R0F4.3
user-api.service.spec.ts	R0F3
user-api.service.ts	R0F3
api-home.service.spec.ts	R0F6
api-home.service.ts	R0F6
search.service.spec.ts	R0F5 R0F5.1 R1F5.3
search.service.ts	R0F5 R0F5.1 R1F5.3
api-comment.css	R1F6.9
api-comment.html	R1F6.9
api-comment.spec.ts	R1F6.9
api-comment.ts	R1F6.9
user-profile.component.html	ROF1 R0F4
user-profile.component.css	ROF1 R0F4
user-profile.component.spec.ts	ROF1 R0F4
user-profile.component.ts	ROF1 R0F4

user-profile.component2.html	ROF1 R0F4
view-api-user.component.css	R0F6
view-api-user.component.html	R0F6
view-api-user.component.spec.ts	R0F6
view-api-user.component.ts	R0F6
buy-api.component.css	R0F6
buy-api.component.html	R0F6
buy-api.component.spec.ts	R0F6
buy-api.component.ts	R0F6
buy-credits.css	R1F4.4.3 R0F6.6.3
buy-credits.html	R1F4.4.3 R0F6.6.3
buy-credits.spec.ts	R1F4.4.3 R0F6.6.3
buy-credits.ts	R1F4.4.3 R0F6.6.3
compare.componet.css	R0F8
compare.componet.html	R0F8
compare.componet.spec.ts	R0F8
compare.componet.ts	R0F8
new-api.component.css	R0F3
new-api.component.html	R0F3
new-api.component.spec.ts	R0F3
new-api.component.ts	R0F3
password-recovery.component.css	R1F2.4 R1F2.4.1
password-recovery.component.html	R1F2.4 R1F2.4.1
password-recovery.component.spec.ts	R1F2.4 R1F2.4.1
password-recovery.component.ts	R1F2.4 R1F2.4.1
ap.ts	R0F3
apiBuy.ts	R0F6.6
comment.ts	R1F6.9
data.ts	R0F6
policy.ts	R0F3.4
stat.ts	R1F6.2 R0F6.5
api-user.service.spec.ts	R0F3
api-user.service.ts	R0F3
buy-api.service.spec.ts	R0F6.6
buy-api.service.ts	R0F6.6

buy-credits.service.spec.ts	R1F4.4.3
buy-credits.service.ts	R1F4.4.3
compare.service.spec.ts	R0F6.8
compare.service.ts	R0F6.8
logged-in.services.spec.ts	R0F2
logged-in.services.ts	R0F2
logged-out.services.spec.ts	R0F2.5
logged-out.services.ts	R0F2.5
new-api.service.spec.ts	R0F3
new-api.service.ts	R0F3
session.service.spec.ts	
session.service.ts	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6 R0F7 R0F9
user-comments.service.spec.ts	R1F6.9
user-comments.service.ts	R1F6.9
view-api-user.service.spec.ts	R0F6
view-api-user.service.ts	R0F6
user.service.ts	ROF1 R0F4
polyfills.ts	R0F6.6.1
APIManagerInterface	R0F3 R0F3.1 R0F4.3 R0F5 R0F6 R2F9.4
APIManager	R0F3 R0F4.3 R0F5 R0F6 R2F9.4
AnalyzerInterface	R0F4.3.2 R0F6.5 R0F6.8
Analyzer	R0F4.3.2 R0F6.5
GatewayInterface	
Gateway	
CommentManagerInterface	R1F6.9 R2F9.3
CommentManager	R1F6.9 R2F9.3

DAOApiInterface	R0F3 R0F5 R0F6
DAOApi	R0F3 R0F5 R0F6 R2F9.4
DAOCommentInterface	R1F6.9 R2F9.3
DAOComment	R1F6.9 R2F9.3
DAOInterface	R0F1 R0F2 R0F3 R0F4 R0F5 R0F6 R0F7
DAOKeyInterface	R0F7 R0F1 R0F6.6
DAOKey	R0F7 R0F1 R0F6.6
DAOUserInterface	R0F1 R0F2 R0F2.5 R0F4
DAOUser	R0F1 R0F2 R0F2.5 R0F4
KeyManagerInterface	R0F7 R0F1 R0F6.6
KeyManager	R0F7 R0F7.1.1 R0F7.2 R0F1 R0F6.6
UserManagerInterface	R0F1 R0F2 R0F2.5 R0F4 R1F9 R2F9.2
UserManager	R0F1 R0F2 R0F2.5 R0F4 R1F9 R2F9.2

Tabella 2: Requisiti Classi Requisiti