



Manuele sviluppatore

Progetto Ingegneria Del Software

Versione: 1.0.0

Albertin Enrico
Davide Spada
Bettin Michele

Marcatti Pietro
Marco Andrea Limongelli
Matteo Raccanello

Dipartimento di Matematica
Università degli Studi di Padova

16 maggio 2022



Registro delle Modifiche

| Versione | Modifica | Ruolo | Esecutore | Data |
|----------|--|------------------------|---|----------|
| 1.0.0 | Approvazione del documento | Responsabile | Marcatti Pietro | 28/04/22 |
| 0.1.0 | Verifica complessiva | Verificatore | Spada Davide | 24/04/22 |
| 0.0.5 | Stesura sezione 4 Architettura del prodotto da 4.3 a 4.5 e verifica | Analista, Verificatore | Raccanello Matteo, Bettin Michele | 23/04/22 |
| 0.0.4 | Stesura sezioni 2 Tecnologie utilizzate e 3 Installazione e verifica | Analista, Verificatore | Raccanello Matteo, Bettin Michele | 21/04/22 |
| 0.0.3 | Stesura sezione 4 Architettura del prodotto da 4.1 e 4.2 e verifica | Analista, Verificatore | Limongelli Marco, Raccanello Matteo, Andrea | 19/04/22 |
| 0.0.2 | Stesura sezione 1 Introduzione e verifica | Analista, Verificatore | Raccanello Matteo, Spada Davide | 12/04/22 |
| 0.0.1 | Stesura iniziale dello scheletro del documento e verifica | Analista, Verificatore | Raccanello Matteo, Spada Davide | 10/04/22 |

Indice

| | | |
|----------|--|----------|
| 1 | Introduzione | 4 |
| 1.1 | Scopo del documento | 4 |
| 1.2 | Scopo del prodotto | 4 |
| 1.3 | Glossario | 4 |
| 1.4 | Riferimenti | 4 |
| 2 | Tecnologie utilizzate | 5 |
| 3 | Installazione | 6 |
| 3.1 | Requisiti minimi sistema | 6 |
| 3.1.1 | Requisiti Software | 6 |
| 3.1.2 | Requisiti Hardware | 6 |
| 3.1.3 | Browser | 6 |
| 3.2 | Installazione | 6 |
| 4 | Architettura del prodotto | 8 |
| 4.1 | Design patterns | 8 |
| 4.2 | Descrizione generale | 8 |
| 4.2.1 | Model-View-ViewModel | 8 |
| 4.2.2 | Strategy | 8 |
| 4.2.3 | Observer | 8 |
| 4.3 | Diagramma dei package | 9 |
| 4.4 | Diagramma delle classi | 10 |
| 4.5 | Diagrammi di sequenza | 11 |
| 4.5.1 | Visualizzazione Scatter Plot | 11 |
| 4.5.2 | Caricamento CSV | 12 |
| 4.5.3 | Riduzione dimensionale | 13 |

1 Introduzione

1.1 Scopo del documento

Lo scopo del seguente documento è quello di andare a predisporre uno strumento che fissi le principali scelte di design e codifica del prodotto Login Warrior in modo da rendere più semplice l'espansione (attraverso nuove feature) e la manutenzione al rilascio. Questa necessità deriva dal fatto che non vi è totale certezza che il team che prenderà di nuovo in mano il prodotto sia lo stesso che l'ha progettato, perciò tale documento fornirà grande valore aggiunto alla futura manutenzione dell'applicativo.

1.2 Scopo del prodotto

Lo scopo del prodotto Login Warrior è quello di andare a mettere a disposizione uno strumento a supporto degli analisti del dato. Permette infatti di andare a visualizzare dataset contenenti dati multidimensionali attraverso diverse tipologie di grafico opportunamente personalizzabili. L'applicazione inoltre permette di eseguire un'operazione di riduzione dimensionale per semplificare la comprensione dei dati e il riconoscimento di anomalie.

1.3 Glossario

Al fine di minimizzare le ambiguità il team ha prodotto **Glossario 2.0.0** che raccoglie termini di particolare importanza o ai quali è assegnato un significato particolare individuati nel testo da '*' ad apice.

1.4 Riferimenti

- **Model-View patterns** - Materiale didattico del corso di Ingegneria del Software
- **Diagrammi dei package** - Materiale didattico del corso di Ingegneria del Software
- **Diagrammi di sequenza** - Materiale didattico del corso di Ingegneria del Software
- **Diagrammi delle classi** - Materiale didattico del corso di Ingegneria del Software
- *Norme di progetto 3.0.0*

2 Tecnologie utilizzate

| Tecnologia | Versione | Descrizione |
|------------|----------|---|
| HTML | 5 | Per lo sviluppo dell'interfaccia front end dell'applicazione. |
| CSS | 3 | Usato per definire lo stile grafico dell'interfaccia front end. |
| Javascript | ES2015 | Come richiesto dal committente, usato per lo sviluppo dell'applicazione. |
| React | 18 | Libreria JavaScript per lo sviluppo front end di interfacce utente. |
| D3.js | 7.3.0 | Come richiesto dal committente è stata usata la libreria D3.js per la rappresentazione dinamica dei dati attraverso grafici personalizzabili. |
| Druid.js | | Libreria JavaScript che mette a disposizione gli algoritmi di riduzione che verranno utilizzati per il progetto. |
| Papa Parse | 5.3.2 | Utilizzata per eseguire il parsing di file in formato CSV in JSON. |

3 Installazione

Nella seguente sezione sono presenti i requisiti minimi per l'utilizzo dell'applicativo Login Warrior e verrà esposta la modalità di installazione dell'applicazione per l'utilizzo o per la manutenzione.

3.1 Requisiti minimi sistema

3.1.1 Requisiti Software

I software necessari per l'esecuzione della web app sono:

- Node.js v16.14.0 LTS
- Npm v8.0.0

3.1.2 Requisiti Hardware

Non vi sono particolari requisiti minimi per l'utilizzo del software. Vi è da notare comunque che più il file da caricare è grande, maggiori saranno le risorse richieste.

3.1.3 Browser

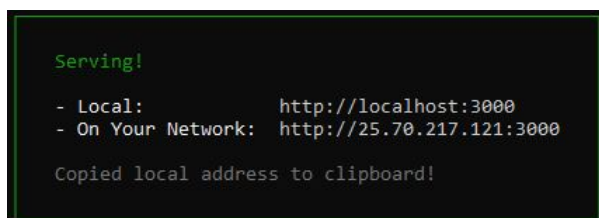
Secondo quanto definito nell'analisi dei requisiti, i browser compatibili con l'applicativo e su cui è stato testato sono:

- Safari dalla v15
- Mozilla Firefox dalla v91.7.0
- Google Chrome dalla v87
- Microsoft Edge dalla v79
- Tutti i browser basati su Chromium

3.2 Installazione

Per poter utilizzare la web app Login Warrior è necessario seguire i seguenti passaggi:

- Per prima cosa scaricare dal branch di release nel repository* su Github la cartella che contiene l'applicativo. Il link per poterla scaricare è il seguente:
<https://github.com/PietroMarcatti/CodeSix/releases>
- Dopo aver estratto il file in formato .zip è necessario avviare il web server dell'applicazione. Per fare ciò bisogna entrare all'interno della cartella appena estratta aprire il terminale ed eseguire il comando:
serve -s build
- Infine dopo avere eseguito il precedente passaggio verrà avviato il server come è possibile vedere in figura:



```
Serving!  
- Local:          http://localhost:3000  
- On Your Network: http://25.70.217.121:3000  
  
Copied local address to clipboard!
```

Sarà necessario ora accedere a uno dei due indirizzi tramite un browser compatibile (elencati nella sezione precedente)e verrà aperta la pagina iniziale della web app.

4 Architettura del prodotto

4.1 Design patterns

Sono di seguito presentati i design patterns implementati nel prodotto software.

- Model-View-ViewModel*
- Strategy*
- Observer*

4.2 Descrizione generale

4.2.1 Model-View-ViewModel

Il pattern architetturale al quale il gruppo CodeSix ha scelto di attenersi è il *Model-View-ViewModel**. Questa particolare tipologia di pattern permette di scrivere codice facilmente manutenibile e riutilizzabile grazie al forte disaccoppiamento tra la logica di presentazione (UI) e di business. Inoltre questo pattern è risultato essere il migliore per essere utilizzato in React, libreria per la creazione di interfacce utente single-page che rende-
rizza le componenti in base al loro stato interno ed unicamente ai cambiamenti che avvengono su queste.

Per permettere il passaggio dei dati dal Model alle varie componenti grafiche è stato utilizzato il Context React, al quale viene passato un'istanza del ViewModel (che nel nostro caso corrisponde al RootStore). Questo permette di accedere al valore del ViewModel in qualsiasi parte della View evitando di passare i dati di componente in componente. Ciò avverrebbe altrimenti attraverso le props, cioè gli argomenti della vista, che verrebbero passati fino a livelli potenzialmente profondi della gerarchia. Per evitare questo viene creata un'istanza del ViewModel passata ad un Context.Provider, ossia il contenitore della View. All'interno di questo ogni componente può utilizzare un hook per accedere al Context React ed utilizzare il valore più recente del ViewModel. Il Context React evita quindi di passare i dati per molti componenti rischiando, nel caso peggiore, di doverli utilizzare nell'ultimo livello della gerarchia.

4.2.2 Strategy

Per quanto riguarda il processo di riduzione dimensionale viene utilizzato il design pattern strategy*. La scelta di implementare questa tipologia di design pattern deriva dall'osservazione secondo la quale l'unica principale differenza era determinata dalla tipologia di algoritmo applicato ai fini della riduzione. Esso permette quindi di definire una famiglia di algoritmi e isolarli all'interno di un oggetto che ha permesso di renderli interscambiabili dinamicamente ed evitare duplicazione di codice permettendo quindi l'integrazione di ulteriori algoritmi di riduzione dimensionale derivando nuove classi concrete dall'interfaccia esposta. Nel nostro caso abbiamo:

- **DimReducer.js**: è la classe concreta che invoca la ConcreteStrategy richiesta del client;
- **ReductionAlgorithm.js**: è l'interfaccia comune a tutti gli algoritmi ed utilizzata da DimReducer per impostarne i parametri ed invocarli;
- **tSNEAlgorithm.js, UMAPAlgorithm.js**: rappresentano gli algoritmi concreti di riduzione dimensionale ed espongono l'interfaccia comune.

4.2.3 Observer

Inoltre, per poter fare in modo che una componente della View si renderizzi non solo al cambiamento del suo stato interno ma anche al cambiamento dei dati nel Model, abbiamo utilizzato la libreria *Mobx**. Questa

permette di implementare l'*observer pattern** la quale fornisce la possibilità di segnare delle classi (o attributi di esse) come "observable" e di costruire dei componenti della View come "observer". Questi ultimi vengono automaticamente ri-renderizzati al cambiamento di un qualsiasi attributo observable.

4.3 Diagramma dei package

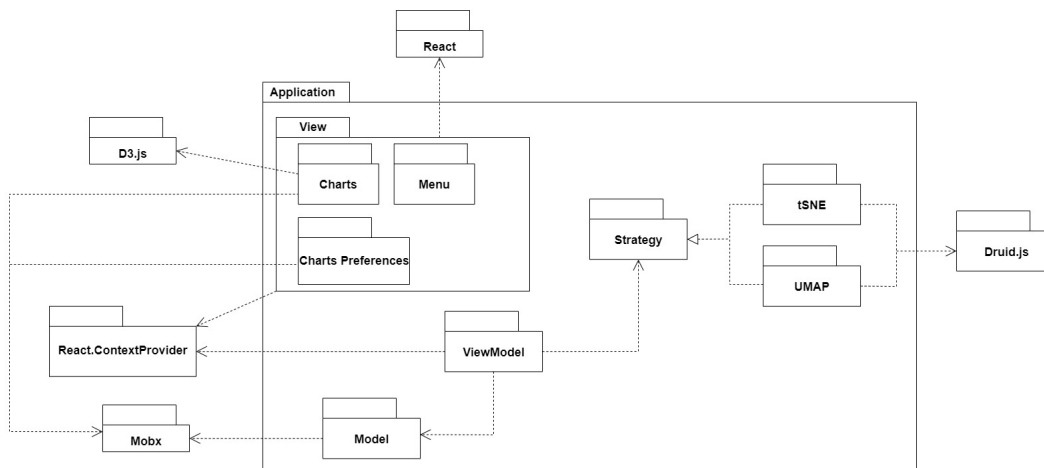


Figura 1: Diagramma dei packages

4.4 Diagramma delle classi

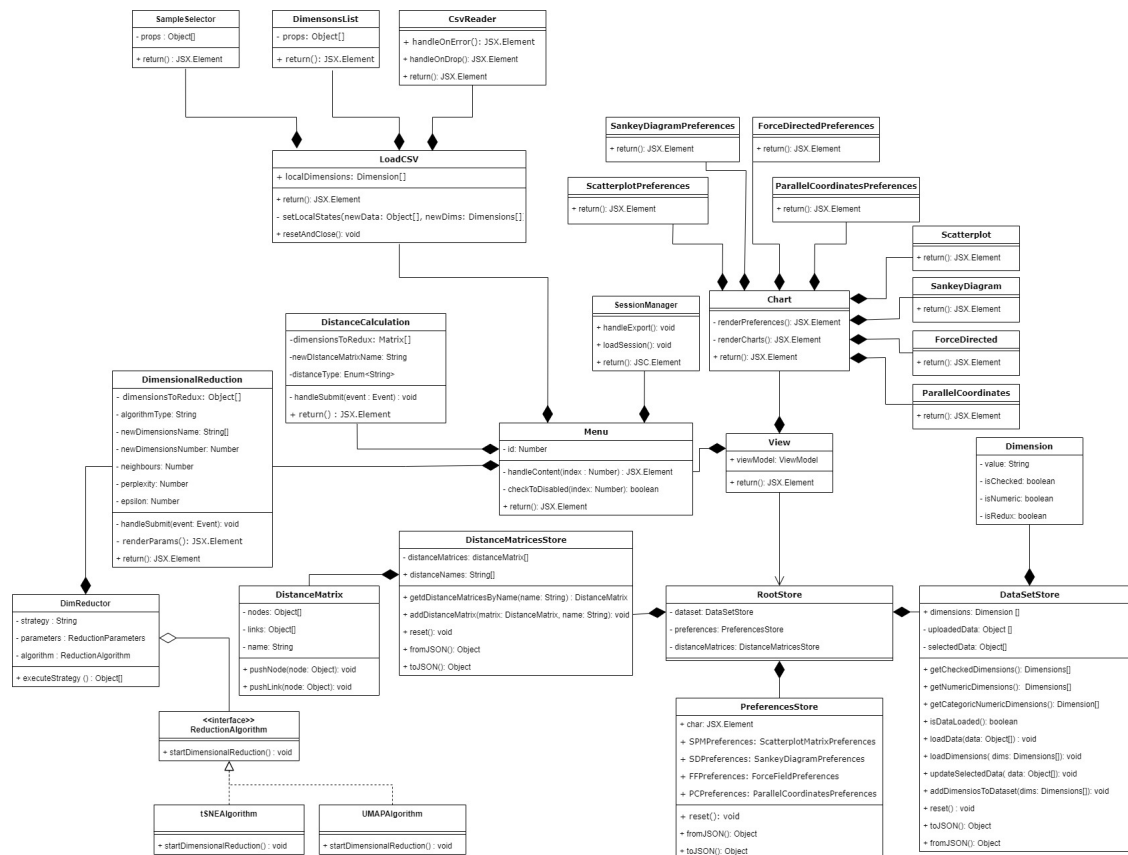


Figura 2: Diagramma delle classi

In questa sezione verranno definite le classi che compongono l'architettura dell'applicazione e saranno descritte le relazioni tra di esse attraverso il diagramma delle classi in figura. Come detto in precedenza è stato adottato il pattern architetturale Model-View-ViewModel per permettere al modello, alla logica e alla vista di essere indipendenti. Analizzando più attentamente il diagramma delle classi è possibile individuare due "sezioni" principali:

- **View** composta dal Menu e da una classe Graph. Il primo permette all'utente di avere accesso alle varie funzionalità dell'applicazione: il caricamento del dataset con le informazioni da visualizzare (CSVLoader), le funzionalità di caricamento e salvataggio della sessione (Handle Sessione) e l'operazione di riduzione dimensionale (Distance Reduction). Inoltre è presente la classe Distance Calculation utilizzata per la rappresentazione grafica Force Directed. La classe Graph gestisce invece la visualizzazione dei grafici selezionati dall'utente. Al suo interno contiene un attributo che specifica la tipologia di grafico selezionato (RenderGraph) e le personalizzazioni (RenderPreferences). Oltre alla visualizzazione del grafico gestisce anche la finestra di personalizzazione dello stesso.
- La parte del modello invece è composta dalla classe RootStore che è legata da una relazione di aggregazione alle rispettive classi (di cui dispone al suo interno anche un riferimento):
 - **DataSetStore**: è la classe utilizzata per la gestione del caricamento del dataset e per il salvataggio dei dati sui login. Mette inoltre a disposizione dei metodi che permettono di rendere migliore la

user experience come per esempio la possibilità di sapere se è già presente un dataset caricato (`isDataLoaded`);

- `PreferencesStore`: è la classe che permette di salvare le specifiche di personalizzazione dei vari grafici. In questo modo quando l'utente ricaricherà una precedente sessione di lavoro non dovrà reimpostare tali specifiche nuovamente;
- `DistanceMatricesStore`: utilizzata dal grafico Force Directed per il salvataggio della matrice delle distanze.

4.5 Diagrammi di sequenza

Nella seguente sezione verranno descritte attraverso l'utilizzo di diagrammi di sequenza alcune funzionalità del prodotto.

4.5.1 Visualizzazione Scatter Plot

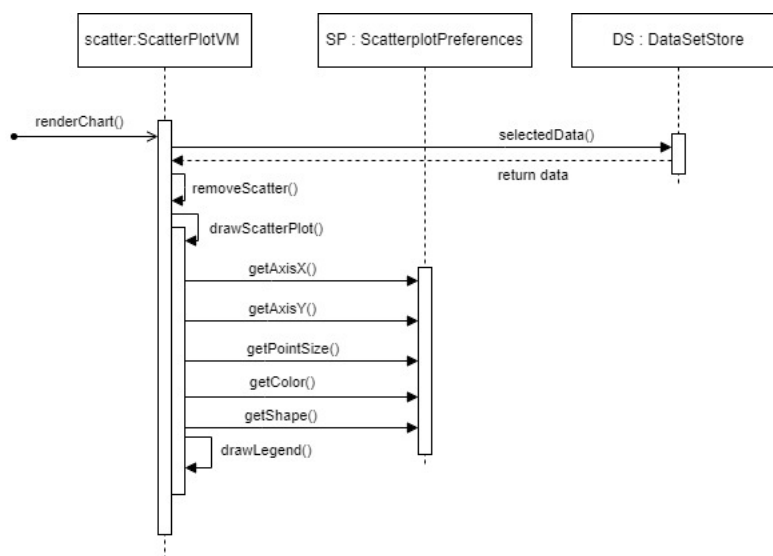


Figura 3: Diagramma dei package visualizzazione Scatter Plot

Un'istanza dello `ScatterPlot` richiede tramite un messaggio sincrono al `DataSetStore` di inviargli i dati precedentemente caricati e rimane in attesa fintanto che non li riceve. Successivamente `scatter` chiama su se stesso il metodo `removeScatter` nell'eventualità sia già stato rappresentato un grafico in precedenza. Infine con la chiamata di `drawScatterPlot` vengono eseguite le richieste delle personalizzazione selezionate dall'utente attraverso dei messaggi (metodi `getter`) all'istanza di `ScatterPlotPreferences` e viene visualizzata la legenda (metodo `drawLegend`).

4.5.2 Caricamento CSV

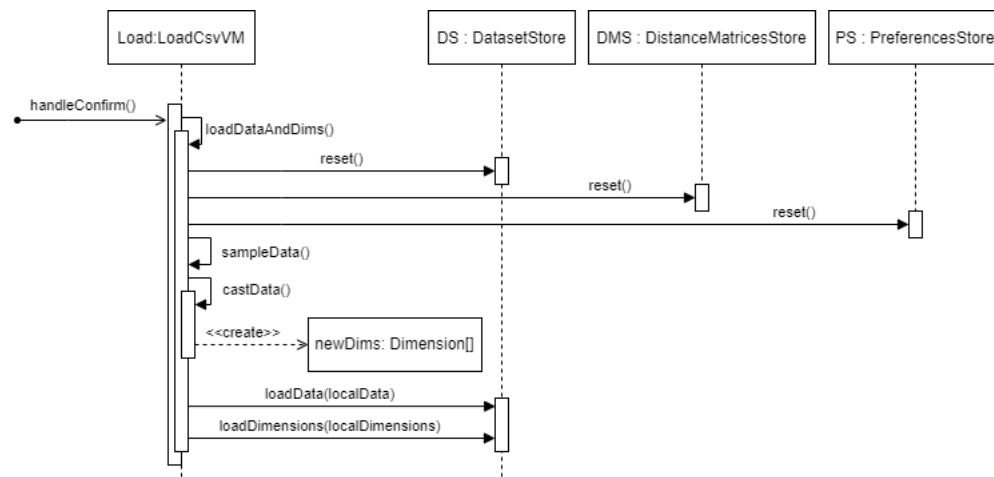


Figura 4: Diagramma dei package caricamento CSV

A seguito del caricamento del CSV con i dati per prima cosa vengono eliminati i dati precedentemente memorizzati in DataSetStore, DistanceMatricesStore e PreferencesStore tramite un messaggio di delete. Successivamente viene eseguito il cast dei dati caricati (castData) e viene creato un array con le nuove dimensioni del dataset. Infine vengono caricati i nuovi dati e le dimensioni sul DataSetStore.

4.5.3 Riduzione dimensionale

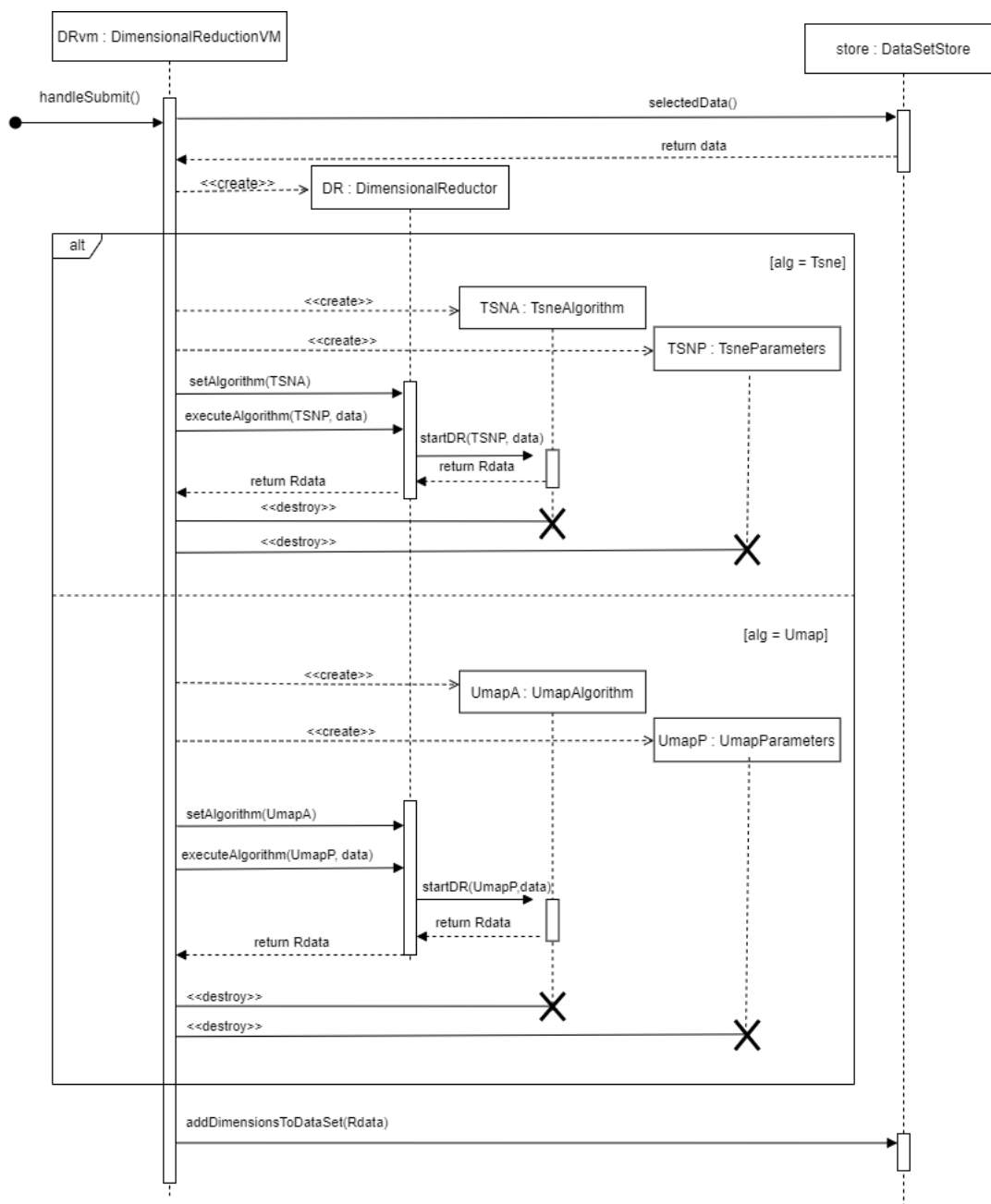


Figura 5: Diagramma dei package riduzione dimensionale

Il seguente diagramma descrive l'operazione di riduzione sui dati a seguito del caricamento del dataset e la selezione di tale funzionalità da parte dell'utente. Appena viene eseguito il submit l'istanza di DimensionalReduction invia un messaggio sincrono a DataSetStore per richiedere i dati caricati in precedenza e resta in attesa di risposta. Successivamente DimensionalReduction crea attraverso un messaggio di create un'i-

stanza di DimensionalRedactor. A questo punto a seconda della scelta effettuata dall'utente verrà utilizzato un diverso algoritmo per eseguire la riduzione dimensionale sui dati:

- Tsne Algorithm*
- Umap Algorithm*

In ogni caso le operazioni eseguite saranno le stesse ad eccezione dell'algoritmo di riduzione utilizzato. Per questo motivo tali azioni saranno descritte una sola volta. Per prima cosa vengono creati tramite un messaggio di create un'istanza di Algorithm e una Parameters (sia per Tsne che per Umap). Successivamente viene inviato un messaggio a DimensionalRedactor coi metodi per settare l'algoritmo selezionato, i parametri e passare i dati. A questo punto da DimensionalRedactor invoca il metodo startDR su Algorithm per eseguire la riduzione dimensionale e rimane in attesa di ricevere i dati ottenuti dopo l'operazione per poi ritomarli a sua volta a DimensionalRedaction. Terminata l'operazione vengono eliminati Algorithm e Parameters tramite un messaggio di delete e infine i nuovi dati ottenuti con la riduzione dimensionale vengono salvati nel DataSetStore.