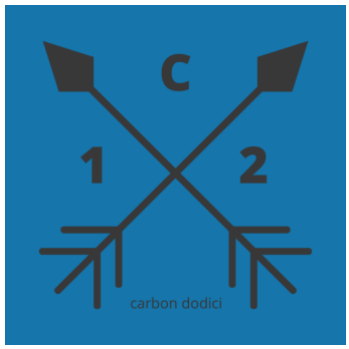

Carbon12 – Predire in Grafana

Allegato tecnico
Product Baseline

Informazioni sul documento

Data di approvazione	2020/05/11
Uso	Esterno
Destinatari	Carbon12 Zucchetti SPA Prof. Tullio Vardanega Prof. Riccardo Cardin
E-mail di riferimento	carbon.dodici@gmail.com

Scopo del documento

Il presente documento ha lo scopo di presentare la progettazione del prodotto realizzato da Carbon12 per il progetto Predire in Grafana.

Indice

1 Introduzione

Il presente documento ha lo scopo di presentare i diagrammi dei package, delle classi e di sequenza che illustrino la contestualizzazione dei design pattern adottati entro l'architettura del sistema.

Per maggiore facilità di lettura il documento è stato diviso in due capitoli dedicati rispettivamente alla descrizione del plug-in e a quella del programma di addestramento.

Per ciascun capitolo verranno riportati i diagrammi di:

- Package;
- Classi;
- Sequenza

sviluppati dal team Carbon12.

Nella sezione seguente invece verrà presentata la contestualizzazione dei design pattern adottati entro l'architettura del sistema.

1.1 Design pattern

1.1.1 Programma di addestramento

1.1.1.1 MVC pull model

Il programma di addestramento è costituito da un'applicazione web che si basa sulla tecnologia Node.js e sul framework Express. Viene utilizzato poi il linguaggio di markup HTML in accoppiata con il template engine EJS per la realizzazione delle viste con cui l'utente interagisce. Pertanto, per la sua realizzazione è stata adottata un'architettura MVC pull model dove il server rappresenta il controller che gestisce le richieste http provenienti dalla vista, smistandole utilizzando le route del framework Express e chiamando gli opportuni metodi sulle classi che gestiscono il modello. Quest'ultimo è rappresentato dalle classi responsabili della gestione dei file dati in ingresso dall'utente e dalle librerie che implementano gli algoritmi di machine learning. Non vi è quindi diretto contatto tra il modello e la vista, quest'ultima viene aggiornata sempre dal server che ottenuti i dati elaborati li fornisce alla vista per la loro visualizzazione utilizzando il template engine EJS. (Figura 2)

1.1.1.2 Object Adapter

Per la gestione delle librerie esterne che implementano gli algoritmi per RL e SVM si è utilizzato il design pattern Object Adapter. Sono stati raccolti quindi in un'unica interfaccia i soli metodi necessari al server per ottenere i risultati dell'allenamento e per configurare i modelli con la configurazione ottenuta da un eventuale predittore in ingresso ottenuto da un allenamento precedente, nel caso si tratti di un aggiornamento. Tale interfaccia è opportunamente implementata da due classi adapter: RL_Adapter e SVM_Adapter che richiamano i metodi necessari all'interno delle librerie utilizzate. Tale pattern è stato adottato per permettere in futuro l'aggiunta di nuove librerie che implementano nuovi algoritmi. (Figura 2)

1.1.2 Plug-in

1.1.2.1 MVVM

L'interfaccia web visualizzata dall'utente è stata implementata utilizzando il framework AngulaJS, di conseguenza viene usato per il plug-in il pattern MVVM, con le pagine HTML che fungono da view, ImportCtrl e PredictCtrl da view-model ed InfluxDB, SVM_Adapter e RL_Adapter da model.

1.1.2.2 Object Adapter

Le librerie di Machine Learning per effettuare la previsione sono state gestite come nell'applicazione di addestramento, pertanto mediante l'utilizzo del pattern Object Adapter esponendo un'interfaccia che permetta di configurare i modelli con la configurazione ottenuta dall'addestramento e di richiamare poi i metodi per effettuare la previsione sui dati. (Figura 8)

1.1.2.3 Builder

Per la gestione dei pannelli è stato utilizzato il pattern Builder, implementato dall'insieme delle classi Director, Builder, Target, View e Panel. In questo modo è stato possibile separare le responsabilità di costruzione delle componenti di un pannello in classi dedicate, riducendo la complessità della classe Panel.

2. Programma di addestramento

2.1 Diagrammi dei package

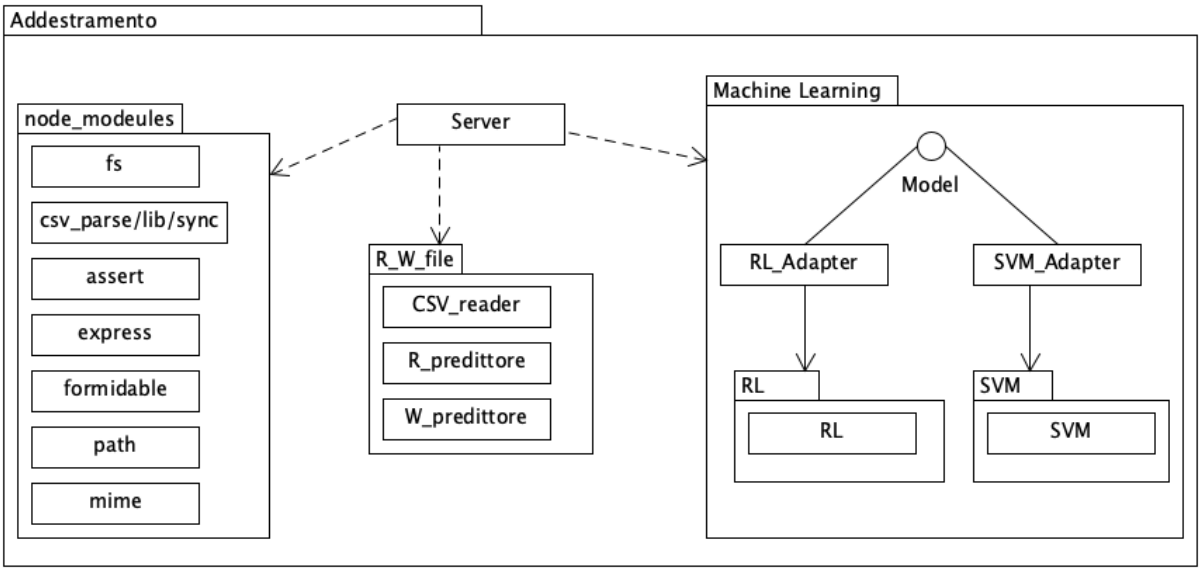

FIGURA 1: DIAGRAMMA DEI PACKAGE DI ADDESTRAMENTO

2.2 Diagrammi delle classi

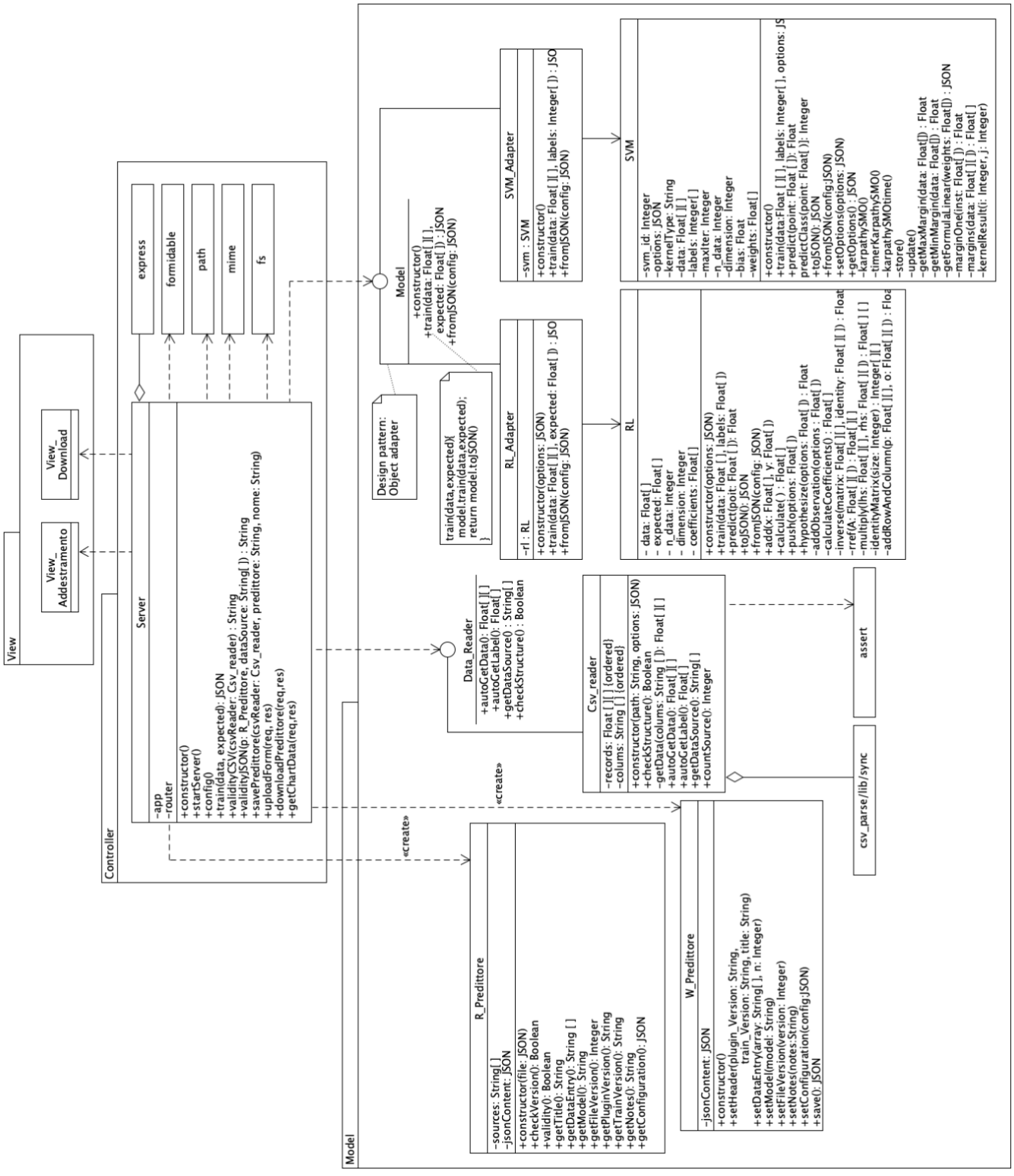

FIGURA 2: DIAGRAMMA DELLE CLASSI DI ADDESTRAMENTO

2.3 Diagrammi di sequenza

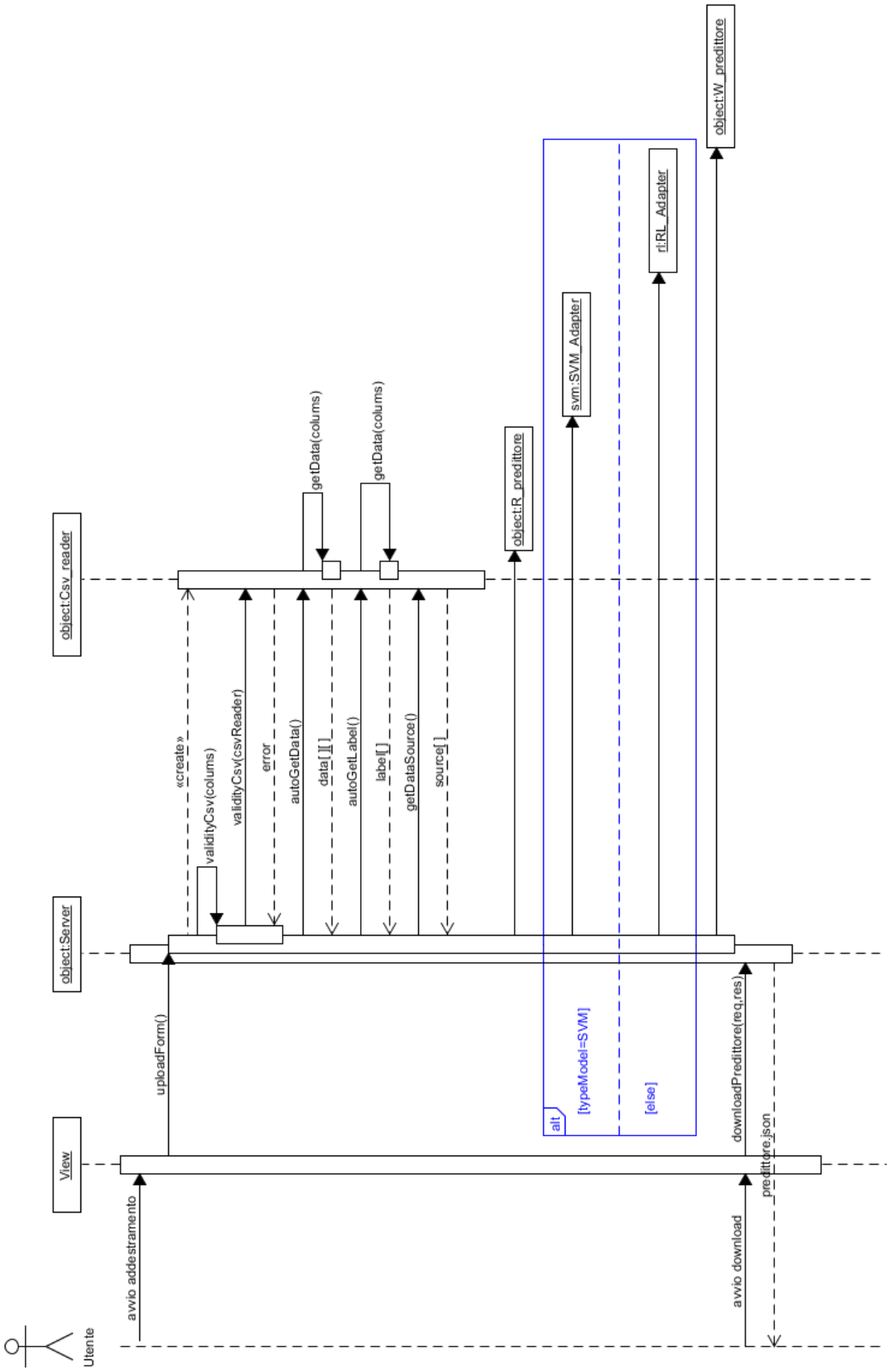

FIGURA 3: DIAGRAMMA DI SEQUENZA CARICAMENTO CSV

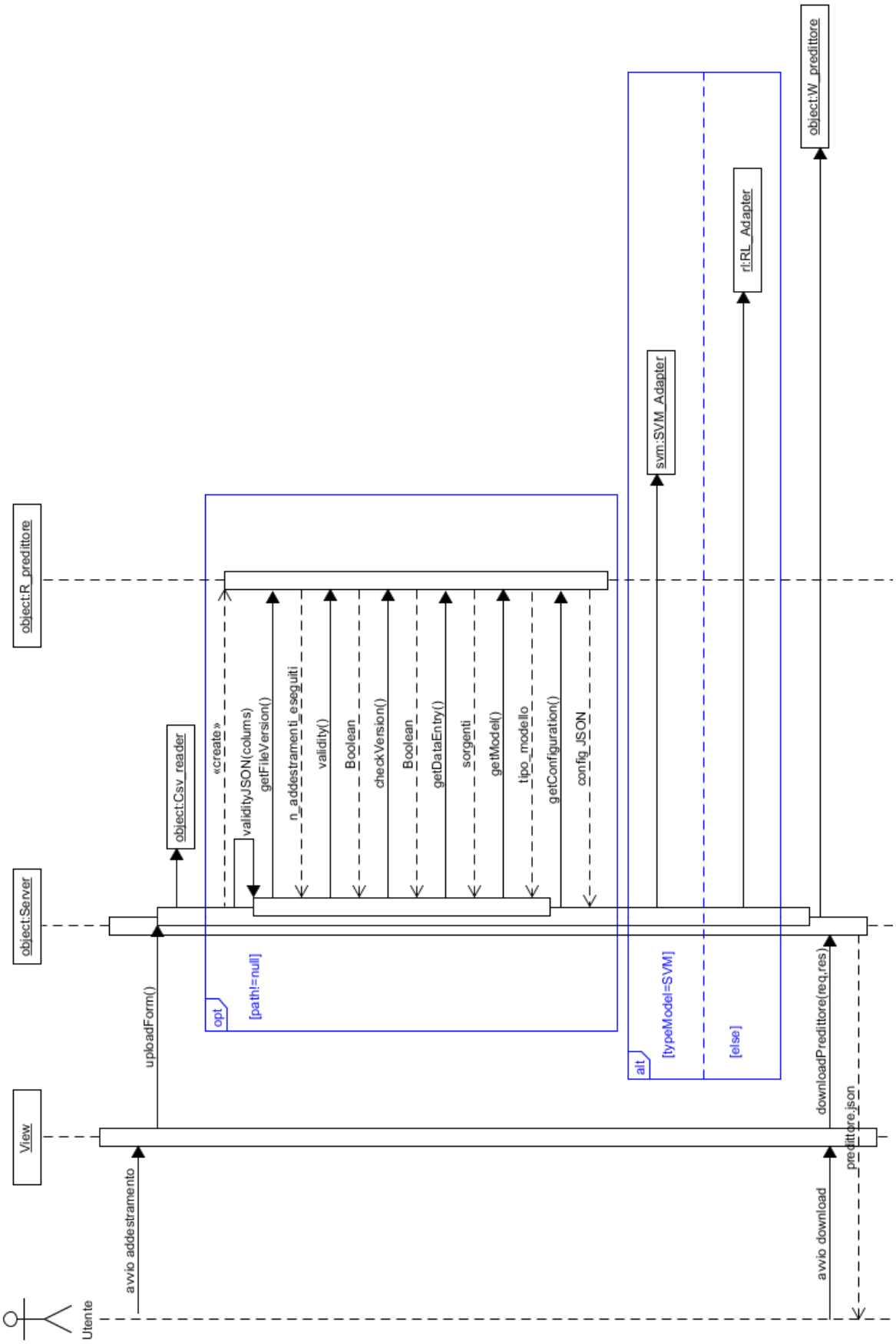

FIGURA 4: DIAGRAMMA DI SEQUENZA CARICAMENTO JSON

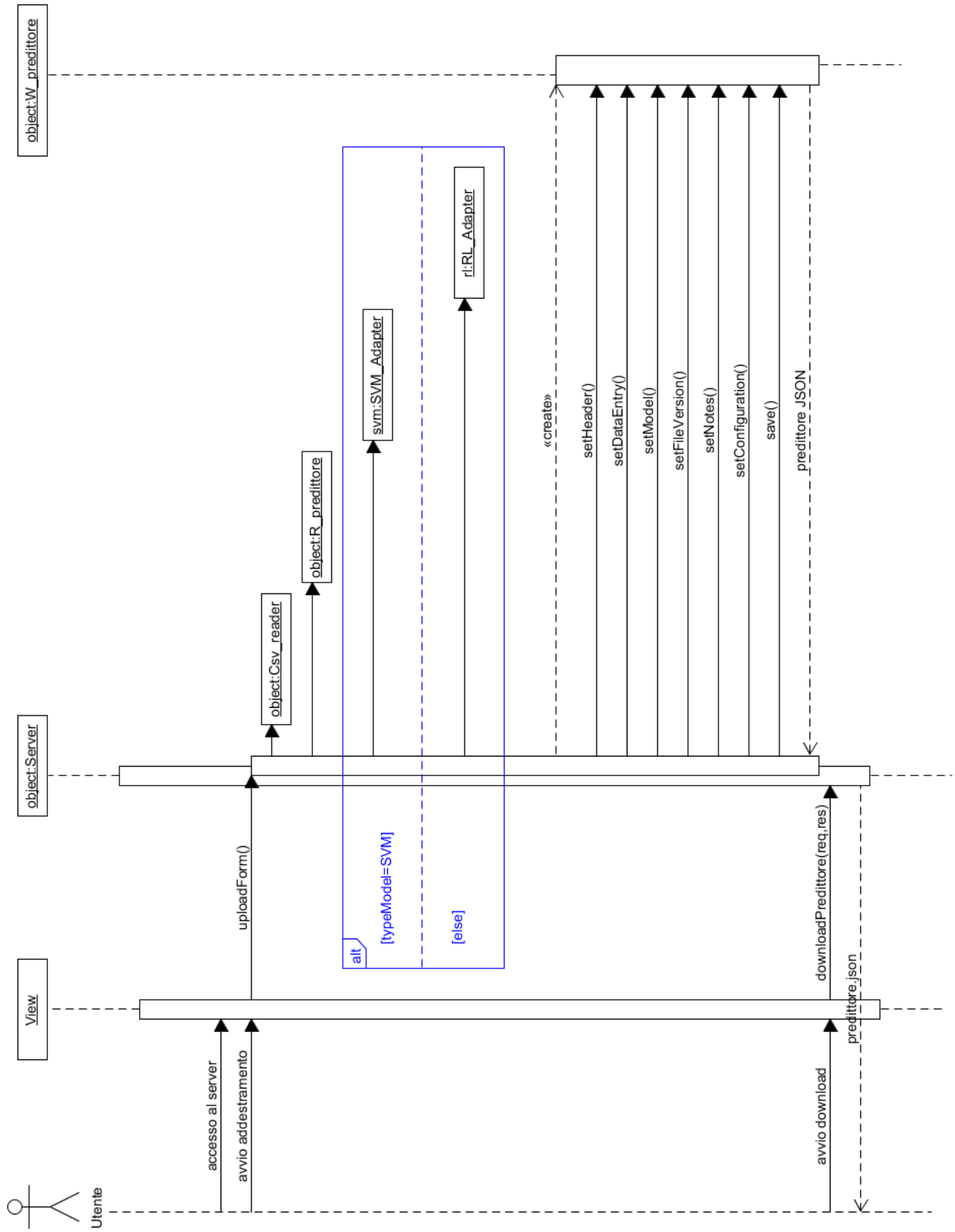

FIGURA 5: DIAGRAMMA DI SEQUENZA SALVATAGGIO ADDESTRAMENTO

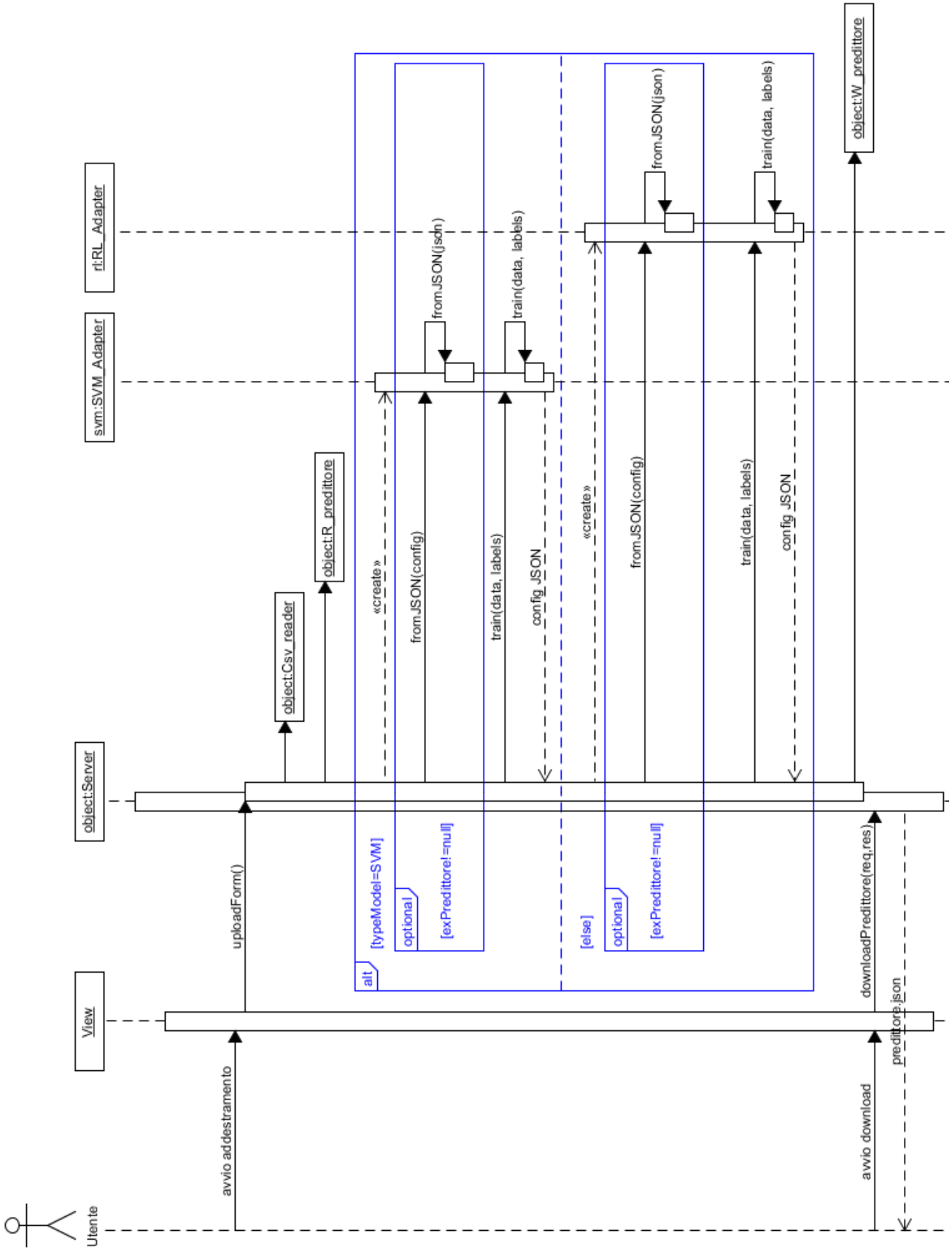

FIGURA 6: DIAGRAMMA DI SEQUENZA ADDESTRAMENTO

3 Plug-in

3.1 Diagrammi dei package

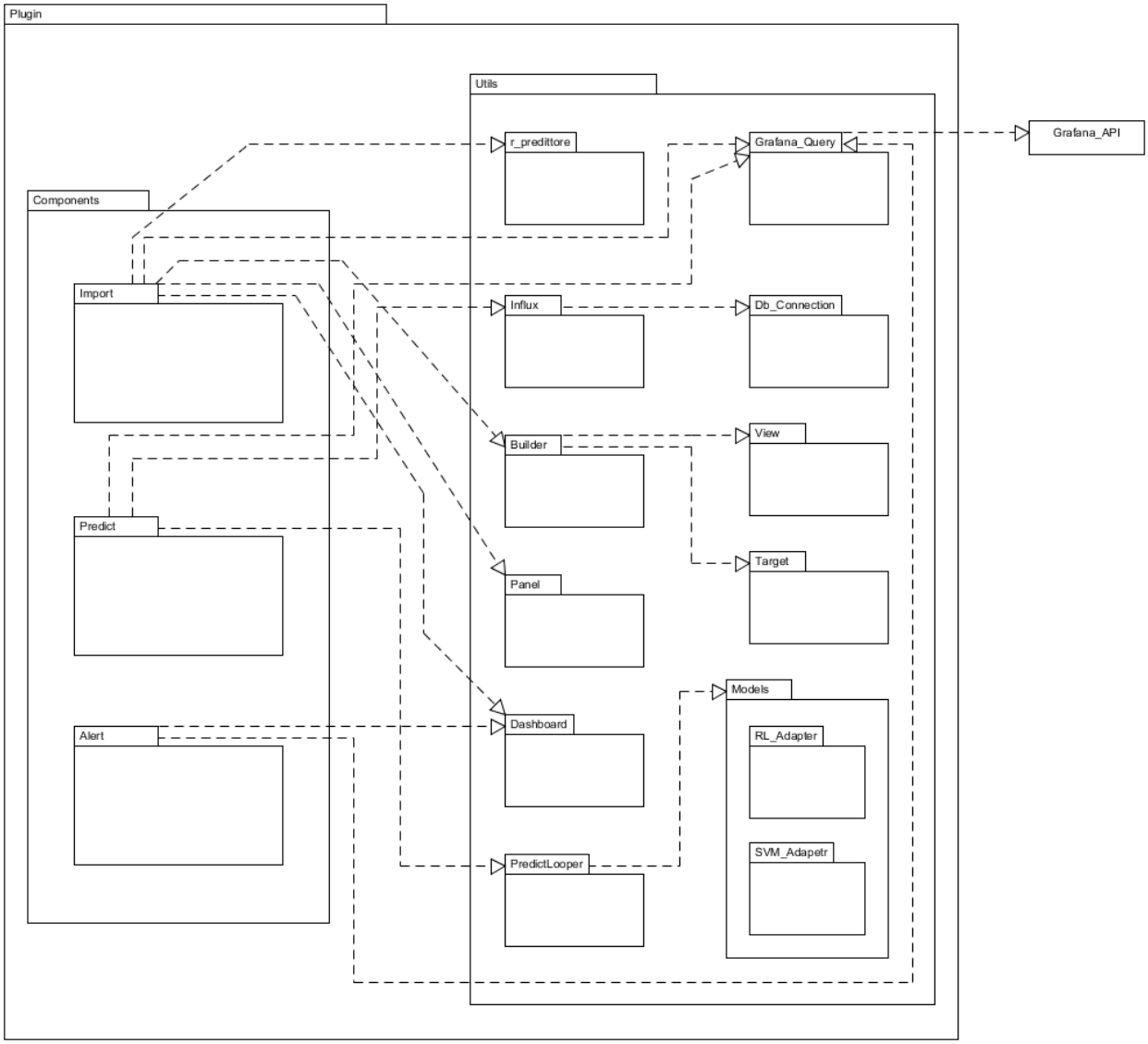

FIGURA 7: DIAGRAMMA DI PACKAGE DI PLUGIN

3.2 Diagrammi delle classi

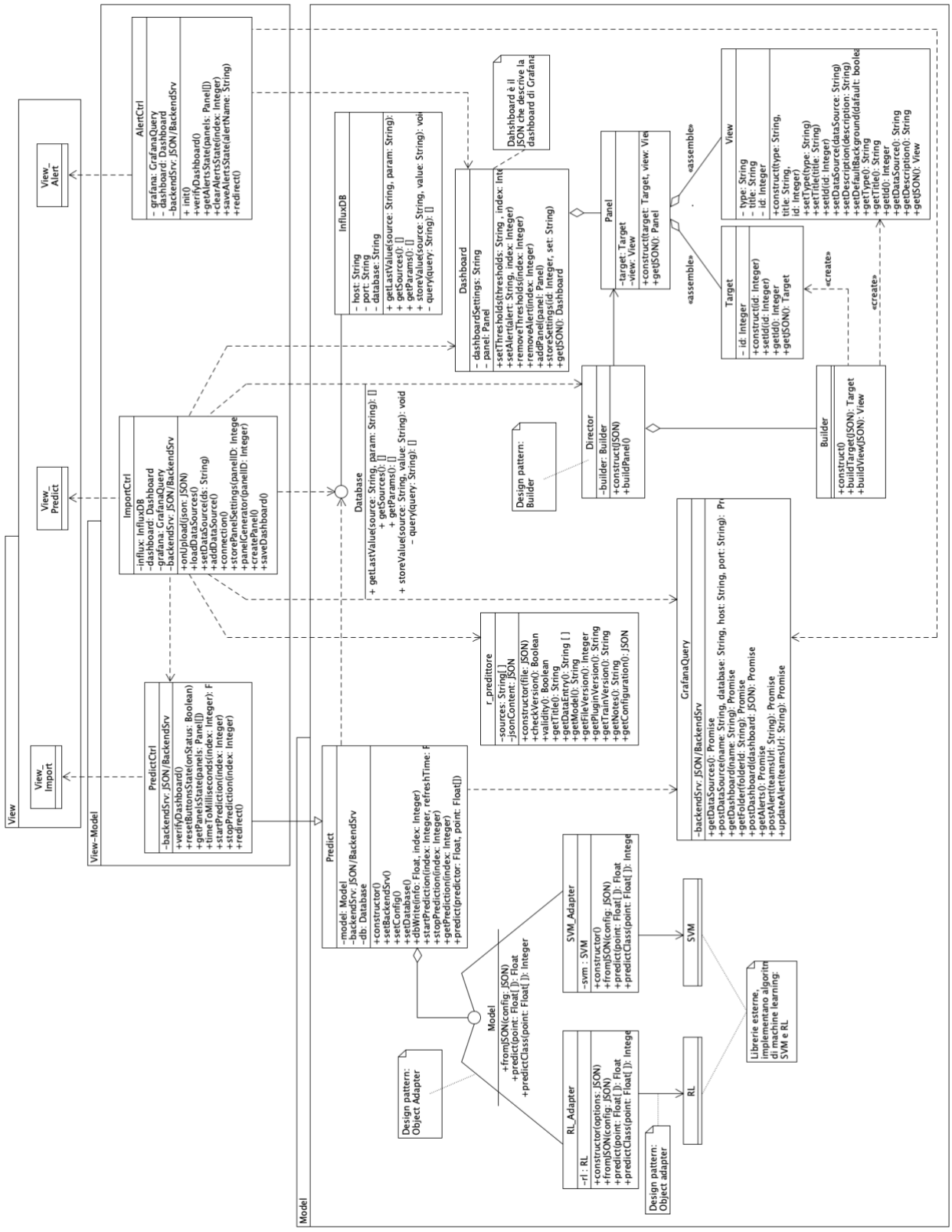

FIGURA 8: DIAGRAMMA DELLE CLASSI DI PLUGIN

3.3 Diagrammi di sequenza

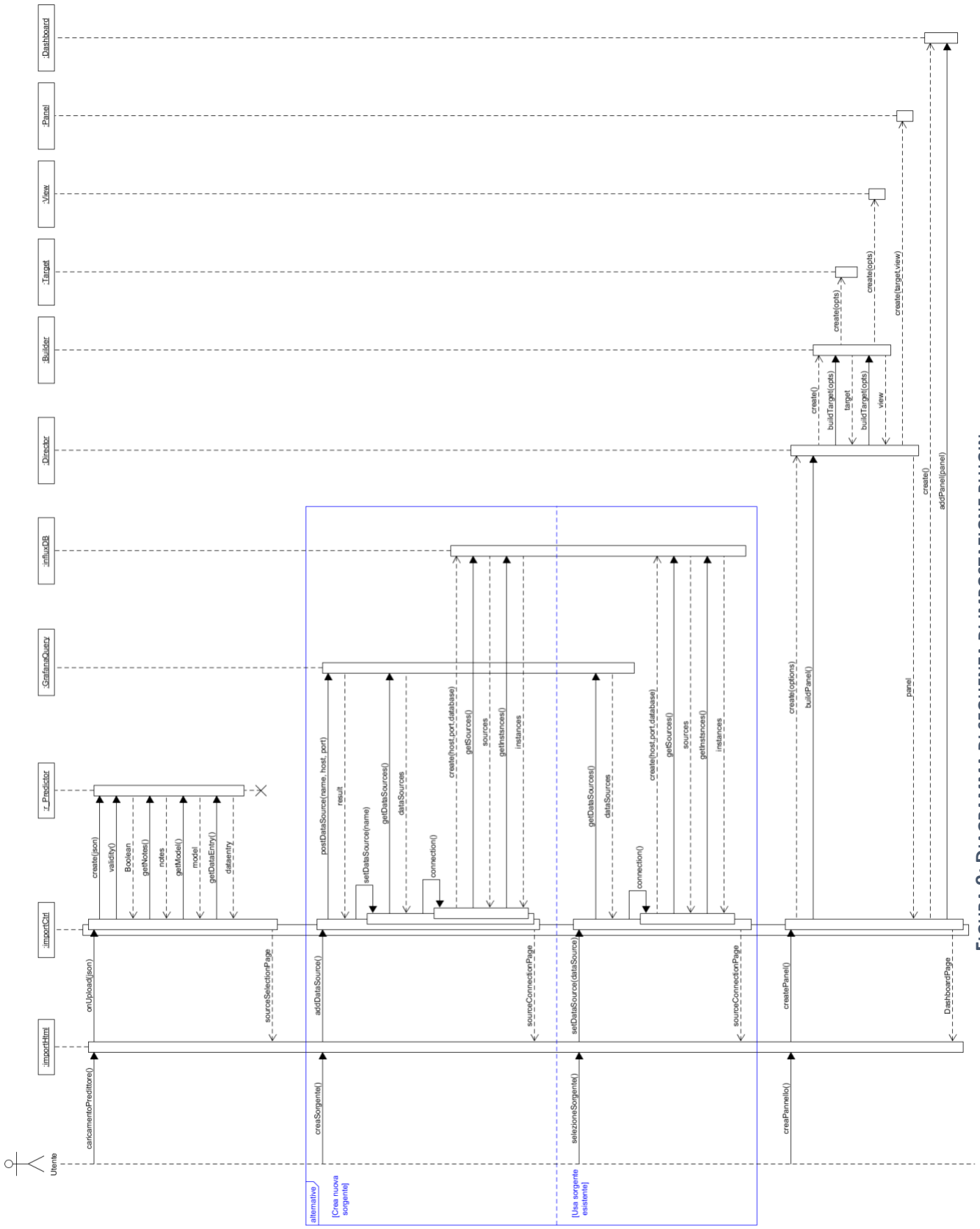

FIGURA 9: DIAGRAMMA DI SEQUENZA DI IMPOSTAZIONE PLUGIN

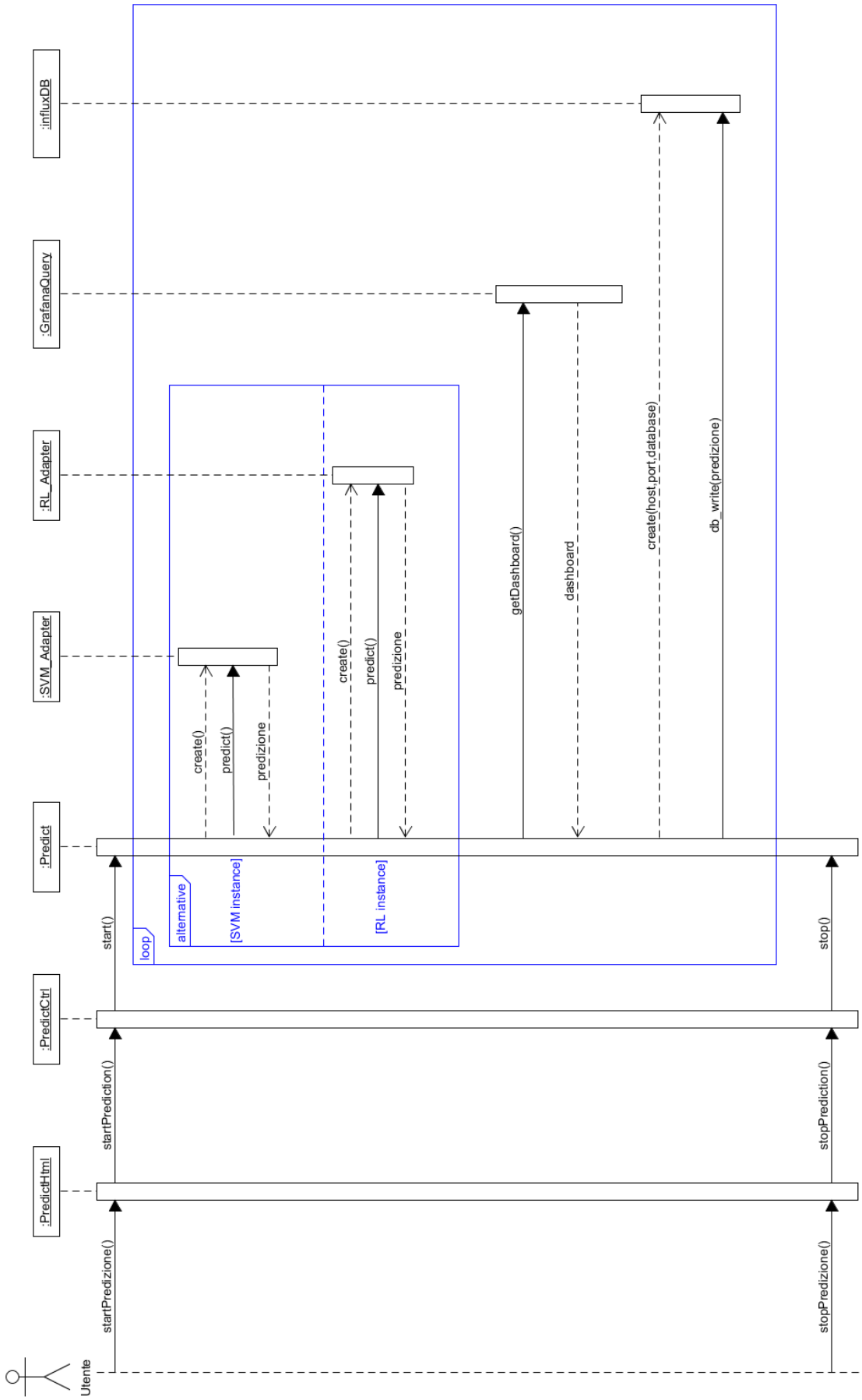

FIGURA 10: DIAGRAMMA DI SEQUENZA DI PREDIZIONE DI PLUGIN

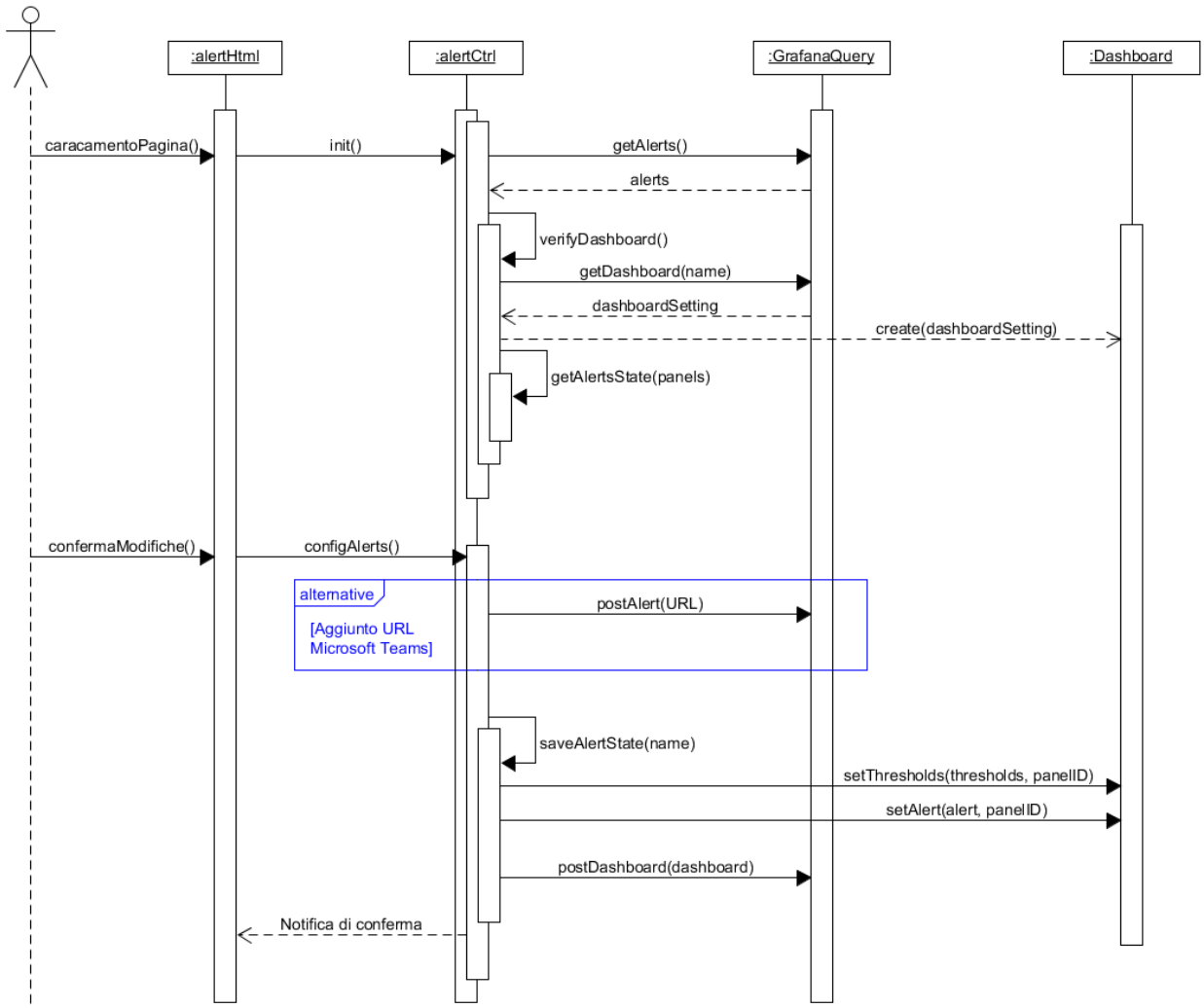

FIGURA 11 - DIAGRAMMA DI SEQUENZA DI ALERT PLUG-IN