



# DOCUMENTAZIONE

A CURA DI GIORGIO CHIRICO E RAFFAELE GIACOMO GIOVANNI DI MAIO



# LEGENDA

Capitolo 1: Software Engineering Management

Capitolo 2: Software Life Cycle

Capitolo 3: Configuration Management

Capitolo 4: People Management and Team Organization

Capitolo 5: Software Quality

Capitolo 6: Requirement Engineering

A pair of black-rimmed glasses rests on an open book. A red bookmark is visible on the left page. The background is a soft, out-of-focus blue-grey.

# LEGENDA

Capitolo 7: Modelling

Capitolo 8: Software Architecture

Capitolo 9: Software Design

Capitolo 10: Software Testing

Capitolo 11: Software Maintenance



# CAPITOLO 1: Software Engineering Management

## - Project Plan

E' possibile prendere visione del nostro project plan all'interno della repository su GitHub al seguente link, andandolo poi a scaricare.

<https://github.com/giorgio-hash/progettoSE/blob/master/SoftwareEngineeringProject/documentazione/ProjectPlan.docx>

# CAPITOLO 2: Software Life Cycle

## - Metodi Agile

Per la realizzazione del progetto è stata utilizzata una filosofia di tipo Agile, mirata a seguire tutti i punti presenti nel manifesto.

In particolare troviamo:

- Concentrarsi su persone e interazioni invece che su processi e strumenti;
- Concentrarsi sul funzionamento del software invece che sulla comprensività della documentazione;
- Concentrarsi sulla collaborazione con il cliente invece che sulla negoziazione dei contratti;
- Concentrarsi sulle risposte al cambiamento del software invece che sul seguire un piano.

# CAPITOLO 2: Software Life Cycle

## - XP: Xtreme Programming

All'interno del gruppo di metodi Agile abbiamo scelto di utilizzare l'Xtreme Programming.

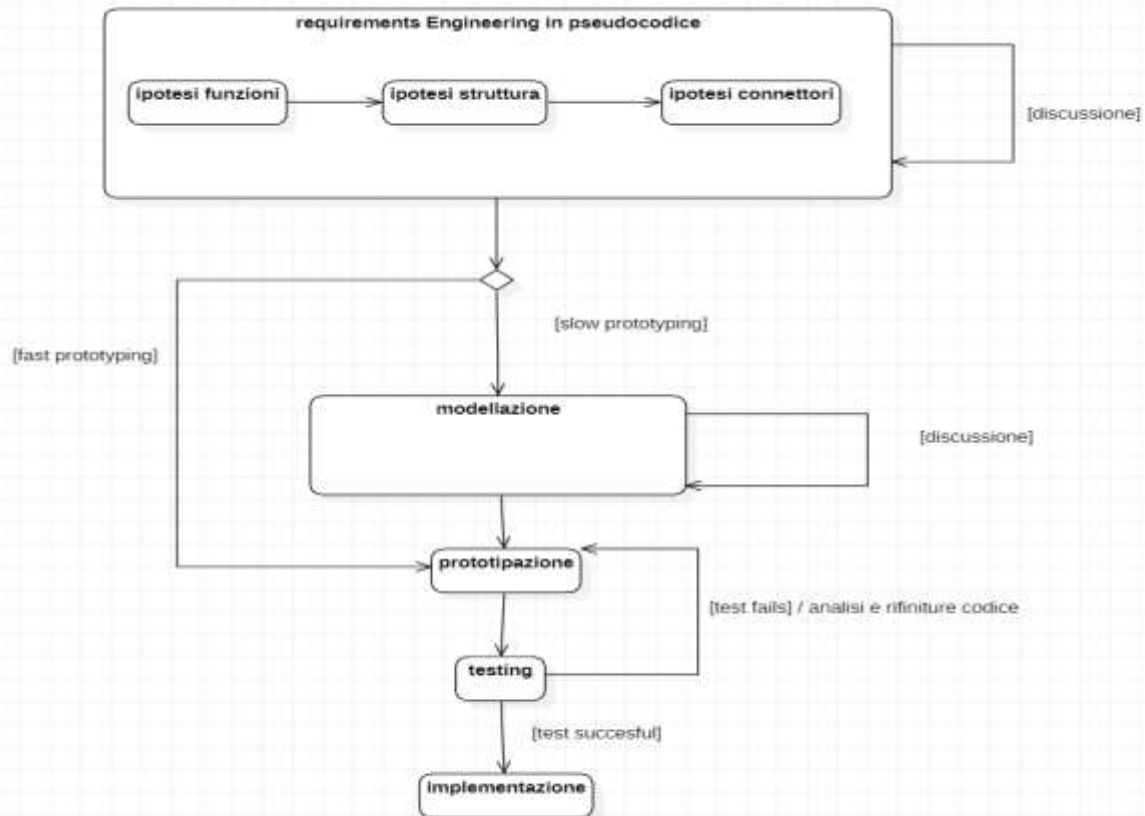
I punti chiave di questo processo di sviluppo sono:

- Il codice viene scritto a piccoli passi;
- il sistema deve SEMPRE compilare e deve essere SEMPRE eseguibile;
- il cliente deve essere coinvolto nella realizzazione del progetto;
- tutti gli sviluppatori hanno la stessa responsabilità per quanto riguarda il software e la metodologia utilizzata.



# CAPITOLO 2: Software Life Cycle

Flow diagram for prototyping process:



# CAPITOLO 3: Configuration Management

Il configuration management si basa sulla possibilità di gestire qualsiasi tipo di «strumento» durante lo sviluppo del software.

Esso è cruciale per quanto riguarda progetti di larga scala.

- GitHub

Come sistema per il configuration management del nostro progetto abbiamo scelto di utilizzare GitHub che ci ha permesso di condividere le varie modifiche ed implementazioni successive del nostro progetto sulla baseline.



# CAPITOLO 3: Configuration Management

Le quattro operazioni principali che abbiamo utilizzato riguardante GitHub sono:

- Branch, è utilizzato per isolare lo sviluppo del lavoro senza influire su altri branch nella repository. E' possibile unire due branch usando una pull request;
- Commit, un record salvato che contiene tutti i cambiamenti fatti ad uno o più file nella repository;
- Pull request, fa sapere agli altri collaboratori i cambiamenti che sono stati inseriti nella repository. Una volta che una pull request è aperta, è possibile discutere dei cambiamenti con i collaboratori.
- Issue, viene creata per tenere traccia di idee, bugs, tasks, ecc.

# CAPITOLO 3: Configuration Management (Branches)

progettoSE/SoftwareEngineeringProject

github.com/giorgio-hash/progettoSE/tree/diversificazione-prodotti/SoftwareEngineeringProject/Codice

Search or jump to... Pull requests Issues Marketplace Explore

giorgio-hash / progettoSE Private

Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights Settings

diversificazio... progettoSE / SoftwareEngineeringProject / Codice /

Go to file Add file ...

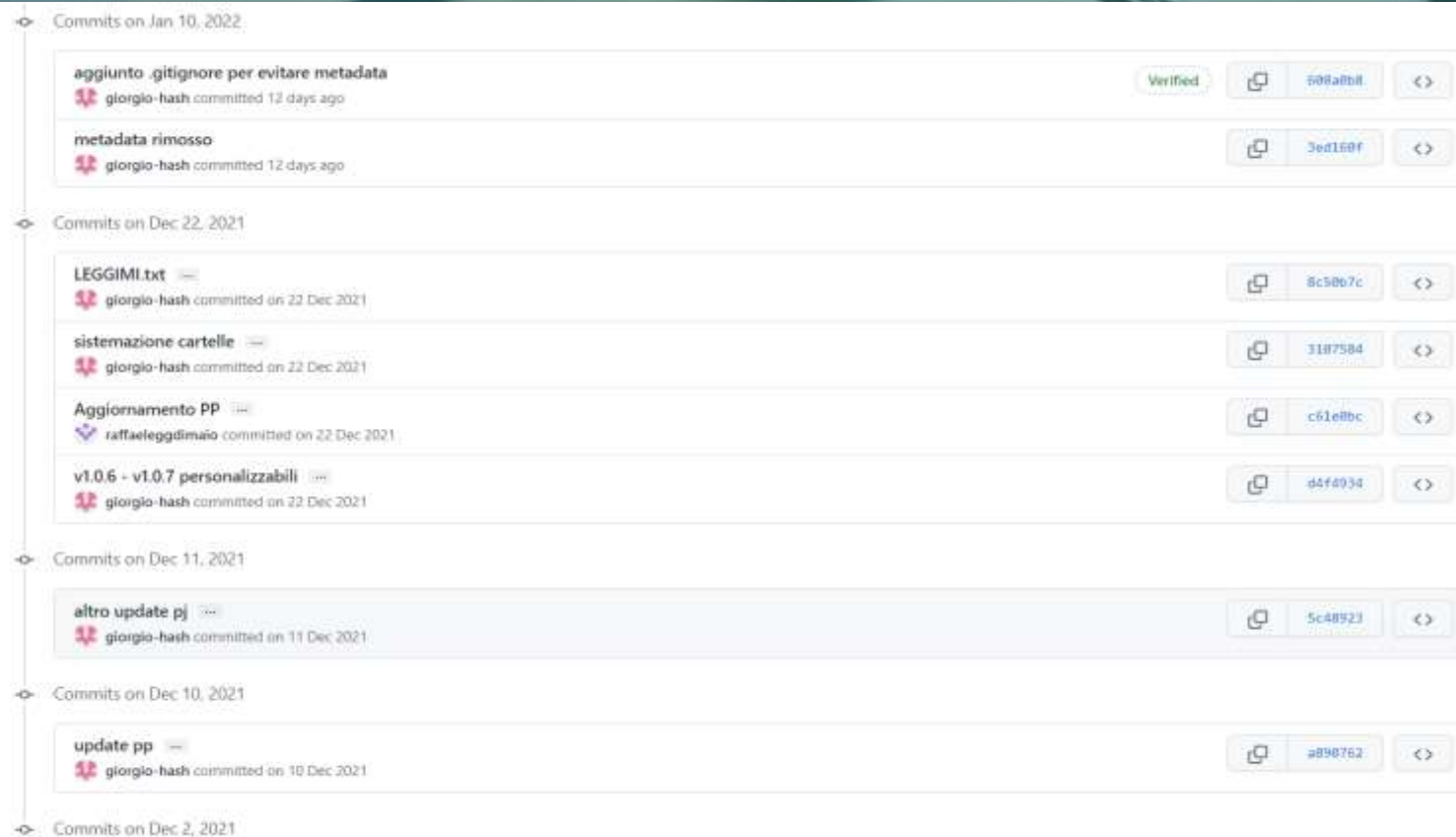
This branch is 3 commits ahead of master. Contribute

giorgio-hash FeatureIDE per diversificazione prodotti 2 seconds ago 14 seconds ago History

FeaturizedGestoreParcheggio	FeatureIDE per diversificazione prodotti	14 seconds ago
prototipi	v1.0.0 completo	2 months ago
v1.0.7_gestoreParcheggio	v1.0.6 - v1.0.7 personalizzabili	last month
versioni precedenti	v1.0.6 - v1.0.7 personalizzabili	last month

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

# CAPITOLO 3: Configuration Management (Commits)

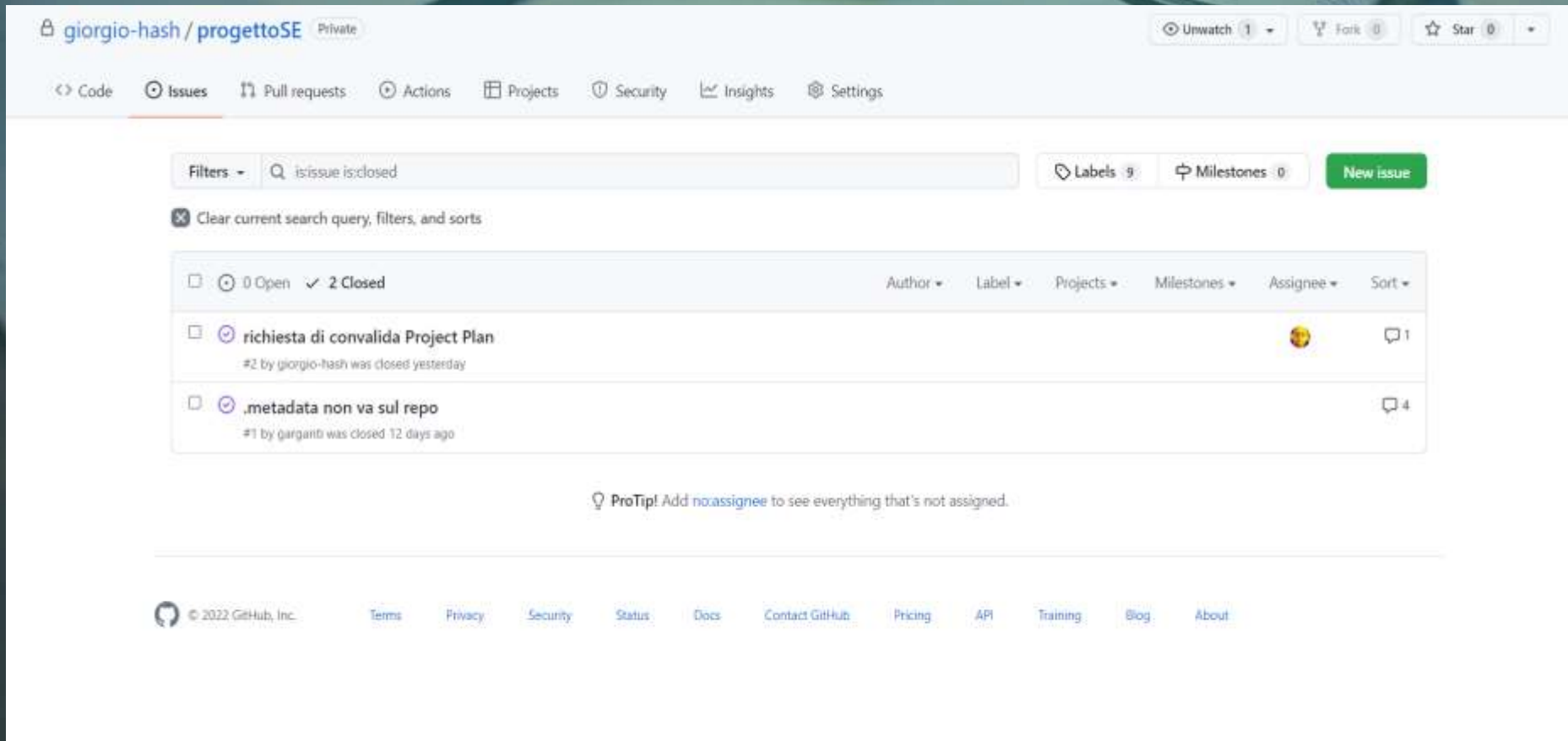


The screenshot displays a Git commit history interface. It is organized into sections by date, with expandable/collapsible arrows on the left. The sections are:

- Commits on Jan 10, 2022**
  - Commit: `aggiunto .gitignore per evitare metadata` by `giorgio-hash` (committed 12 days ago). Hash: `508a8b1f`. Status: Verified.
  - Commit: `metadata rimosso` by `giorgio-hash` (committed 12 days ago). Hash: `3ed160f`.
- Commits on Dec 22, 2021**
  - Commit: `LEGGIMI.txt` by `giorgio-hash` (committed on 22 Dec 2021). Hash: `8c50b7c`.
  - Commit: `sistemazione cartelle` by `giorgio-hash` (committed on 22 Dec 2021). Hash: `318758d`.
  - Commit: `Aggiornamento PP` by `raffaeleggidimaio` (committed on 22 Dec 2021). Hash: `c61e8bc`.
  - Commit: `v1.0.6 - v1.0.7 personalizzabili` by `giorgio-hash` (committed on 22 Dec 2021). Hash: `d4f403d`.
- Commits on Dec 11, 2021**
  - Commit: `altro update pj` by `giorgio-hash` (committed on 11 Dec 2021). Hash: `5c48923`.
- Commits on Dec 10, 2021**
  - Commit: `update pp` by `giorgio-hash` (committed on 10 Dec 2021). Hash: `a890762`.
- Commits on Dec 2, 2021**



# CAPITOLO 3: Configuration Management (Issues)



giorgio-hash / progettoSE Private

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights Settings

Filters  Labels 9 Milestones 0 [New issue](#)

Clear current search query, filters, and sorts

<input type="checkbox"/>	0 Open ✓ 2 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	<a href="#">richiesta di convalida Project Plan</a> #2 by giorgio-hash was closed yesterday						1
<input type="checkbox"/>	<a href="#">.metadata non va sul repo</a> #1 by garganti was closed 12 days ago						4

ProTip! Add `no:assignee` to see everything that's not assigned.

© 2022 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

# CAPITOLO 3: Configuration Management (Pull request)

The screenshot shows a GitHub pull request interface. At the top, the browser address bar displays `github.com/giorgio-hash/progettoSE/pull/3`. The page title is "Diversificazione prodotti #3". Below the title, a green "Open" button is visible, followed by the text "giorgio-hash wants to merge 3 commits into master from diversificazione-prodotti".

The main content area shows a list of activities:

- A comment by giorgio-hash: "Ho pensato a una diversificazione dei prodotti utilizzando FeatureIDE con precompilatore ANTENNA."
- A commit by giorgio-hash: "added 3 commits 10 minutes ago". The commit list includes:
  - FeatureIDE per Domain Engineering (Verified, c16b67)
  - errore (Verified, 75b6bd)
  - FeatureIDE per diversificazione prodotti (Verified, 2a89c6e)
- An assignment by giorgio-hash: "assigned raffeledimaio 1 minute ago".
- Labels added by giorgio-hash: "enhancement" and "question" (13 seconds ago).

On the right side, there are sections for "Reviewers" (No reviews), "Assignees" (raffeledimaio), "Labels" (enhancement, question), "Projects" (None yet), "Milestone" (No milestone), and "Linked issues" (Successfully merging this pull request may close these issues).

At the bottom, a message states: "Add more commits by pushing to the diversificazione-prodotti branch on giorgio-hash/progettoSE."

# CAPITOLO 4: People Management and Team Organization

Le persone coinvolte nel progetto sono stati due studenti dell'Università degli studi di Bergamo:

- Giorgio Chirico;
- Raffaele Giacomo Giovanni Di Maio.

Il team è stato sempre composto da due sviluppatori che hanno ricoperto simultaneamente diversi ruoli, talvolta analoghi, utilizzando inoltre anche il paradigma del pair-programming.



# CAPITOLO 4: People Management and Team Organization

Per questa fase sono stati utilizzati 2 stili di organizzazione di team e persone:

- Relation style

Questa metodologia si basa sul lavoro innovativo, complesso e specializzato con decision-making basato su riunioni con il team.

- Integration style

Questa metodologia viene utilizzata quando il lavoro è di natura esplorativa e le varie task del progetto sono molto interdipendenti tra di loro.

# CAPITOLO 4: People Management and Team Organization

Il Relation style è stato utilizzato per creare l'idea del «Parcheggio a pagamento» e, attraverso una serie di riunioni settimanali, prendere decisioni su come il lavoro sarebbe poi stato portato avanti.

L'integration style invece è stato utilizzato nell'ambito della suddivisione del lavoro per la creazione di parti di codice che sarebbero poi state integrate tra di loro.

Questo per fare in modo che i pezzi di codice implementati non fossero slegati completamente rispetto al resto del programma.

# CAPITOLO 5: Software Quality

Nell'implementazione del nostro progetto ci siamo basati sul seguire la tabella dei Quality Factors.

In particolare troviamo:

- Correttezza

Ci siamo impegnati a rispettare quello che era stato deciso in sede di project plan centrando quindi tutti gli obiettivi prefissati volti alla realizzazione del nostro progetto.

- Affidabilità

Abbiamo fatto in modo che il programma risultasse affidabile eseguendo una serie di test utilizzando i vari prototipi da noi creati durante la realizzazione del progetto.



# CAPITOLO 5: Software Quality

- Efficienza

La realizzazione del nostro progetto ha portato alla scrittura di molte linee di codice che, nel corso dei vari refactoring effettuati, sono state snellite e quindi rese più efficienti.

- Integrità

Per l'accesso al database contenente il registro dei ticket erogati è presente l'accesso tramite autenticazione.

- Usabilità

L'applicazione è stata realizzata per risultare più user friendly possibile.

# CAPITOLO 5: Software Quality

- Manutenibilità

Per la manutenibilità del nostro progetto è richiesta una conoscenza di base della realtà implementata. Presa coscienza della realtà in esame, esso risulta facilmente manutenibile.

- Testabilità

E' stata creata una struttura modulare in modo da rendere più semplice eseguire test mirati alle singole zone di interesse.

- Flessibilità

E' stato fatto in modo che il codice fosse abbastanza flessibile per poter, in futuro, implementare il software per la gestione di un parcheggio a pagamento «barrier-less».

# CAPITOLO 5: Software Quality

- Portabilità

Il nostro software è portabile in quanto non dipende da nessun hardware (è richiesto l'utilizzo un qualsiasi computer general-purpose) nonostante sia comunque necessario utilizzare un ambiente di sviluppo Java.

- Riusabilità

Il codice risulta essere riusabile in quanto parti del nostro codice completo derivano, a loro volta, da altre parti di codice esistente.

- Interoperabilità

E' richiesta stabilità di connessione tra i nodi del sistema realizzato, tra server-database e tra server-circuito bancario.



# CAPITOLO 6: Requirement Engineering

L'ingegneria dei requisiti è descritta utilizzando lo standard IEEE 830.

## 1. Introduction

1.1 Purpose: l'obiettivo del nostro progetto è quello di realizzare un software che sia in grado di gestire i flussi di entrate e uscite da un parcheggio a pagamento introducendo un sistema di interfacce utente per i terminali e la possibilità di pagare con metodi diversi.

1.2 Scope: il software del parcheggio a pagamento contiene interfacce utente per i terminali di facile comprensione così da facilitare il pagamento, ritiro e consegna del ticket.

Inoltre è fatto in modo che esso sia fault tolerance grazie all'introduzione di un database che possa contenere i dati dei ticket in caso di blackout o malfunzionamenti generali.

# CAPITOLO 6: Requirement Engineering

## 2. Overall description

2.1 Product perspective: l'obiettivo del software è quello di permettere ai vari clienti di accedere al parcheggio attraverso il ritiro del biglietto presso il terminale di entrata e di uscire dallo stesso attraverso il terminale di uscita consegnando il biglietto e pagando la somma richiesta per la sosta al suo interno. Tutto questo meccanismo è regolato da un semaforo e da due sbarre posizionate rispettivamente presso il terminale di entrata e di uscita. Inoltre per garantire la fault tolerance è stato aggiunto un database che contiene al suo interno tutti i ticket degli autisti ancora presenti all'interno del parcheggio in modo da non perdere i dati.

# CAPITOLO 6: Requirement Engineering

2.2 Product function: il sistema prevede 4 funzioni principali:

- Erogazione del ticket per cliente in entrata;
- Ritiro e pagamento del ticket per cliente in uscita;
- Salvataggio del ticket erogato in entrata all'interno di un database;
- Controllo del semaforo e delle sbarre per la gestione corretta dei flussi di veicoli in entrata ed in uscita dal parcheggio.

2.3 User characteristics: gli utenti che si rivolgeranno a questo tipo di sistema sono coloro che guidano veicoli di ogni tipo e dimensione che necessita di un parcheggio dove sostare per qualsiasi motivo, da andare a fare una passeggiata fino ad andare a lavorare passando magari per fare delle compere.



# CAPITOLO 6: Requirement Engineering

2.4 Constrains: gli utenti sono autorizzati soltanto a ritirare, pagare e consegnare il ticket. Di conseguenza si limitano semplicemente ad usufruire del parcheggio per svolgere le loro attività senza mettere mano a questioni interne al software.

# CAPITOLO 6: Requirement Engineering

## 3. Specific requirements

### 3.1 External interface requirements

3.1.1 User Interfaces: le due user interfaces create svolgono due funzioni diverse. In particolare quella che si trova sul terminale d'entrata si occupa di erogare il ticket al guidatore nel caso in cui quest'ultimo preme il bottone per l'erogazione (e ci sia almeno un posto libero nel parcheggio), mentre quella sul terminale di uscita si occupa di ritirare il biglietto e di far pagare la somma prestabilita.

3.1.2 Hardware interfaces: l'interfaccia utente è screen-oriented.

# CAPITOLO 6: Requirement Engineering

## 3.1 External interface requirements

3.1.3 External Software Interfaces: a livello di interfacce software esterne al progetto sono state utilizzate:

- la libreria prog4ed per la simulazione dell'input ticket e pagamento;
- la libreria Connectors di Java per la comunicazione con il database su MySQL.

3.1.4 Communication interfaces: l'interfacciamento database è stato creato utilizzando MySQL con la libreria Connectors di Java. La comunicazione, su MySQL, è di tipo socket TCP con autenticazione. Inoltre il terminale d'entrata/uscita comunica con il GestoreParcheggio attraverso porte socket.



# CAPITOLO 6: Requirement Engineering

## 3.1 Functional requirements

3.1.1 Interfaccia d'entrata: uso di libreria Swing con implementazione CardLayout;

3.1.2 Interfaccia d'uscita: uso di libreria Swing con implementazione CardLayout;

# CAPITOLO 6: Requirement Engineering

## 3.1 Functional requirements

3.1.3 Erogazione ticket: comunicazione socket con server per controllare disponibilità posti così da erogare il ticket in caso di riscontro positivo.

3.1.4 Ritiro e pagamento ticket: comunicazione socket per validazione ticket e pagamento di quest'ultimo.

# CAPITOLO 6: Requirement Engineering

## 3.1 Functional requirements

3.1.5 Semafori: lo sviluppo di un'applicazione real time ha portato all'uso dei semafori per la gestione dei flussi di entrata/uscita dal parcheggio.



# CAPITOLO 6: Requirement Engineering

3.3 Performance requirements: il sistema dovrà supportare un massimo di due flussi simultanei.

E' necessaria una connessione TCP stabile tra i vari componenti interconnessi tra loro.

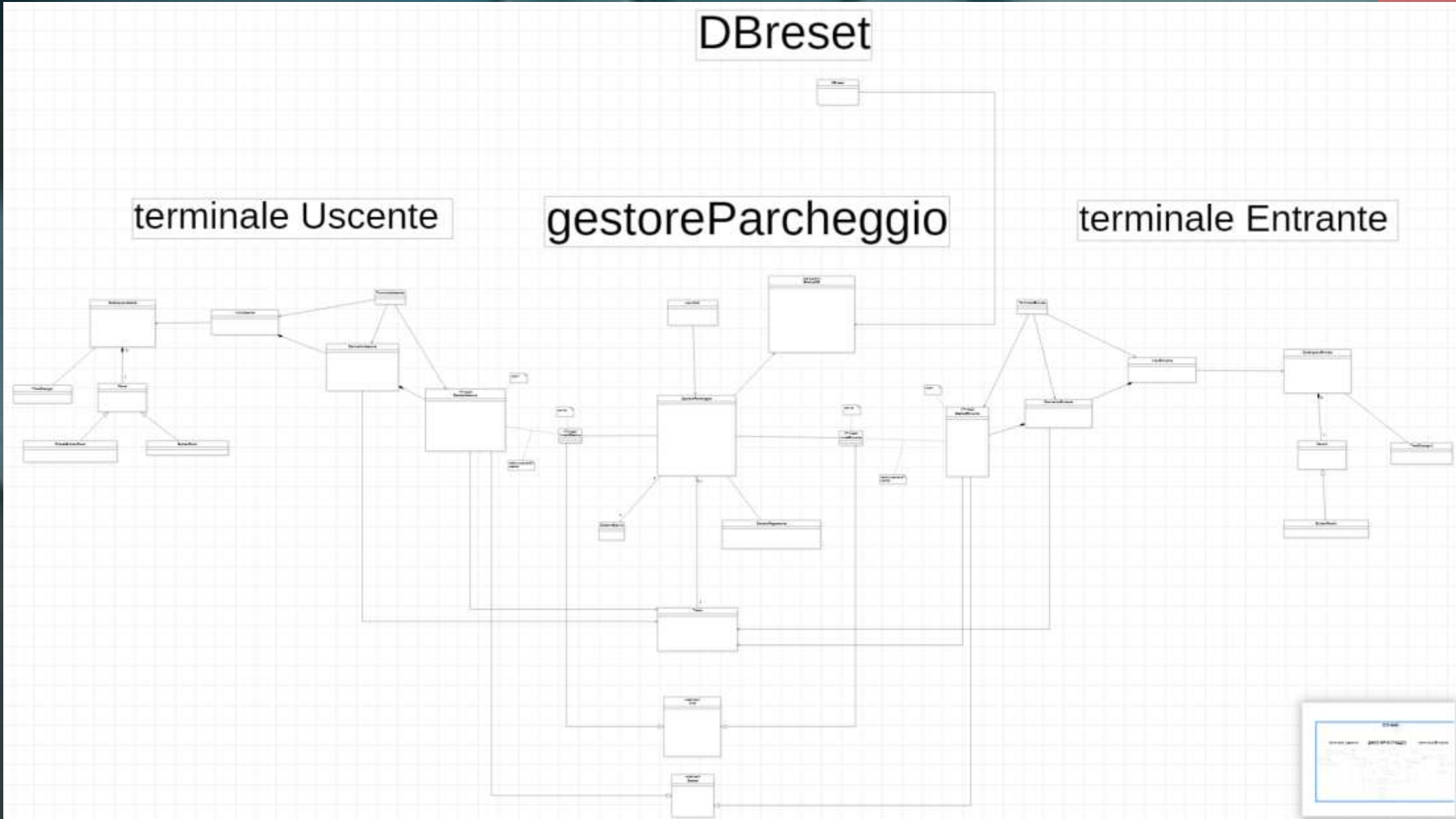
Si vuole evitare una possibile starvation per i due terminali d'entrata e d'uscita.

The background of the slide is a blurred image of a desk. A pair of black-rimmed glasses is resting on a stack of papers. A red pen is visible on the left side of the papers. In the top right corner, there is a solid red rectangular block.

# CAPITOLO 6: Requirement Engineering

3.5 Software system attributes: tutte le caratteristiche di correttezza, affidabilità, efficienza, ecc legate al nostro software sono disponibili al Capitolo 5: Software Quality.

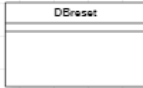
# CAPITOLO 7: Modelling(Completo)



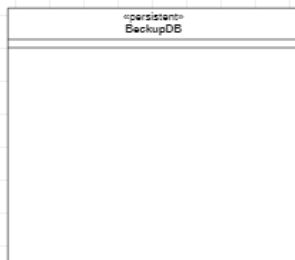


# CAPITOLO 7: Modelling(DBReset)

## DBreset



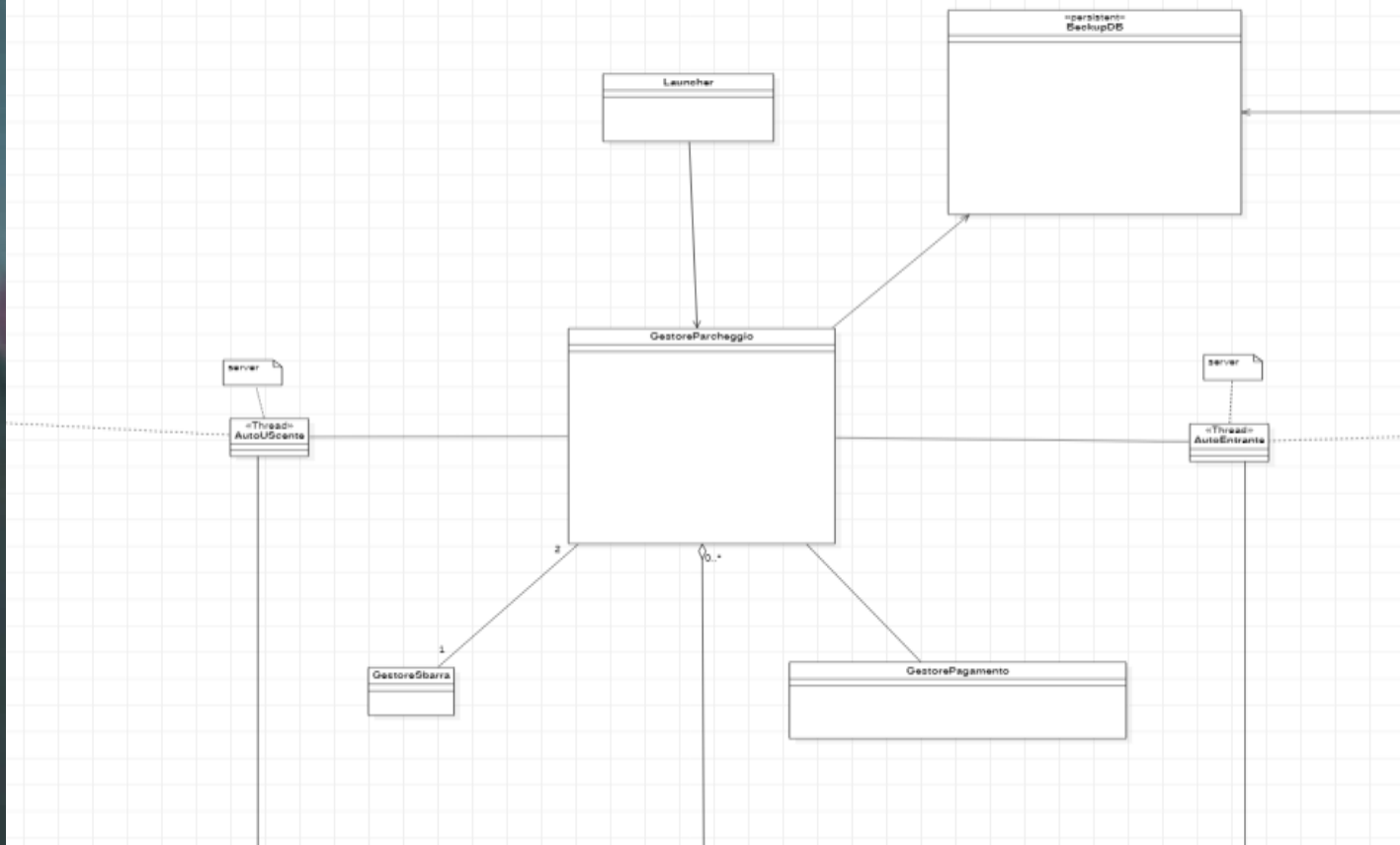
## Parcheggio



# CAPITOLO 7:

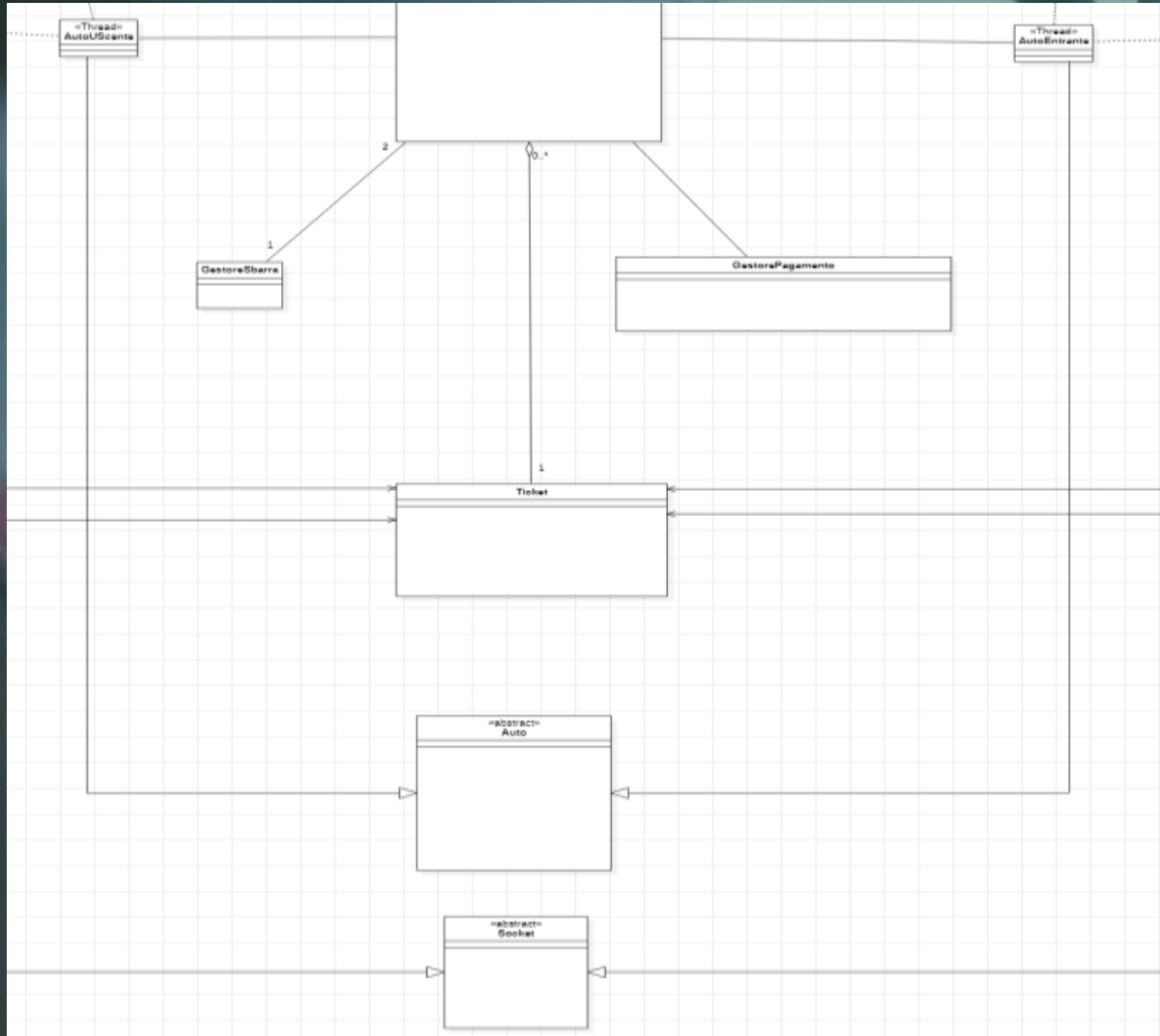
## Modelling (GestoreParcheggio 1/2)

### gestoreParcheggio



# CAPITOLO 7:

## Modelling (GestoreParcheggio2/2)

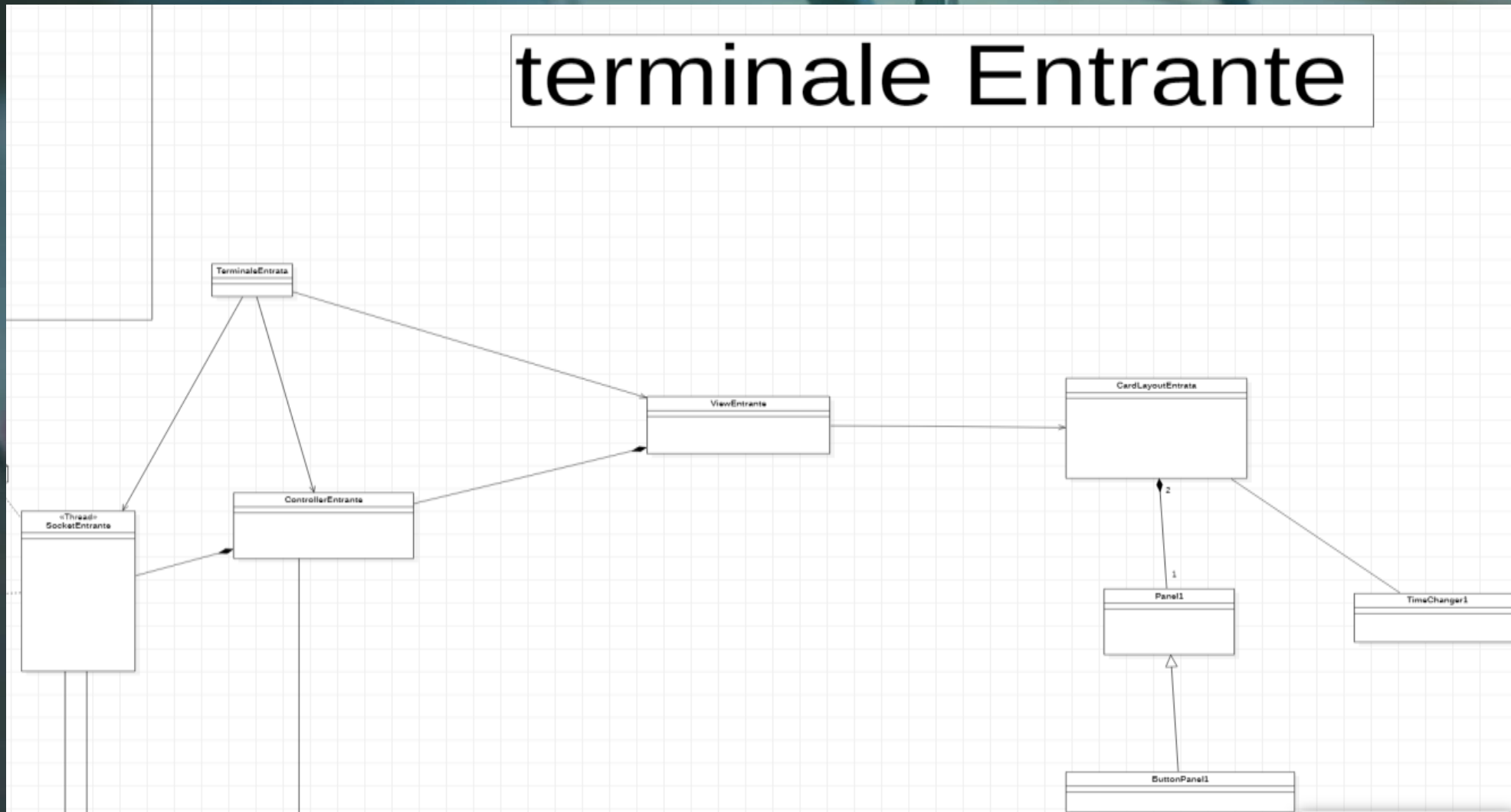




# CAPITOLO 7:

## Modelling(TerminaleEntrante)

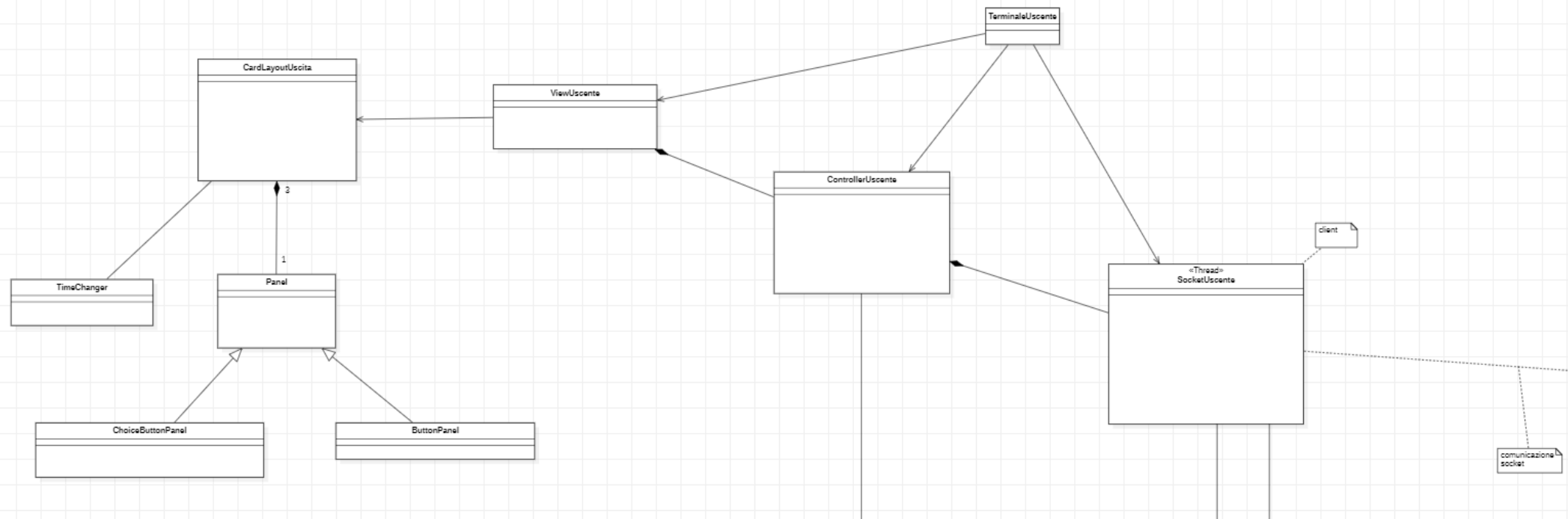
terminale Entrante



# CAPITOLO 7:

## Modelling(TerminaleUscente)

### terminale Uscente



# CAPITOLO 8: Software Architecture

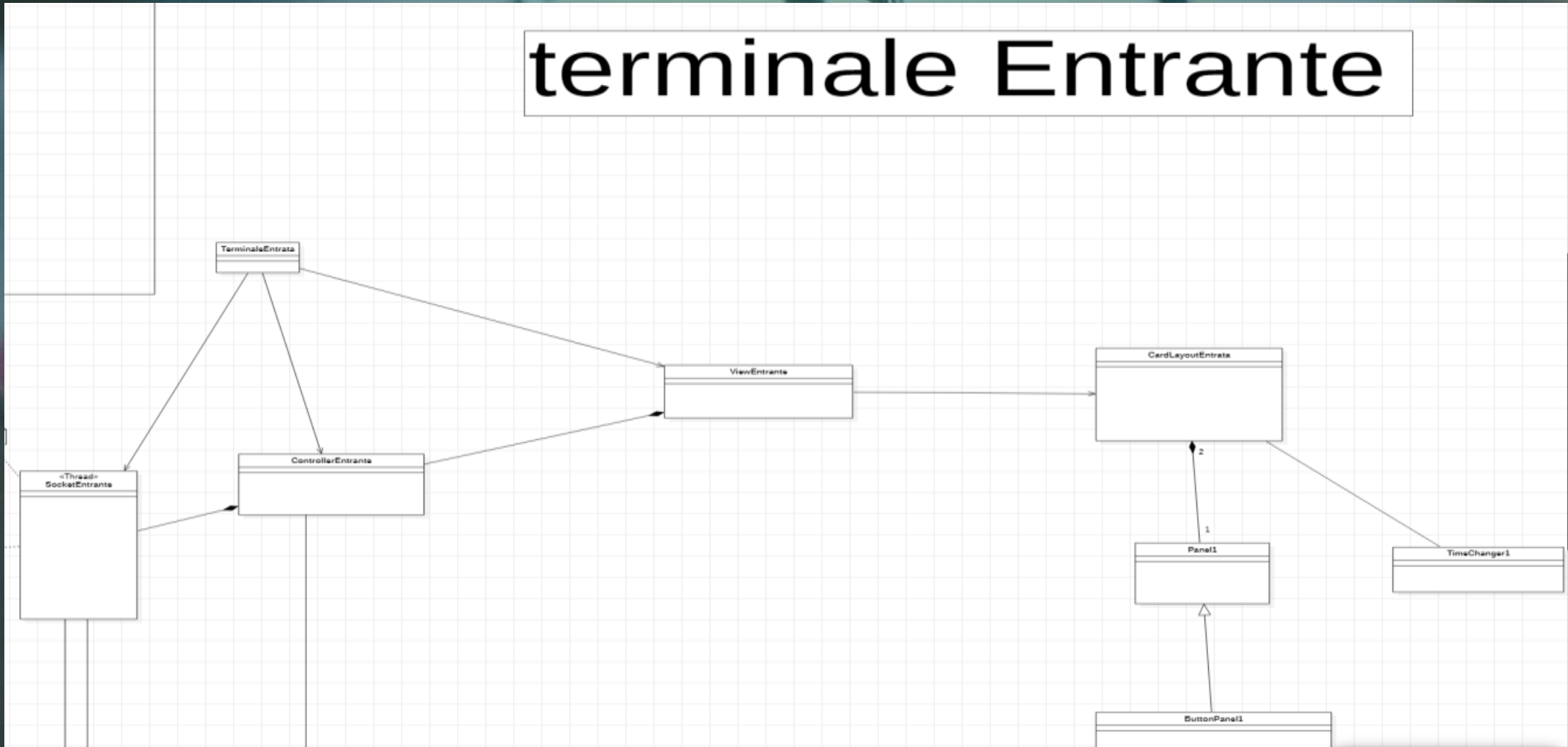
Come modello architetturale per la realizzazione del nostro progetto abbiamo scelto il Model-View-Controller. MVC prevede un'architettura composta da tre parti diverse: i dati (Model), la visualizzazione dei dati (View) e la gestione degli input (Controller). Questi tre componenti sono interconnessi: lo stato del sistema (Model) viene mostrato tramite la View all'utente, il quale produce gli input con cui il Controller aggiorna il Model. Nel nostro caso, la View è la parte di interfaccia utente che si trova sui due terminali di entrata e uscita; il Controller è la parte di gestione del parcheggio che si occupa dello scambio di messaggi tra i terminali ed il GestoreParcheggio per regolare i flussi di entrata e di uscita; il Model è la parte contenente tutti i ticket che sono stati erogati e che sono ancora presenti all'interno del database e quindi rappresenta tutti quei clienti non ancora usciti dal parcheggio.



# CAPITOLO 8: Software Architecture

## Architectural View Terminale Entrante

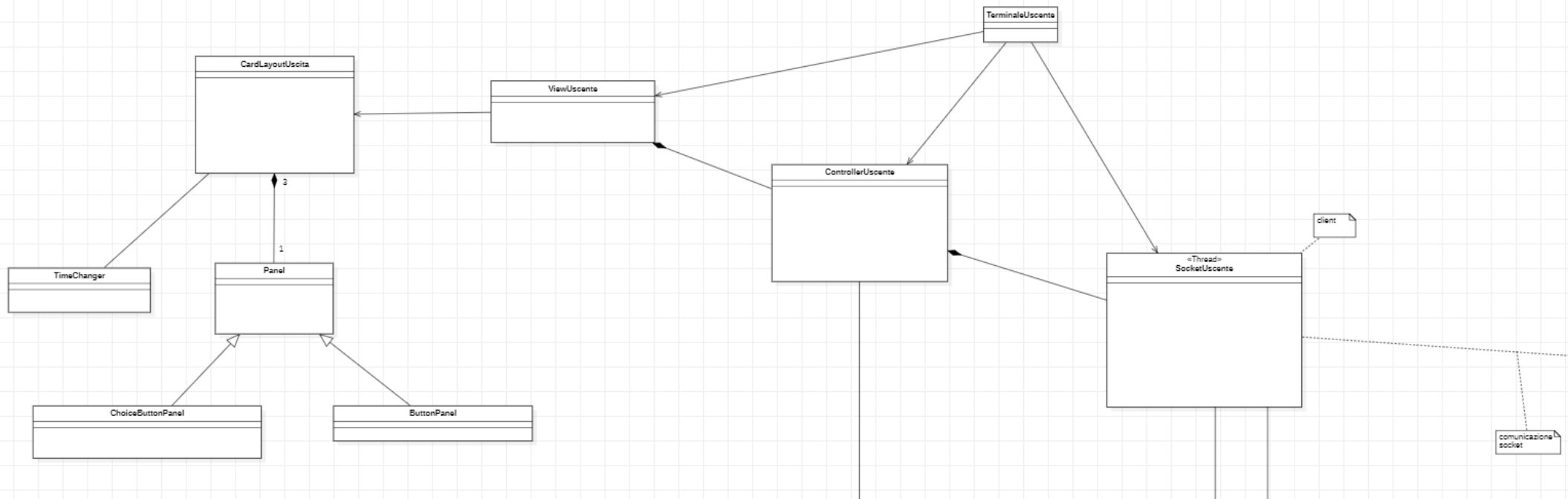
### terminale Entrante



# CAPITOLO 8: Software Architecture

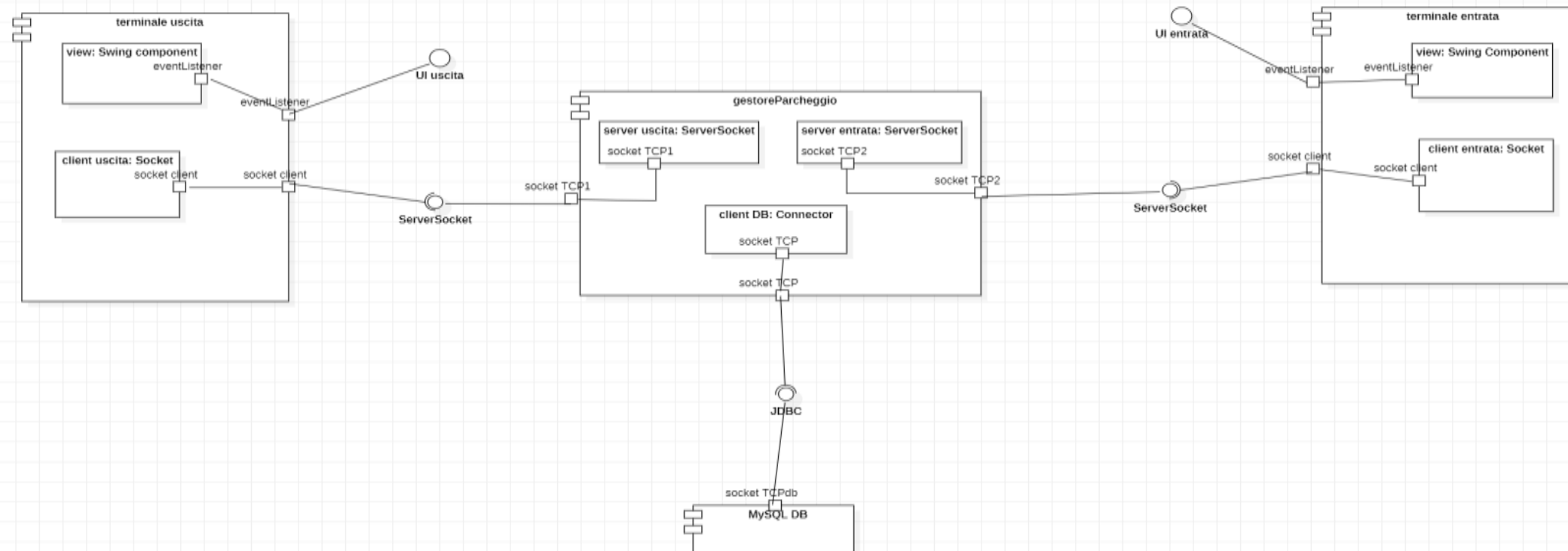
Architectural View Terminale Uscente

## terminale Uscente



# CAPITOLO 8: Software Architecture

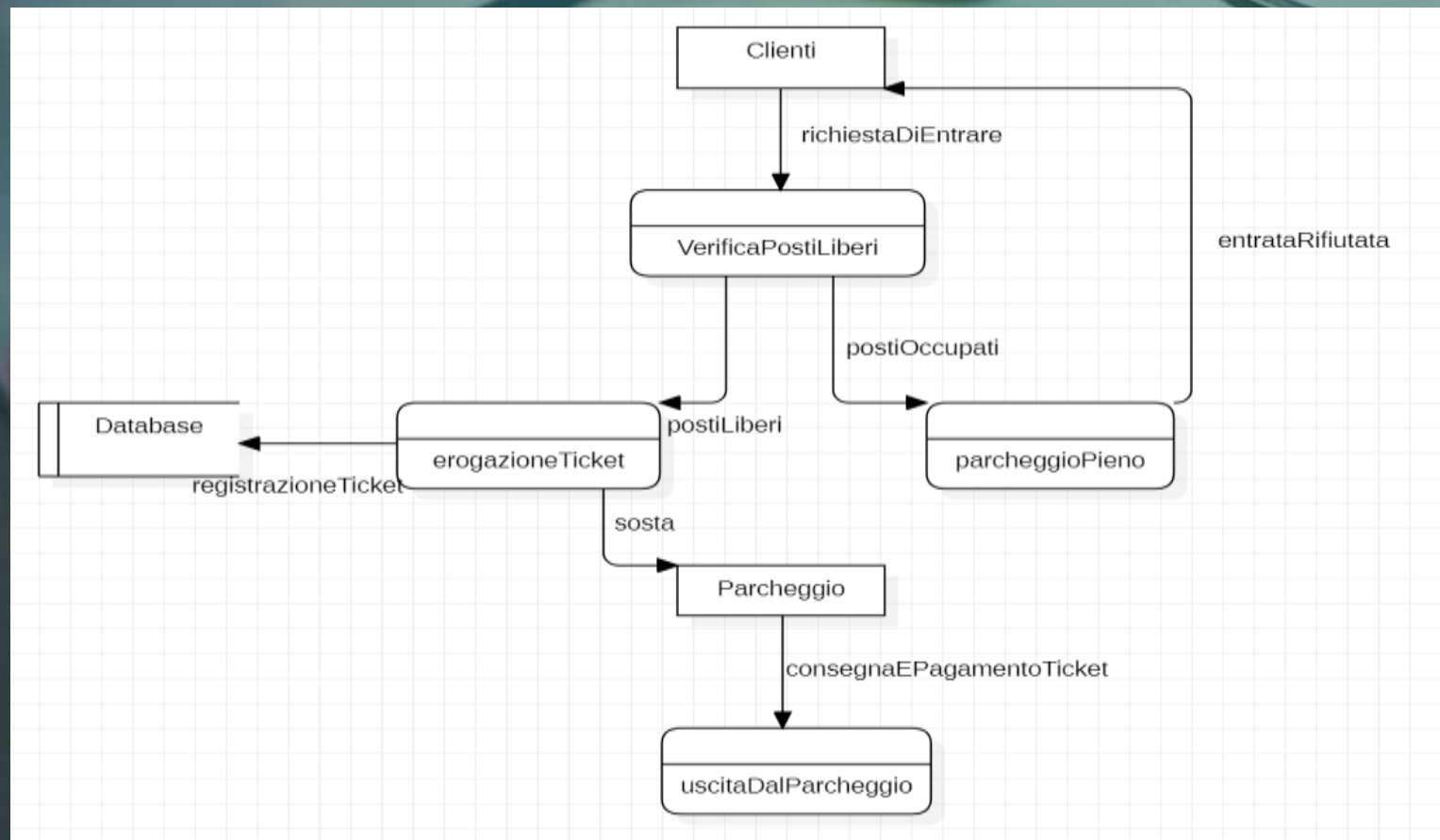
Vista connettori e componenti





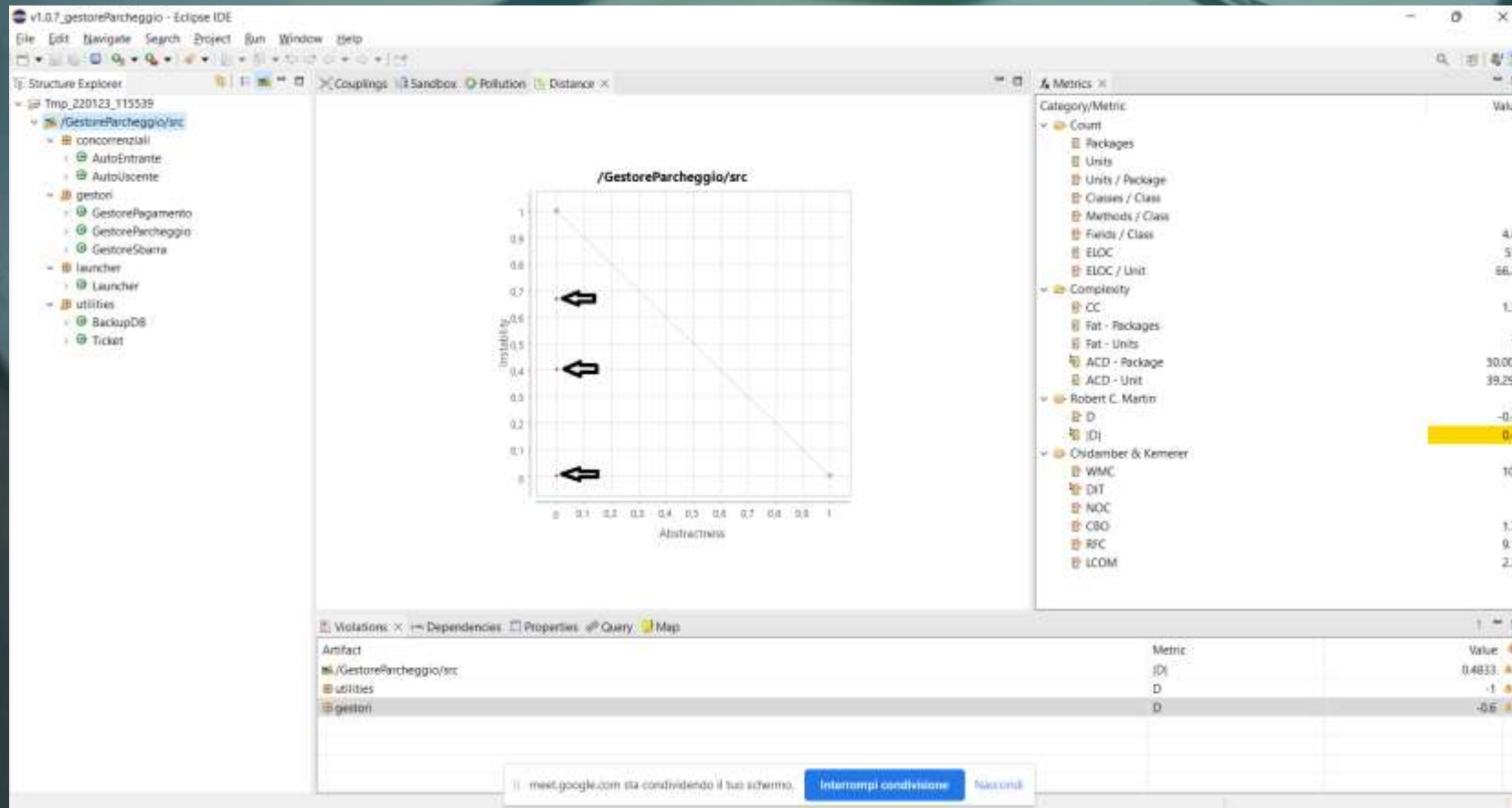
# CAPITOLO 9: Software Design

## Design Architecture:



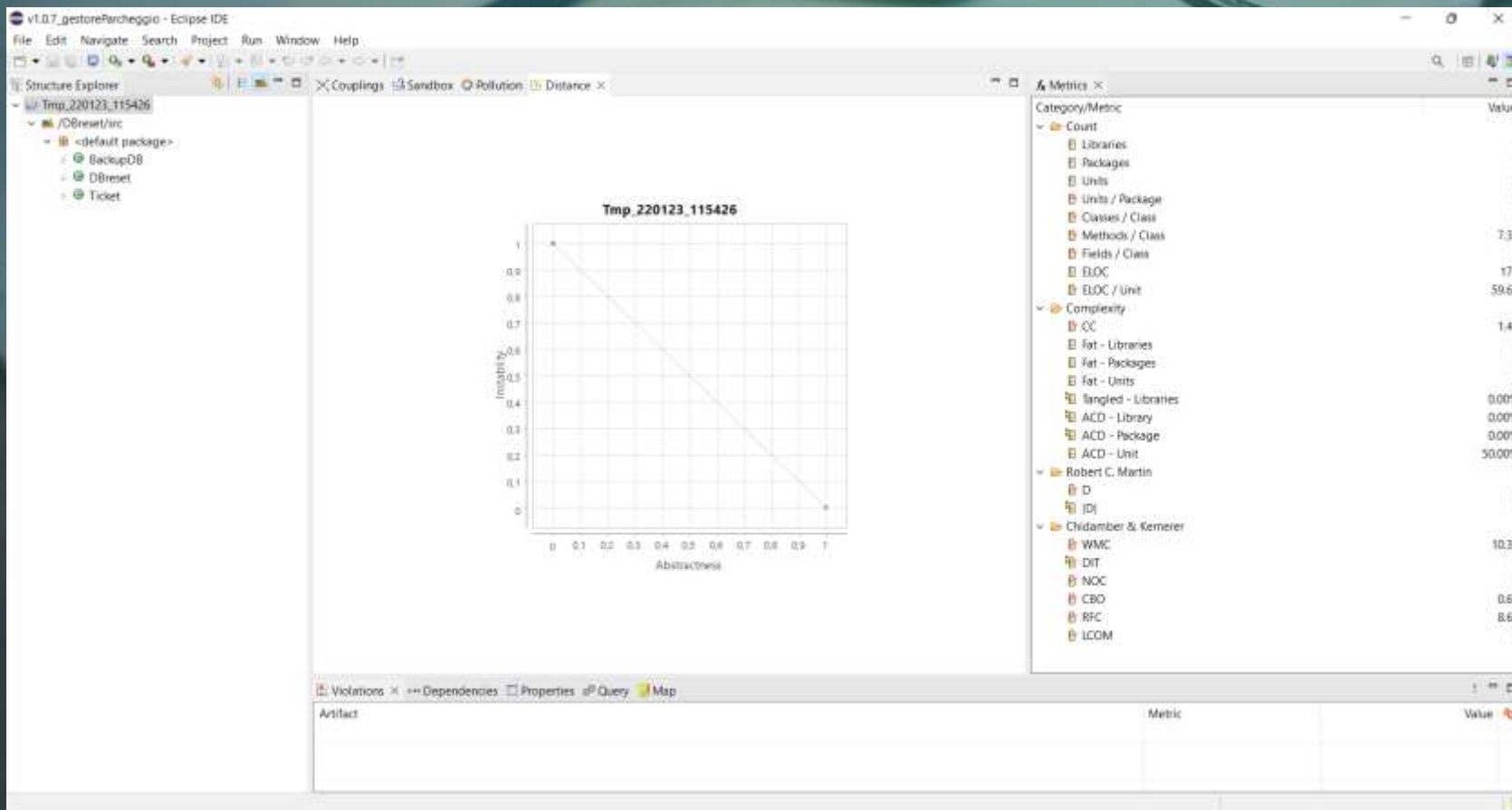
# CAPITOLO 9: Software Design

## Analisi STAN IDE parcheggio:



# CAPITOLO 9: Software Design

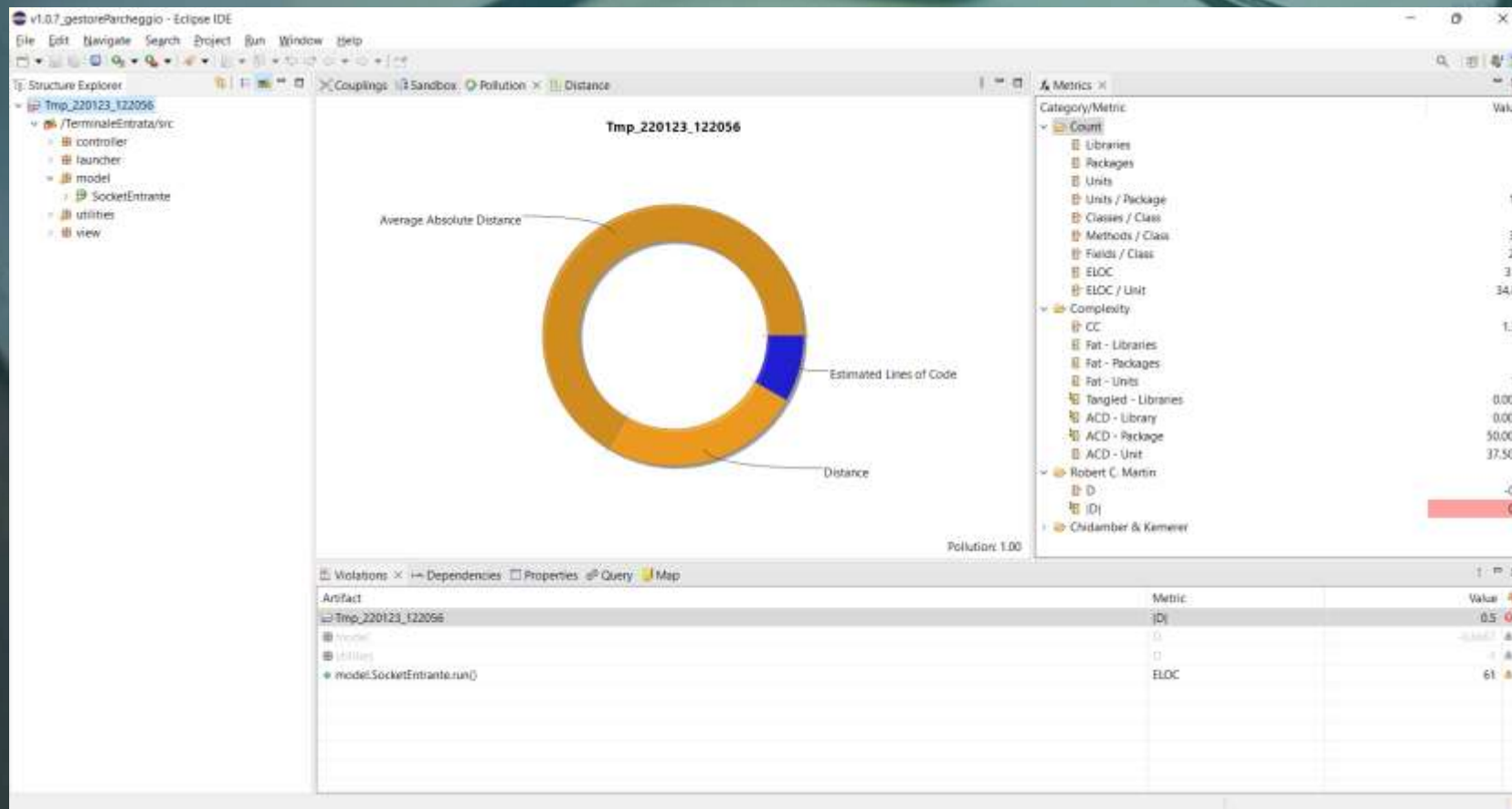
## Analisi STAN IDE DBreset:





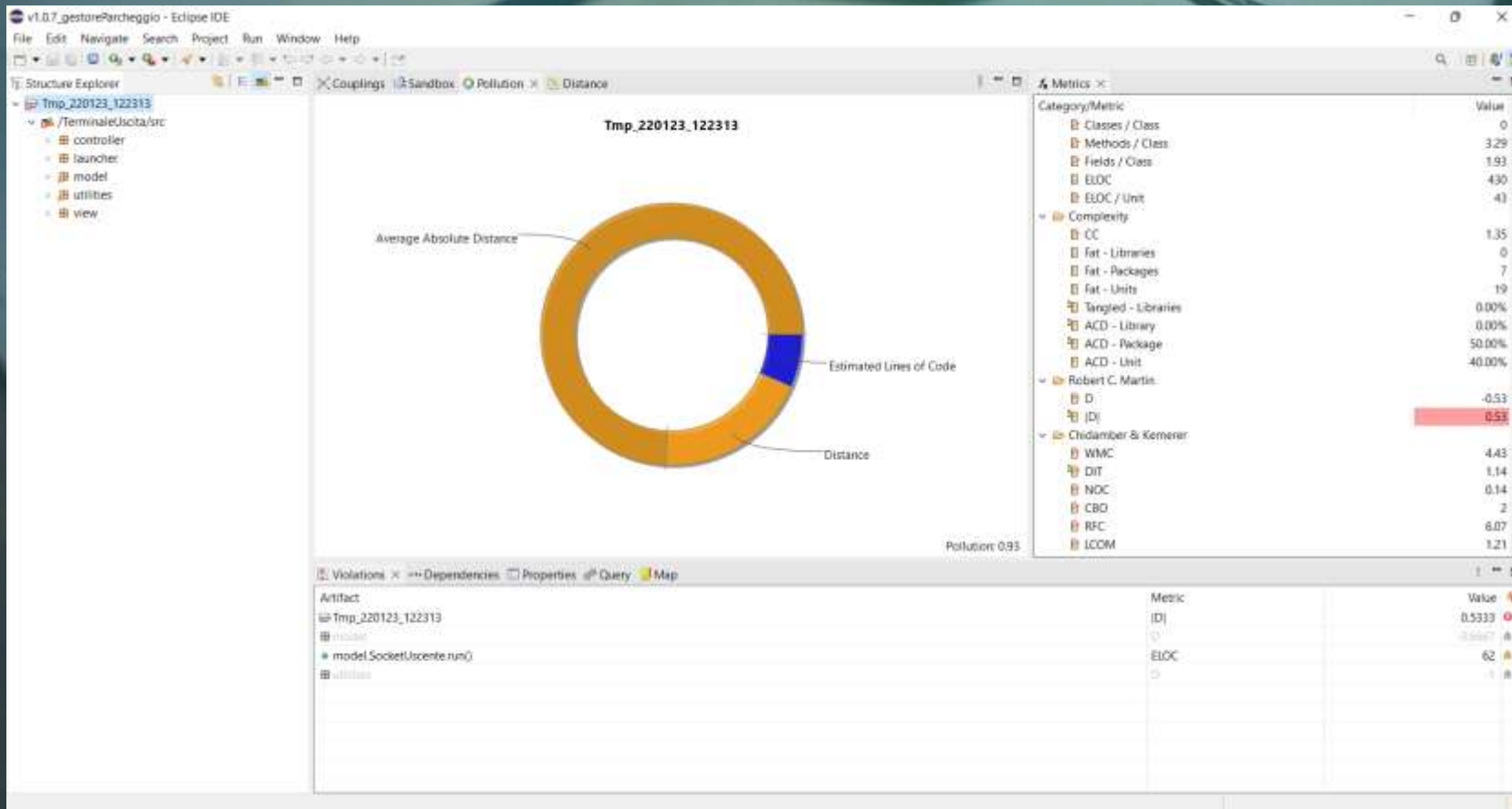
# CAPITOLO 9: Software Design

Analisi STAN IDE terminale entrante:



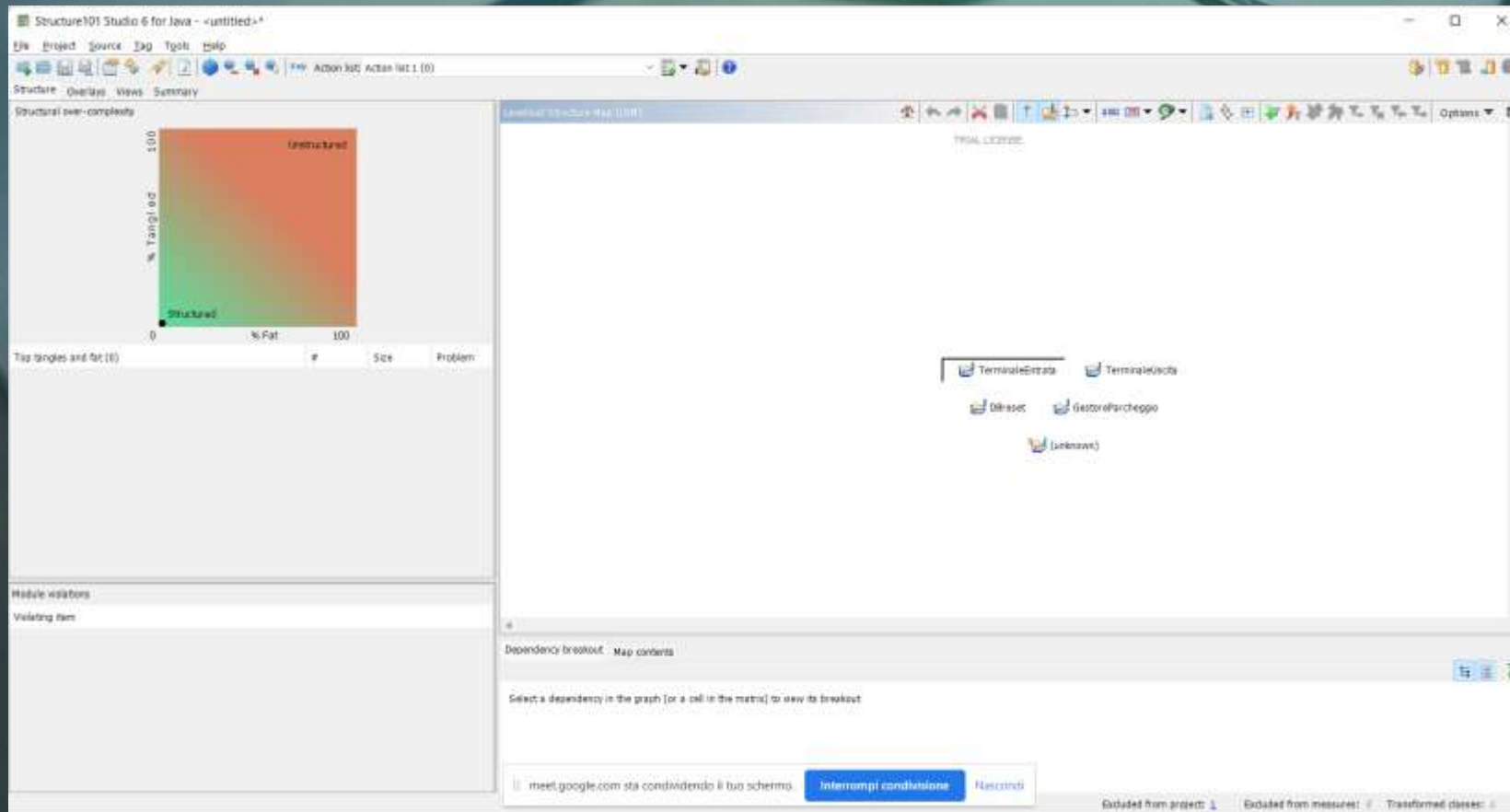
# CAPITOLO 9: Software Design

Analisi STAN IDE terminale uscente:



# CAPITOLO 9: Software Design

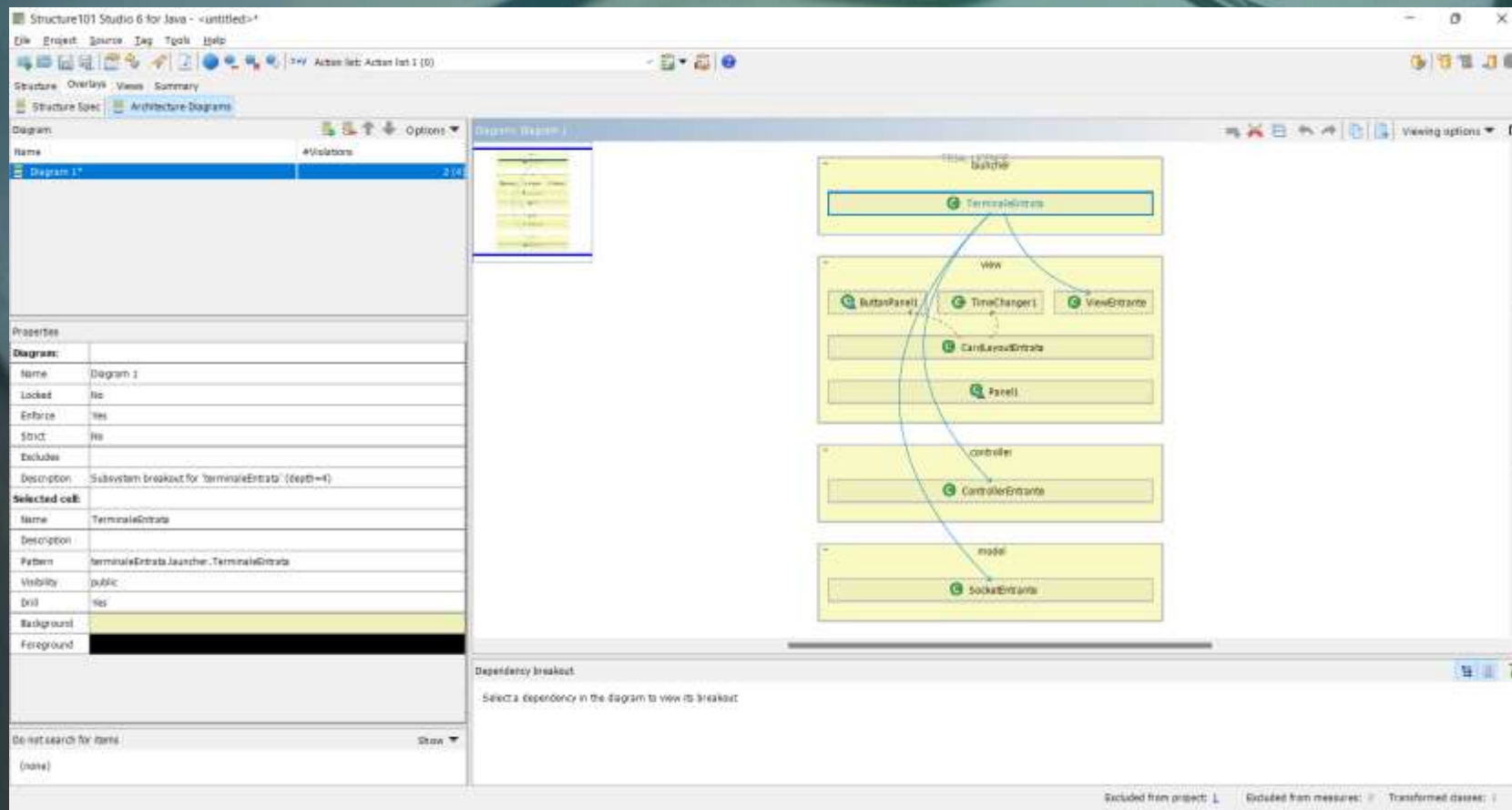
Analisi structure101 generica:





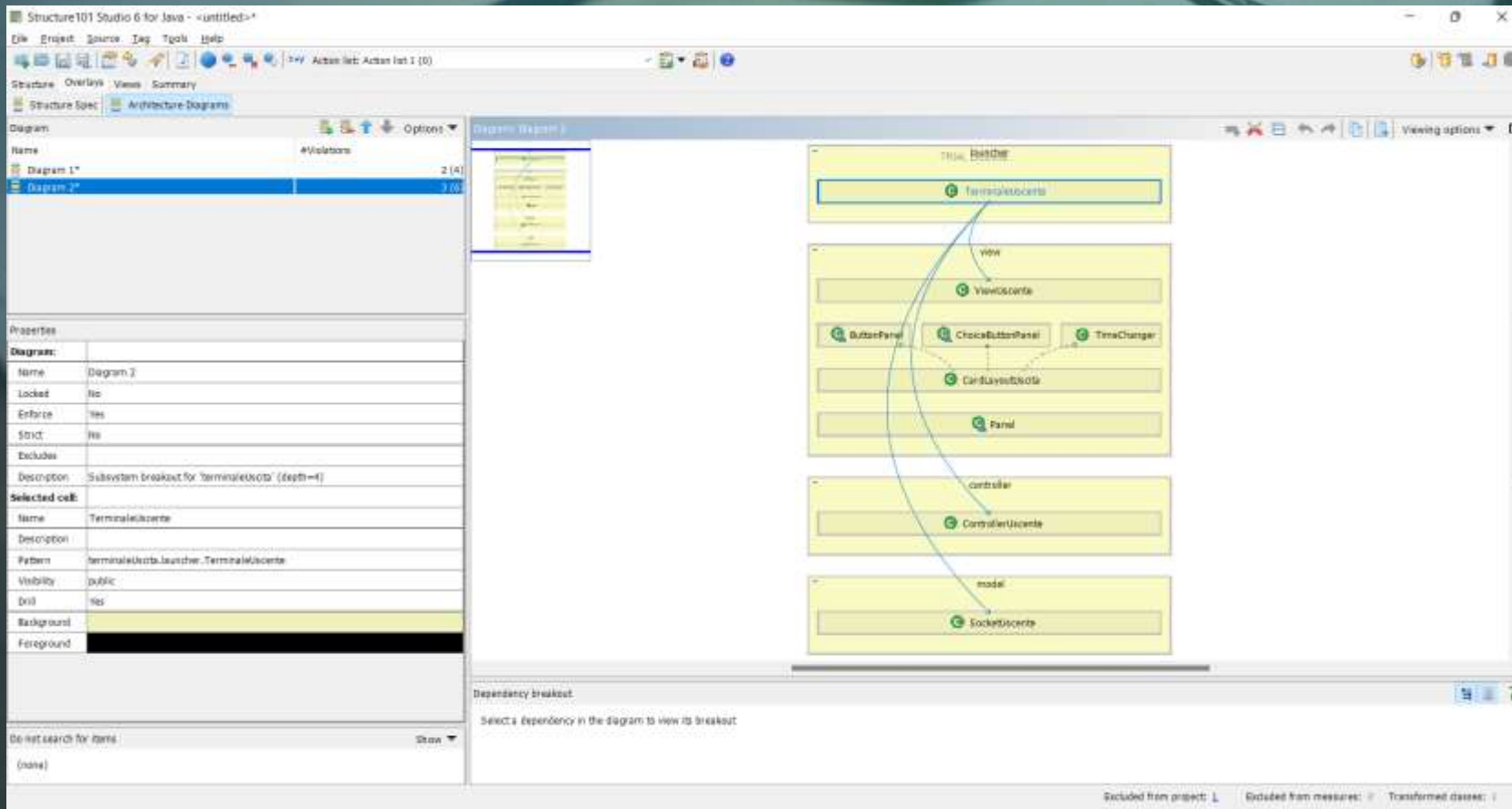
# CAPITOLO 9: Software Design

Analisi structure101 terminale entrata:



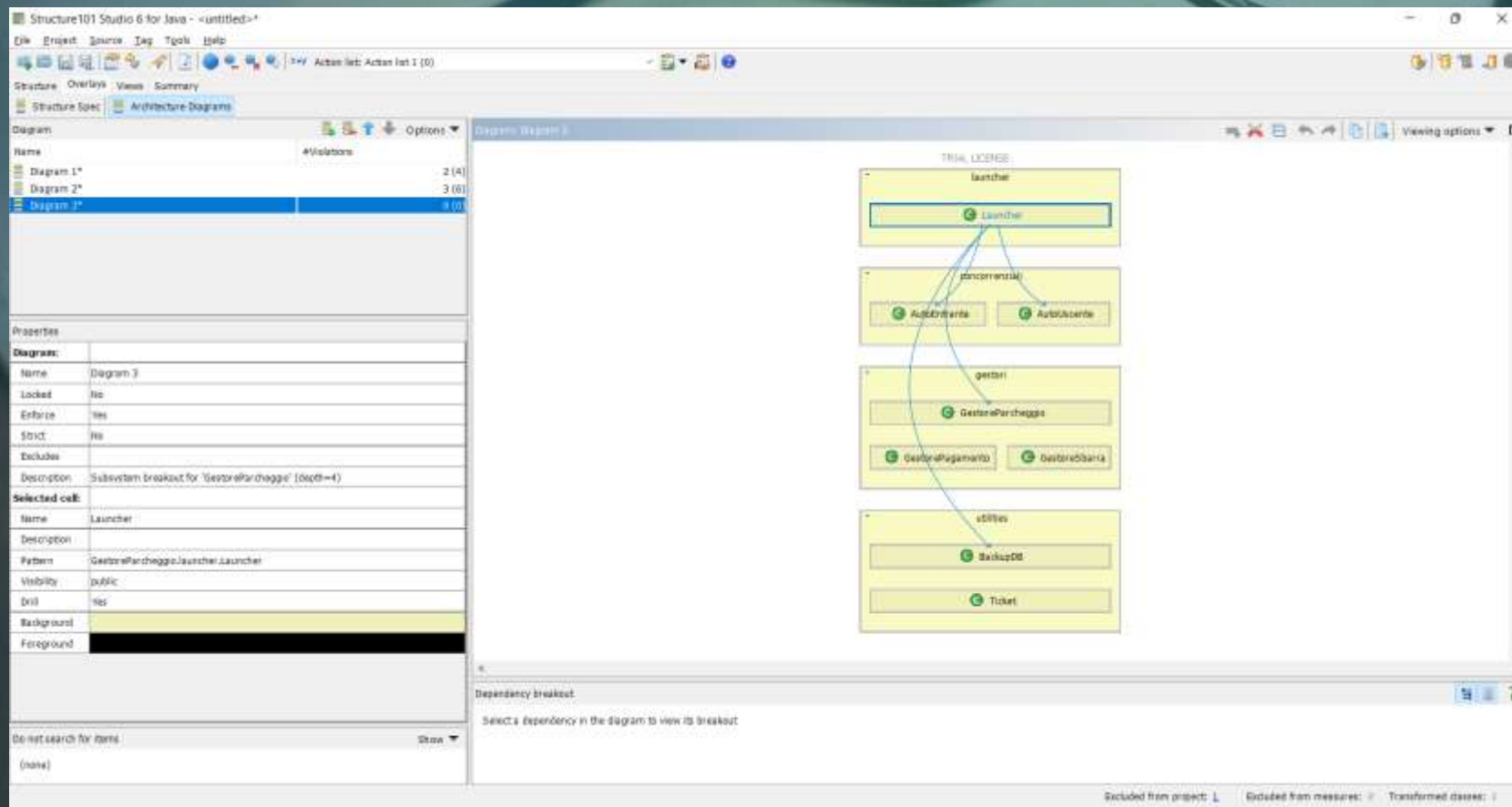
# CAPITOLO 9: Software Design

Analisi structure101 terminale uscente:



# CAPITOLO 9: Software Design

Analisi structure101 server:





# CAPITOLO 9: Software Design

Analisi structure101 totale:

The screenshot displays the Structure101 Studio 6 for Java interface. The main window shows a software design diagram with three columns representing different components: **terminaInizio**, **GestoreParcheggi**, and **terminaEntrata**. Each column contains a hierarchy of elements: **launcher**, **view**, **controller**, and **model**. Below these, there are **DBreset** and **BackupDB** elements. At the bottom, there are **com**, **prog**, and **Ticket** elements. The diagram is connected by numerous dashed lines representing dependencies.

On the left side, the **Diagram** pane lists several diagrams, with **Diagram 5** selected. Below this, the **Properties** pane shows the details for **Diagram 5**, including its name, locked status, enforced status, strict status, excluded status, and description (**Top-level breakout**). The **Selected cell** pane shows the details for the selected cell, including its name, description, pattern, visibility, drill status, background, and foreground.

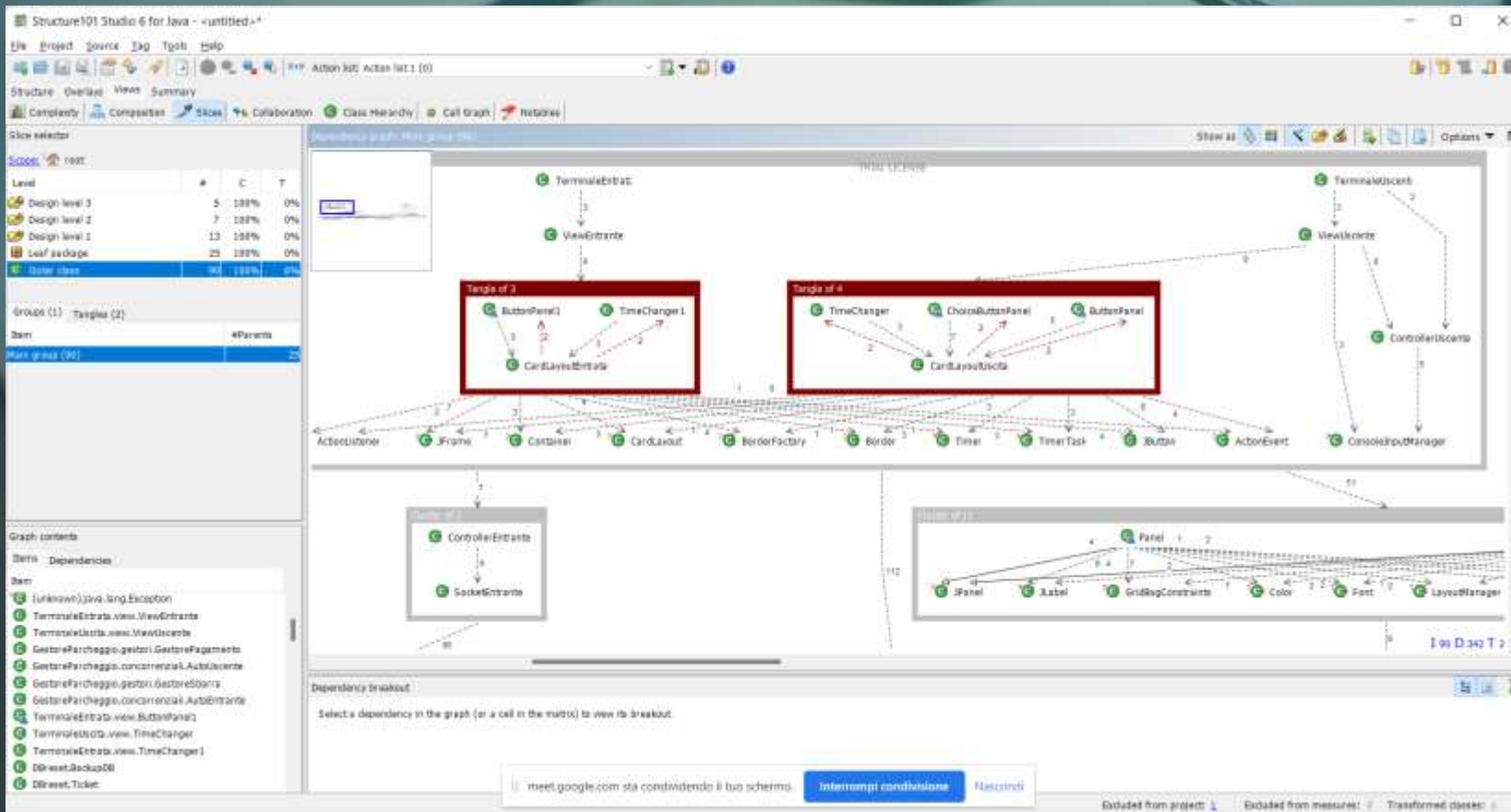
At the bottom, the **Dependency breakout** pane shows a list of dependencies, with a note to "Select a dependency in the diagram to view its breakout".

The **Diagram** pane also includes a table showing the number of violations for each diagram:

Diagram	#Violations
Diagram 1*	2 (4)
Diagram 2*	3 (6)
Diagram 3*	0 (0)
Diagram 4*	0 (0)
Diagram 5*	3 (11)

A tooltip indicates that the number in parentheses represents the **Number of weighted violations (number of unique violations)**.

# Analisi structure101 cicli controllati CardLayout:



# CAPITOLO 9: Software Design(Pattern applicati)

Delegation Pattern: l'obiettivo è quello di minimizzare il costo di sviluppo del software andando a riutilizzare dei metodi già creati. L'uso di questo pattern è presente nella creazione delle due interfacce utente presenti sui terminali di entrata ed uscita.

Observer Pattern: l'obiettivo è quello di massimizzare la flessibilità del sistema permettendo a tutti gli oggetti, dipendenti da un altro oggetto, di essere notificati e quindi cambiare il loro stato se il «super-oggetto» lo cambia.

L'uso di questo pattern è presente nella creazione dei terminali di entrata ed uscita.



# CAPITOLO 10: Software Testing

Nel corso dello sviluppo del software sono state effettuate una serie di attività di testing.

In particolare è stato adottato il Manual Testing cioè una procedura per cui i test sul software sono eseguiti manualmente dai programmatori senza utilizzo di strumenti automatizzati.

Il testing viene utilizzato per identificare eventuali bug, problemi e difetti nell'applicativo software.

Il Manual testing ha come caratteristica quella di aiutare particolarmente a trovare bug critici.

Lo svantaggio è che necessita di un maggiore sforzo per valutare la fattibilità dell'automazione.

L'obiettivo principale del Manual Testing è che l'applicazione sia priva di errori e che il software soddisfi i functional requirements.



# CAPITOLO 10: Software Testing

La maggior parte dei testing effettuati si sono basati sui paradigmi dell'Integration testing e del White Box testing.

Integration testing → Un livello di test che si concentra sulle interazioni tra componenti o sistemi.

L'Integration Testing è un'attività che serve a verificare:

- l'integrazione tra le varie unità/moduli sviluppati;
- l'interazione dei moduli con il sistema;
- che i requisiti software sia a basso livello che ad alto livello siano soddisfatti.

# CAPITOLO 10: Software Testing

Gli obiettivi dell'attività di Integration Testing sono i seguenti:

- testare l'integrazione tra componenti appena possibile in modo da ridurre il rischio;
- verificare che i comportamenti funzionali e non funzionali dei moduli siano progettati come da specifiche;
- trovare eventuali difetti nelle integrazioni tra moduli, interfacce, API, micro-servizi, database, strumenti di terze parti, sistema, sotto-sistemi;
- trovare eventuali difetti quando vengono apportate modifiche a moduli, componenti, interfacce, ecc...;
- cercare di evitare di far passare i difetti ai livelli successivi di test che ne aumenterebbero la complessità.

# CAPITOLO 10: Software Testing

White Box testing → Un livello di test in cui le strutture interne, il design ed il codice del software sono testati per verificare i flussi di input-output e per migliorare la sicurezza e l'usabilità.

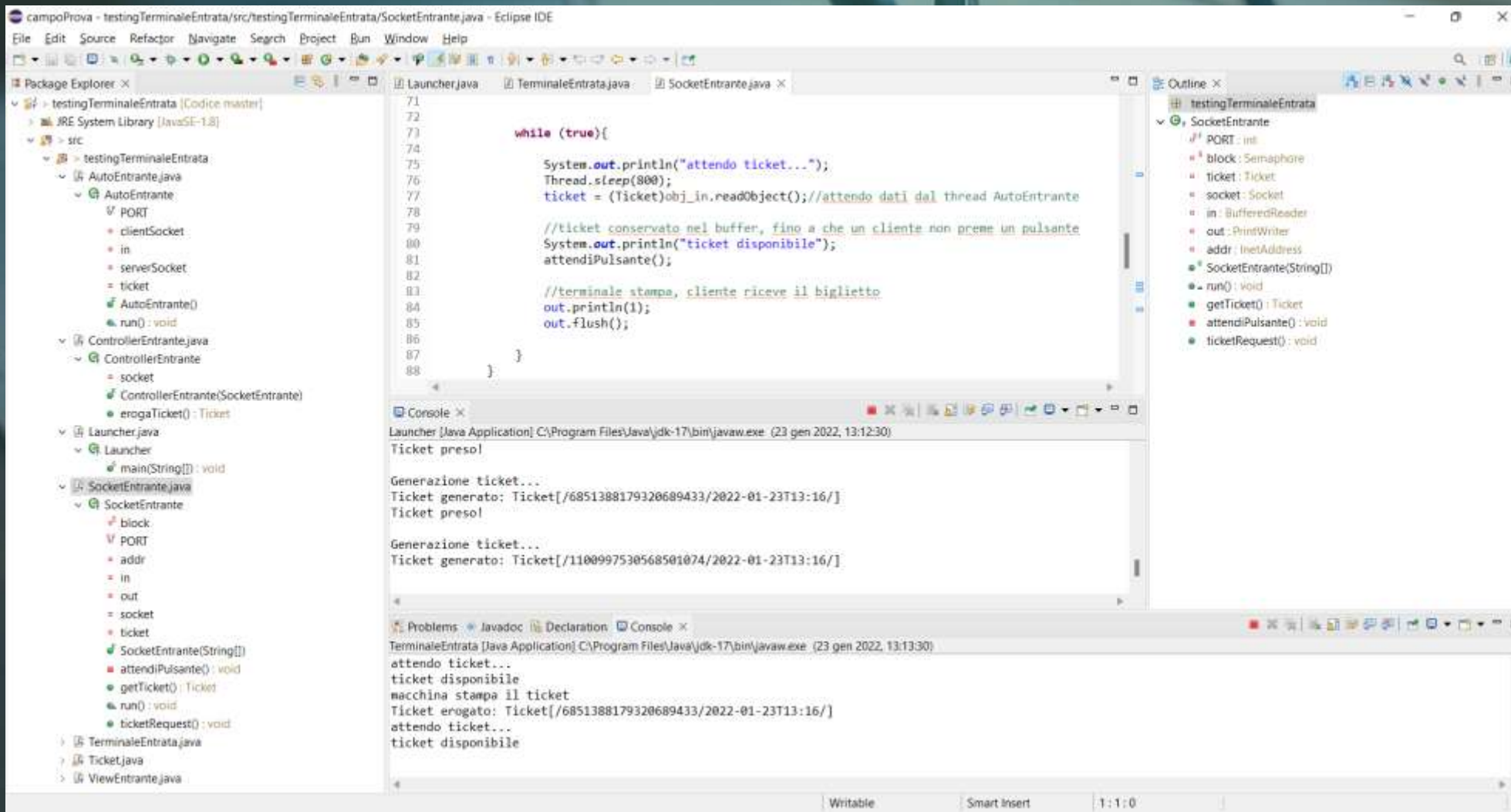
In questo tipo di testing il codice è visibile a coloro che lo testano.

Il White Box testing è un'attività che serve a verificare:

- eventuali problemi di sicurezza interni al software;
- il flusso degli input;
- i valori attesi in output;
- il testare individualmente qualsiasi componente del codice software;
- verificare che il codice sia scritto bene;
- alcune funzionalità all'interno di possibili cicli.



# CAPITOLO 10: Software Testing



The screenshot displays the Eclipse IDE interface for a project named 'campoProva'. The package explorer on the left shows the project structure, including the 'testingTerminaleEntrata' package and its sub-packages. The main editor window shows the 'SocketEntrante.java' file, which contains a while loop that generates and prints tickets. The console window at the bottom shows the output of the program, including the generation of tickets and the availability of tickets.

```
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

```
while (true){  
    System.out.println("attendo ticket...");  
    Thread.sleep(800);  
    ticket = (Ticket)obj_in.readObject();//attendo dati dal thread AutoEntrante  
  
    //ticket conservato nel buffer, fino a che un cliente non preme un pulsante  
    System.out.println("ticket disponibile");  
    attendiPulsante();  
  
    //terminale stampa, cliente riceve il biglietto  
    out.println(1);  
    out.flush();  
}
```

Launcher [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (23 gen 2022, 13:12:30)  
Ticket preso!  
  
Generazione ticket...  
Ticket generato: Ticket[/6851388179320689433/2022-01-23T13:16/]  
Ticket preso!  
  
Generazione ticket...  
Ticket generato: Ticket[/1100997530568501074/2022-01-23T13:16/]

TerminaleEntrata [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (23 gen 2022, 13:13:30)  
attendo ticket...  
ticket disponibile  
macchina stampa il ticket  
Ticket erogato: Ticket[/6851388179320689433/2022-01-23T13:16/]  
attendo ticket...  
ticket disponibile



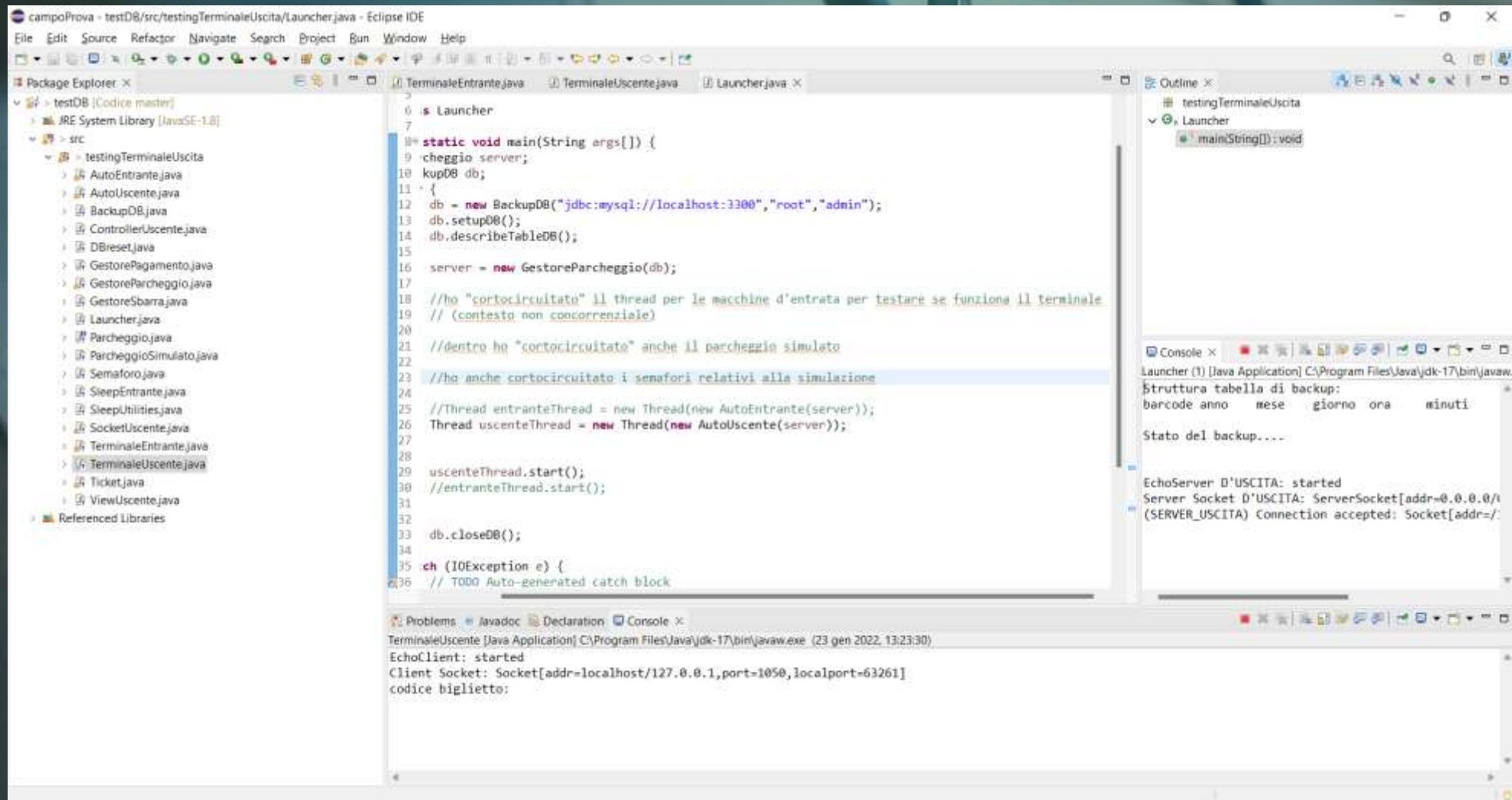
# CAPITOLO 10: Software Testing

Nell'immagine precedente è stato testato il prototipo `v0.4.0_testingTerminaleEntrata_v2.0`.

In questo caso di testing abbiamo verificato il corretto funzionamento dello scambio di messaggi tra terminale d'entrata e server, ovvero della corretta generazione e passaggio dei ticket.

L'esecuzione del test è automatizzata.

# CAPITOLO 10: Software Testing



```
campoProva - testDB/src/testingTerminaleUscita/Launcher.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer:
- testDB [Codice master]
  - JRE System Library [JavaSE-1.8]
  - src
    - testingTerminaleUscita
      - AutoEntrante.java
      - AutoUsciente.java
      - BackupDB.java
      - ControllerUsciente.java
      - DBreset.java
      - GestorePagamento.java
      - GestoreParcheggio.java
      - GestoreSbarra.java
      - Launcher.java
      - Parcheggio.java
      - ParcheggioSimulato.java
      - Semaforo.java
      - SleepEntrante.java
      - SleepUtilities.java
      - SocketUsciente.java
      - TerminaleEntrante.java
      - TerminaleUsciente.java
      - Ticket.java
      - ViewUsciente.java
  - Referenced Libraries

TerminaleEntrante.java TerminaleUsciente.java Launcher.java
6 Launcher
7
8 static void main(String args[]) {
9     chggio server;
10    kupDB db;
11    {
12        db = new BackupDB("jdbc:mysql://localhost:3300","root","admin");
13        db.setupDB();
14        db.describeTableDB();
15    }
16    server = new GestoreParcheggio(db);
17
18    //ho "cortocircuitato" il thread per le macchine d'entrata per testare se funziona il terminale
19    // (contesto non concorrenziale)
20
21    //dentro ho "cortocircuitato" anche il parcheggio simulato
22
23    //ho anche cortocircuitato i semafori relativi alla simulazione
24
25    //Thread entranteThread = new Thread(new AutoEntrante(server));
26    Thread uscenteThread = new Thread(new AutoUsciente(server));
27
28    uscenteThread.start();
29    //entranteThread.start();
30
31
32    db.closeDB();
33
34
35    ch (IOException e) {
36        // TODO Auto-generated catch block
37    }
38 }

Outline:
- testingTerminaleUscita
  - Launcher
    - main(String[]): void

Console:
Launcher (1) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Struttura tabella di backup:
barcode anno mese giorno ora minuti
Stato del backup....

EchoServer D'USCITA: started
Server Socket D'USCITA: ServerSocket[addr=0.0.0.0/0.0.0.0, port=1050]
(SERVER_USCITA) Connection accepted: Socket[addr=0.0.0.0/0.0.0.0, port=63261]

Problems: /javadoc Declaration Console
TerminaleUsciente [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (23 gen 2022, 13:23:30)
EchoClient: started
Client Socket: Socket[addr=localhost/127.0.0.1,port=1050,localport=63261]
codice biglietto:
```

# CAPITOLO 10: Software Testing

Nell'immagine precedente è stato testato il prototipo `v0.4.0_testingTerminaleUscita_v2.0_reteScarica`.

In questo caso di testing abbiamo verificato il corretto funzionamento dello scambio di messaggi tra server e terminale d'uscita, cortocircuitando il terminale d'entrata.

Lo scopo del testing era di verificare il riconoscimento dei ticket e la corretta cancellazione di quest'ultimi dal database in caso di utente che esce dal parcheggio.



# CAPITOLO 10: Software Testing

The screenshot displays the Eclipse IDE interface for a project named 'campoProva'. The Package Explorer on the left shows the project structure, including a 'src' directory with a 'progettoParcheggio' package. The main editor shows the 'Launcher.java' file, which contains the following code:

```
9 Parcheggio server;  
10 BackupDB db;  
11 try {  
12     db = new BackupDB("jdbc:mysql://localhost:3300","root","admin");  
13     db.setupDB();  
14     db.describeTableDB();  
15  
16     server = new GestoreParcheggio(db);  
17  
18     Thread entranteThread = new Thread(new AutoEntrante(server));  
19     Thread uscenteThread = new Thread(new AutoUscente(server));  
20  
21  
22     uscenteThread.start();  
23     entranteThread.start();  
24  
25  
26     db.closeDB();  
27 }  
28 catch (IOException e) {  
29     // TODO Auto-generated catch block  
30     e.printStackTrace();  
31 }  
32 catch (SQLException e1) {  
33     // TODO Auto-generated catch block  
34     e1.printStackTrace();  
35 }
```

The Console window on the right shows the output of the application, including the following messages:

```
TerminaleEntrante (1) [Java Application] C:\Program Files\Java\jdk-17\bin\java.exe  
Ticket stampato  
attendo ticket...  
ticket disponibile  
Ticket stampato  
attendo ticket...  
ticket disponibile  
Ticket stampato  
attendo ticket...  
ticket disponibile  
  
TerminaleUscente (1) [Java Application] C:\Program Files\Java\jdk-17\bin\java.exe  
pagamento concluso, buona giornata  
  
esito: 0  
pagamento concluso, buona giornata  
  
esito: 0  
pagamento concluso, buona giornata
```

The Problems window at the bottom shows the following messages:

```
Launcher (2) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (23 gen 2022, 13:27:06)  
Ticket consegnato: Ticket[/5722008180435480236/2022-01-23T13:29/]  
sono arrivato qui  
Il conto da pagare è: 0.0 0  
Innalzamento sbarra in corso..  
ticket rimosso da parcheggio virtuale. size : 16  
Una macchina è uscita alle: 2022-01-23T13:29:22.574858200  
Posti occupati nel parcheggio = 16  
Ticket eliminato: 5722008180435480236 --- entrato alle 2022-01-23T13:29  
Abbassamento sbarra in corso..
```



# CAPITOLO 10: Software Testing

Nell'immagine precedente è stato testato il prototipo v0.3.1\_prototipoDB\_v2.0.

In questo caso di testing abbiamo verificato il corretto funzionamento del database in un contesto concorrenziale e inter-procedurale.

La concorrenzialità è risultata centrale in quanto si voleva valutare eventuali starvation e deadlock all'interno dell'esecuzione del software.



# CAPITOLO 11: Software Maintenance

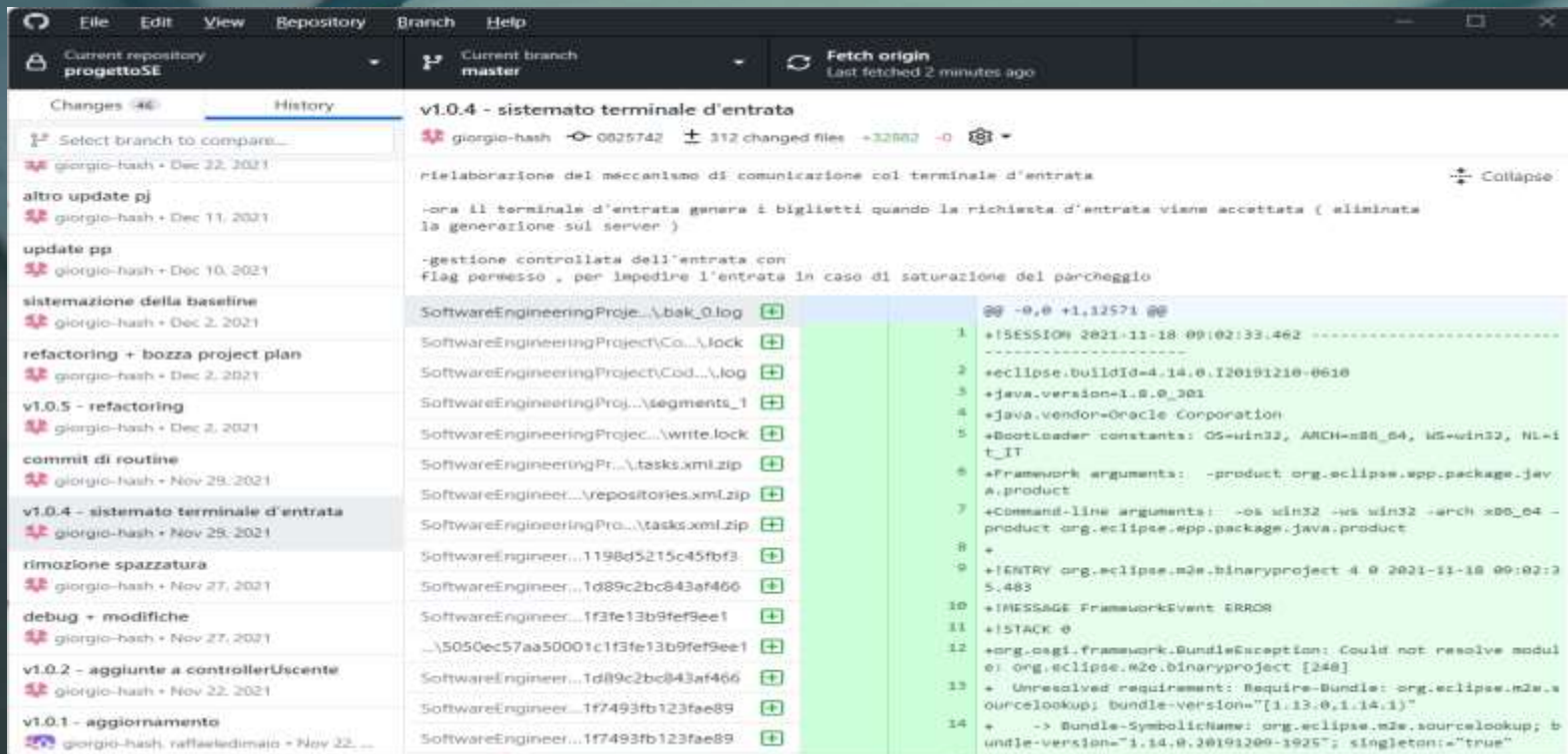
Nel corso dello sviluppo del software sono state effettuate una serie di attività di refactoring.

In particolare i due metodi utilizzati sono stati:

- Manutenzione correttiva;
- Manutenzione preventiva.

# CAPITOLO 11: Software Maintenance

Manutenzione correttiva:



The screenshot displays the Eclipse IDE interface with a dark theme. The top menu bar includes File, Edit, View, Repository, Branch, and Help. Below the menu, the 'Current repository' is set to 'progettoSE' and the 'Current branch' is 'master'. The 'Fetch origin' button indicates the last fetch was 2 minutes ago.

The left sidebar shows the 'Changes' view with a list of commits. The commit 'v1.0.4 - sistemato terminale d'entrata' by 'giorgio-hash' on Nov 28, 2021, is selected. The main area shows the diff for this commit, highlighting changes in the 'SoftwareEngineeringProject\...\bak\_0.log' file. The diff shows a series of log entries, including session information, build details, and a stack trace indicating a 'BundleException: Could not resolve module org.eclipse.m2e.binaryproject [248]'.

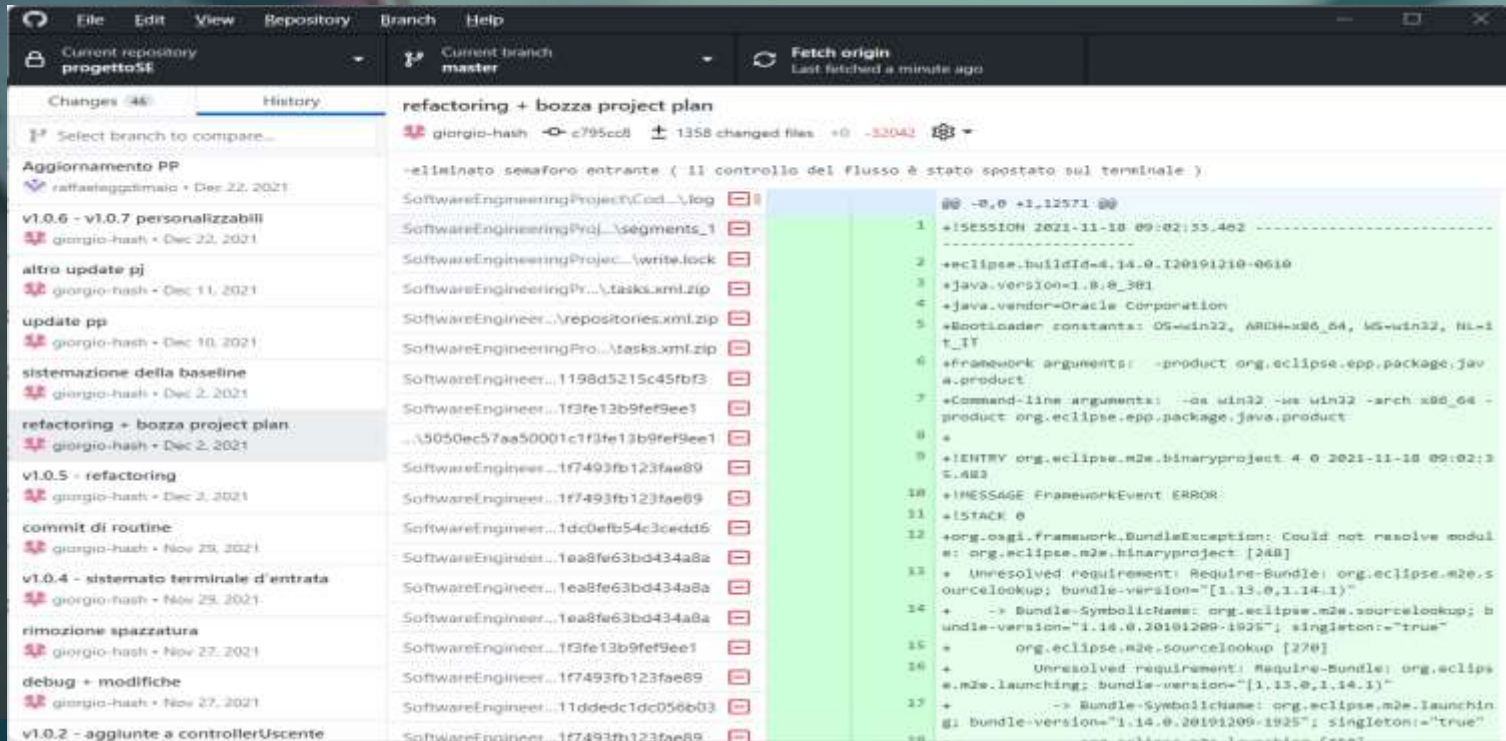
The commit history on the left includes the following entries:

- Select branch to compare...
- giorgio-hash • Dec 22, 2021
- altro update pj
- giorgio-hash • Dec 11, 2021
- update pp
- giorgio-hash • Dec 10, 2021
- sistemazione della baseline
- giorgio-hash • Dec 2, 2021
- refactoring + bozza project plan
- giorgio-hash • Dec 2, 2021
- v1.0.5 - refactoring
- giorgio-hash • Dec 2, 2021
- commit di routine
- giorgio-hash • Nov 29, 2021
- v1.0.4 - sistemato terminale d'entrata
- giorgio-hash • Nov 28, 2021
- rimozione spazzatura
- giorgio-hash • Nov 27, 2021
- debug + modifiche
- giorgio-hash • Nov 27, 2021
- v1.0.2 - aggiunte a controllerUscente
- giorgio-hash • Nov 22, 2021
- v1.0.1 - aggiornamento
- giorgio-hash, raffaeledimato • Nov 22, 2021



# CAPITOLO 11: Software Maintenance

Manutenzione preventiva:



The screenshot displays the Eclipse IDE interface. The top menu bar includes File, Edit, View, Repository, Branch, and Help. Below the menu, the 'Current repository' is set to 'progetto54', and the 'Current branch' is 'master'. The 'Fetch origin' button indicates the last fetch was a minute ago. The main workspace is divided into three panes. The left pane shows the 'Changes' view with a list of commits, including 'refactoring + bozza project plan' by 'giorgio-hash' on Dec 2, 2021. The middle pane shows the 'History' view for the selected commit, listing files such as 'SoftwareEngineeringProjectCod...log' and 'SoftwareEngineeringProj...segments\_1'. The right pane shows the diff view for the selected commit, displaying changes to the 'SoftwareEngineeringProjectCod...log' file. The diff shows a series of changes, including the removal of a semaphore and the addition of a new project plan.

```
refactoring + bozza project plan
giorgio-hash c795cc8 1358 changed files +0 -32042

-eliminato semaforo entrante ( il controllo del flusso è stato spostato sul terminale )
SoftwareEngineeringProjectCod...log
SoftwareEngineeringProj...segments_1
SoftwareEngineeringProjec...write.lock
SoftwareEngineeringPr...tasks.xml.zip
SoftwareEngineer...repositories.xml.zip
SoftwareEngineeringPro...tasks.xml.zip
SoftwareEngineer...1198d5215c45fbf3
SoftwareEngineer...1f3fe13b9fef9ee1
...5050ec57aa50001c1f3fe13b9fef9ee1
SoftwareEngineer...1f7493fb123fae89
SoftwareEngineer...1f7493fb123fae89
SoftwareEngineer...1dc0efb54c3cedd5
SoftwareEngineer...1ea8fe63bd434a8a
SoftwareEngineer...1ea8fe63bd434a8a
SoftwareEngineer...1ea8fe63bd434a8a
SoftwareEngineer...1f3fe13b9fef9ee1
SoftwareEngineer...1f7493fb123fae89
SoftwareEngineer...11ddcd1dc056b03
SoftwareEngineer...1f7493fb123fae89
```