

Software Design - CMMTickets



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Caprinali Michele 1087210
Mazzoleni Gabriel 1086530
Mazzoleni Raphael 1086531

Indice a:

1. [Viste architettoniche](#)
2. [Architettura del software](#)
3. [Libreria Esterna con Maven](#)

Indice b:

1. [Diagramma delle classi](#)
2. [Composizione \(STAN4J\)](#)
3. [Metriche](#)
4. [Design Patterns](#)

Software Architecture

1- Viste architettoniche

I punti di vista all'interno dell'architettura del software sono:

- *Punto di vista Use*: quando un elemento viene modificato, devono essere modificati anche tutti gli elementi da cui è utilizzato a sua volta.
- *Punto di vista dei dati condivisi*: descrive come vengono prodotti e consumati i dati persistenti
- *Punto di vista dell'incarico di lavoro*: mostra chi sta facendo cosa e aiuta a determinare quale conoscenza è necessaria e dove. Tutti i membri del team sono stati coinvolti nelle varie fasi del progetto, come spiegato nel documento 'Gestione di progetto'.

2- Architettura del software

Stile architettonico *MVC*:

- *Modello*: la funzione logica del sistema, per eseguire i calcoli e la gestione dei dati nel database
- *View*: mostra l'output agli utenti
- *Controller*: gestisce l'input del sistema, quali la creazione di eventi o l'acquisto di biglietti.
- *Componenti*: raccolte di procedura (modulo)
- *Connettori*: chiamate di procedura
- *Struttura di controllo*: single thread

3- Libreria Esterna con Maven

Per il progetto, sono state integrate diverse librerie con Maven:

a) Libreria utilizzata: *JUnit*

Scopo: Test di unità e integrazione del codice sviluppato

Configurazione Maven:

```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <scope>test</scope>
</dependency>
<!-- Optionally: parameterized tests support -->
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-params</artifactId>
  <scope>test</scope>
</dependency>
```

b) Libreria utilizzata: *SQLite*

Scopo: Gestione delle tabelle e dati del database

Configurazione Maven:

```
<dependency>
  <groupId>org.xerial</groupId>
  <artifactId>sqlite-jdbc</artifactId>
  <version>3.43.2.2</version>
</dependency>
```

c) Libreria utilizzata: *Log4j*

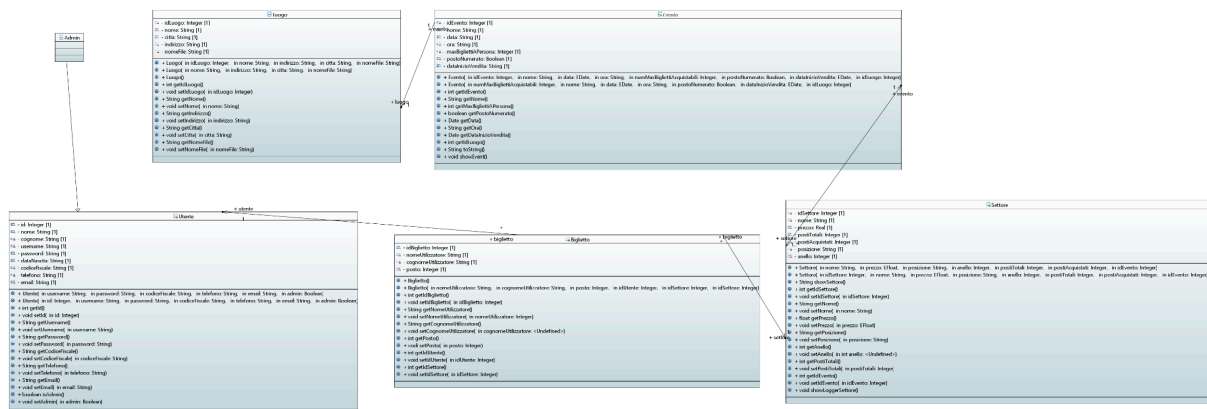
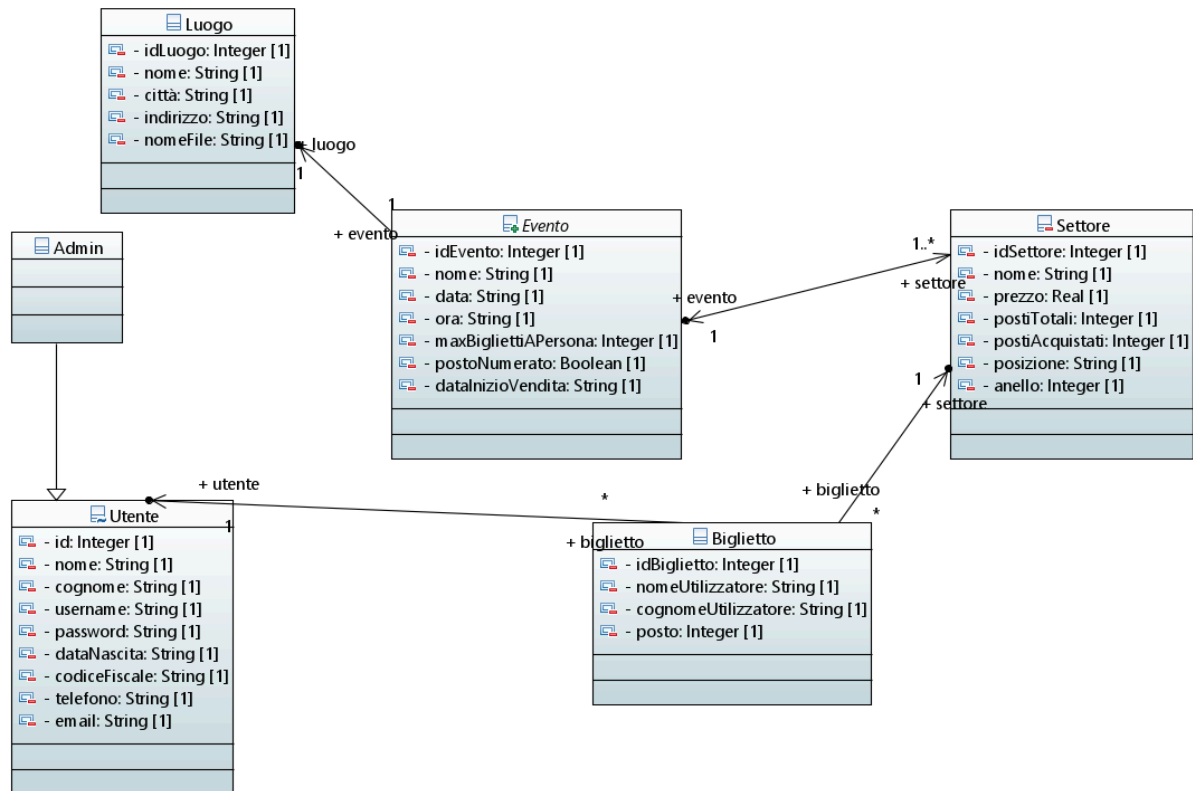
Scopo: utilizzo dei log nel codice

Configurazione Maven:

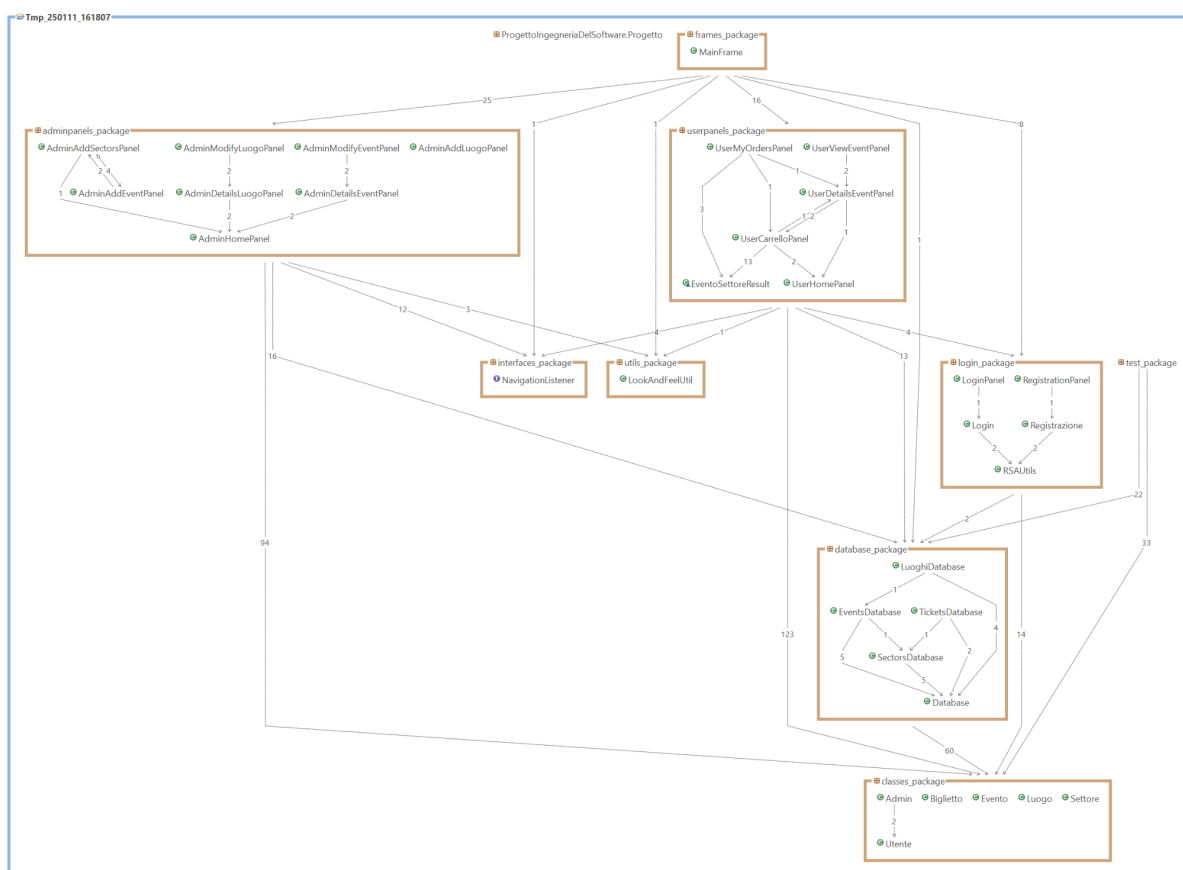
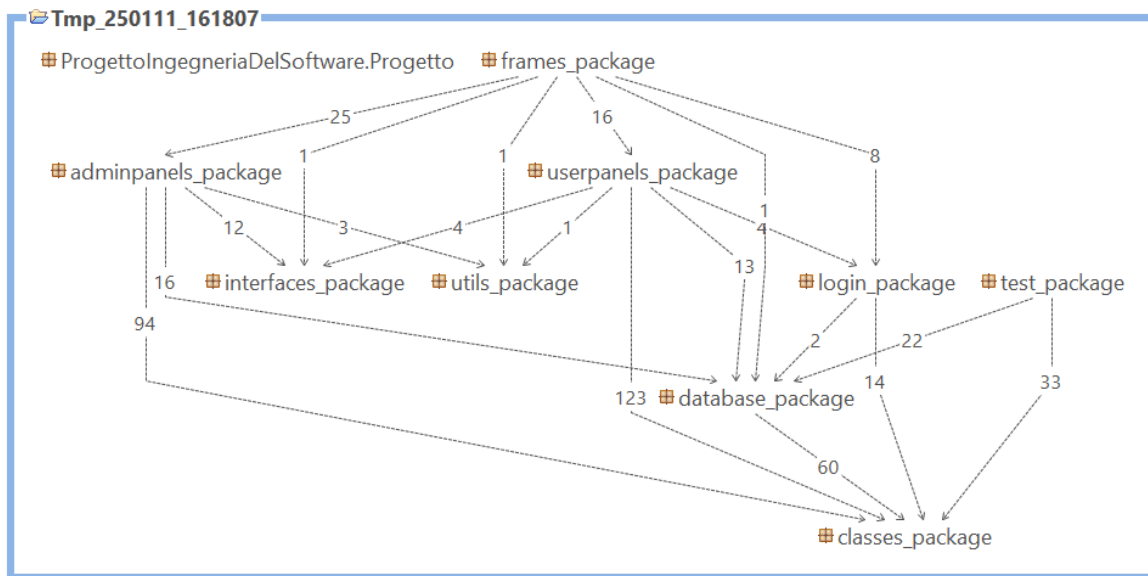
```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.6.1</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.6.1</version>
</dependency>
```

Software Design

1- Diagramma delle classi



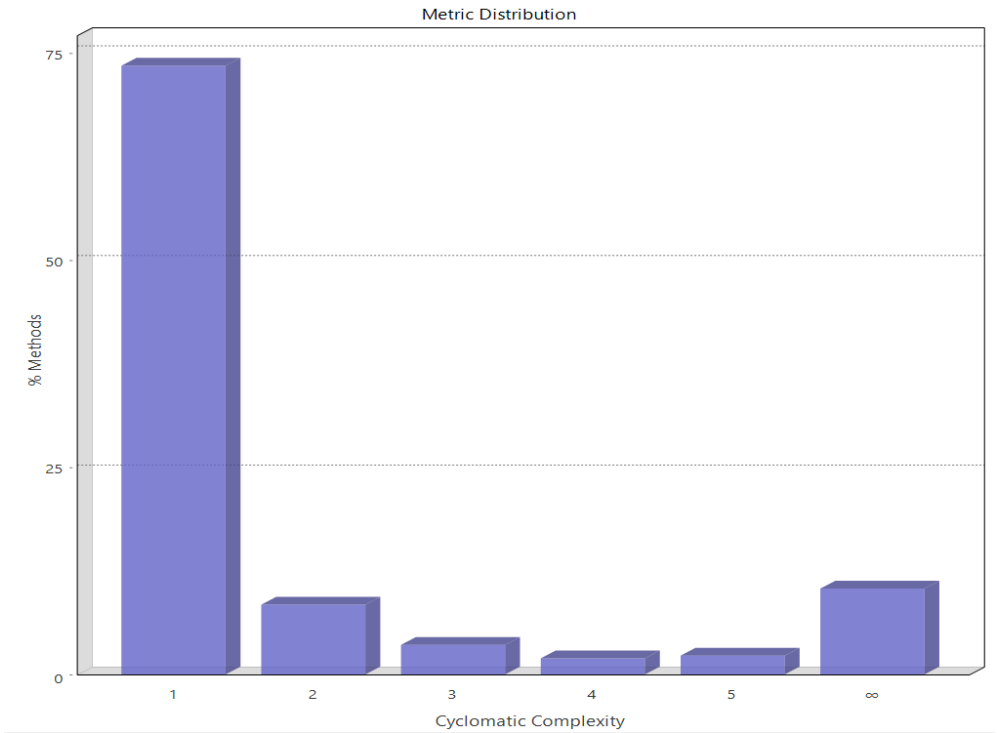
2- Composizione (STAN4J)



3- Metriche

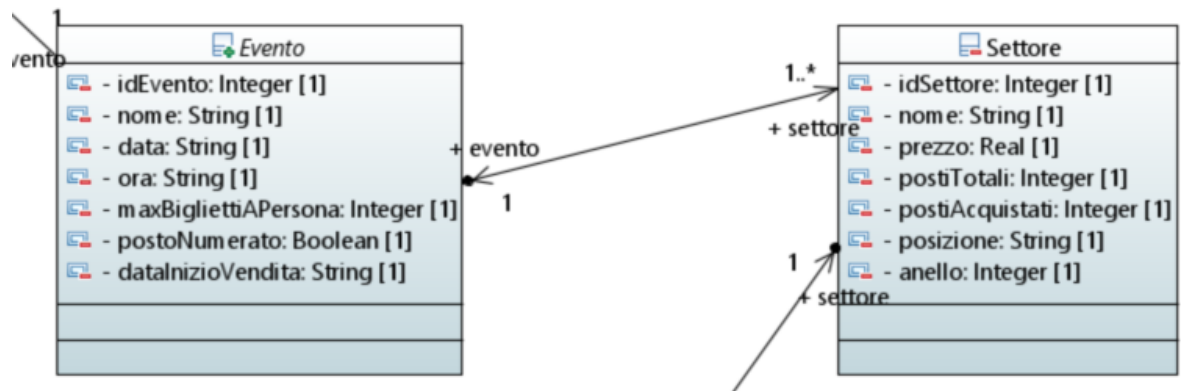
Qui di seguito verranno mostrate alcune metriche del progetto:

Category/Metric	Value
<ul style="list-style-type: none"> Count <ul style="list-style-type: none"> Libraries 3 Packages 9 Units 38 Units / Package 4.22 Classes / Class 0.05 Methods / Class 5.18 Fields / Class 3.71 ELOC 5155 ELOC / Unit 135.66 Complexity <ul style="list-style-type: none"> CC 2.36 Fat - Libraries 1 Fat - Packages 20 Fat - Units 123 Tangled 1.10% Tangled - Libraries 0.00% ACD - Library 16.67% ACD - Package 31.94% ACD - Unit 24.32% Robert C. Martin <ul style="list-style-type: none"> D -0.45 D 0.45 Chidamber & Kemerer <ul style="list-style-type: none"> WMC 12.19 DIT 1.02 NOC 0.02 CBO 5.26 RFC 9.87 LCOM 8.56 	



4- Design Patterns

- *Abstraction-Occurrence Pattern*: l'abbiamo utilizzato per rappresentare l'associazione Settore-Evento in quanto ad ogni settore abbiamo associato il proprio evento:



- *Immutable Pattern*: nel contesto dei Panel che abbiamo sviluppato, possiamo affermare di aver utilizzato l'Immutable Pattern, in quanto tutte le componenti grafiche (JButton, JTextField, JPanel, JScrollPane, ...) sono tutte inizializzate e configurate all'interno del costruttore della classe (in alcune classi che richiedono una costruzione dinamica delle componenti, per esempio in base al numero di biglietti come in UserCarrelloPanel, lo scheletro statico della pagina è comunque inizializzato nel costruttore e non viene più modificato). I metodi pubblici o privati delle classi non alterano lo stato delle componenti, al massimo accedono ai valori o aggiornano i dati visualizzati senza cambiare la struttura o la configurazione del pannello stesso.