# Manuale Sviluppatore

7DOS - 12 Aprile 2019

**Informazioni sul documento**

| | |
|---|---|
| **Versione** | 1.0.0 |
| **Responsabile** | Andrea Trevisin |
| **Verifica** | Lorenzo Busin<br>Nicolò Tartaggia |
| **Redazione** | Giacomo Barzon<br>Marco Costantino<br>Michele Roverato<br>Giovanni Sorice |
| **Stato** | Approvato |
| **Uso** | Esterno |
| **Destinato a** | Prof. Tullio Vardanega<br>Prof. Riccardo Cardin<br>7DOS |
| **Email** | 7dos.swe@gmail.com |

**Descrizione**

Questo documento contiene il manuale sviluppatore relativo al prodotto G&B del gruppo 7DOS.

# Diario delle modifiche

| Modifica | Autore | Ruolo | Data | Versione |
|---|---|---|---|---|
| *Approvazione del documento* | Andrea Trevisin | Responsabile | 2018-12-11 | 1.0.0 |
| *Verifica del documento* | Nicolò Tartaggia | Verificatore | 2018-12-10 | 0.7.0 |
| *Verifica del documento* | Lorenzo Busin | Verificatore | 2018-12-9 | 0.6.0 |
| *Stesura capitolato C4* | Giovanni Sorice | Analista | 2018-12-08 | 0.5.0 |
| *Stesura capitolati C1 e C2* | Giacomo Barzon | Analista | 2018-12-07 | 0.4.0 |
| *Stesura capitolato C3* | Giovanni Sorice | Analista | 2018-12-06 | 0.3.0 |
| *Stesura capitolato C6* | Michele Roverato | Analista | 2018-12-06 | 0.2.0 |
| *Stesura capitolato C5* | Marco Costantino | Analista | 2018-12-05 | 0.1.0 |
| *Stesura della sezione Introduzione* | Giovanni Sorice | Analista | 2018-12-05 | 0.0.2 |
| *Stesura dello scheletro del documento* | Giovanni Sorice | Analista | 2018-12-05 | 0.0.1 |

# Indice

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento rappresenta il Manuale dello Sviluppatore relativo al *prodotto software*$_g$ G&B sviluppato dal gruppo 7DOS. Il suo scopo principale è fornire tutte le informazioni necessarie per usufruire delle funzionalità fino ad ora implementate ed, eventualmente, per estendere e migliorare il prodotto.

## 1.2 Scopo del prodotto

Lo scopo del prodotto è la creazione di un *plug-in*$_g$ per il software di monitoraggio *Grafana*$_g$ in grado di collegare dati provenienti da una specifica *datasource*$_g$ ad un sistema probabilistico definito in una *rete Bayesiana*$_g$. In seguito, i risultati ottenuti vengono memorizzati in un database, nel nostro caso *influxDB*$_g$, e letti attraverso un *panel*$_g$ standard di Grafana. In questo modo è possibile evidenziare eventi non visibili ma con alta *likelihood*$_g$.

## 1.3 Riferimenti

### 1.3.1 Informativi

- **Rete Bayesiana**
  https://en.wikipedia.org/wiki/Bayesian_network

### 1.3.2 Installazione

- **TypeScript**
  https://www.typescriptlang.org/index.html#download-links
- **Node.js**
  https://nodejs.org/it/
- **Grafana**
  http://docs.grafana.org/installation/
- **Grafana Plug-in**
  http://docs.grafana.org/plugins/installation/
- **InfluxDB**
  https://portal.influxdata.com/downloads/
- **VSCode**
  https://code.visualstudio.com/
- **JetBrains WebStorm**
  https://www.jetbrains.com/webstorm/
- **JetBrains WebStorm for students**
  https://www.jetbrains.com/student/

### 1.3.3 Legali

- **Licenza MIT**
  https://opensource.org/licenses/MIT

## 1.4 Glossario

Per rendere la lettura del documento più semplice, chiara e comprensibile viene allegato il *Glossario_v3.0.0* nel quale sono contenute le definizioni dei termini tecnici, dei vocaboli ambigui, degli acronimi e delle abbreviazioni. La presenza di un termine all'interno del Glossario è segnalata con una 'g' posta come pedice (esempio: $Glossario_{g}$).

# 2 Requisiti di sistema

## 2.1 Requisiti hardware

Per poter utilizzare il plug-in e tutto ciò che permette la sua esecuzione sono presenti dei requisiti hardware da soddisfare:

- RAM
- Memoria interna
- Processore
- Altro?

## 2.2 Browser compatibili

Di seguito vengono riportate le versioni minime dei browser sui quali è garantito il funzionamento del nostro plug-in:

- Google Chrome v.73 (quella nel pc di tartizz)
- Mozilla Firefox v.66 (quella nel pc di tartizz)
- Safari v.
- Microsoft Internet Explorer v.11 (quella nel pc di tartizz)
- Microsoft Edge v.41 (quella nel pc di tartizz)

Affinché tutte le funzionalità offerte dal plug-in vengano eseguite correttamente, è necessario che *JavaScript*$_g$ sia abilitato.

# 3 Installazione

## 3.1 Requisiti

Nel file *package.json*$_g$ nella *root directory*$_g$ definisce tutte le dipendenze necessarie per l'esecuzione del plug-in. Esse, ad esempio *TypeScript*$_g$, *Node*$_g$ e *NPM*$_g$, vengono installati automaticamente attraverso il comando

```
npm install
```

## 3.2 Installazione Grafana

Per poter installare e utilizzare il plug-in è necessario scaricare e configurare precedentemente Grafana, al link URL http://docs.grafana.org/installation/.
Il plug-in è stato sviluppato utilizzando la versione 5.4.3. Per evitare errori nell'esecuzione, si raccomanda di avvalersi della stessa versione. Inoltre, il prodotto è stato testato con successo su versioni precedenti, tra le quali 5.x.x ecc. Una volta terminata la configurazione descritta al riferimento precedente, avviare l'eseguibile `grafana-5.x.x/bin/grafana-server.exe` . A questo punto, aprire una finestra del browser e spostarsi all'indirizzo URL `localhost:8080`. Sottolineiamo che la porta 8080 è quella suggerita nella pagina di installazione di Grafana.

## 3.3 Installazione plugin

## 3.4 Installazione e configurazione InfluxDB

# 4 Configurazione dell'ambiente di lavoro

## 4.1 Obbiettivo del capitolo

Il seguente capitolo si impone l'obbiettivo di spiegare al lettore come configurare il proprio ambiente di lavoro in modo tale che sia lo stesso dei membri del team 7DOS.

## 4.2 Requisiti

### 4.2.1 WebStorm

Per lo sviluppo del plug-in il team ha scelto di utilizzare l'*IDE*$_g$ WebStorm, sviluppato da JetBrains. L'IDE è a pagamento, tuttavia esso può essere scaricato gratuitamente dal loro sito ufficiale connettendo al proprio account un'e-mail universitaria. Il software è disponibile per Microsoft Windows, Linux e MacOs.

### 4.2.2 VSCode

VSCode è una valida alternativa a WebStorm, che il team consiglia di utilizzare nel caso in cui quest'ultimo non fosse reperibile. Il software può essere scaricato molto facilmente dal sito ufficiale ed è disponibile per Microsoft Windows, Linux e MacOs.

### 4.2.3 TSLint e ESLint

TSLint e ESLint verranno automaticamente installati con l'esecuzione del comando

```
npm install
```

Una volta installati correttamente WebStorm li rileverà automaticamente tra le dipendenze presenti all'interno del package.json e procederà a segnalare tutti gli errori relativi all'analisi statica rilevati senza la necessità di dover eseguire il comando

```
npm run build
```

# 5 Test

## 5.1 Obiettivo del capitolo

Questo capitolo descrive le procedure per eseguire i test sul funzionamento e la sintassi del codice.

## 5.2 Test sul codice TypeScript

Per avviare i test sul funzionamento del codice TypeScript, è sufficiente eseguire da terminale il comando:

```
npm run test
```

Il comando avvierà l'esecuzione di tutti i test presenti nella cartella `/src/test`. Per avviare i test sulla sintassi del codice è necessario lanciare i comandi:

```
npm run eslint
```

per verificare la sintassi JavaScript, e

```
npm run tslint
```

per verificare la sintassi TypeScript.
Alternativamente questi due comandi possono essere eseguiti assieme lanciando:

```
npm run eslint && npm run tslint
```

È importante notare che il comando congiunto non è compatibile con *Windows Powershell*.
Al termine verranno segnalate le parti di codice che non rispettano le linee guida.

## 5.3 Test sul codice HTML/CSS

Verrà fatto per la RA

## 5.4 Code Coverage

# 6 Architettura

All'interno della seguente sezione verrà descritta l'architettura dell'intero sistema. Per la realizzazione dell'architettura del plugin il team ha deciso di adottare uno stile architetturale a layer.
I principali layer che sono stati individuti nel corso dello sviluppo dell'applicazione sono i seguenti:

- **Presentation layer**: si occupa di gestire la componente grafica del panel;
- **Persistence layer**: si occupa di mantenere la struttura della rete;
- **Buisness layer**: incorpora tutta la logica relativa all'intero processo di ricalcolo delle probabilità;
- **Database layer**: composto dall'insieme di tutti i database esterni forniti dall'utente per il reperimento delle informazioni.

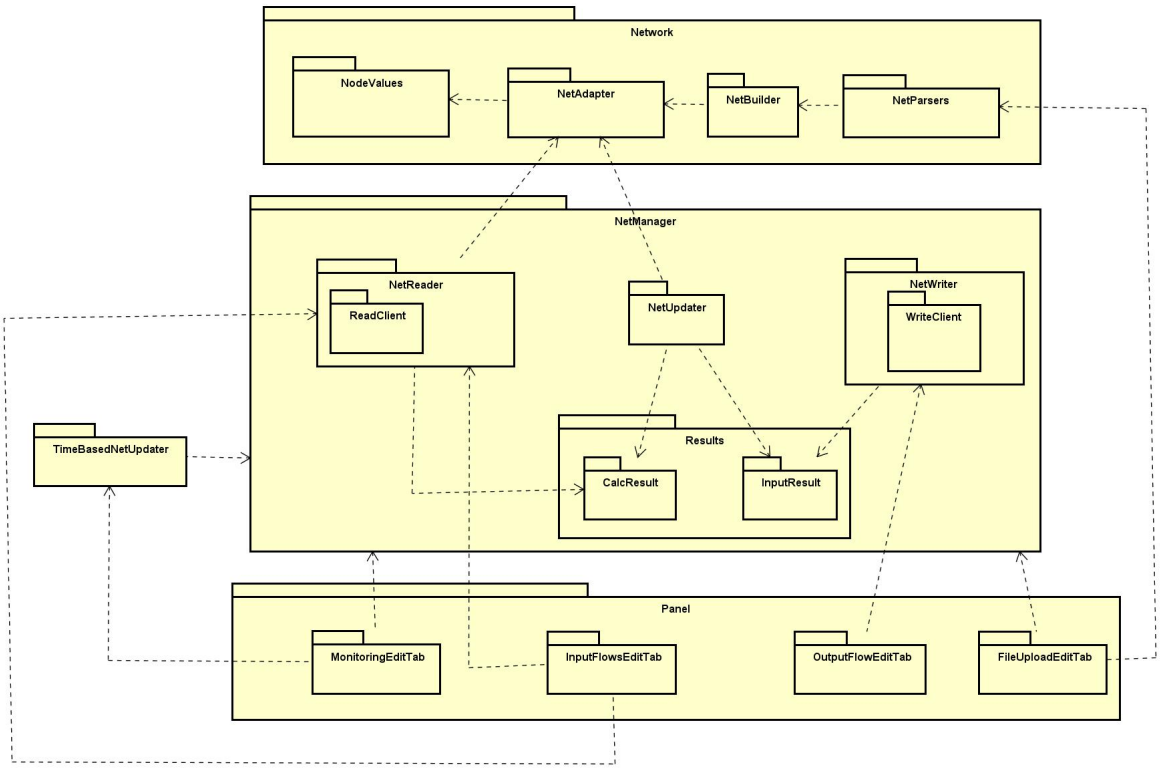


Figura 1: Diagramma dei package

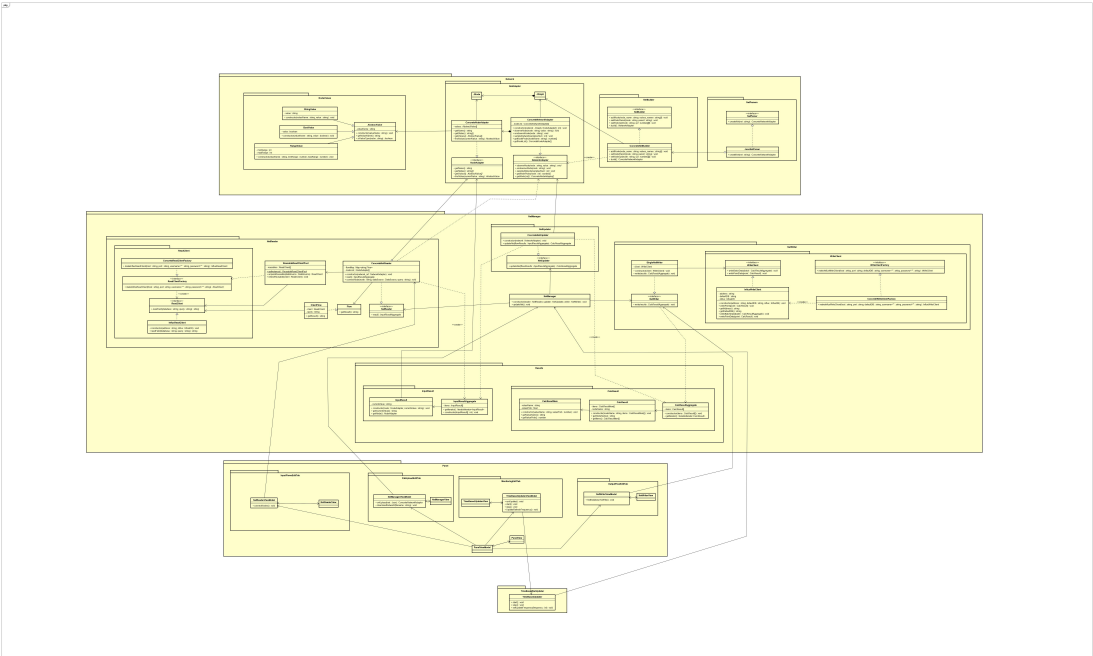

Figura 2: Diagramma delle classi complessivo

Analizziamo ora più in dettaglio ciascuno dei layer precedentemente descritti.

## 6.1 Persistence Layer

L'intera nostra applicazione è basata sulla libreria Javascript JSBayes consigliata dal proponente, la quale permette di mantenere l'intera struttura della rete ed esegue su di essa il calcolo delle probabilità di ogni stato associato ad ogni nodo della rete. Questa libreria tuttavia presenta non pochi problemi, tra i quali una pressochè totale assenza della gestione degli errori, una difficile interazione con altri oggetti, ed una scarsa estendibilità. Il persistence layer è composto principalmente da un insieme di classi che permettono di ovviare ai problemi precedentemente descritti.

### 6.1.1 NetworkAdapter

La classe NetworkAdapter ha il compito di schermare tutte le operazioni non sicure di JSBayes, cioè quelle che vanno a modificare la struttura della rete rischiando quindi di minare la potenziale integrità dei dati. Essa espone solamente i metodi necessari per effettuare il ricalcolo delle probabilità e che quindi non modificano in alcun modo lo stato.

Figura 3: Diagramma UML NetworkAdapter

### 6.1.2 NodeAdapter

Per lo sviluppo delle funzionalità del plugin il team ha avuto la necessità di aggiungere informazioni ai singoli nodi della rete bayesiana memorizzata da JSBayes. Estendere dalle

classi offerte dalla libreria era impensabile a causa dei molteplici problemi descritti all'inizio di questa sezione. Per questo motivo abbiamo deciso di realizzare un adapter anche per i singoli nodi di JSBayes ed includere all'interno di questi tutte le informazioni aggiuntive necessarie, come gli AbstractValue che verranno descritti in dettaglio all'interno della sezione seguente.

Figura 4: Diagramma UML NodeAdapter

### 6.1.3 AbstractValue

Per poter determinare in quale stato il nodo si trova sulla base dei dati prelevati da una qualsiasi fonte dati, come un database, il semplice nome che JSBayes associa ad ogni stato non era sufficiente.
Il compito degli AbstractValue è proprio quello di colmare questa lacuna. Invocando su un oggetto AbstractValue il metodo `isValueType(value:string):bool` e passandogli come parametro il risultato di una lettura da una qualsiasi fonte dati l'AbstractValue ritorna un boolean indicando all'invocatore se lo stato associato è attualmente vero o meno.
Il team ha realizzato solamente tre implementazioni concrete di abstract value, rispettivamente RangeValue, StringValue e BoolValue, tuttavia è possibile realizzare altre implementazioni semplicemente estendendo dall'interfaccia AbstractValue.

Figura 5: Diagramma UML AbstractValue

### 6.1.4 NetBuilder

Per garantire l'integrità dei dati abbiamo deciso di esternare il processo di creazione della rete JSBayes all'interno di un builder esterno il quale ha il compito di effettuare tutti i controlli necessari e ritornare un riferimento ad un NetworkAdapter solo al termine dell'intero processo di creazione invocando il metodo `build()` .

Figura 6: Diagramma UML NetBuilder

### 6.1.5 NetParser

Il NetParser ha il compito di realizzare un NetAdapter, sfruttando le funzionalità esposte dal NetBuilder, a partire dal contenuto di un file di testo. Il team fino ad ora ha realizzato un unica implementazione, quella relativa a file di tipo JSON in quanto era l'unico formato richiesto dal proponente. Nel caso in cui un giorno fosse necessario utilizzare formati differenti è sarà sufficiente realizzare una differente implementazione dell'interfaccia NetParser.

Figura 7: Diagramma UML NetParser

## 6.2 Buisness Layer

Il buisness layer ingloba all'interno di esso tutta la logica relativa al processo di calcolo delle probabilità. Dopo un attenta analisi del problema il team è riuscito a scomporre il suddetto processo in 3 macro-fasi principali:

- La lettura dei dati dai database associati ai nodi delle rete;
- Il processo di calcolo delle probabilità vero e proprio partendo dai dati letti precedentemente.
- La scrittura delle probabilitá appena calcolate su un database che l'utente poi potrá utilizzare per fare la visualizzazione dei dati su grafana

Ci è sembrato dunque abbastanza naturale modellare l'architettura del buisness layer in tre oggetti distinti ognuno dei quali si occupa di gestire una specifica fase del processo dall'inizio alla fine. Analizziamo ora più in dettaglio gli oggetti precedentemente citati.

### 6.2.1 NetReader

Il NetReader è collegato ad un ReadClient, di cui parleremo successivamente all'interno della sezione dedicata al database layer. NetReader è responsabile di collegare un nodo ad un flusso di dati e di richiedere a ReadClient di interrogare il database per ottenere i dati necessari. Per non dover istanziare più client di lettura per la stessa sorgente dati o database, abbiamo implementato una Object Pool per i client di lettura: quando si connette un nodo ad un flusso di dati, se il client richiesto esiste viene estratto dal pool, altrimenti viene creato ad-hoc.

Figura 8: Diagramma UML NetReader

### 6.2.2 NetUpdater

NetUpdater ha il compito di effettuare il ricalcolo utilizzando i dati letti da NetReader e le funzionalità esposte dal NetworkAdapter.

Figura 9: Diagramma UML NetUpdater

### 6.2.3 NetWriter

Figura 10: Diagramma UML NetWriter

### 6.2.4 NetManager

Il NetManager non è altro che un facade il quale ha un riferimento ad un istanza di NetReader, NetUpdater e NetWriter ciascuna. Esso ha il compito di coordinare l'intero processo di calcolo delle probabilità invocando sequenzialmente le operazioni giuste nell'ordine corretto.

Figura 11: Diagramma UML NetManager

#### 6.2.5 Results

## 6.3 Database Layer

## 6.4 Presentation Layer