



Agents of S.W.E.

A SOFTWARE COMPANY

Agents of S.W.E. - Progetto "G&B"

Manuale dello Sviluppatore

xx Versione	0.0.3
Approvazione	??
Redazione	Marco Chilese Luca Violato
Verifica	??
Stato	Work in Progress
Uso	Esterno
Destinato a	Agents of S.W.E. Prof. Tullio Vardanega Prof. Riccardo Cardin Zucchetti S.p.A.

agentsofswe@gmail.com



Registro delle Modifiche

Versione	Data	Ruolo	Autore	Descrizione
0.0.3	2019-04-09	Progettista	Marco Chilese	Stesura di §1, §1.3, §2, §3.1, §3, §3.2, §5.2, §5.5, §6.2, §6.3, §8
0.0.2	2019-04-06	Analista	Luca Violato	Aggiunte sezioni §2 e §6
0.0.1	2019-04-04	Progettista	Marco Chilese	Strutturazione del Documento. Prima stesura di §4.1, §7.1, §5.1, §5.3, §7.2, §7.3, §3.2

Tabella 1: Registro delle Modifiche



Indice

1	Introduzione	4
1.1	Scopo del Documento	4
1.2	Scopo del Prodotto	4
1.3	Riferimenti	4
2	Principali Tecnologie Utilizzate	6
2.1	Scopo del capitolo	6
2.2	Tecnologie Lato Client	6
2.3	Tecnologie Lato Server	6
2.4	Tecnologie per il Testing	6
3	Installazione	7
3.1	Requisiti	7
3.2	Esecuzione	7
4	Impostare l'Ambiente di Lavoro	8
4.1	Scopo del Capitolo	8
5	Requisiti	8
5.1	WebStorm	8
5.2	NPM	8
5.3	ESLint	9
5.4	Jest	9
5.5	Webpack	10
6	Architettura	11
6.1	Scopo del Capitolo	11
6.2	Pannello	11
6.3	Server	14
7	Test	17
7.1	Scopo del Capitolo	17
7.2	Test sul Codice JavaScript	17
7.3	Code Coverage	17
8	Licenza	18

Elenco delle tabelle

1	Registro delle Modifiche	1
---	--	---

Elenco delle figure

1	Diagramma dei Package del Pannello	12
2	Diagramma delle classi del Pannello	13
3	Diagramma dei Package del Server	15
4	Diagramma delle classi del Server	16

1 Introduzione

1.1 Scopo del Documento

Lo scopo del documento è quello di fornire tutte le informazioni necessarie agli sviluppatori che intenderanno estendere o migliorare il plug-in G&B.

Verranno fornite informazioni relative anche ad un possibile ambiente di sviluppo il più completo possibile da cui sarà possibile partire. In particolare sarà illustrato l'ambiente di sviluppo utilizzato dal team **Agents of S.W.E.** per lo sviluppo del prodotto in oggetto.

La seguente guida può essere utilizzata indifferente sia da utenti Microsoft Windows, Linux e Apple MacOS.

1.2 Scopo del Prodotto

Lo scopo del prodotto è la creazione di un plug-in per la piattaforma open source_G di visualizzazione e gestione dati, denominata Grafana_G, con l'obiettivo di creare un sistema di alert dinamico per monitorare la "liveliness"_G del sistema a supporto dei processi DevOps_G e per consigliare interventi nel sistema di produzione del software. In particolare, il plug-in_G utilizzerà dati in input forniti ad intervalli regolari o con continuità, ad una rete bayesiana_G per stimare la probabilità di alcuni eventi, segnalandone quindi il rischio in modo dinamico, prevenendo situazioni di stallo.

1.3 Riferimenti

1.3.1 Referimenti per l'Installazione

- <https://www.npmjs.com/>.

1.3.2 Referimenti Legali

- <https://grafana.com/docs/contribute/cla/>.

1.3.3 Referimenti Informativi

- <https://grafana.com/>;
- <https://www.influxdata.com/time-series-platform/telegraf/>;
- <https://www.influxdata.com/time-series-platform/influxdb/>;
- <https://jestjs.io/>;



- <https://www.jetbrains.com/webstorm/>;
- <https://webpack.js.org/>.
- <https://angularjs.org/>;
- <http://codecov.io>;
- <https://jquery.com/>.

2 Principali Tecnologie Utilizzate

2.1 Scopo del capitolo

In questo capitolo verranno espone e contestualizzate le tecnologie impiegate all'interno del progetto, in modo tale da offrire agli sviluppatori un quadro più completo del progetto.

2.2 Tecnologie Lato Client

Lato client vengono adottate le seguenti tecnologie:

- *EcmaScript 6*: principale linguaggio con cui il plug-in è strutturato, in particolare modo viene utilizzato per lo sviluppo del pannello;
- *AngularJS*: framework adottato da *Grafana* per l'interazione con l'utente;
- *HTML & CSS*: rispettivamente nelle versioni 5 e 3, utilizzati per modellare l'interfaccia del pannello.

2.3 Tecnologie Lato Server

Lato server vengono adottate le seguenti tecnologie:

- *NodeJS*: tecnologia utilizzata per implementare il server.

2.4 Tecnologie per il Testing

La tecnologia utilizzata per testare il codice è *Jest*, framework di test per codice *JavaScript*.

3 Installazione

Per poter utilizzare il prodotto all'interno di Grafana, e poiché quest'ultimo sia riconosciuto come plug-in, è necessario per prima cosa eseguire la build del prodotto. La build del prodotto avviene attraverso due fasi: la prima che va a risolvere le dipendenze, la seconda che esegue la build.

Il processo di build avviene attraverso l'intervento del module bundler *WebPack*. È necessario quindi eseguire i seguenti comandi dalla directory che ospita il plug-in:

```
npm install  
npm run build.
```

3.1 Requisiti

I requisiti minimi richiesti per il funzionamento del plug-in non sono dovuti al prodotto in sé, ma sono dovuti alle tecnologie che vengono utilizzate. Pertanto si rimanda ai requisiti minimi delle seguenti tecnologie:

- *InfluxDB*;
- *Telegraf*;
- *Grafana*;
- *NodeJS*.

3.2 Esecuzione

Per poter eseguire il prodotto è necessario seguire i seguenti passi:

1. Arrestare l'esecuzione di Grafana, qualora fosse in esecuzione;
2. Copiare la directory del progetto all'interno della cartella predestinata da Grafana per ospitare i plug-in aggiuntivi da installare;
3. Eseguire i comandi descritti in §3;
4. Assicurarsi che il server esterno a cui sono delegati i calcoli sia attivo e raggiungibile;
5. Avviare Grafana e procedere all'aggiunta del plug-in.

Per informazioni relative al funzionamento del plug-in si rimanda al Manuale Utente.

4 Impostare l'Ambiente di Lavoro

4.1 Scopo del Capitolo

In questa sezione viene riportata una guida per la corretta configurazione dell'ambiente di sviluppo, in modo che sia la stessa utilizzata dal gruppo **Agents of S.W.E.**.

Per contribuire al progetto non è strettamente necessario seguire questa sezione, tuttavia è consigliato al fine di ottenere un ambiente di lavoro pronto e correttamente impostato per lo sviluppo.

5 Requisiti

Per i requisiti minimi di funzionamento dell'ambiente di lavoro consigliato, si rimanda al sito del produttore di ogni singola tecnologia.

5.1 WebStorm

WebStorm è l'ambiente di sviluppo integrato (IDE) di JetBrains utilizzato dal team per lo sviluppo del progetto. Esso può essere ottenuto mediante download dal sito ufficiale nella formula di prova gratuita se non si dispone di licenza, che può essere ottenuta attraverso l'indirizzo email universitario.

Tale software è disponibile per i sistemi operativi Microsoft Windows, Linux e Apple MacOS.

Per ulteriori informazioni si rimanda al sito ufficiale.

5.2 NPM

NPM è il gestore di pacchetti utilizzato per gestire il progetto dal team **Agents of S.W.E.**. Per la relativa installazione si rimanda al sito del produttore.

Per l'effettivo utilizzo è necessario avere all'interno della directory del progetto un file denominato "package.json". Esso permette di esprimere le dipendenze e le caratteristiche del prodotto, oltre che alla definizione dei vari comandi da eseguire con radice "`npm run ...`".

Di seguito è riportato, nei punti salienti, il file "package.json" utilizzato durante lo sviluppo del progetto:

```
1 {  
2   "name": "GrafanaAndBayes",
```



```
3  "version": "2.0.0",
4  "description": "Plug-in per Grafana",
5  "main": "src/module.js",
6  "scripts": {
7    "test": "jest",
8    "build": "webpack --config build/webpack.prod.conf.js",
9    "dev": "webpack --mode development",
10   "eslint": "eslint ./src",
11   "codecov": "codecov"
12 },
13 "jest": {
14   "verbose": true,
15   "collectCoverage": true,
16   "coverageDirectory": "./coverage/"
17 },
18 "author": "Agents Of S.W.E.",
19 "license": "ISC",
20 "devDependencies": {
21   ...
22 },
23 "dependencies": {
24   ...
25 }
26 }
```

5.3 ESLint

ESLint verrà installato automaticamente attraverso il comando `npm install`.

Per abilitarlo all'interno di WebStorm è necessario lanciare l'IDE e recarsi in: File > Settings > ESLint e scegliere "Enable". All'interno del campo "Node Interpreter" è necessario inserire il percorso alla directory in cui si trovano i file eseguibili di Node. Se non rilevato in automatico, specificare la posizione del file di configurazione ".eslintrc" all'interno della directory del progetto.

5.4 Jest

Jest è il framework di test utilizzato per testare il codice JavaScript. È inoltre il responsabile della generazione dei report di coverage utilizzati in seguito per il calcolo del code coverage.

L'installazione di tale componente consta nell'aggiunta del relativo pacchetto *npm*. In particolare si dovranno eseguire i seguenti comandi:

```
npm install -save-dev jest.
```

A ciò seguirà poi l'aggiunta del relativo comando di script all'interno di "package.json" in modo tale da consentire l'esecuzione dei test all'invocazione del comando `npm run test`. La modifica da apportare al file sopracitato, e sopra riportato, è visibile alla riga 7:

```
"test": "jest",.
```

5.5 Webpack

Webpack è il model bundler utilizzato eseguire la build del progetto.

L'installazione di tale componente avviene tramite NPM. È necessario eseguire il seguente comando:

```
npm install -save-dev webpack.
```

In seguito è necessario aggiungere una riga al file "package.json", come riportato in riga 8:

```
"build": "webpack -config build/webpack.prod.conf.js",.
```

6 Architettura

6.1 Scopo del Capitolo

Lo scopo del capitolo è quello di fornire le informazioni necessarie allo sviluppatore per potersi interfacciare con il prodotto, in modo tale da rendere più agevole l'ampliamento e la modifica.

6.2 Pannello

Per rendere lo sviluppo più semplice, e garantire la manutenibilità del codice il team ha optato per un approccio modulare. In questo modo, avendo moduli separati con compiti distinti, sarà più semplice modificarne o estenderne il comportamento senza dover necessariamente modificare la base comune.

In particolare i moduli separati sono:

- **GBCtrl**: è il modulo principale che utilizza gli altri e svolge le operazioni fondamentali;
- **TemporalPolicyCtrl**: è il modulo che si occupa di gestire ed impostare le politiche temporali all'interno del pannello;
- **ThresholdsCtrl**: è il modulo che si occupa di gestire ed impostare le soglie per il monitoraggio;
- **Parser**: è il modulo che si occupa di controllare e interpretare la rete bayesiana in input, in formato *JSON*;
- **ConnectServerProxy**: è il modulo che si occupa di inoltrare le richieste al server preoccupandosi che i dati siano strutturati nel modo corretto e ricevere i dati;
- **ModalCreatore**: è il modulo che si occupa di visualizzare le finestre che permettono l'interazione con l'utente;
- **GetAPIGrafana**: è il modulo che si occupa di interagire con le API di *Grafana* per ottenere i dati relativi alle sorgenti dati.

6.2.1 UML - Diagramma dei Package

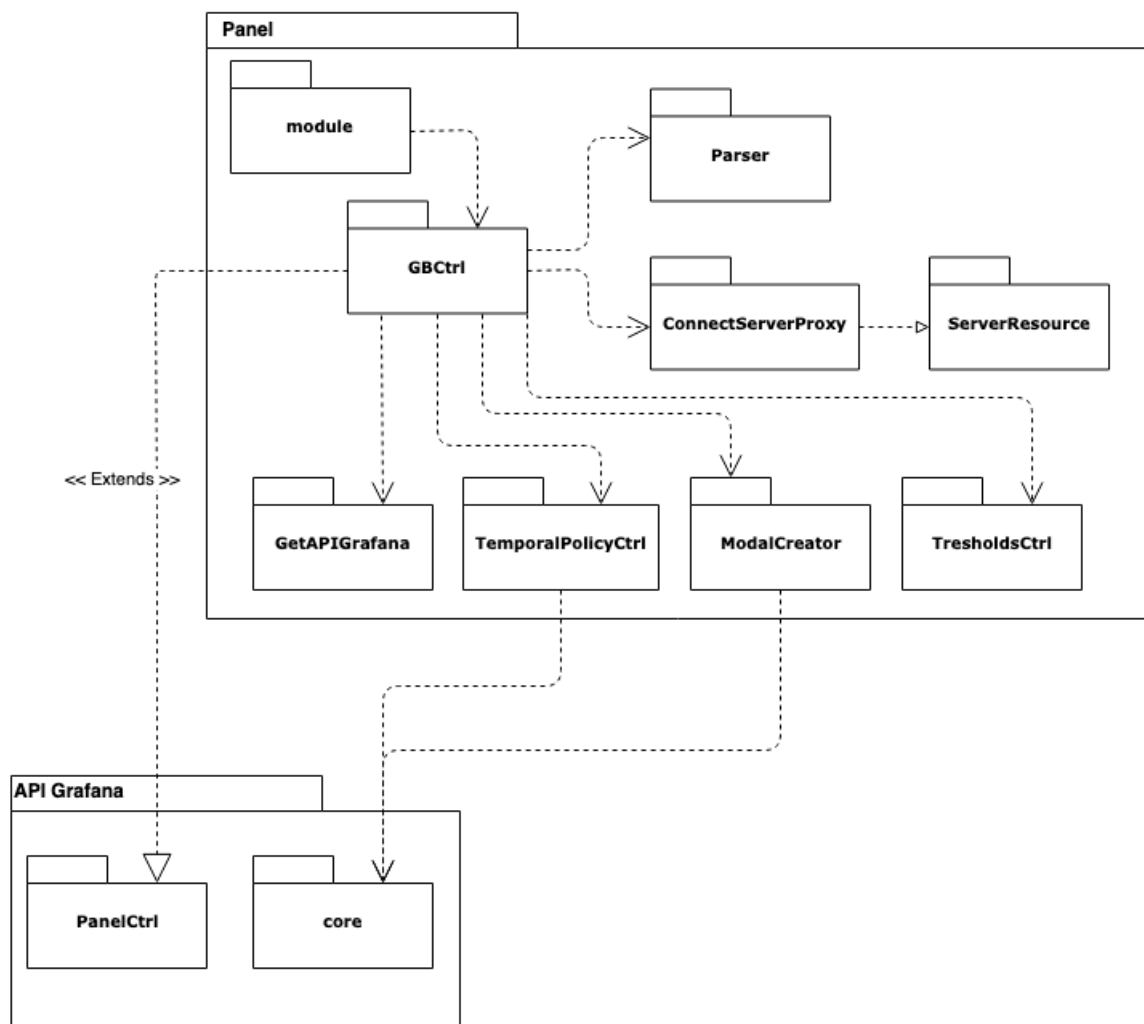


Figura 1: Diagramma dei Package del Pannello



Figura 2: Diagramma delle classi del Pannello



6.3 Server

Per rendere lo sviluppo più semplice, e garantire la manutenibilità del codice il team ha optato per un approccio modulare. In questo modo, avendo moduli separati con compiti distinti, sarà più semplice modificarne o estenderne il comportamento senza dover necessariamente modificare la base comune.

In particolare i moduli presenti per quanto riguarda lo sviluppo del server sono:

- **Server:** è il modulo principale che si occupa di comunicare con gli altri, ed esegue le operazioni fondamentali;
- **InfluxDB:** è il modulo che si occupa di adattare la libreria *Influx* e ci fornisce un set di metodi specializzati;
- **Network:** è il modulo che si occupa di adattare la libreria *jsbaye* e costruisce la rete bayesiana da utilizzare;
- **ProxyServer:** è il modulo che si occupa di filtrare, autenticare le richieste verso il server e protegge lo stesso.

6.3.1 UML - Diagramma dei Package

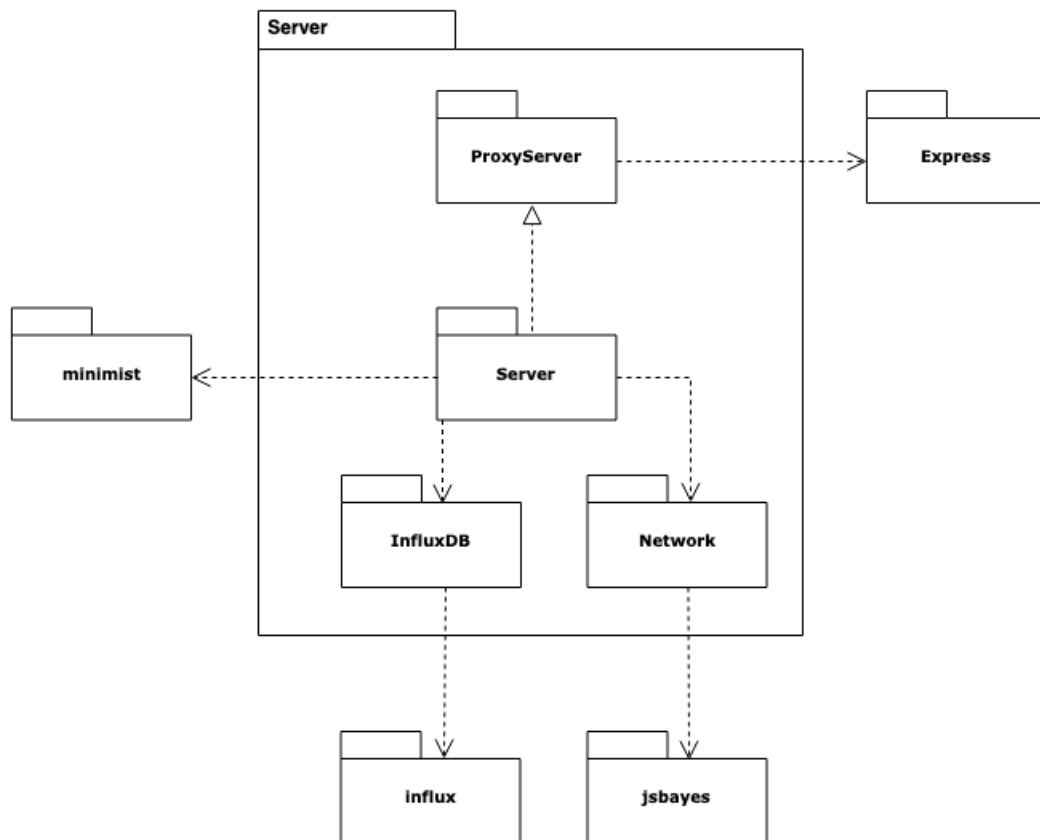
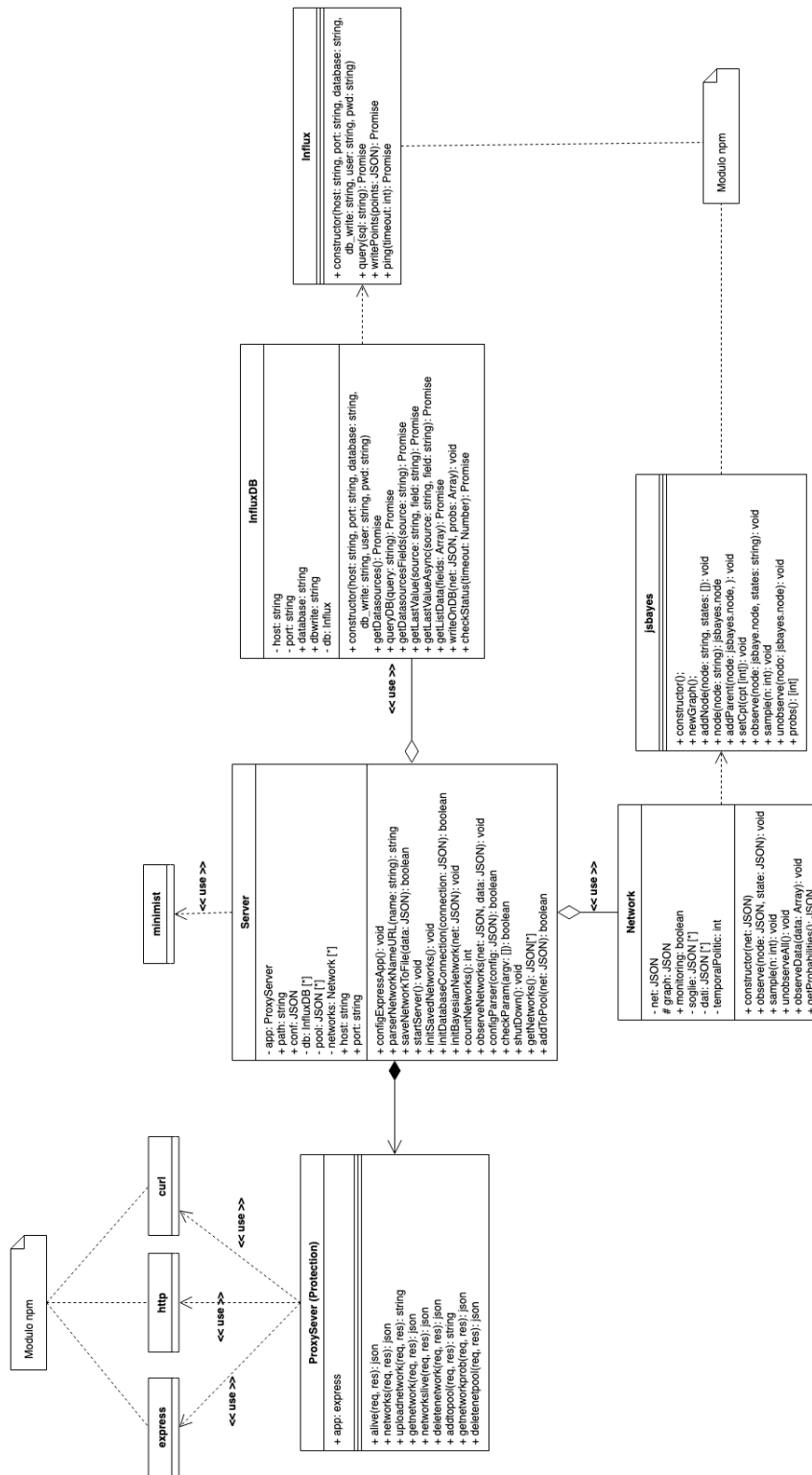


Figura 3: Diagramma dei Package del Server

Figura 4: Diagramma delle classi del Server



7 Test

7.1 Scopo del Capitolo

Questo capitolo ha lo scopo di indicare agli sviluppatori come controllare in modo automatico il proprio codice e la sintassi.

7.2 Test sul Codice JavaScript

Per eseguire i test sul codice è necessario eseguire i seguenti comandi:

```
npm run test  
o  
jest.
```

Per verificare che il codice sia coerente con le linee guida adottate è necessario eseguire:

```
npm run eslint.
```

Per correggere in modo automatico alcuni dei potenziali problemi, eseguire:

```
npm run eslint-fix.
```

Questi script eseguono un controllo del codice all'interno di `./src`, la directory dov'è contenuto il codice.

Se si desidera eseguire *ESLint* su di un'unico file, si rimanda il lettore alla documentazione ufficiale.

7.3 Code Coverage

Per eseguire il controllo di coperta del codice JavaScript, è necessario eseguire il seguente comando:

```
npm run codecov.
```

Nel caso tale comando fallisca la principale motivazione è data dall'assenza di report su cui generare il code coverage. Per rimediare a ciò basterà eseguire il primo comando nella sezione §7.2 il quale provvederà a generare i report necessari.

8 Licenza

MIT License

Copyright (c) 2019 Agents Of S.W.E.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.