# A close up of a sign Description automatically generated Mathematical Expressions Evaluator Server di Francesco Benincasa

## Introduzione

Il progetto si pone l’obiettivo di implementare un server per la valutazione di espressioni matematiche, come descritto nelle specifiche del progetto fornite. La soluzione è stata realizzata con e per OpenJDK 11. Il progetto è stato realizzato con l’IDE IntelliJ.

## A screenshot of a cell phone Description automatically generatedTecnologie utilizzate

Per la gestione del progetto si è utilizzato Maven (versione 3.6.3). I sorgenti del progetto sono organizzati seguendo la convenzione di Maven stesso.

Come piattaforma di source version control e di issue management si è utilizzata la piattaforma GitHub.

Si riportano di seguito un elenco dei principali framework e le librerie utilizzati nel progetto:

* **Spring framework**: framework scelto come container Inversion of Control (IoC). Lo stesso è stato utilizzato per realizzare il componente AOP[[1]](#footnote-1) definito per monitorare le statistiche di esecuzione dei comandi.
* **Spring boot**: utilizzato per semplificare l’utilizzo di Spring.
* **ANTLR**[[2]](#footnote-2): libreria utilizzata per generare l’analizzatore sintattico lessicale necessario ad analizzare le richieste effettuate dai client. La grammatica definita per il progetto con le convenzioni ANTLR è definita nel file src/main/antlr4/org/abubusoft/mee/server/grammar/Commands.g4.
* **JUnit**: framework utilizzato per testare i componenti “core” del progetto.
* Varie librerie di utilità: Google Guava, Apache commons IO, Apache commons lang.

## Struttura del progetto

Si riportano le principali entità definite nel progetto mediante due diagrammi delle classi; uno per il data model ed un altro per le classi definiscono la logica di business. Il class diagram per il modello dati:

A screenshot of a video game

Description automatically generated

Il class diagram della logica di business:

A screenshot of a video game

Description automatically generated

## Compilazione

Si assume che il codice sorgente sia stato scompattato in una cartella. Per compilare il progetto è necessario aver installato il JDK 11 o superiore (OpenJDK) e Maven (versione utilizzata 3.6.3). Una volta aperta una CLI sulla cartella con i sorgenti scompattati, eseguire il seguente comando:

mvn clean package

L’esecuzione di tale comando produce l’artifact ./target/BenincasaFrancesco.jar. Il jar contiene, oltre alle classi del progetto vero e proprio, tutte le classi delle librerie e dei framework utilizzate (escluse ovviamente quelle appartenenti al JDK).

## Esecuzione

Il jar prodotto dall’ultima build del progetto è incluso nello zip assieme a questo. È possibile eseguire l’applicativo con il comando java -jar BenincasaFrancesco.jar.

La porta di default è impostata a 10000. Nel caso si desideri mettere in ascolto l’applicativo su un’altra porta è sufficiente eseguire il comando

java -jar BenincasaFrancesco.jar ${p}

Dove ${p} rappresenta il numero di porta di ascolto del server.

A screenshot of a video game

Description automatically generated

### Log

Il livello di log impostato durante la build è il livello INFO. Con questo livello vengono visualizzate le informazioni inerenti informazioni sull’avvio dell’applicativo, delle nuove connessioni, delle connessioni chiuse e degli eventuali comandi con errori. Per impostare un maggiore livello di dettaglio dei log (ad esempio gli aggiornamenti delle statistiche di esecuzione), è possibile eseguire l’applicazione impostando il livello di log a DEBUG o a TRACE:

java -Dlogging.level.org.abubusoft.mee.server=DEBUG -jar BenincasaFrancesco.jar

Tra le informazioni presenti nelle voci di log si ritrova anche il thread usato. I thread utilizzati per calcolare il valore delle espressioni matematiche sono quelli con il nome con prefisso Compute.

1. https://it.wikipedia.org/wiki/Programmazione\_orientata\_agli\_aspetti [↑](#footnote-ref-1)
2. https://www.antlr.org/ [↑](#footnote-ref-2)