

PROGETTO EVENT

Gestione di una foresteria

GAETANO COMANDATORE

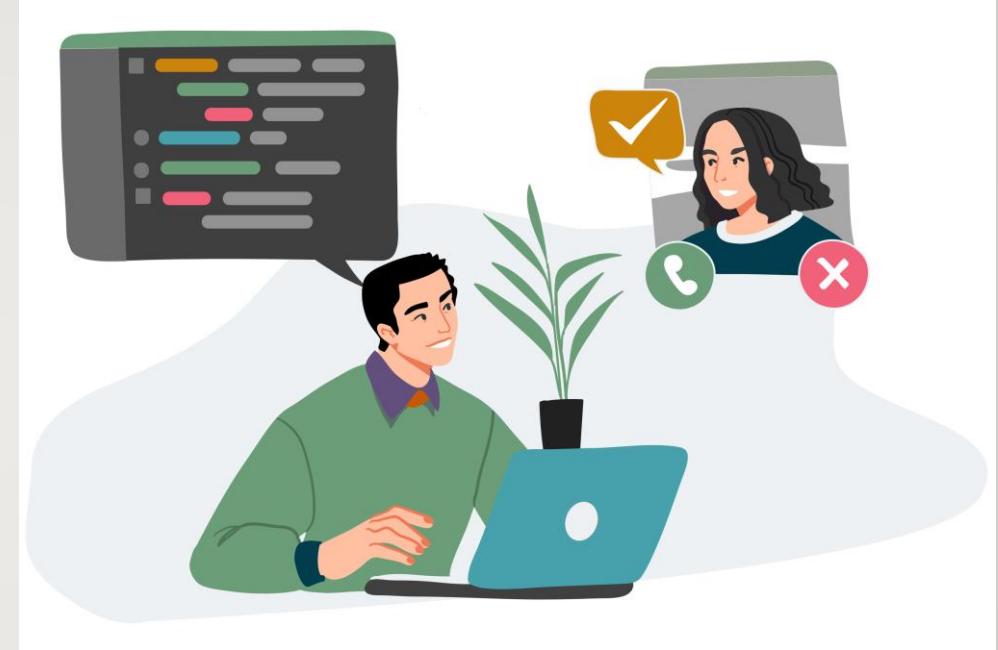
1090952

NICHOLAS IOTTI

1058728

MODELLO ORGANIZZATIVO

- PLANNING GAME
- PAIR PROGRAMMING
- SYSTEM METAPHORE
- SCRUM
- CONTINUOS INTEGRATION
- TRELLO: <https://trello.com/b/CcWDGTUq/progetto-event>

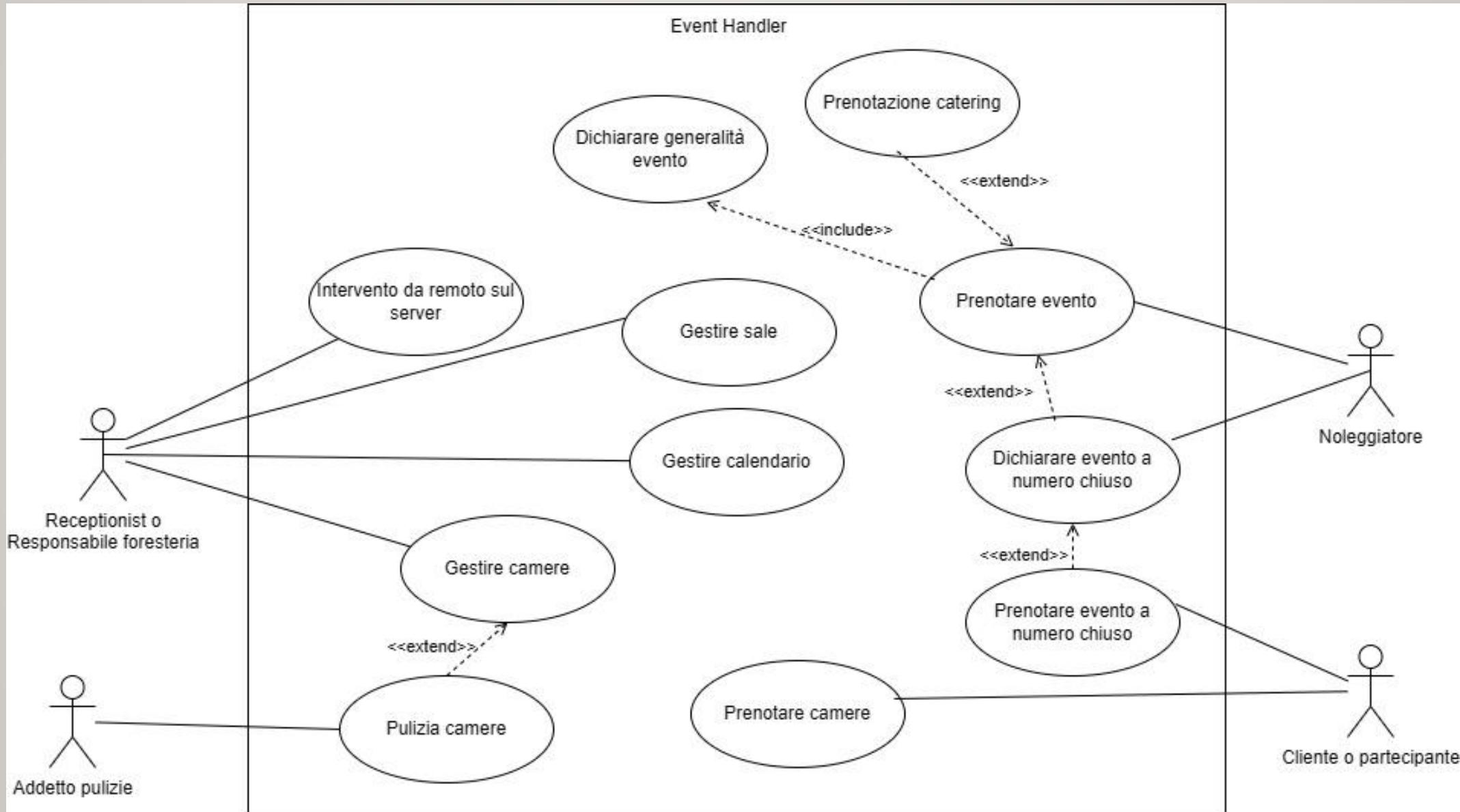


DEFINIZIONE REQUISITI: NON FUNZIONALI

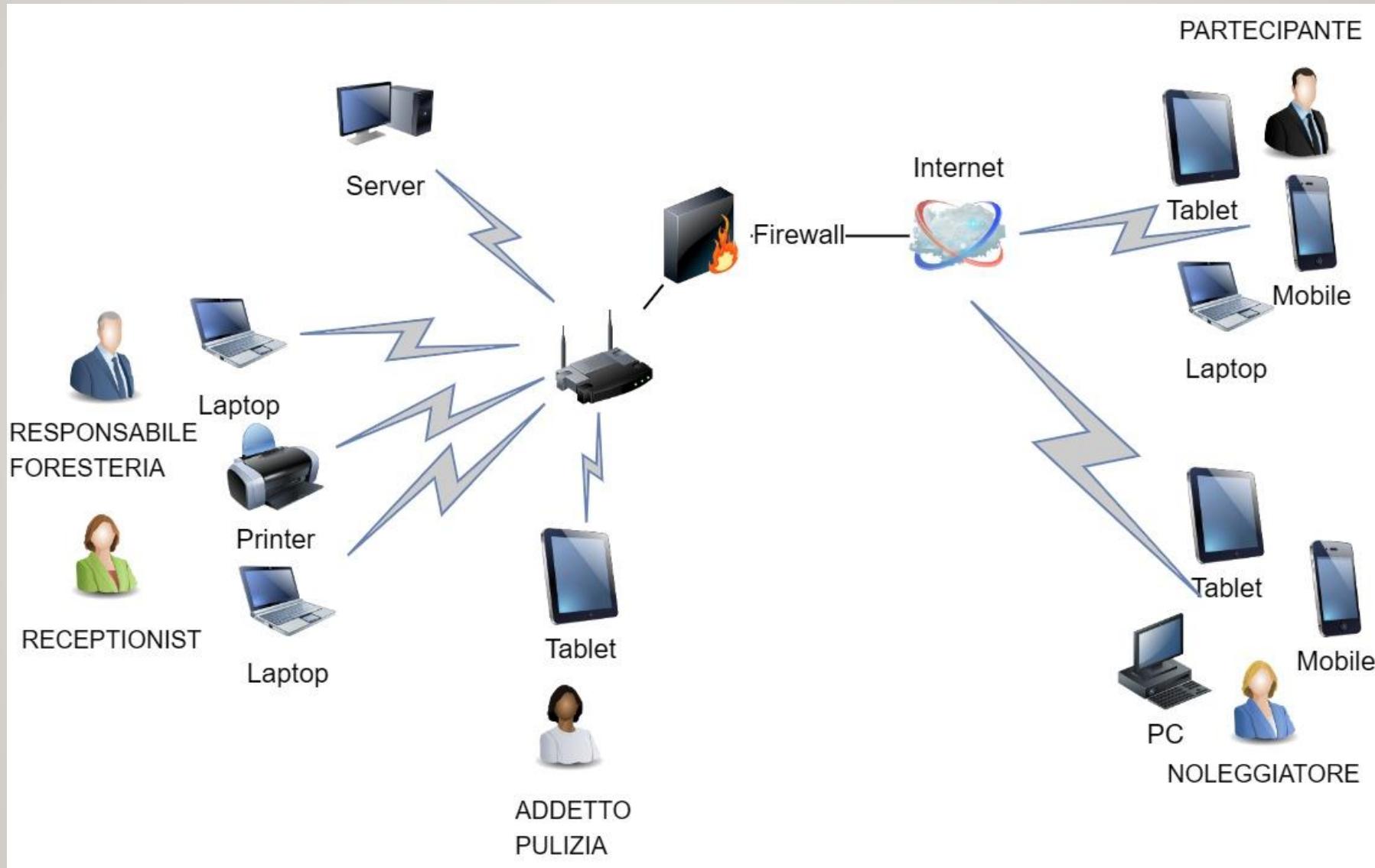
- MVC
- TIERS: SpringBoot
- JavaFX
- API-led
- Design patterns (visitor, singleton, publisher-subscriber)
- Versione 1: No DBMS: file JSON
- *Versione 2: DBMS: MongoDB (sia in locale che su cloud)*



DEFINIZIONE REQUISITI FUNZIONALI:

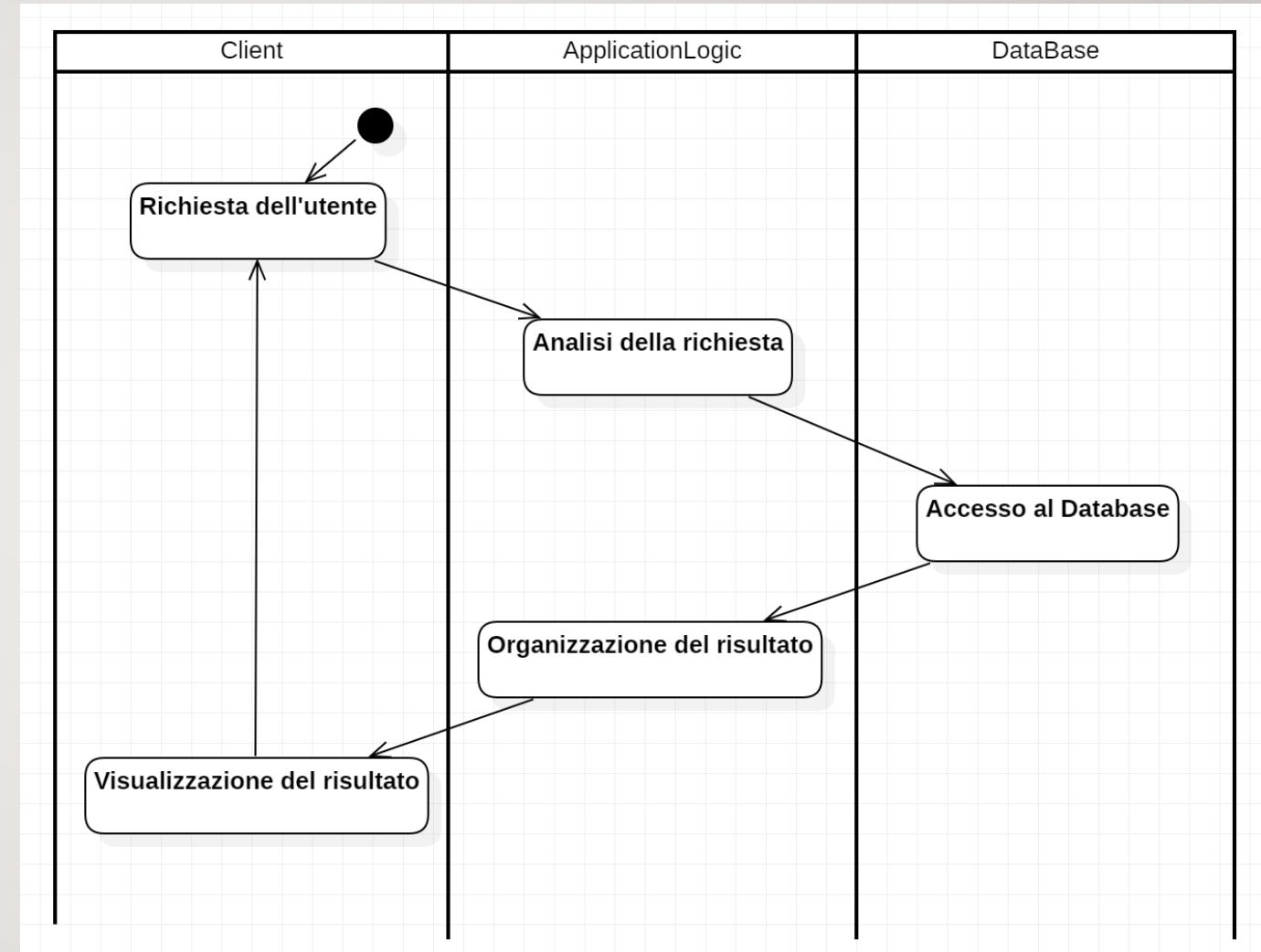


INFORMAL DEPLOYMENT DIAGRAM

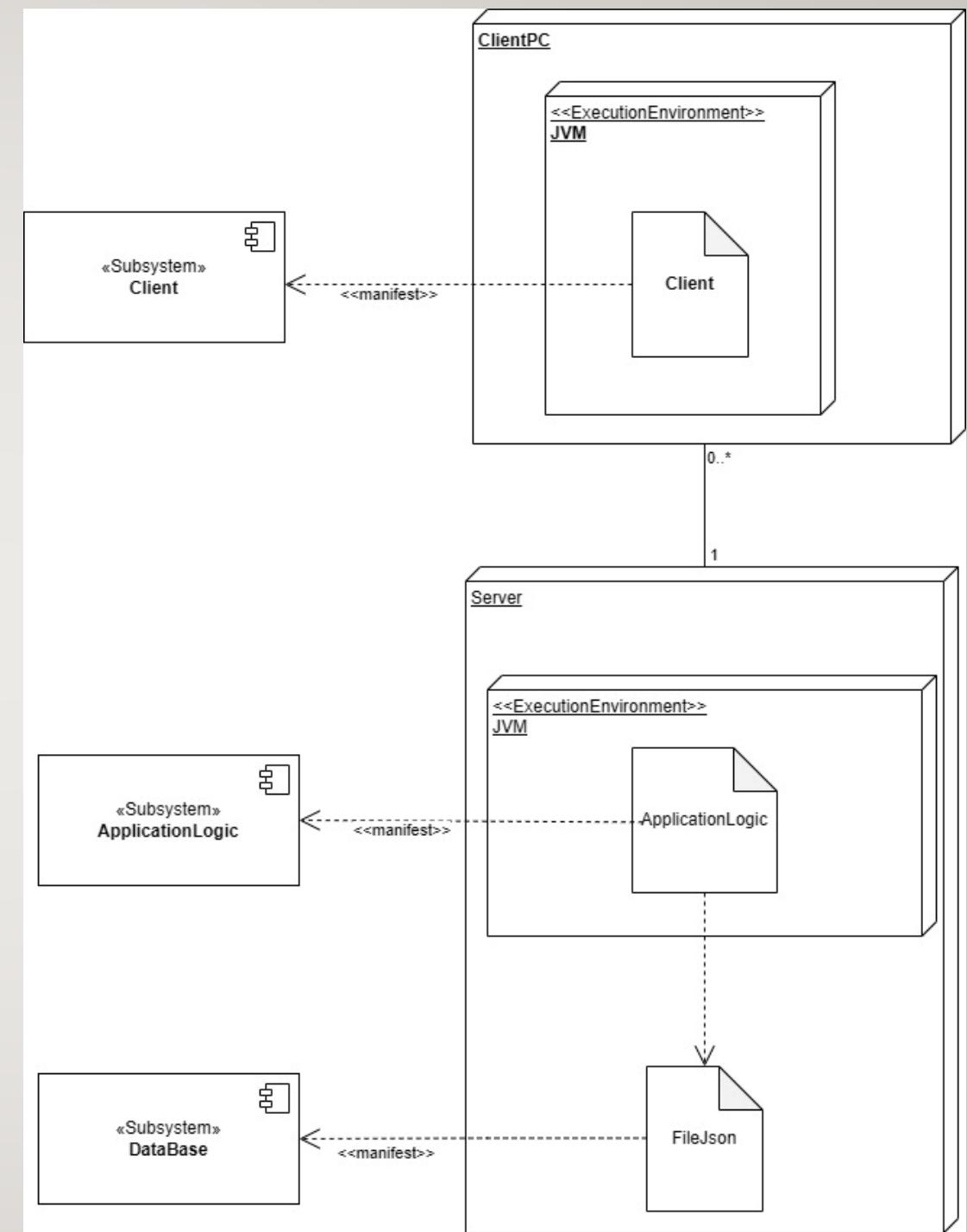


EARLY ARCHITECTURE DESIGN

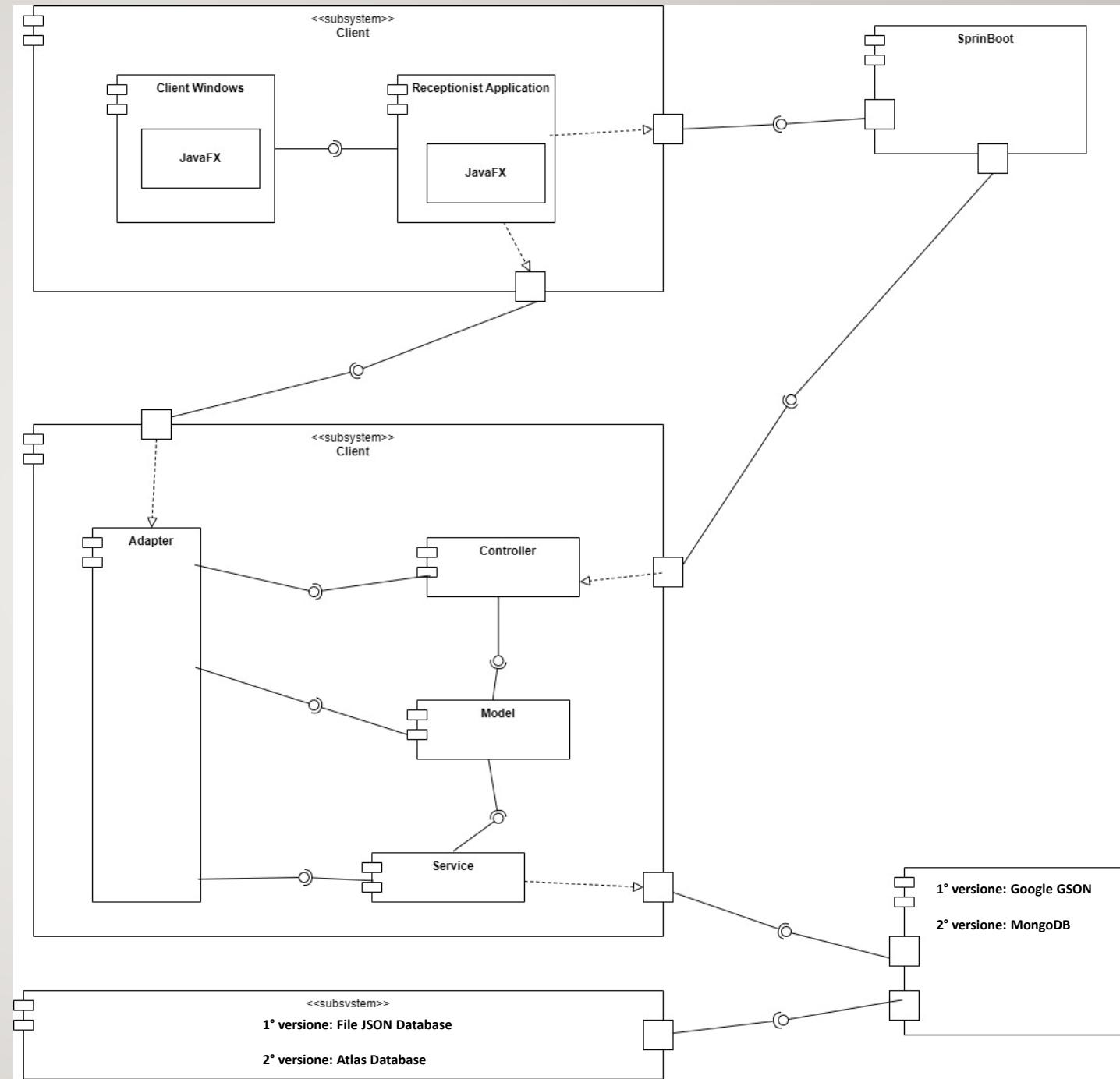
- 2 layers: Client- Server
- Comunicazione con SpringBoot
- RESTful:
 - Richieste gestite tramite HTTP
 - Comunicazioni stateless
 - Trasmissioni risorse complete
 - Informazioni in forma standard



DEPLOYMENT VIEW



FUNCTIONALITY VIEW



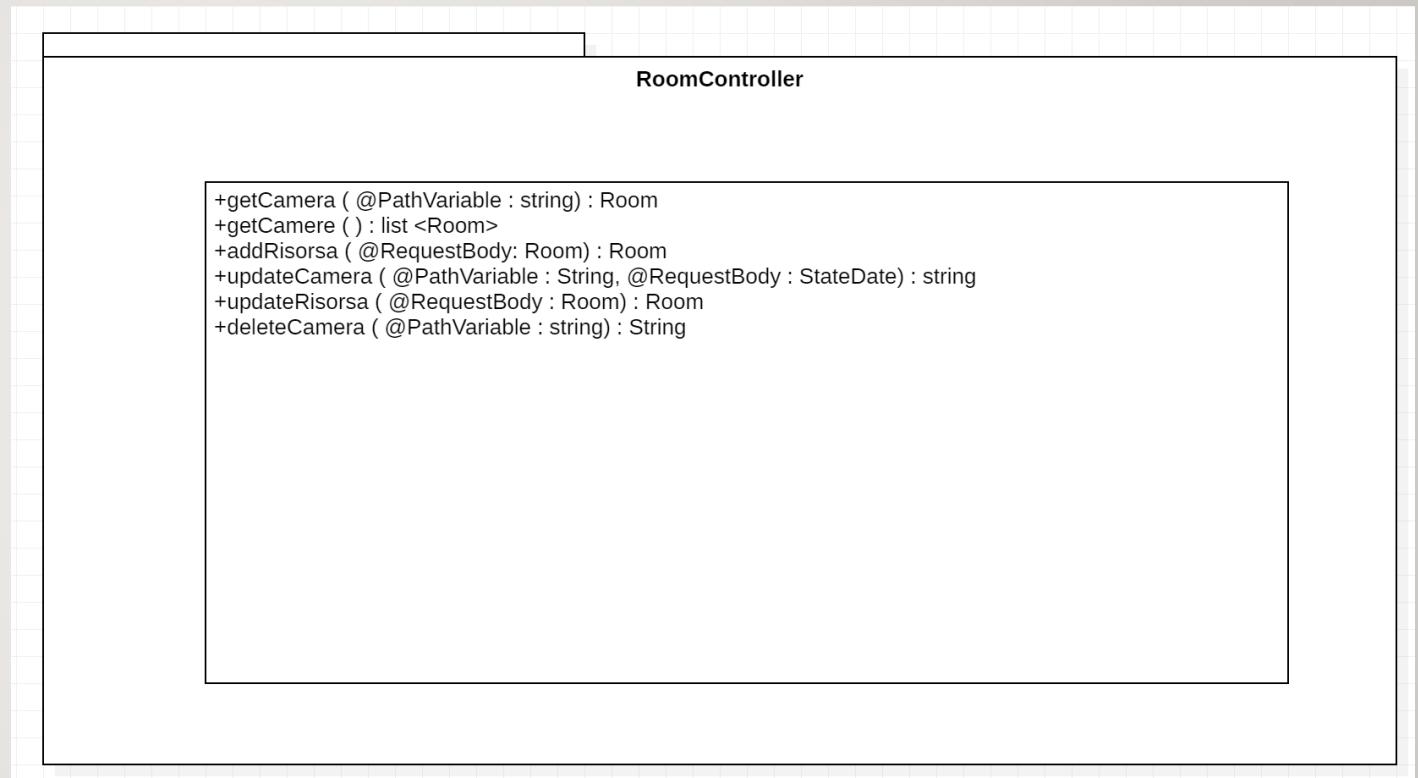
TOOL CHAIN

Nome	Tipo	Funzione	Nome	Tipo	Funzione
Windows	Sistema Operativo	SO	JSON	Formato file	Data Base
Java 17	Linguaggio di programmazione	Language	Gson	Libreria	SerDE formato json
JDK 20	Development Kit	Ambiente Sviluppo	Git Hub	SAAS	WEB Repository
Eclipse 4.29.0	IDE	Ambiente Sviluppo	MongoDB	Software	DBMS NoSQL in locale
Maven 4.0.0	Build Tool	Gest. dipendenze	AtlasDB	SAAS	DBMS NoSQL in cloud
sonarLint 10.5	Plugin Eclipse	Quality + Security	Trello	SAAS	Gestione progetti
codeMR 2020..4.1	Plugin Eclipse	Analisi Statica	Teams	SAAS	Call + chat
JUnit5 5.10.1	Libreria di test	Analisi Dinamica	WhatsApp	SAAS	Call + chat
JavaDoc	Plugin Eclipse	Strumento di Documentazione	Draw.io	SAAS	Grafici e diagrammi UML
SpringBoot 3.2.0	Framework	Framework per Applicazioni WEB	MS Office	Software (Word e PowerPoint)	Documentazione
JavaFX	Libreria	Componenti UI			

IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

- Caso d'uso *Gestire camera*
- Creazione di un listener
RoomController con i metodi in figura
- File JSON Camere.json
- SpringBoot per comunicazione
UI-Controller



IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

ANALISI COMPLESSITÀ

Metodo *WRoom.mettiDati*

2 cicli annidati, ma numero delle camere fisse

$O(kn) \rightarrow O(n)$

```
private void mettiDati() {
    // Invia una richiesta GET al server per ottenere i dati di tutte le camere
    ResourceRoom[] rooms = restTemplate.getForObject("http://localhost:8080/api/room", ResourceRoom[].class);

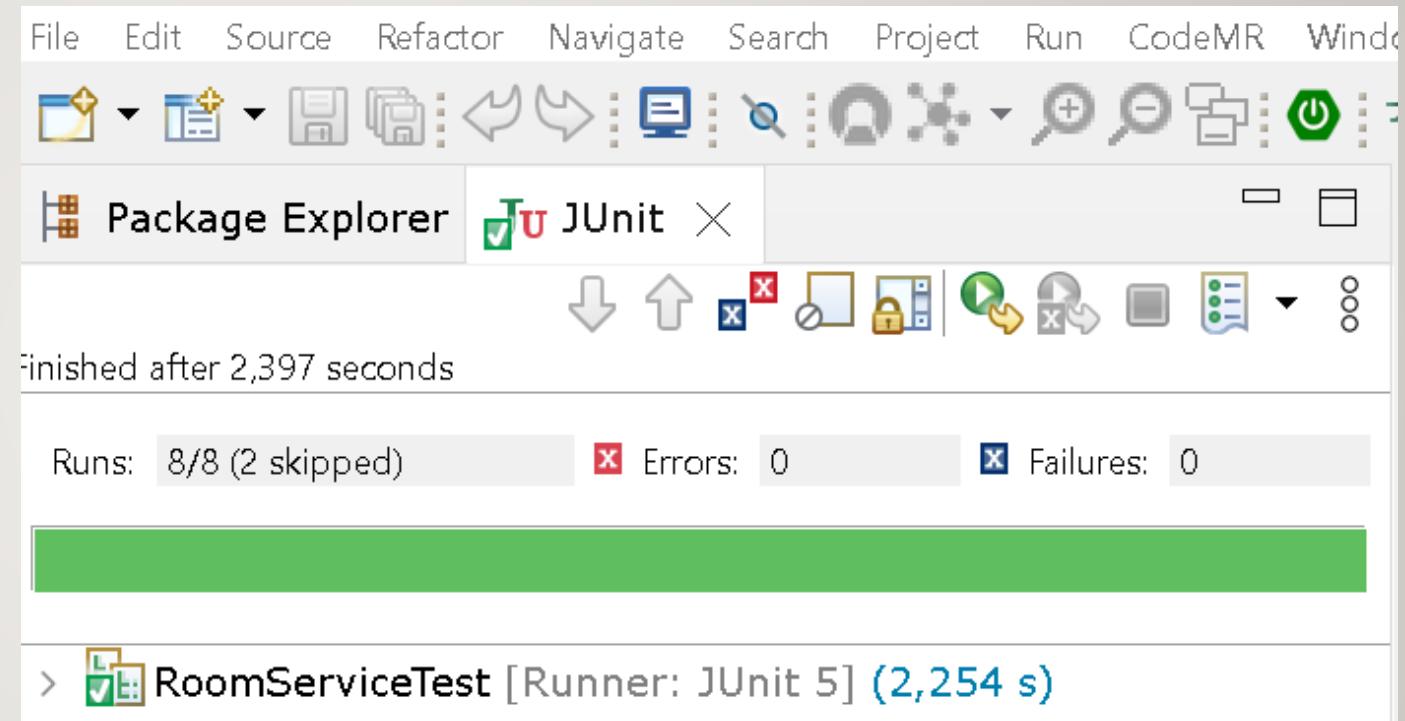
    // Crea un insieme di date uniche
    Set<String> uniqueDates = new HashSet<>();
    for (ResourceRoom room : rooms) {
        for (StateDate stateDate : room.getDisponibilita()) {
            uniqueDates.add(stateDate.getData().toString());
        }
    }
}
```

IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

ANALISI DINAMICA

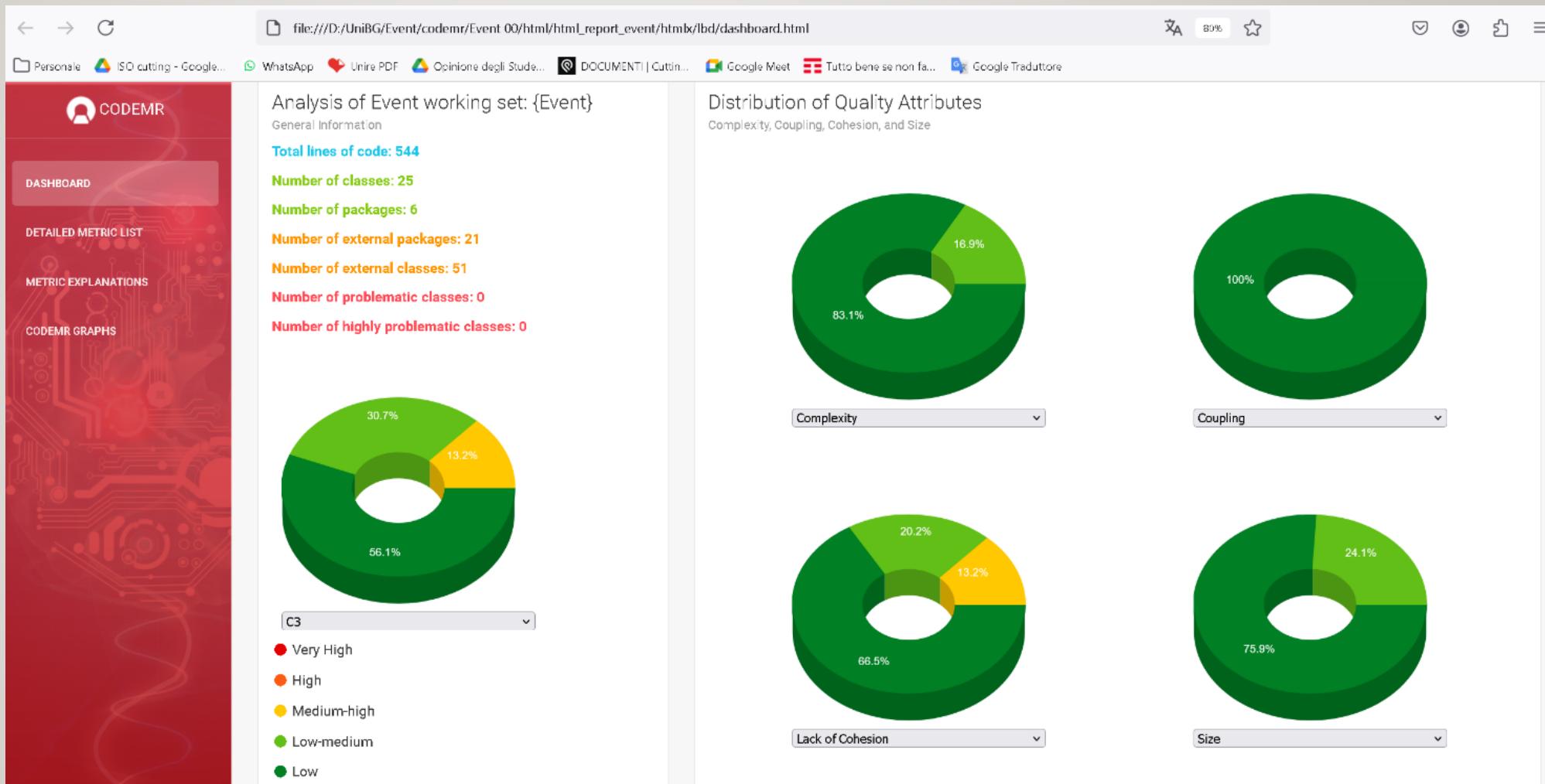
Esito positive di tutti gli 8
test previsti dal codice



IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

ANALISI STATICA



IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

MASCHERA DELLE CAMERE

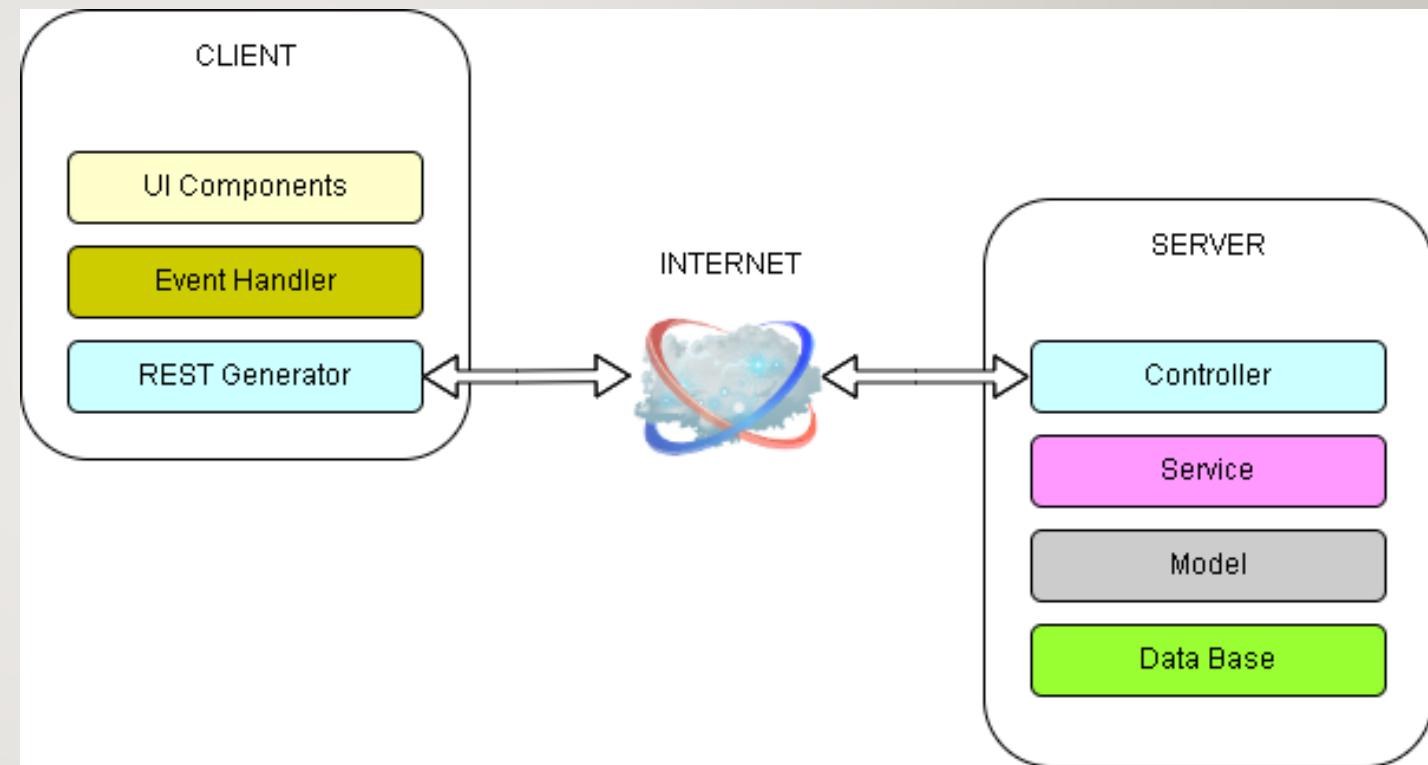
Visualizzazione Camere									
Camera	Tipo	Costo	N. Letti	2024-01-01	2024-01-02	2024-01-03	2024-01-04	2024-01-05	
Room001	SUITE	120.0	2	INUSO	DISPONIBILE	PULIZIA			
Room002	MINISUITE	120.0	2	DISPONIBILE	PRENOTATA	PRENOTATA			
Room003	SUITE	100.0	2	PRENOTATA	DISPONIBILE	INUSO			
Room004	ROYALE	120.0	3	DISPONIBILE	CHIUSO	PULIZIA			
Room005	ROYALE	120.0	3			PRENOTATA	DISPONIBILE	PRENOTATA	

IMPLEMENTAZIONE CODICE:

PRIMA ITERAZIONE

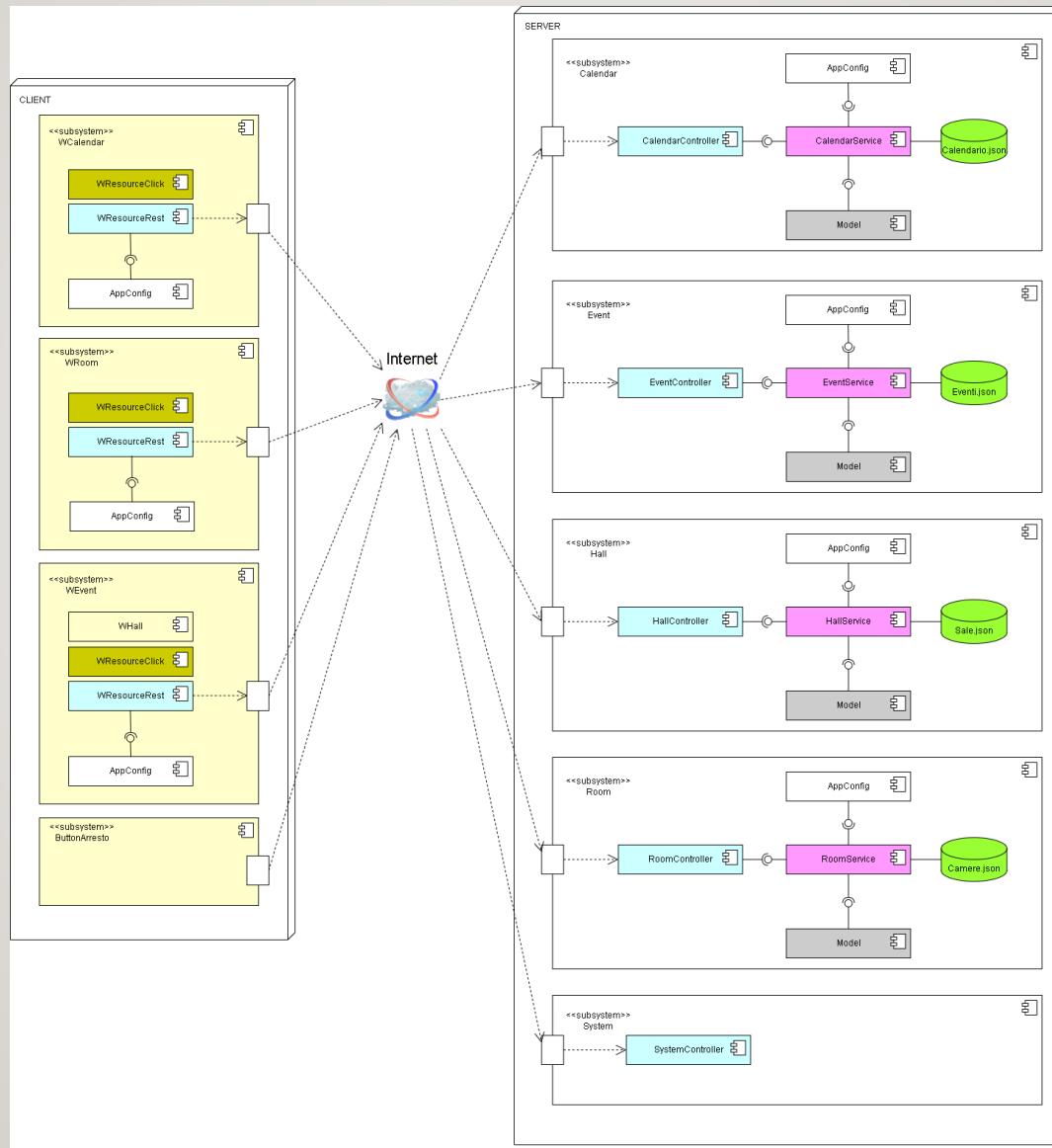
ARCHITECTURAL PATTERN

- Two tiers
- La comunicazione via internet ad opera di un REST Generator e di un Controller
- L'interfaccia utente (JavaFX) separata dal codice
- Il server è suddiviso in controller (listener) ed il service
- Tutte le comunicazioni passano attraverso il controller: MVC passivo (Pull)



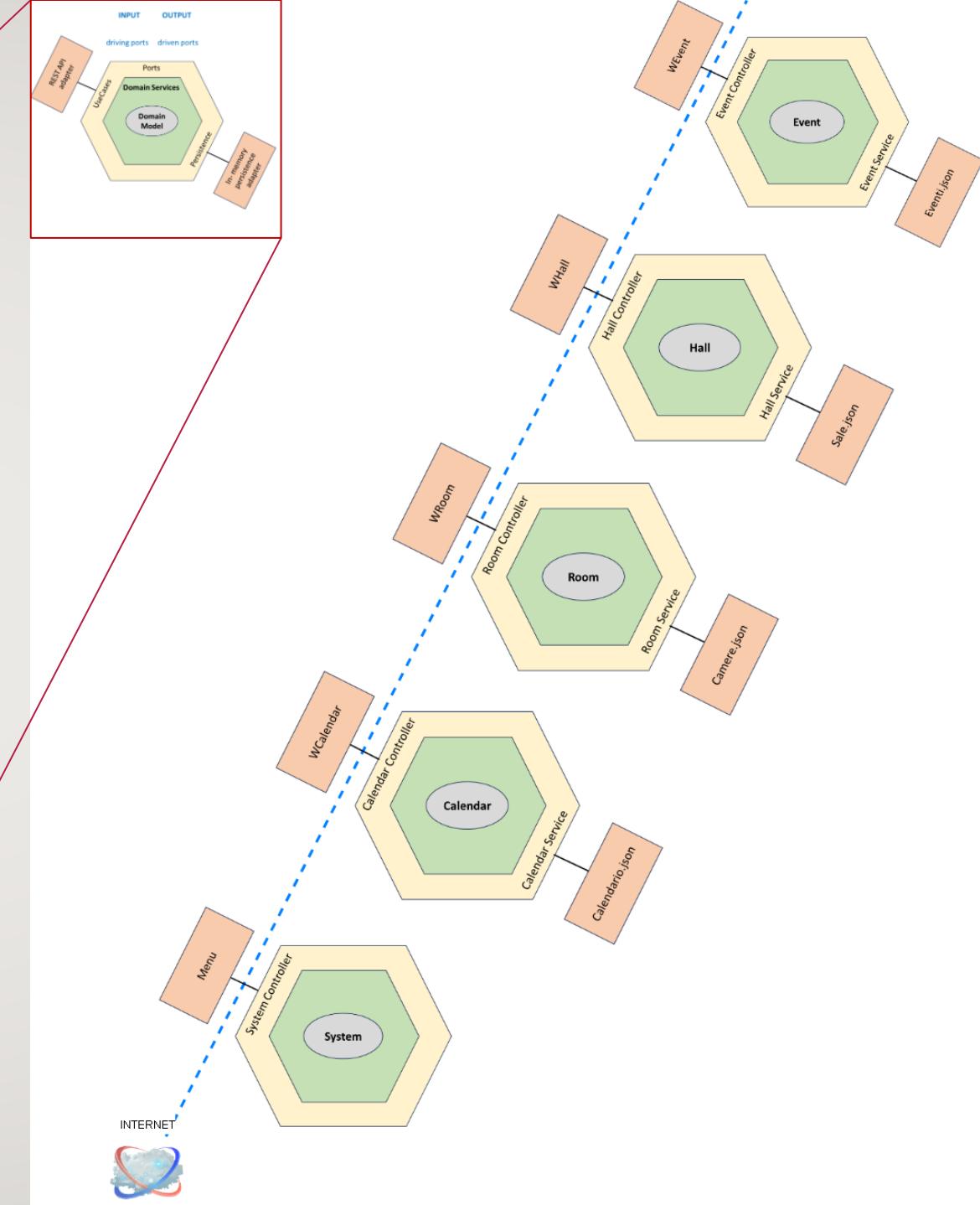
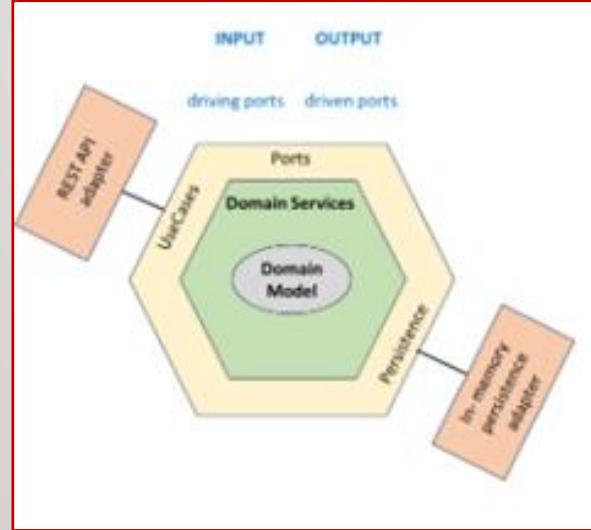
ARCHITECTURAL PATTERN

API LED – VERSIONE I



IMPLEMENTAZIONE MICROSERVIZI

VERSIONE I

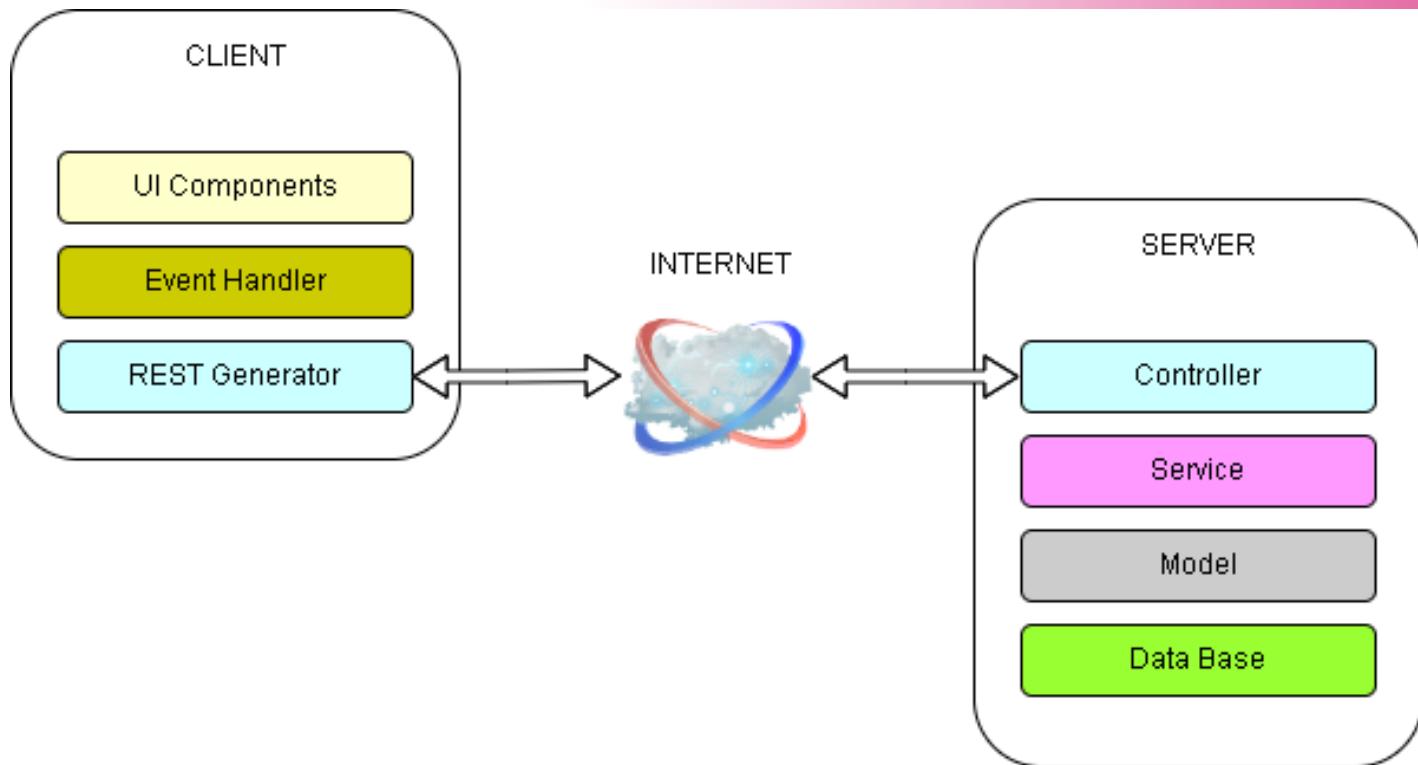


IMPLEMENTAZIONE CODICE:

VERSIONE 2

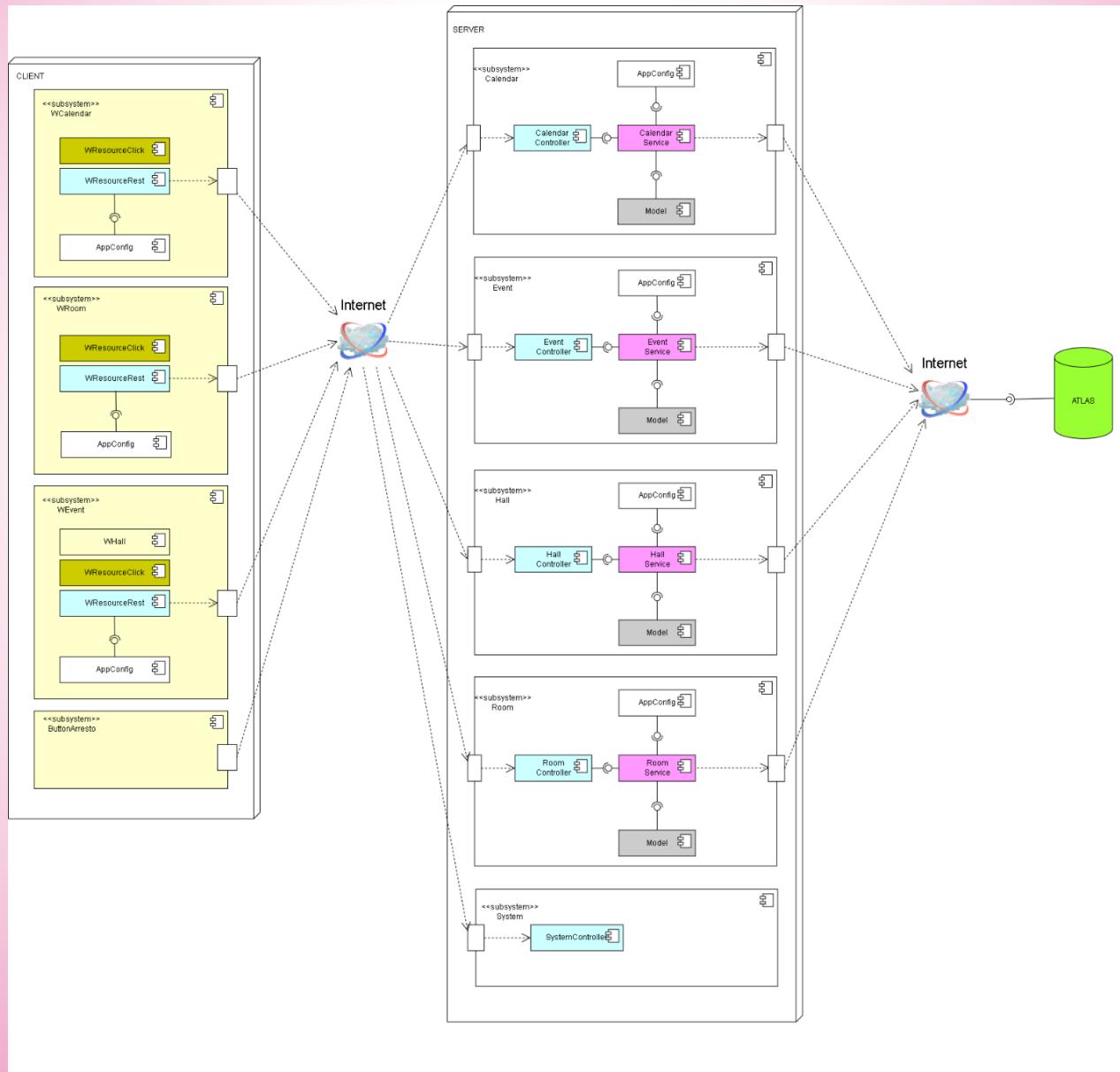
ARCHITECTURAL PATTERN

- Three tiers
- La comunicazione via internet ad opera di un REST Generator e di un Controller
- L'interfaccia utente (JavaFX) separata dal codice
- Il server è suddiviso in controller (listener) ed il service
- Tutte le comunicazioni passano attraverso il controller: MVC passivo (Pull)



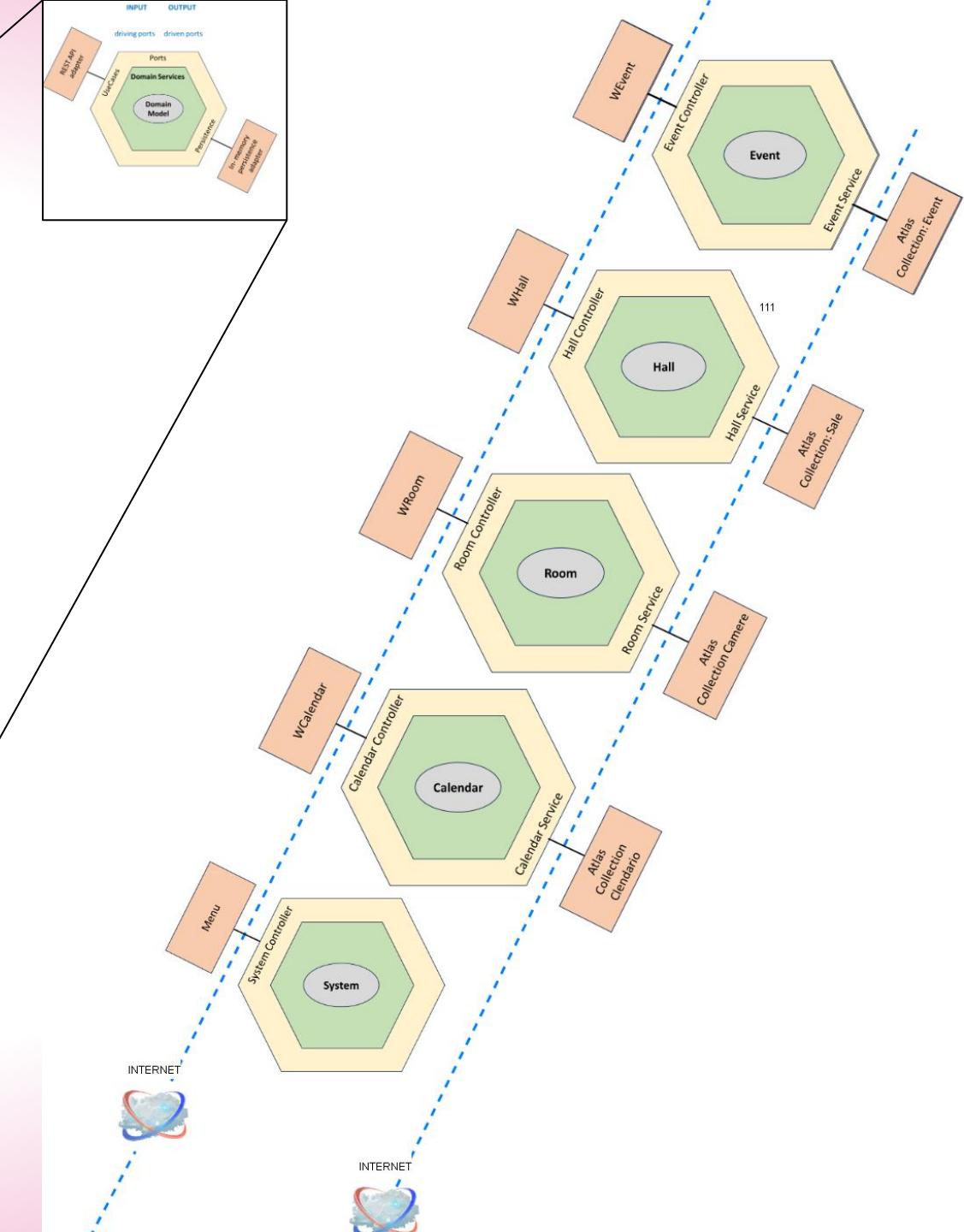
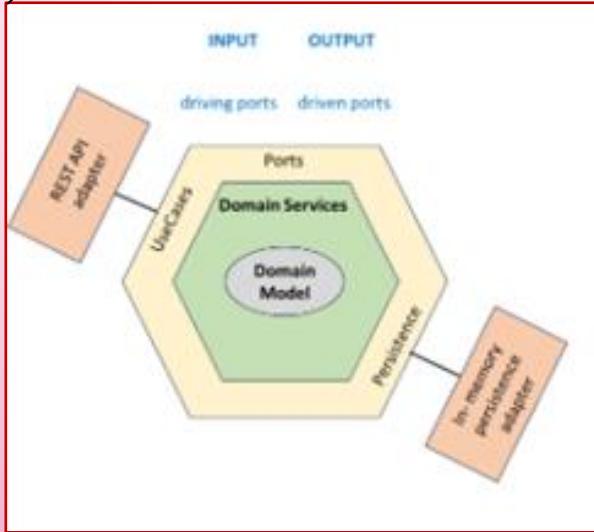
ARCHITECTURAL PATTERN

API LED – VERSIONE 2



IMPLEMENTAZIONE MICROSERVIZI

VERSIONE 2

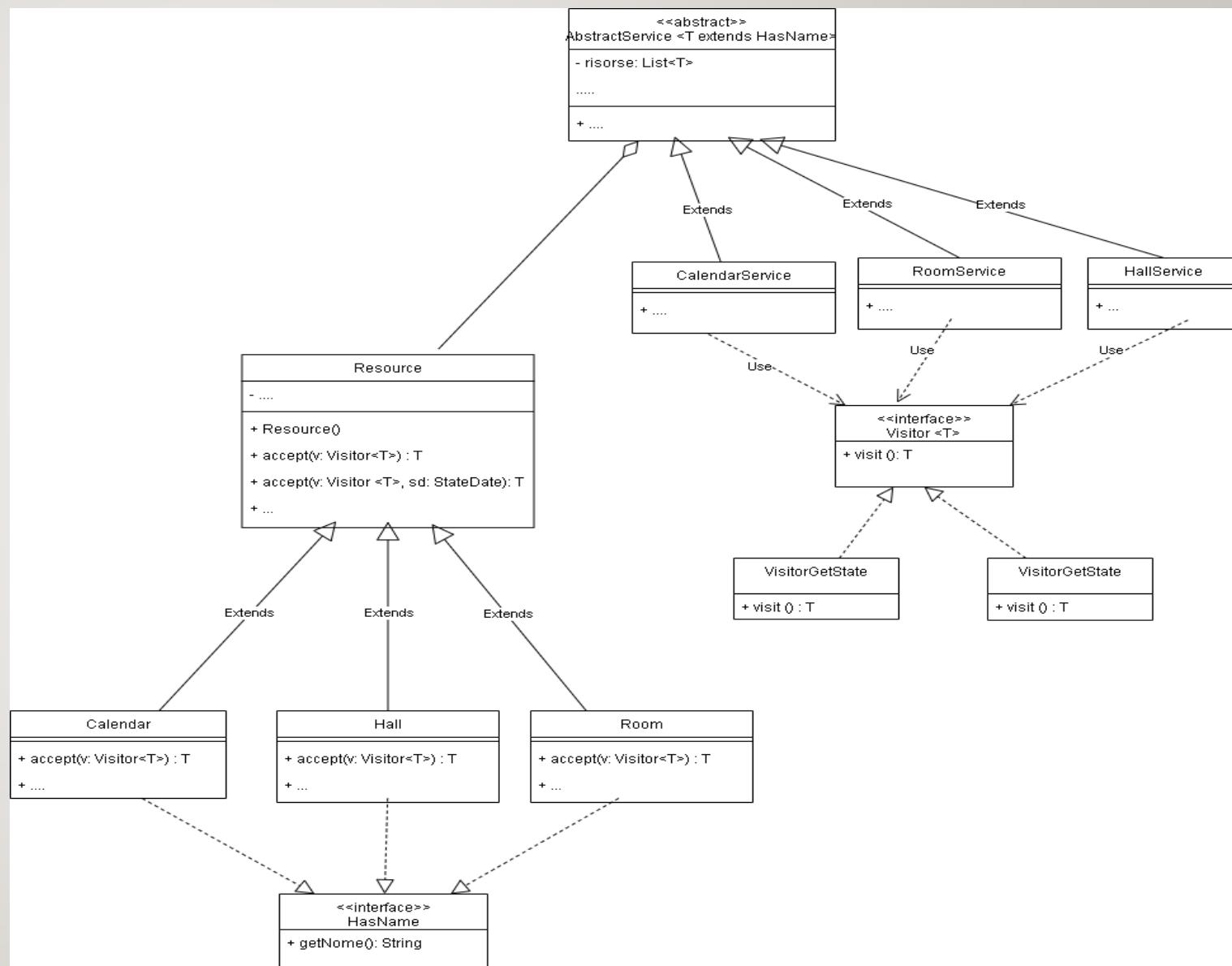


CLASS DIAGRAM: DESIGN PATTERNS

VISITOR

- Procede al cambio degli stati delle risorse
- Potrebbe fornire il prezzo delle camere per l'emissione delle conferme delle prenotazioni o delle fatture

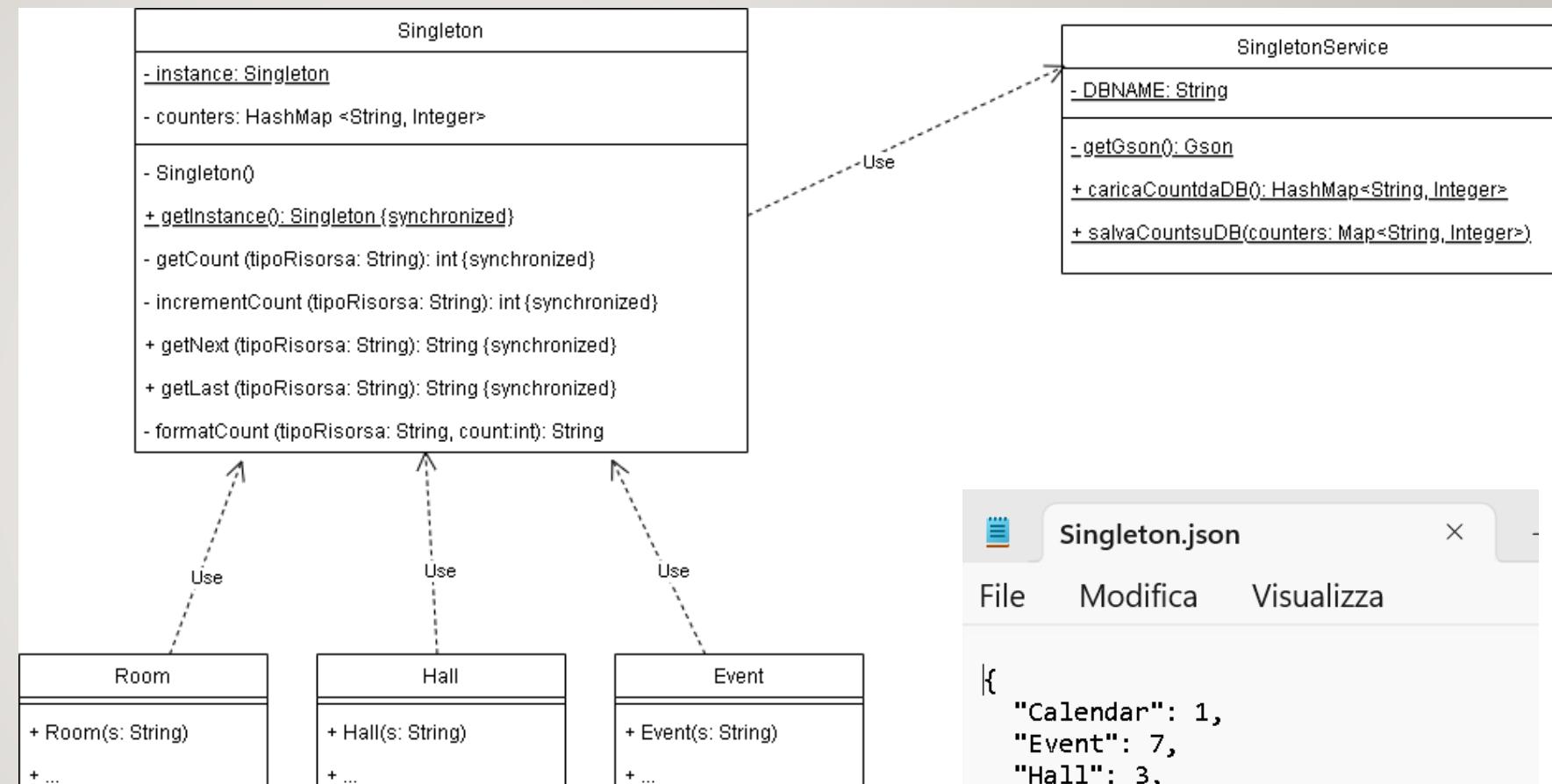
```
1 package server.model;
2
3
4 /**
5  * Interfaccia utilizzata per garantire che le classi che la implementano
6  * abbiano un metodo getName(). Questo metodo è utilizzato nella classe
7  * AbstractService per identificare univocamente ciascuna risorsa.
8 */
9
10 public interface HasName {
11
12     public String getName();
13
14 }
```



CLASS DIAGRAM: DESIGN PATTERNS

SINGLETON

- Gestisce la numerazione delle risorse
- Potrebbe gestire la numerazione delle prenotazioni e delle fatture



Singleton.json

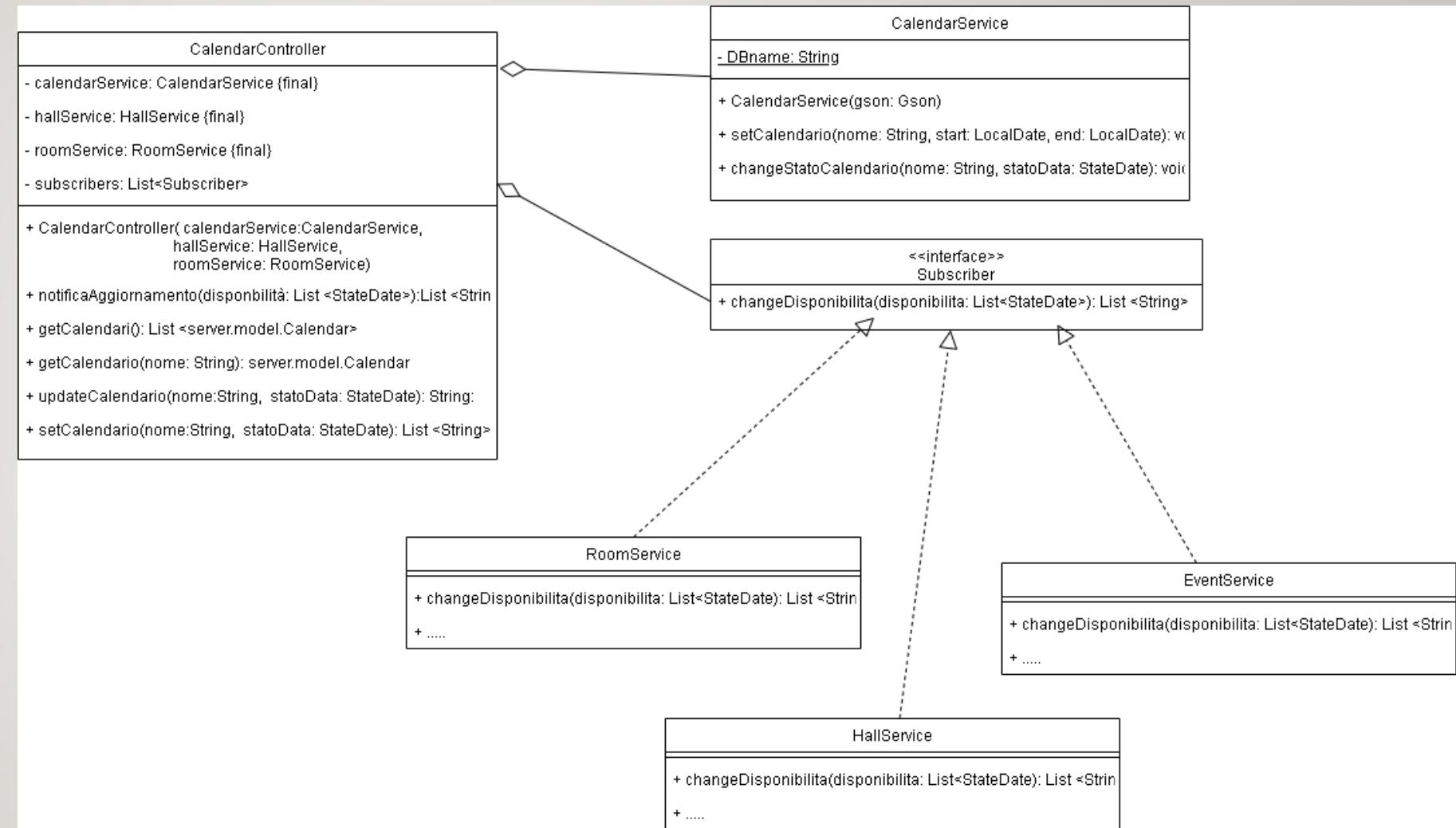
File Modifica Visualizza

```
{  
    "Calendar": 1,  
    "Event": 7,  
    "Hall": 3,  
    "Room": 5  
}
```

CLASS DIAGRAM: DESIGN PATTERNS

OBSERVER

- Notifica a tutti i service che è stato modificato il calendario
- Procede al cambio degli stati delle risorse
- Procede e alla generazione dei report utili per comunicare agli attori desiderati

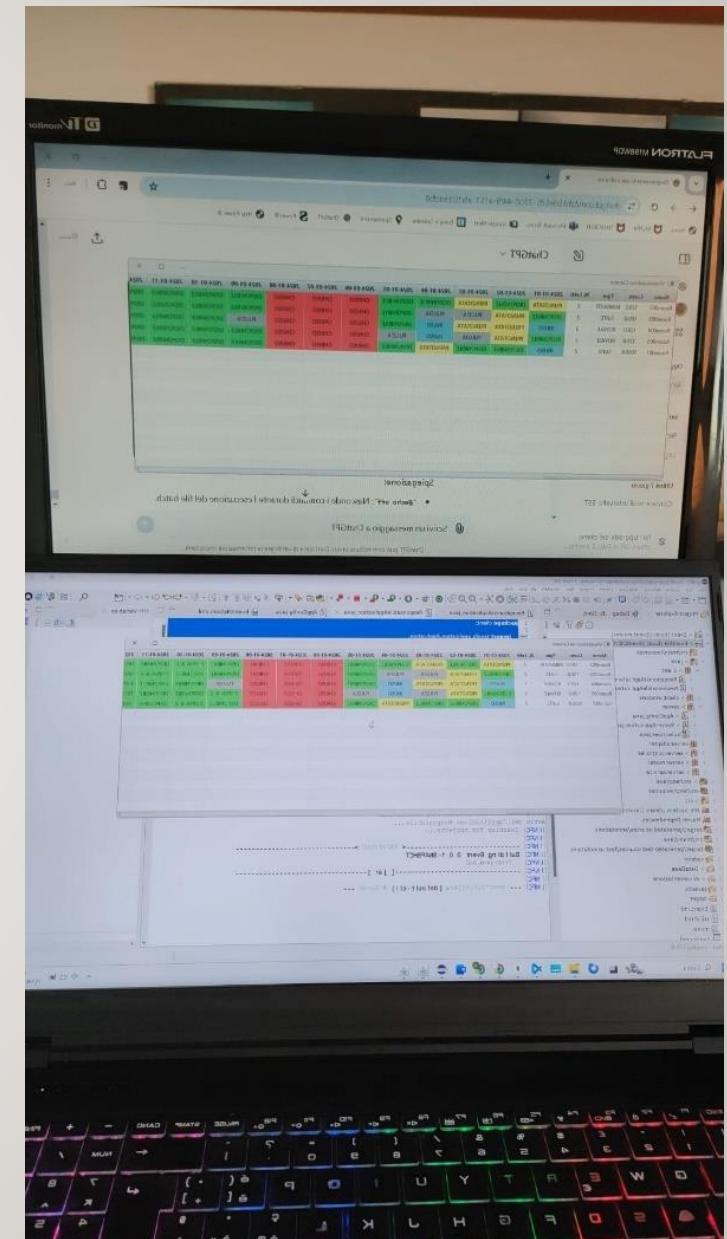


TEST CON DUE CLIENT

Il test con due client ha evidenziato la necessità di un'ulteriore serie di observer: uno per ogni risorsa/API (Calendar, Event, Hall e Room)

- 1. L'inizializzazione di un client dovrebbe utilizzare l'iniezione fornita da Spring per ottenere le istanze dei controller e chiedere di essere inserito nella lista degli Observer.
- 2. I vari controller aggiornano le rispettive liste con i riferimenti alle varie WResource relative
- 3. Ogni qualvolta che un client chiede la modifica di una risorsa il service che la esegue provvede anche alla sua notifica a tutti gli interessati.

Tale modifica è stata solo analizzata ma non realizzata.



INTERFACCE GRAFICHE

Menu

- Gestione Calendario
- Gestione Camere
- Gestione Eventi
- Arresto del sistema

WCalendar

Visualizzazione Calendario

Data inizio: 2024-01-01 Data fine: 2024-02-27 Estendi il Calendario

2024-01-01	2024-01-02	2024-01-03	2024-01-04	2024-01-05	2024-01-06	2024-01-07	2024-01-08	2024-01-09	2024-01-10	2024-01-11
DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE	CHIUSO	CHIUSO	CHIUSO	DISPONIBILE	DISPONIBILE	DISPONIBILE
< >										

click dx

INTERFACCE GRAFICHE

WRoom

INTERFACCE GRAFICHE

WEvent

Sistema - Windows - Help

Gestione Eventi

ID da cercare:

Messaggio:

ID evento: Event001

Nome Organizzatore: Pippo

Costo Partecipazione: 100.0

Partecipanti Previsti: 100

Catering: COFFEE_BREAK

Data: 01/01/2024

Sala: Hall002

Ora Inizio: 09:00

Ora Fine: 19:00

Elenco Interventi:

Nome	Costo	N. Posti	2024-01-01	2024-01-02	2024-01-03	2024-01-04	2024-01-05
Hall001	1200.0	30	PRENOTATA	DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE
Hall002	1500.0	50	PRENOTATA	DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE
Hall003	2200.0	150	DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE	DISPONIBILE

La sala è prenotata per l'evento:

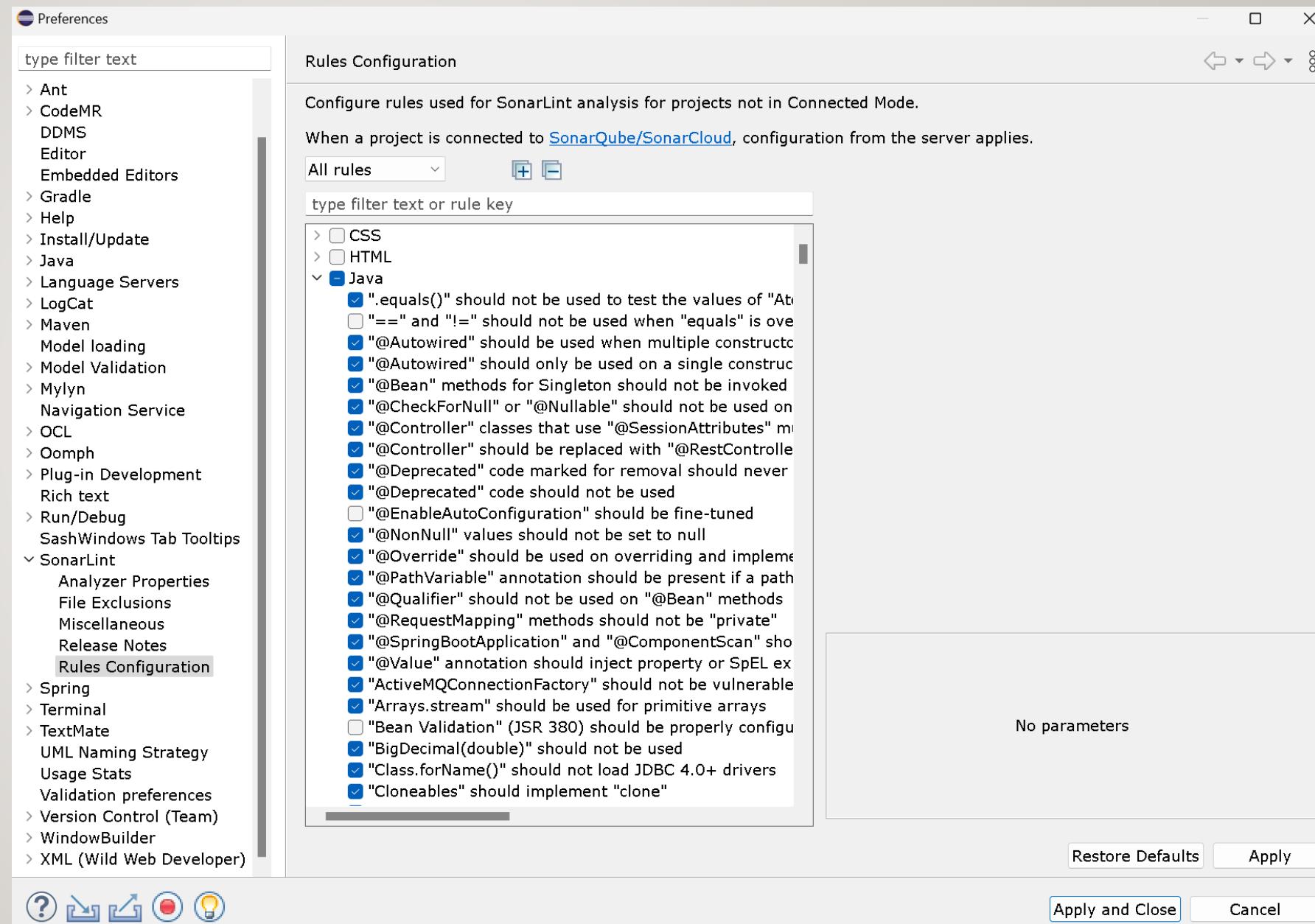
i Evento {
nome: Event002,
data: 2024-01-01,
ora inizio: 09:00,
ora fine: 17:00,
organizzato da: PAC,
partecipanti previsti: 25,
catering: COLAZIONE,
sala: Hall001,
costo partecipazione: 120.0,
elenco interventi: [
Intervento 1,
Relatore 1,
Descrizione dell'intervento 1111,
Intervento 2,
Relatore 2,
Descrizione dell'intervento 2,
qwqwqw,
Relatoreeeee,
bla bla bla,
pippo,
pippoooooooooooo,
pppppppppppppppppppppppp]}
}

Primo Evento Evento Precedente Prossimo Evento Ultimo Evento Crea Nuovo Evento Salva / Modifica Eve...

InterventiLista
(TableView)

ANALISI STATICÀ: SONAR LINT

- In locale: senza SonarQube o SonarCloud
- Disabilitando solo pochissime regole (Java)
- Usando il meno possibile il comando //NOSONAR

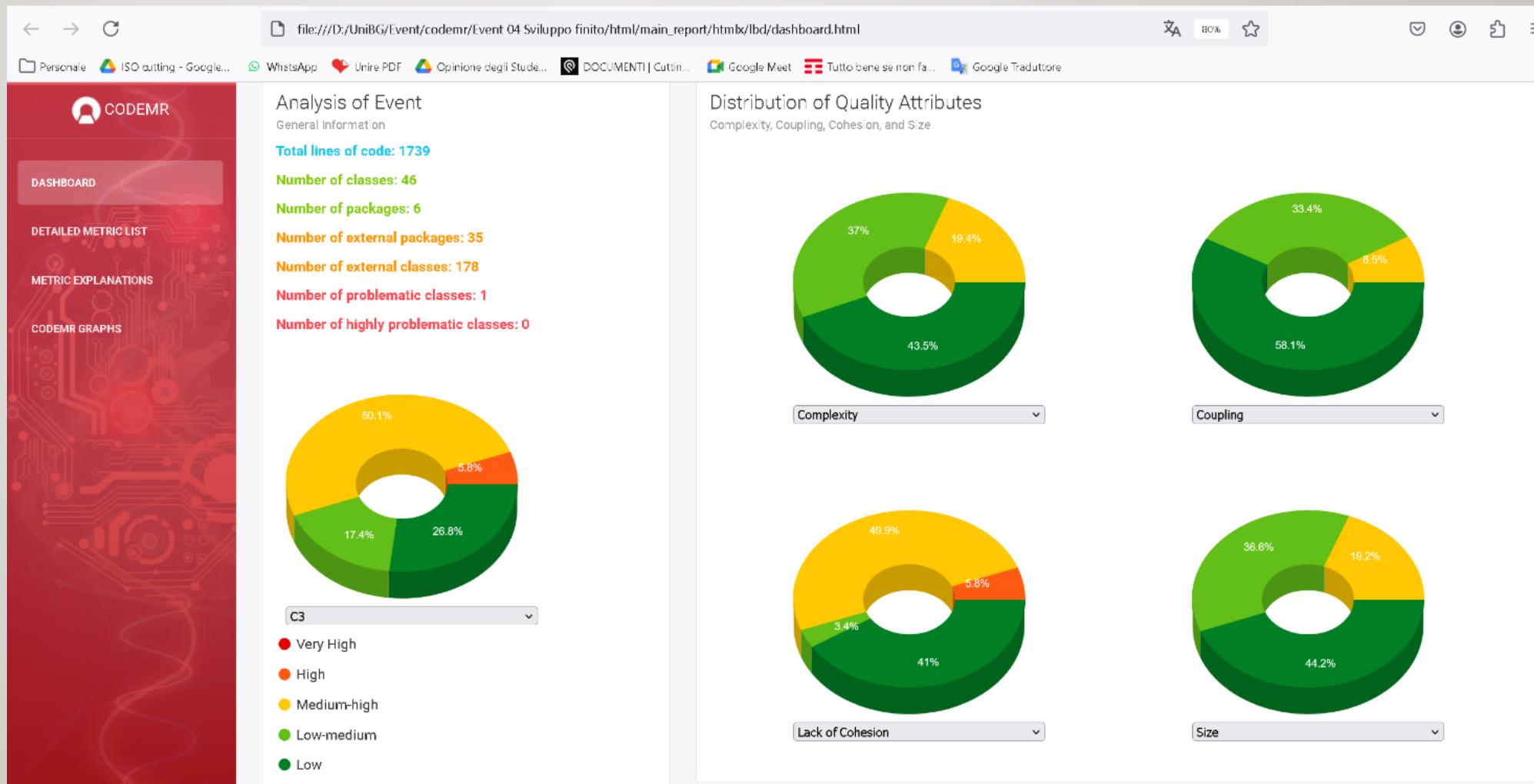


ANALISI STATICÀ: CODEMR

ITERAZIONI INTERMEDI

ITERAZIONE 4

- completato lo sviluppo degli user case
- l'analisi mostra problemi di coesione di tipo "grave"



ANALISI STATICÀ: CODEMR

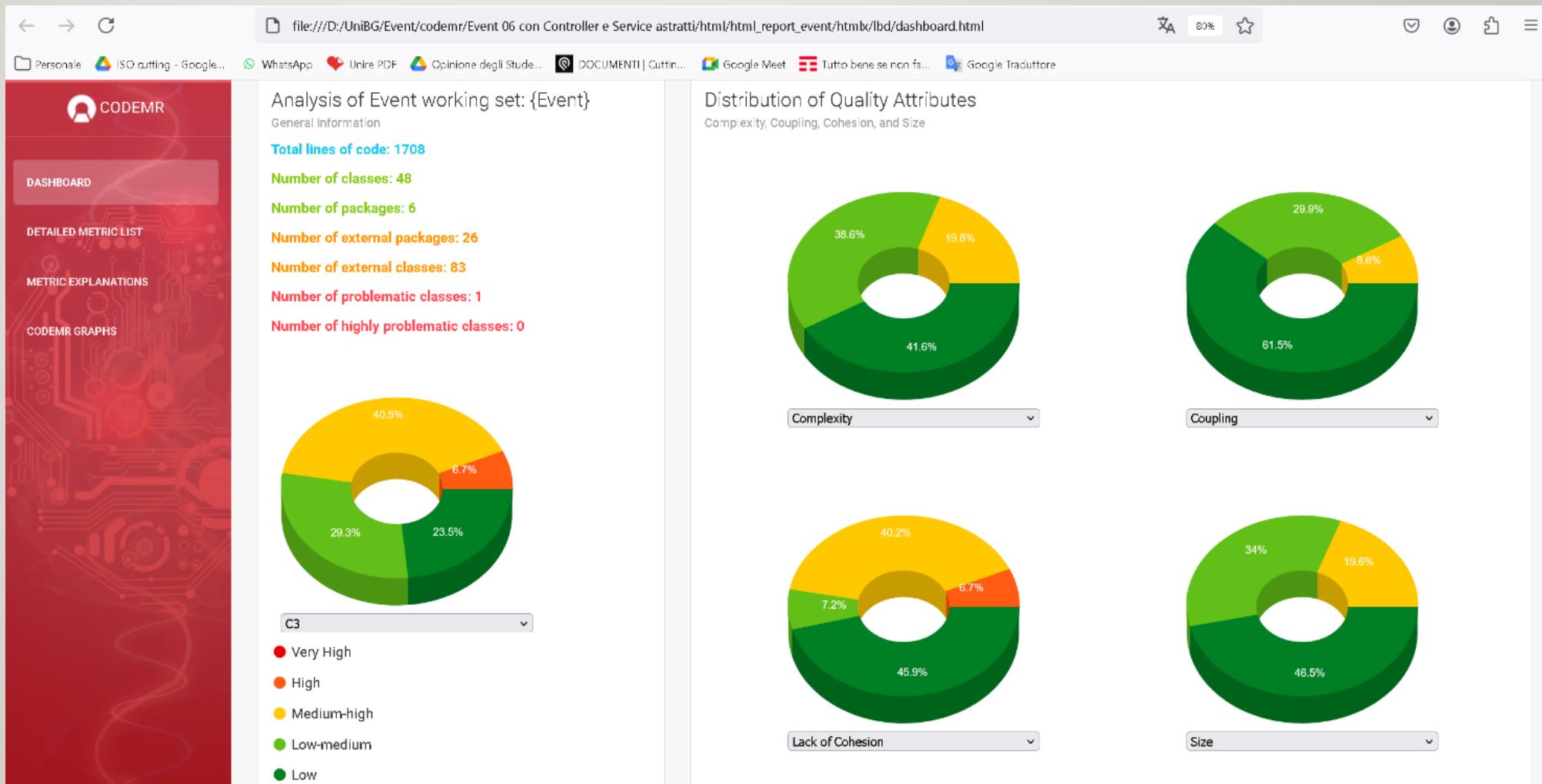
ITERAZIONI INTERMEDI

REFACTORING

- revisione classi del server
- Introduzione classe astratta
- Introduzione observer

RISULTATI

- numero delle classi esterne dimezzato
- coesione non migliorata



ANALISI STATICÀ: CODEMR

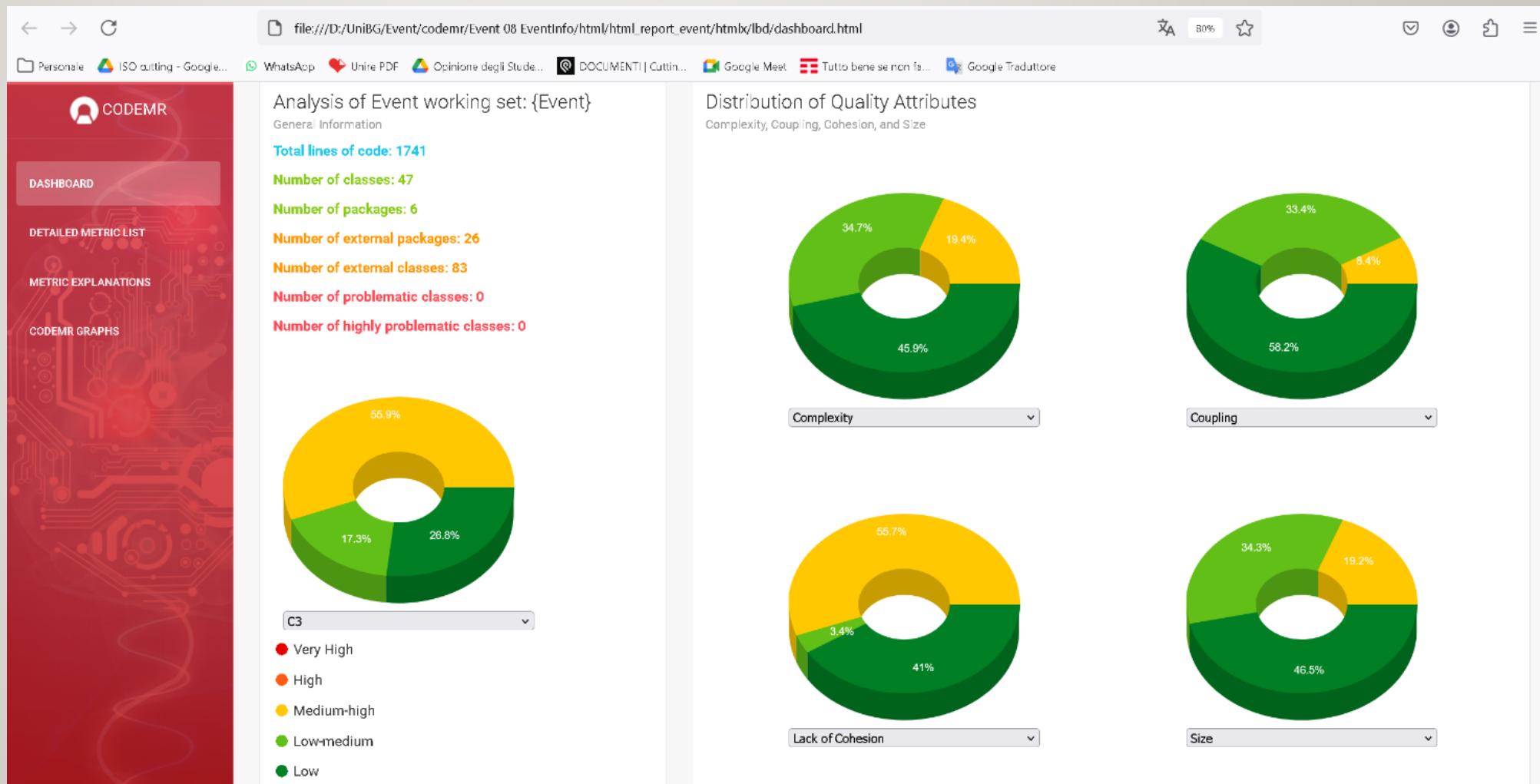
ITERAZIONI INTERMEDI

REFACTORING

- introduzione
WEventInfo

RISULTATO

- coesione
migliorata



ANALISI STATICÀ: CODEMR

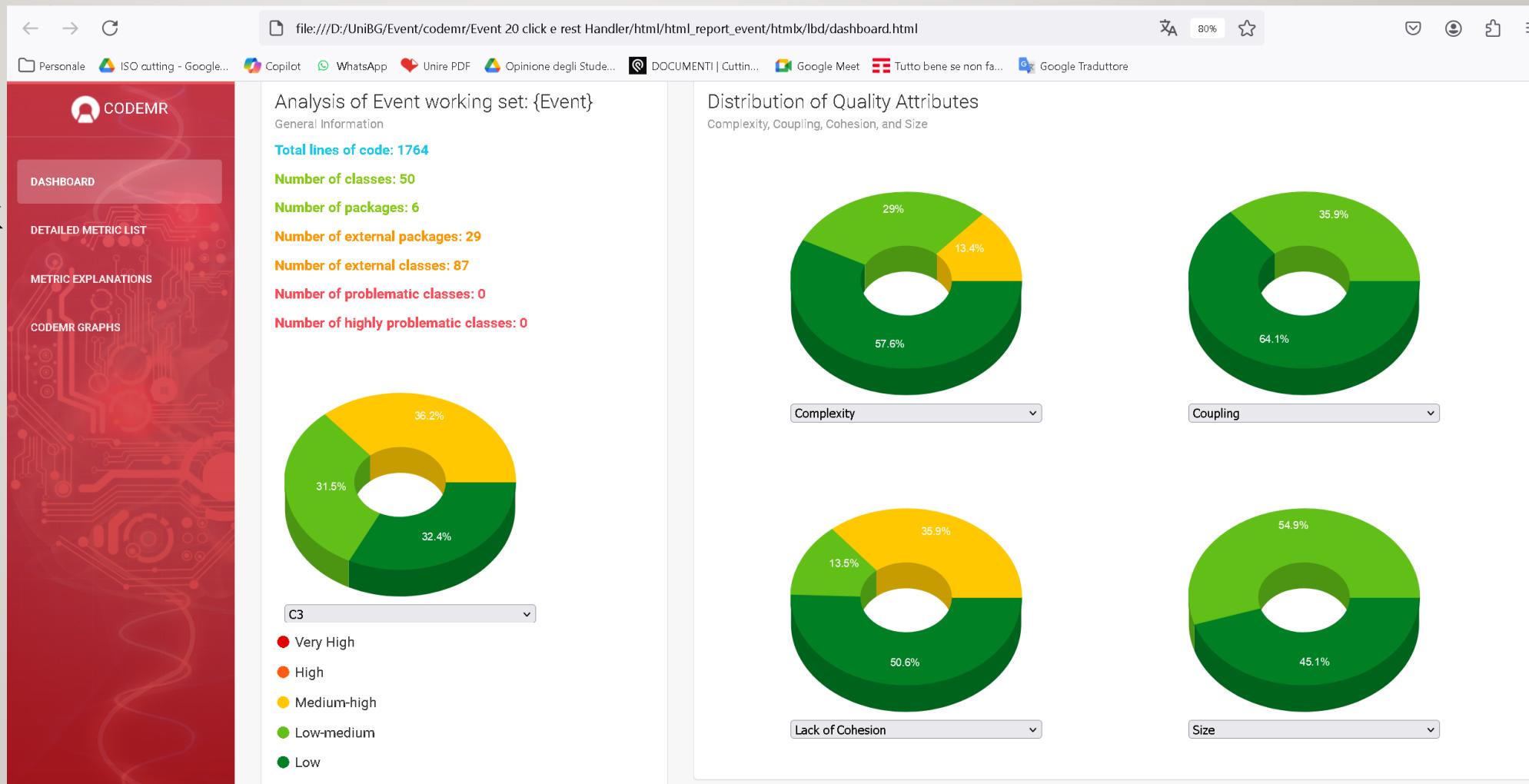
INTERAZIONE FINALE CODEMR/ITERAZIONEFINALE

REFACTORING

- introduzione
- WEventLayout
- WResourceClick
- ... altre minori

RISULTATO

- coupling ottimizzato



ANALISI STATICÀ: CODEMR ITERAZIONE FINALE

- Very High
- High
- Medium-high
- Low-medium
- Low

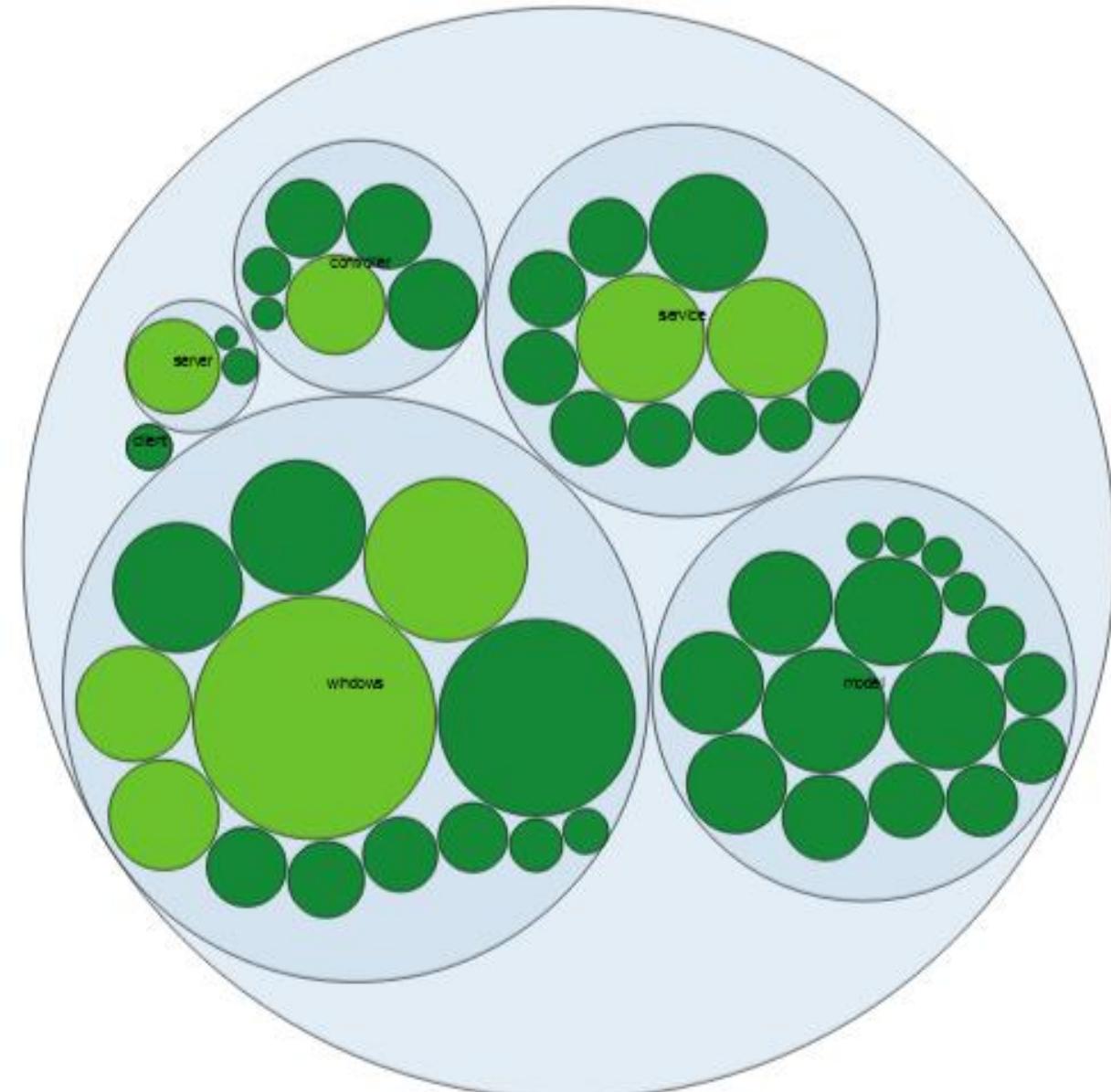
Metric Values by Packages

Select class metric to visualize

Metric Selection

Coupling

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



ANALISI STATICÀ: CODEMR ITERAZIONE FINALE

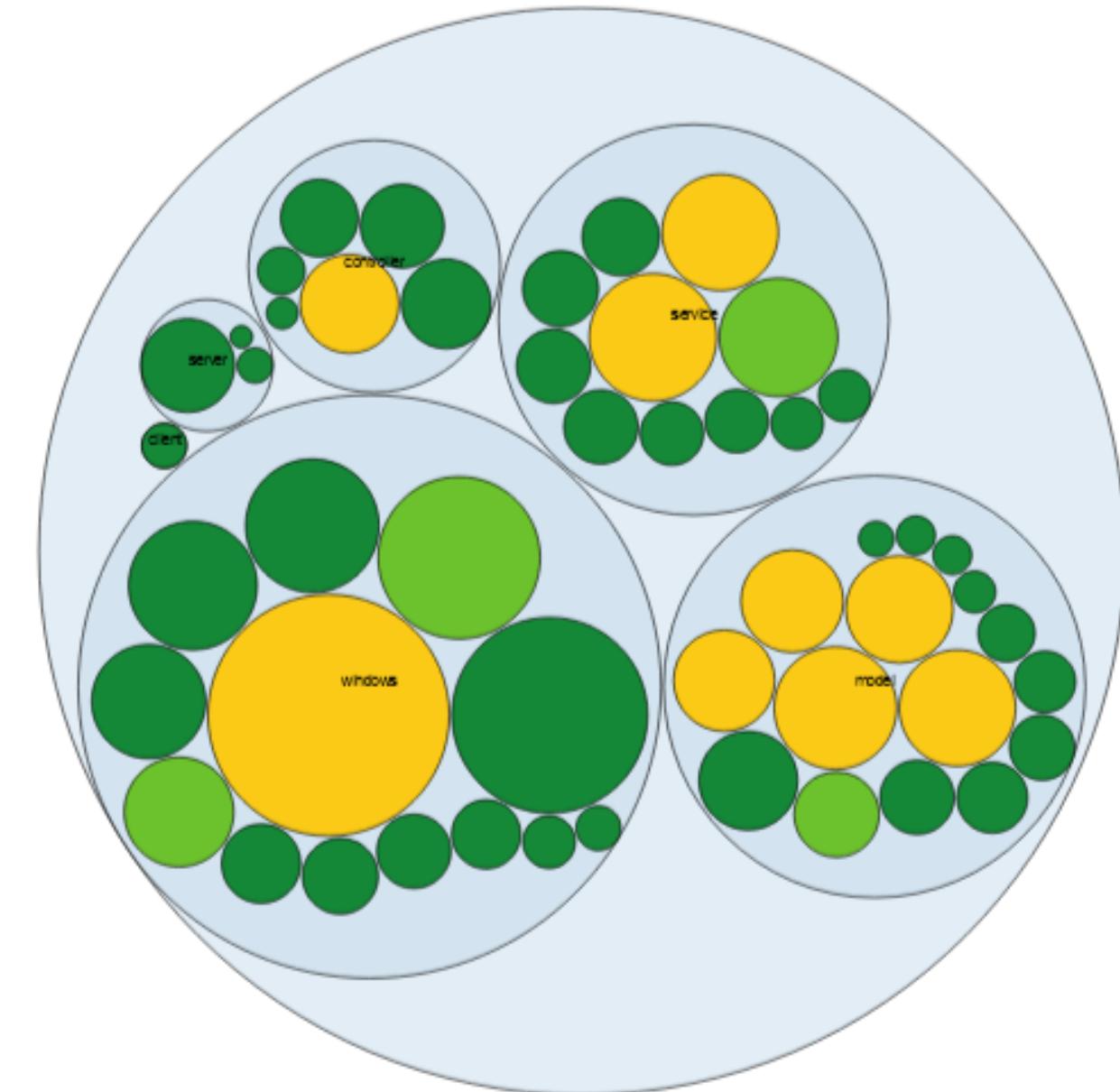
- Very High
- High
- Medium-high
- Low-medium
- Low

Metric Values by Packages
Select class metric to visualize

Metric Selection

Lack of Cohesion

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



ANALISI STATICÀ: CODEMR ITERAZIONE FINALE

- Very High
- High
- Medium-high
- Low-medium
- Low

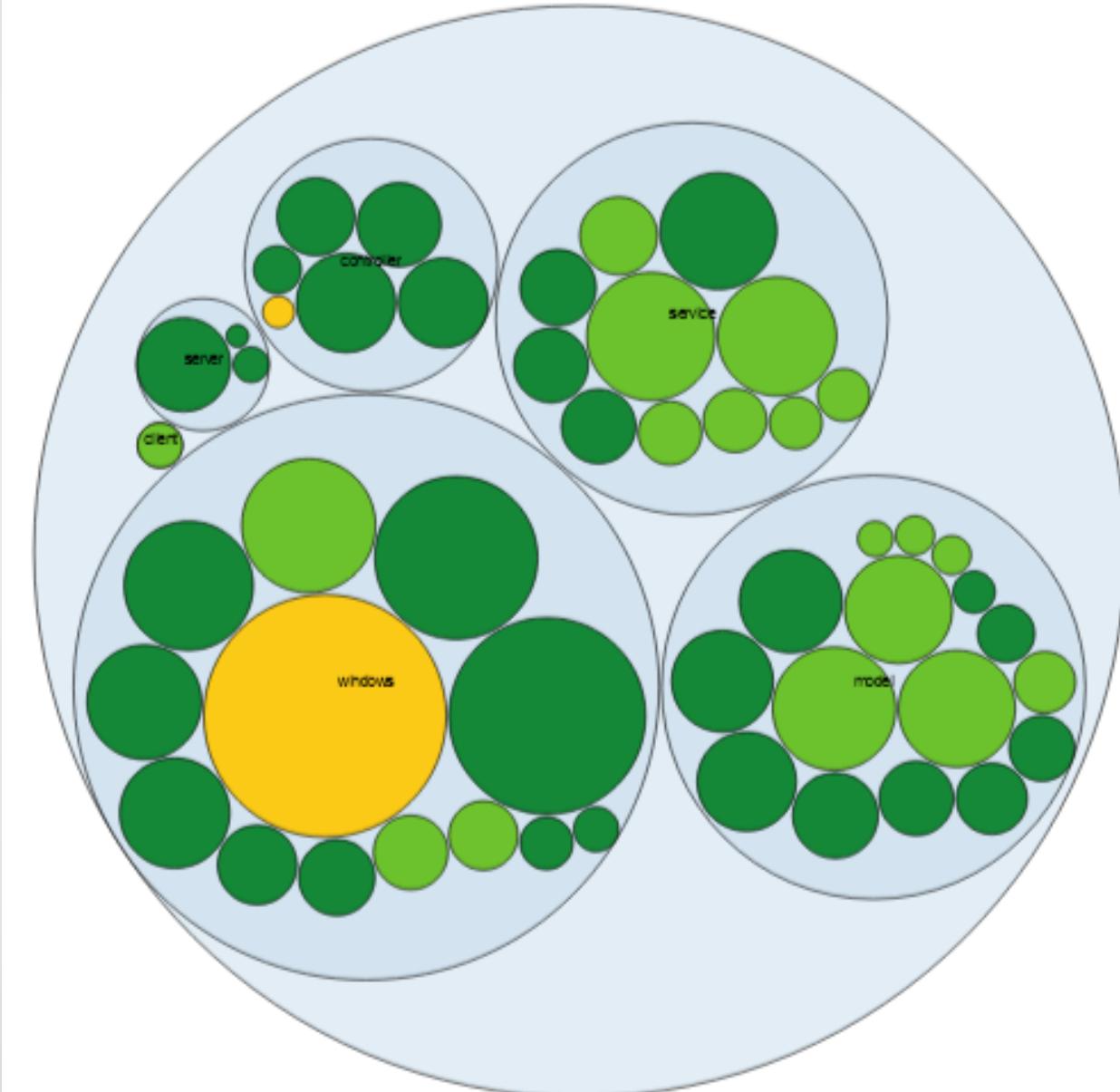
Metric Values by Packages

Select class metric to visualize

Metric Selection

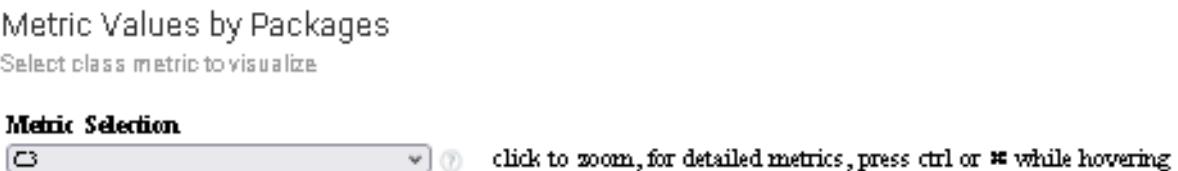
Complexity

ⓘ click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



ANALISI STATICÀ: CODEMR ITERAZIONE FINALE

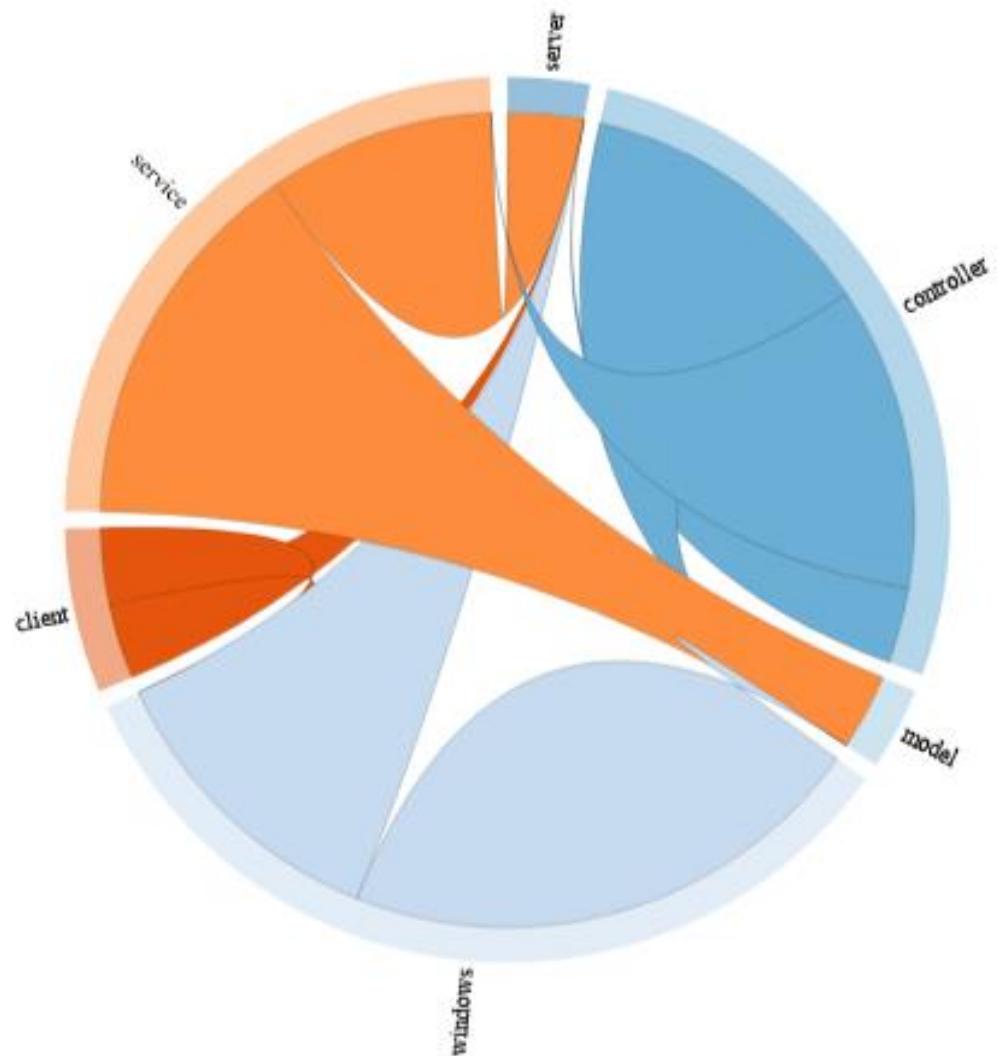
- Very High
- High
- Medium-high
- Low-medium
- Low



ANALISI STATICÀ: CODEMR ITERAZIONE FINALE

- Very High
- High
- Medium-high
- Low-medium
- Low

Package Dependencies
Hover on the wheel to see the details



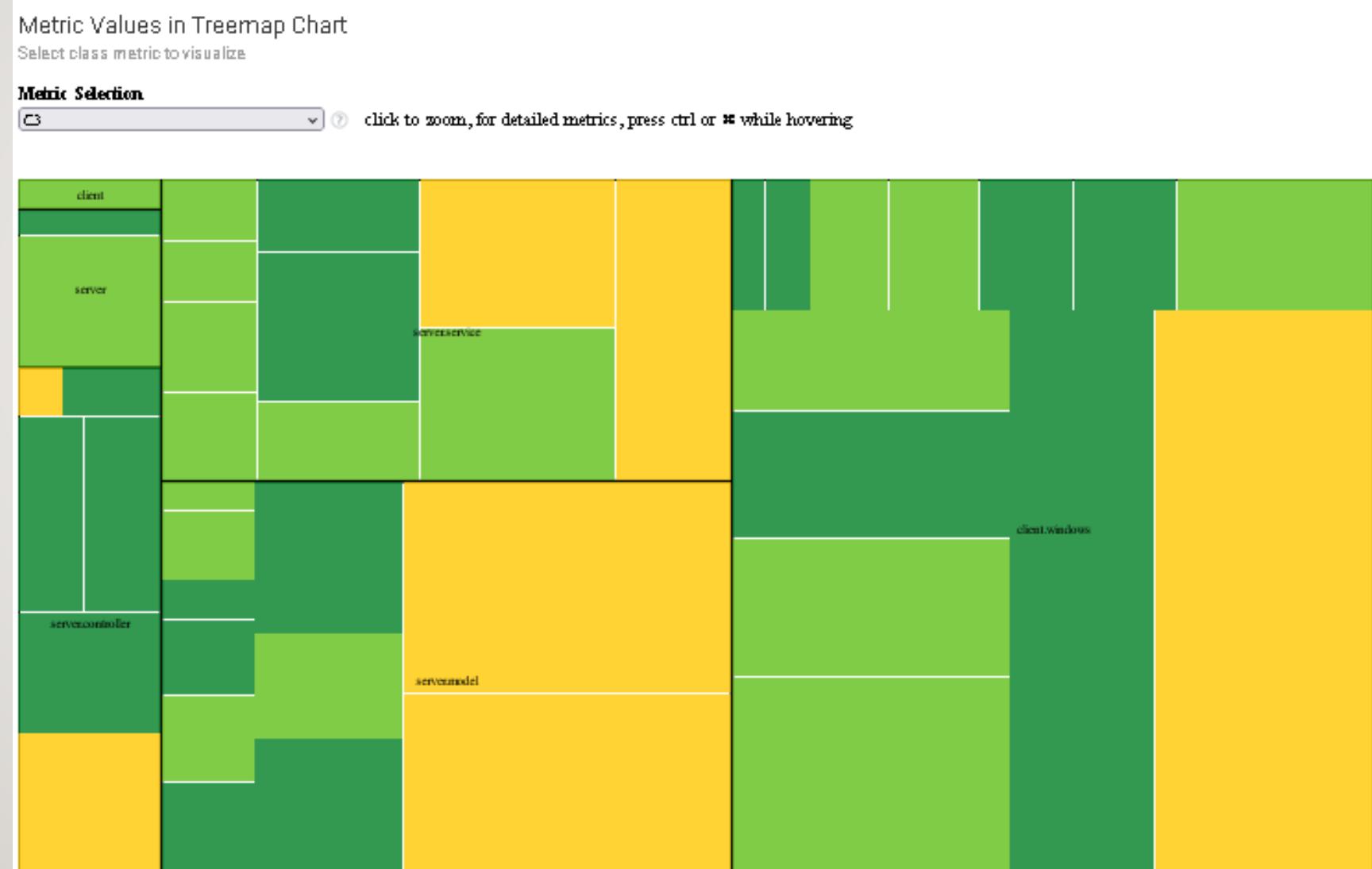
ANALISI STATICÀ: CODEMR

ITERAZIONE FINALE

Per C3 si intende la Related Quality Attributes, ovvero il valore massimo fra

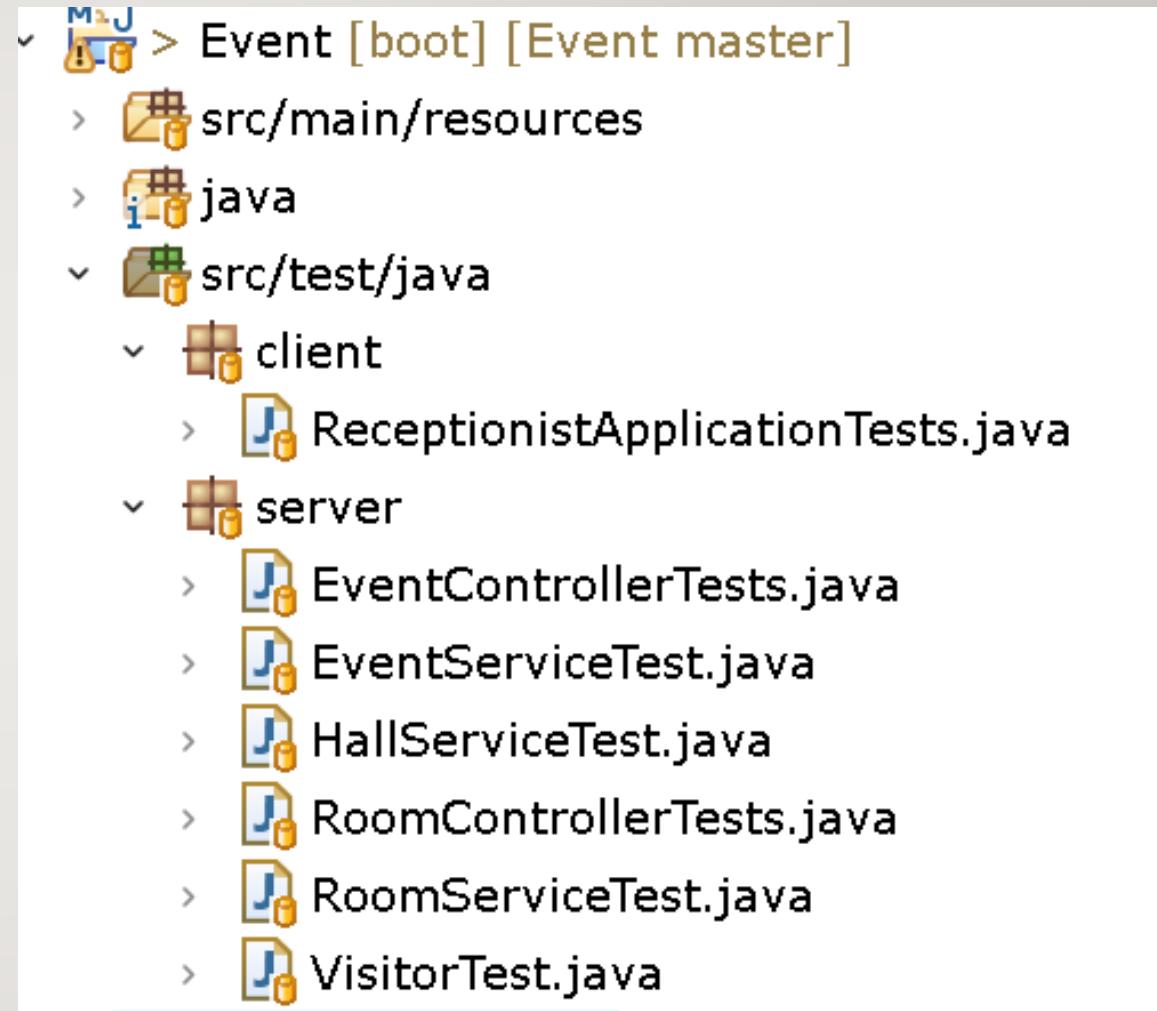
- Coupling,
- Lack of Cohesion,
- Complexity metrics.

- Very High
- High
- Medium-high
- Low-medium
- Low



ANALISI DINAMICA: JUNIT5

- 7 classi di test per un totale di 20 test: non copre tutto il codice, ma i pattern design utilizzati garantiscono comunque una discreta copertura
- La classe meglio testata è la *RoomService*: insieme alla *RoomController* realizzata nella iterazione 2 AMDD
- Tutto il codice relativo alla UI è stato testato verificandone direttamente il funzionamento
- Tutti i test vengono superati dal codice (*keep it green*)



DOCUMENTAZIONE: JAVADOC

JAVADOC/INDEX.HTML

OVERVIEW PACKAGE CLASS USE TREE INDEX HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

SEARCH  Search

All Classes and Interfaces

Interfaces

Classes

Class

Description

[DinamicCol<T extends Resource>](#)

Classe che implementa l'interfaccia `Callback` per la gestione di colonne dinamiche in una tabella.

Menu

La classe rappresenta la finestra di menu principale del client.

StaticCol

Il metodo statico per la creazione di colonne statiche in una tabella.

WAnagrafica

Bozza di finestra per eventuale futura implementazione di ricerca e/o gestione analoga di una camera (da generalizzare alla classe astratta `WResource`).

WCalendar

Classe per la gestione della finestra di visualizzazione del calendario.

WEvent

Classe per la gestione della finestra degli eventi

WEventLayout

Superclasse di `WEvent`, contiene i metodi per la definizione dei campi e del layout della finestra.

WEventRest

Interfaccia per la gestione delle richieste REST effettuate dalla finestra `WEvent`.

WHall

Classe per la gestione della finestra di visualizzazione delle sale.

[WResource<T extends Resource>](#)

Classe generica per la gestione della finestra con l'elenco delle risorse e la loro disponibilità.

[WResourceClick<T extends Resource>](#)

Classe per la gestione degli eventi di click sulle celle della tabella gestita da `WResource`.

[WResourceRest<T extends Resource>](#)

Classe per la gestione delle risorse tramite richieste REST ai vari controller.

WRoom

Classe per la gestione della finestra di visualizzazione delle stanze.

DBMS: MONGODB E ATLASDB

VERSIONE 2

I DB sono suddivisi in Collection (l'equivalente delle relazioni/tabelle di un DBMS relazionale) a loro volta suddivisi in documenti (le tuple/record per un DBMS relazionale). MongoDB assegna un id a ciascun documento.

Su Atlas è necessario, innanzitutto creare un account e un Cluster definendone la dimensione e l'ubicazione (ed altre caratteristiche che concorrono a definirne il costo). Ogni cluster può contenere più DB. L'entità logica di più alto livello è il Progetto (che può usare più cluster).

The screenshot shows the MongoDB Compass interface. At the top, it says "Atlas DB" and "Event > Camere". On the left, there's a sidebar with "Databases" (Event, Calendario, Eventi, Sale, Singleton, admin, local), "My Queries", "Performance", and "Connections". The main area is titled "Documents" with a count of 5. It shows three documents with their IDs, numerical letti values, tipi ("MINISUITE" and "SUITE"), disponibilità arrays, names ("Room001" and "Room002"), and costs (120 and 100). Below the documents, there are buttons for "ADD DATA", "EXPORT DATA", "UPDATE", and "DELETE". At the bottom, there's a search bar and some navigation links.

The screenshot shows the MongoDB Atlas Project Overview page for "Project 0". The left sidebar includes "DEPLOYMENT" (Database selected), "Data Lake", "SERVICES" (Device & Edge Sync, Triggers, Data API, Data Federation, Search, Vector Search, Stream Processing, Migration, SECURITY), "Data Services" (selected), "App Services", "Charts", and "Collections" (selected). The main area shows "DATABASES: 1" and "COLLECTIONS: 5". The "Event.Camere" collection is highlighted. It provides storage details (STORAGE SIZE: 24KB, LOGICAL DATA SIZE: 64.2KB, TOTAL DOCUMENTS: 5, INDEXES TOTAL SIZE: 20KB), and tabs for "Find", "Indexes", "Schema Anti-Patterns", "Aggregation", and "Search Indexes". A "Generate queries from natural language in Compass" button is available. At the bottom, there's a "QUERY RESULTS: 1-5 OF 5" section with the same document details as the first screenshot.

CLASS DIAGRAM

(SET COMPLETO)

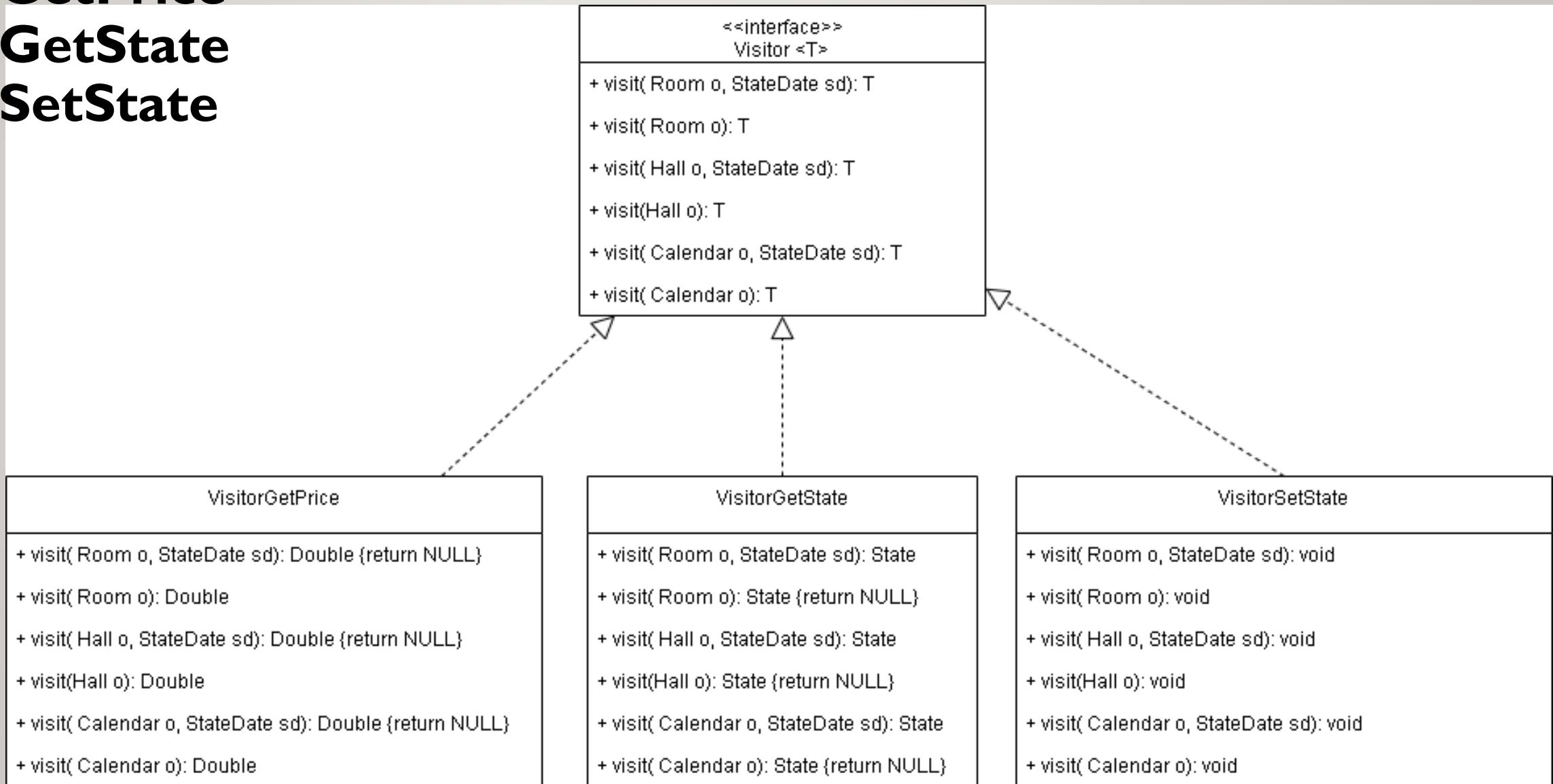
CLASS DIAGRAM:

Visitor

VisitorGetPrice

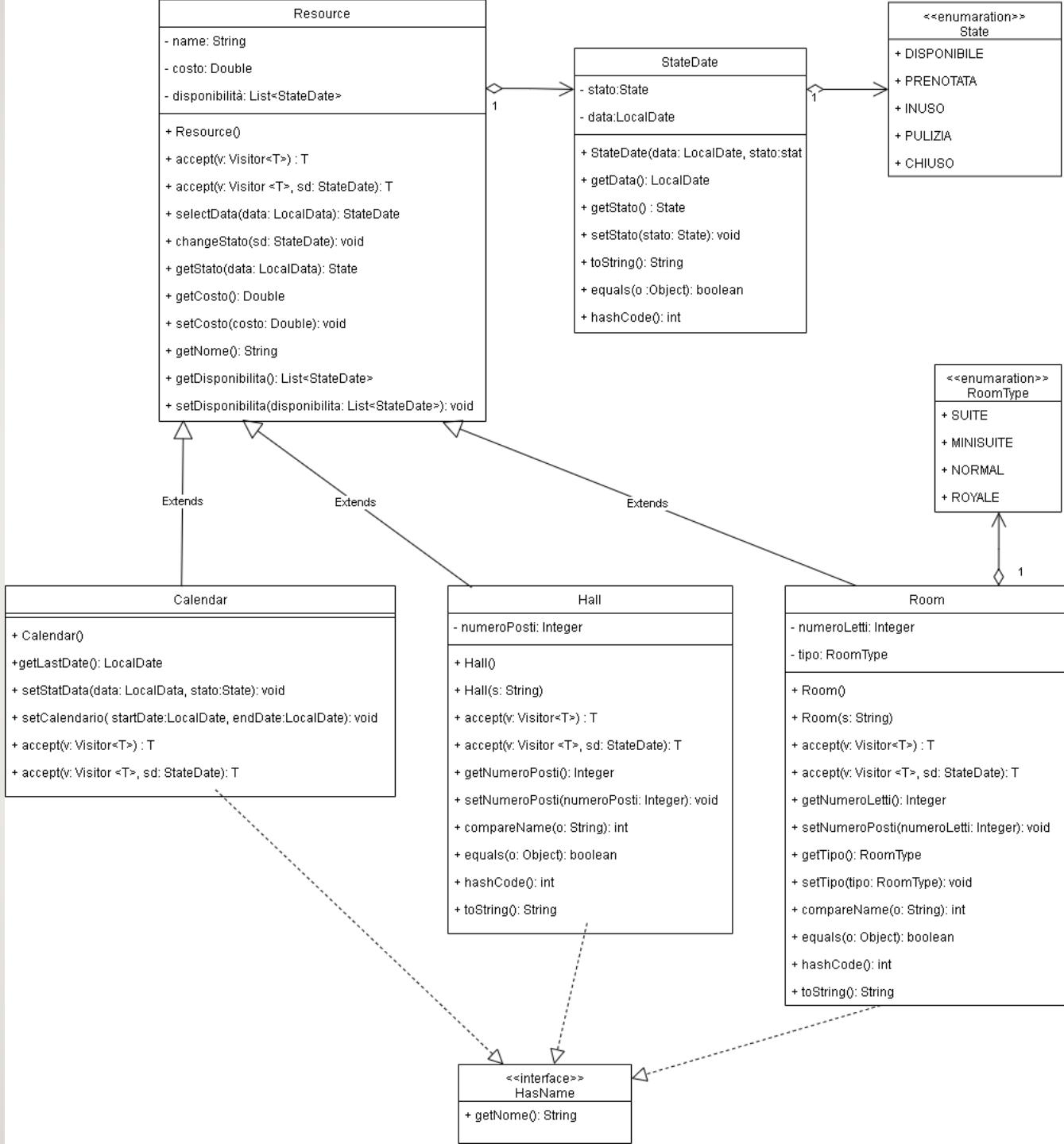
VisitorGetState

VisitorSetState



CLASS DIAGRAM:

Resource Calendar Hall Room

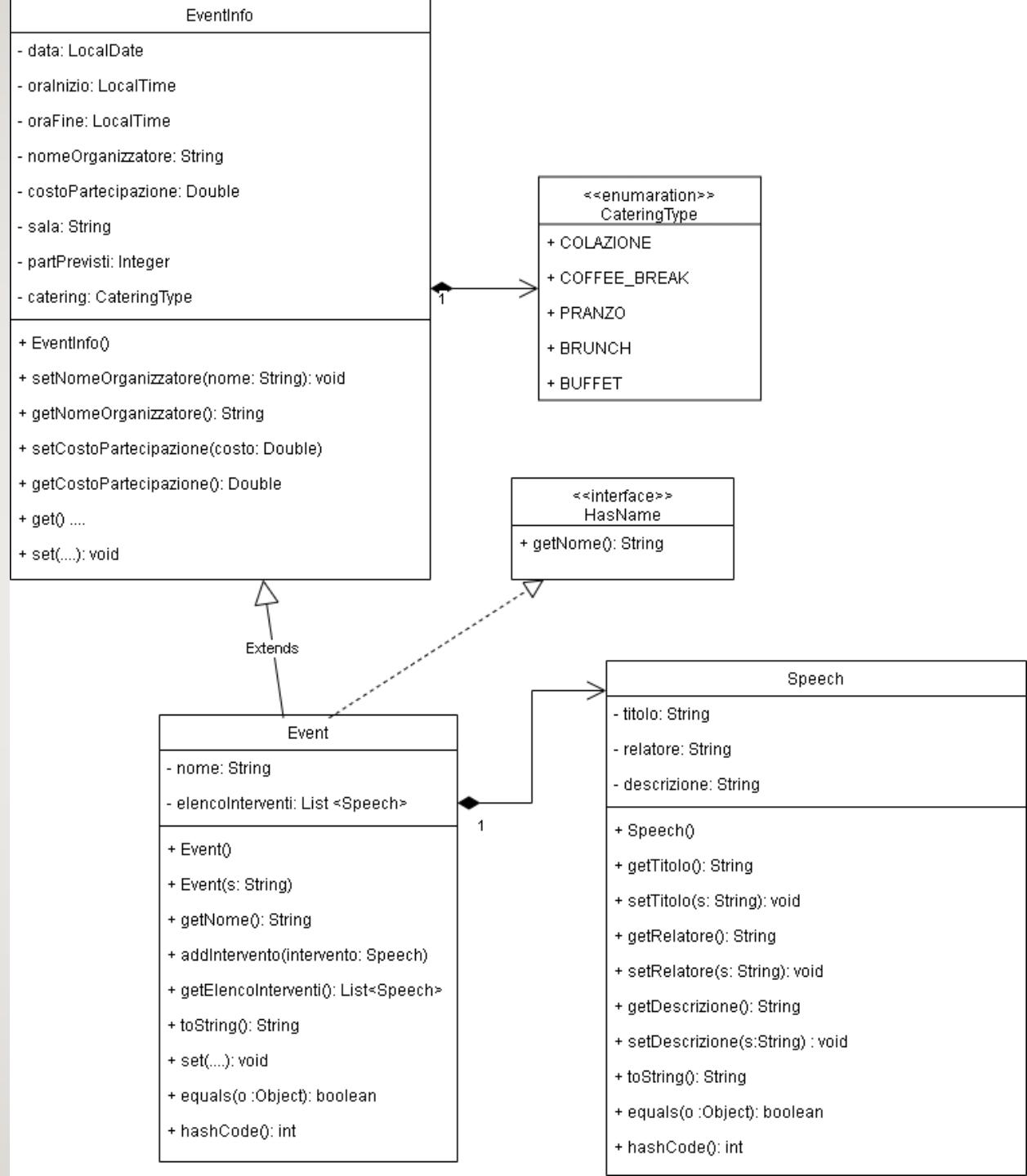


CLASS DIAGRAM:

Event

EventInfo

Speech



CLASS DIAGRAM:

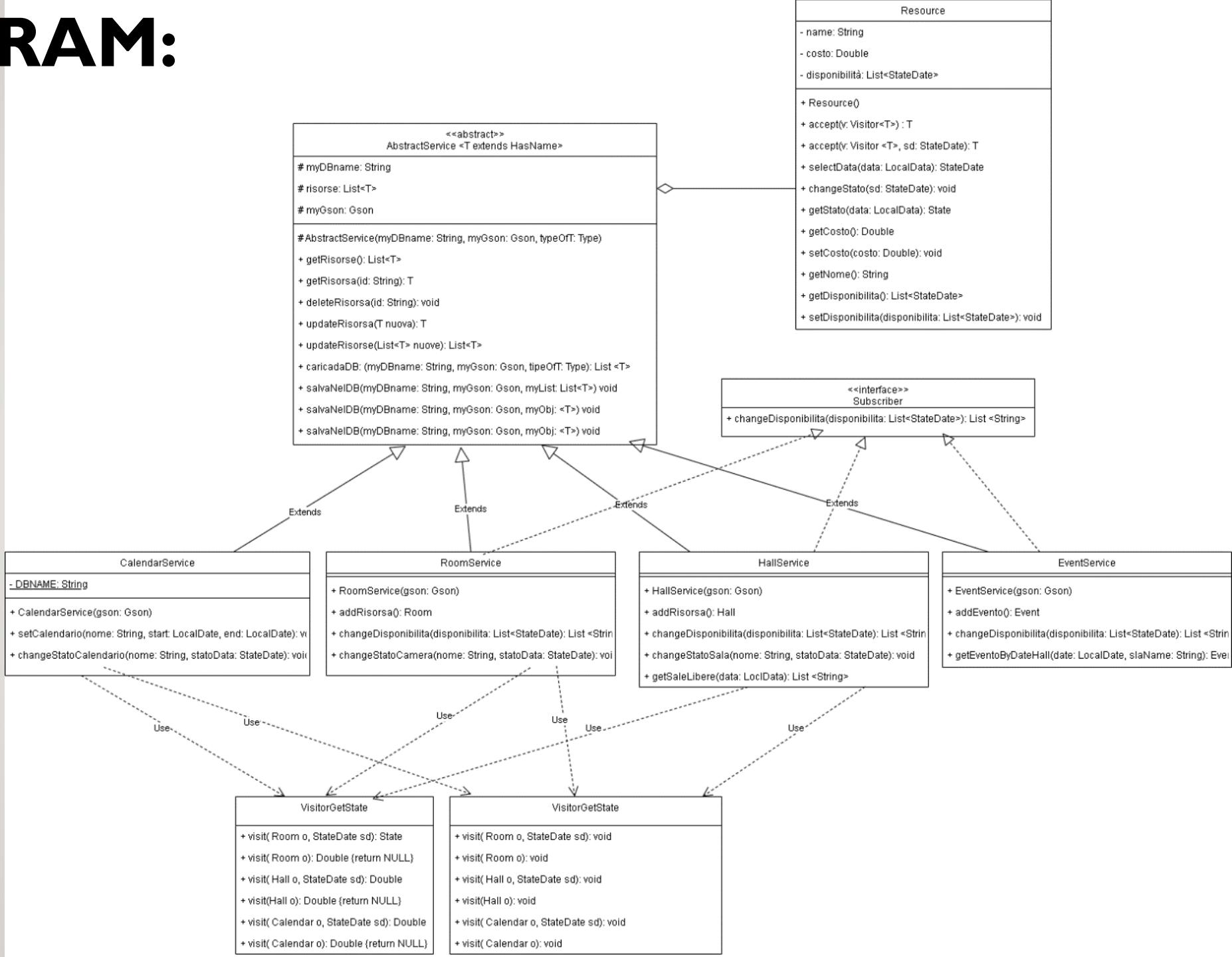
AbstractService

CalendarService

RoomService

HallService

EventService



CLASS DIAGRAM:

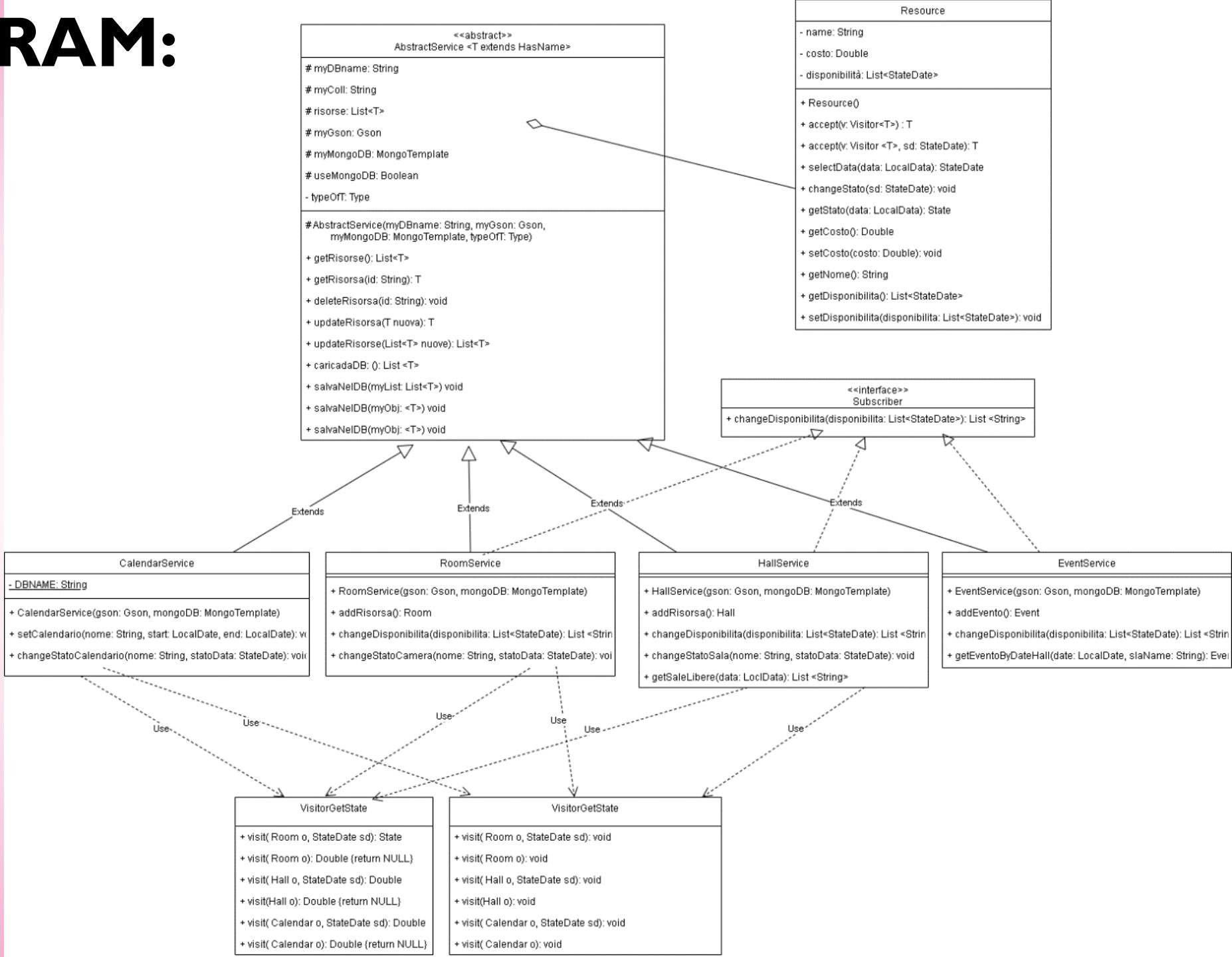
AbstractService

CalendarService

RoomService

HallService

EventService

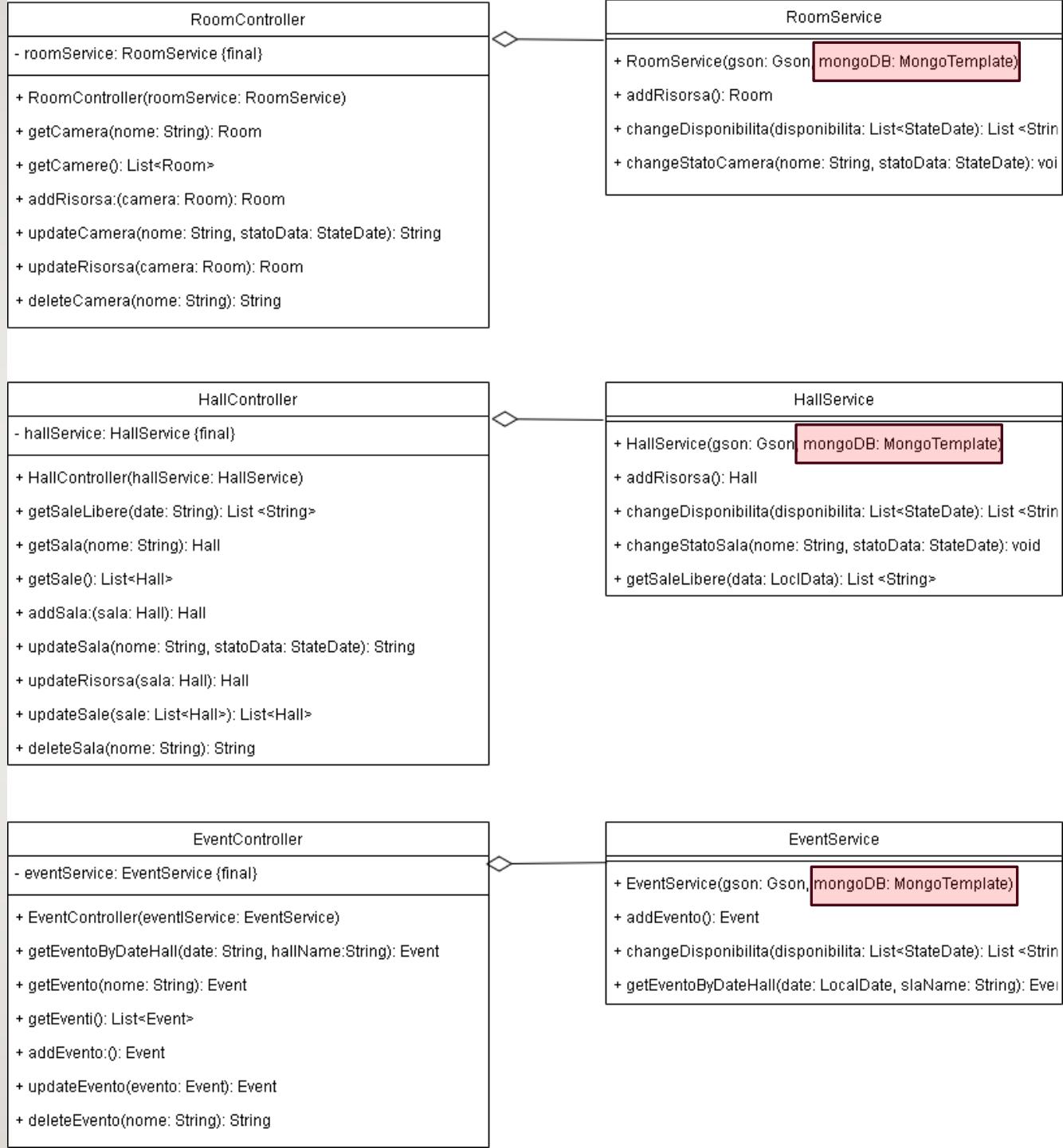


CLASS DIAGRAM:

RoomController

HallController

EventController



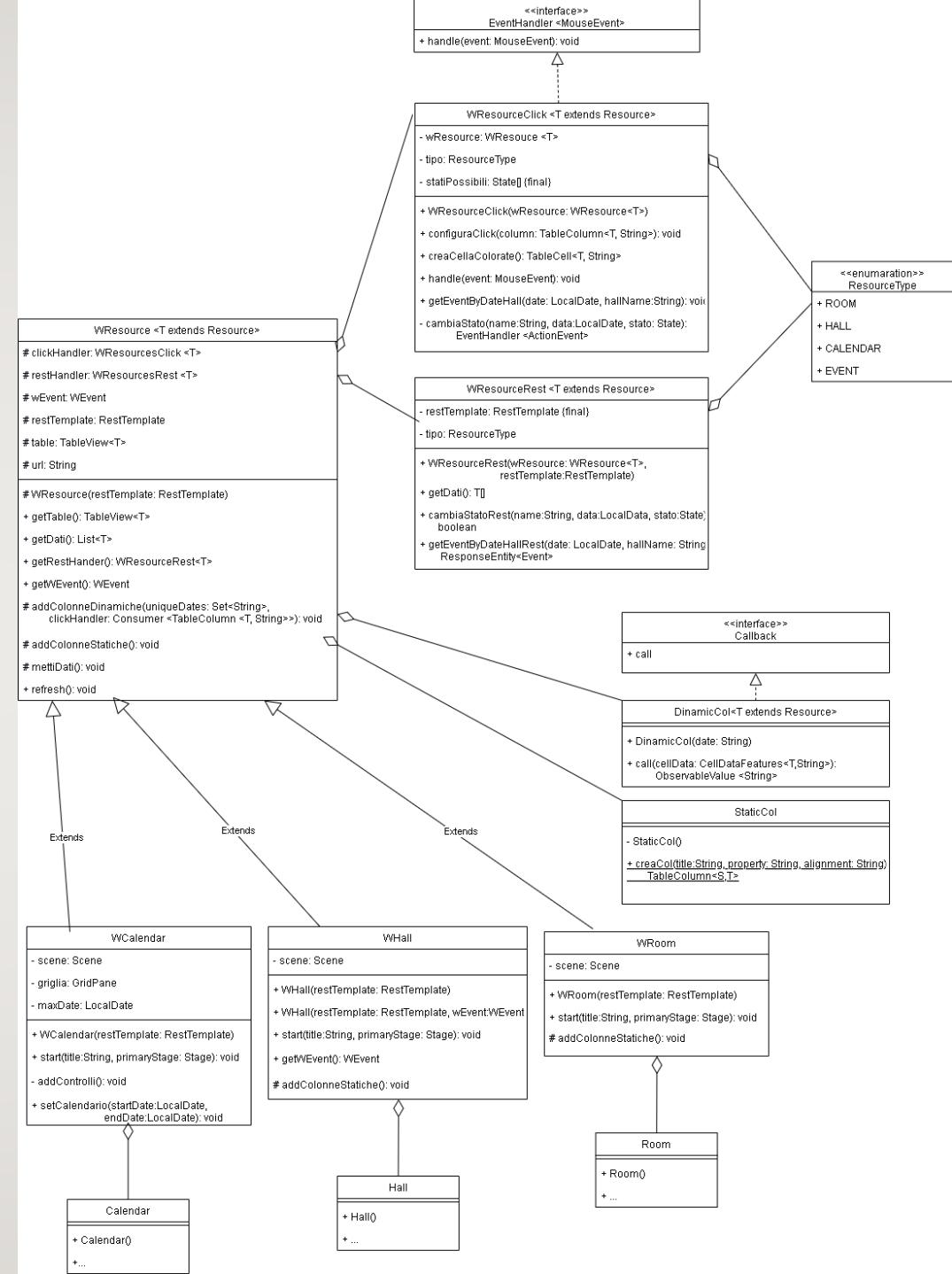
CLASS DIAGRAM:

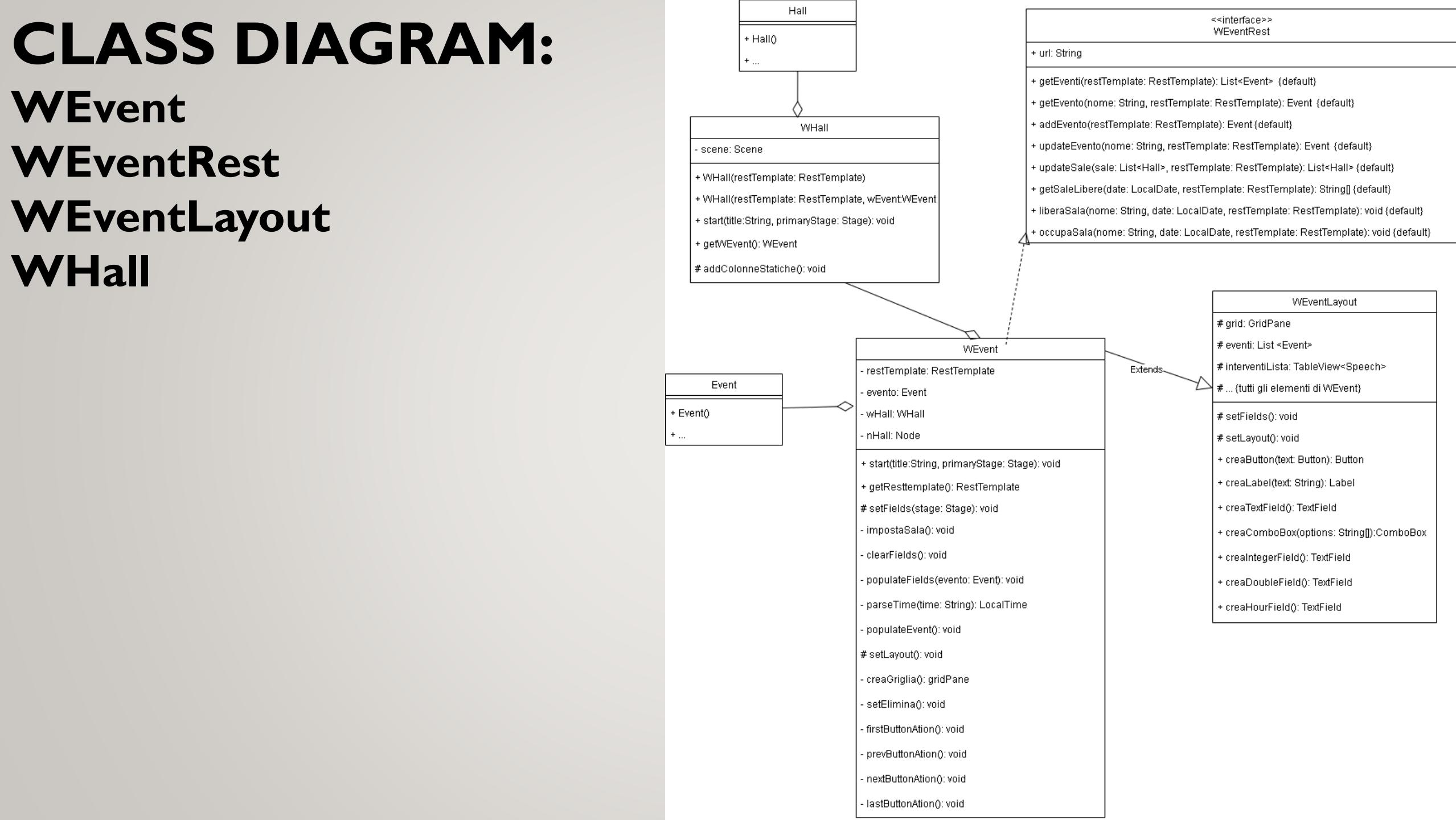
WResource

WCalendar

WHall

WRoom





CLASS DIAGRAM:

ResponsabileApplication

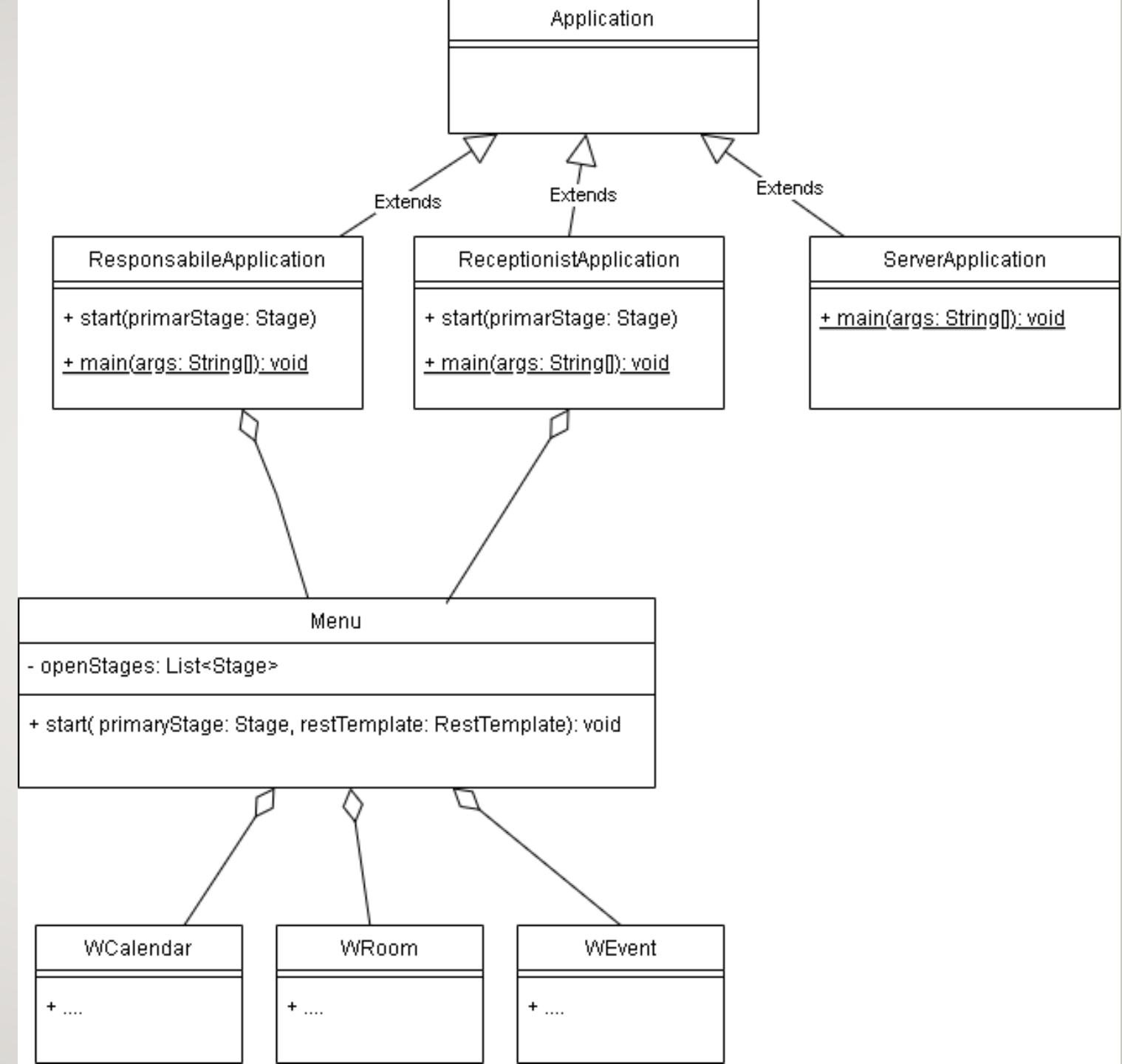
ReceptionistApplication

ServerApplication

Menu

JavaFX non consente l'esecuzione di più applicationi sulla stessa JVM: uno dei due client viene lanciato da terminale con il seguente bat.

```
1 @echo off
2 echo Avvio dell'applicazione Responsabile...
3
4 rem Imposta il percorso del progetto
5 set PROJECT_DIR=%~dp0
6
7 rem Esegui l'applicazione ResponsabileApplication utilizzando Maven
8 mvn exec:java -Dexec.mainClass="client.ResponsabileApplication"
9
10 pause
11
```



CLASS DIAGRAM:

AppConfig

