Università degli Studi di Verona

BOOKS EXCHANGE

Ingegneria del Software

Canditati:
Federico Bianchi - VR369001
Matteo Olivato - VR371734

BOOKS EXCHANGE

È un servizio web che consente di condividere e scambiare il proprio patrimonio librario con altri Utenti registrati al servizio.

La necessità di avere un servizio che faciliti la condivisione di libri ha portato alla realizzazio ne di questa applicazione che ha attraversato delle fasi di sviluppo che si possono riassume re nel seguente **Ciclo di Vita**:

- Analisi;
- Progettazione;
- Scrittura;
- Convalida.

Processo di Sviluppo → **Modello a spirale**

- È un modello di processo in cui ogni iterazione del ciclo produce una versione del software più raffinata;

- Insieme di risultati intermedi che garantiscono che il prodotto sia soddisfacente
  → Meno rischi

Il Processo di sviluppo del progetto ha portato alla realizzazione dei seguenti Documenti:

- Vision → Business needs, Attori del Sistema, Problemi e Soluzioni;
- Caratteristiche → Caratteristiche e Tecnologie utilizzate;
- Use Case → Casi d'uso;
- Project Plan → Pianificazione e tempi di sviluppo del Progetto;
- Risk List → Top 4 dei Rischi.

DOCUMENTO di VISION (Vision BooksExchange.pdf)

Documento introduttivo del Progetto scritto in un linguaggio naturale, individua le **Business Needs** e:

→ Problemi: tipo di problema, chi riguarda, per quale aspetto;
→ Soluzioni: ad ogni problema, quali vantaggi portano, alternative;
→ Perchè utilizzare Books Exchange?
→ Gli Stakeholder;
→ Attori e ruoli nel Sistema con attribuzione agli Stakeholder.

DOCUMENTO di CARATTERISTICHE (Caratteristiche BooksExchange.pdf)

Che identifica a partire dal documento di Vision, le componenti fisiche, logiche e tecnologiche del Sistema e come queste interagiscono tra loro.

Requisiti Funzionali
→ modellati secondo il **FURPS+**
Requisiti non Funzionali

Requisiti Funzionali: Gli attori del Sistema e quali funzionalità possiedono
Utente: Registrazione - Autenticazione - Condivisione - Ricerca
Amministratore: Gestione del Sistema – Gestione dei Correttori di Bozze
Correttore di Bozze: Gestione delle Inserzioni

Documento Use Case BooksExchange.pdf *per una trattazione completa.*

Requisiti non Funzionali:
→ *Usabilità*: Pagina Help Online;
→ *Performance*: Tempo di accesso medio di 5s, Tempo medio di ricerca 3s;
→ *Affidabilità*: Presenza di un Amministratore che monitora il Sistema;
→ *Scalabilità*: Server Remoto a supporto per i dati multimediali;
→ *Supportabilità*: Servizio accessibile a tutti i dispositivi muniti di browser;
→ *Sicurezza*: Servizio protetto con autenticazione e assegnazione privilegi;
→ *Robustezza*: Messaggi di errore in caso di malfunzionamento.

Gestione di Progetto

WBS(Work Breakdown Structure)

È una rappresentazione gerarchica ad albero dove ogni livello rappresenta graficamente la scomposizione del lavoro da svolgere per costruire i deliverables.
Programmazione dei risultati non delle azioni.
Deve rispettare la regola del 100%!

OBS(Organization Breakdown Structure)

È una decomposizione gerarchica delle responsabilità del progetto dove viene individuata la persona o l'ufficio competente per ogni pacchetto di lavoro.

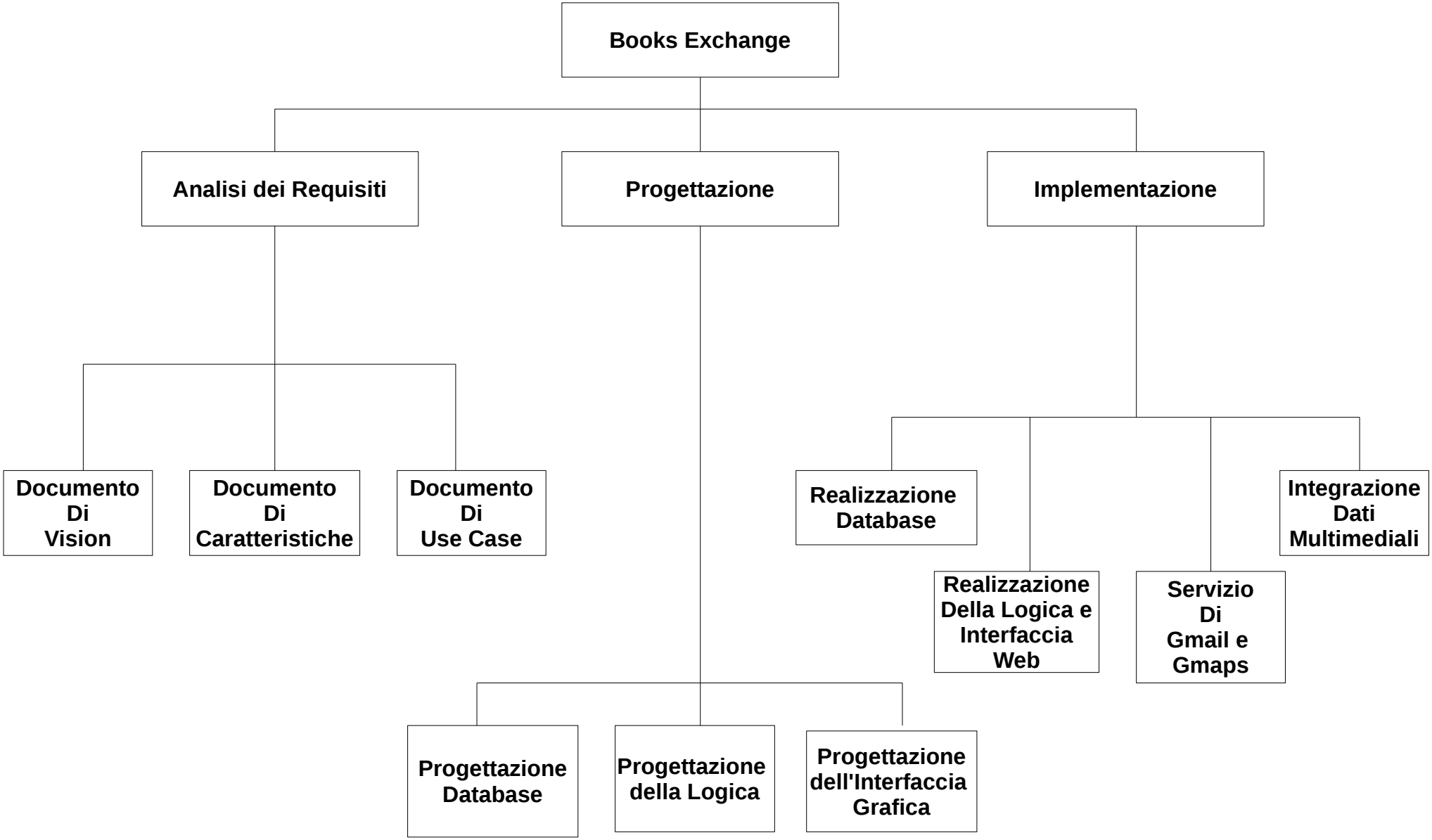
RAM (Responsibility Assigment Matrix)

Deriva dall'incrocio della WBS e della OBS. Si ha quindi la:

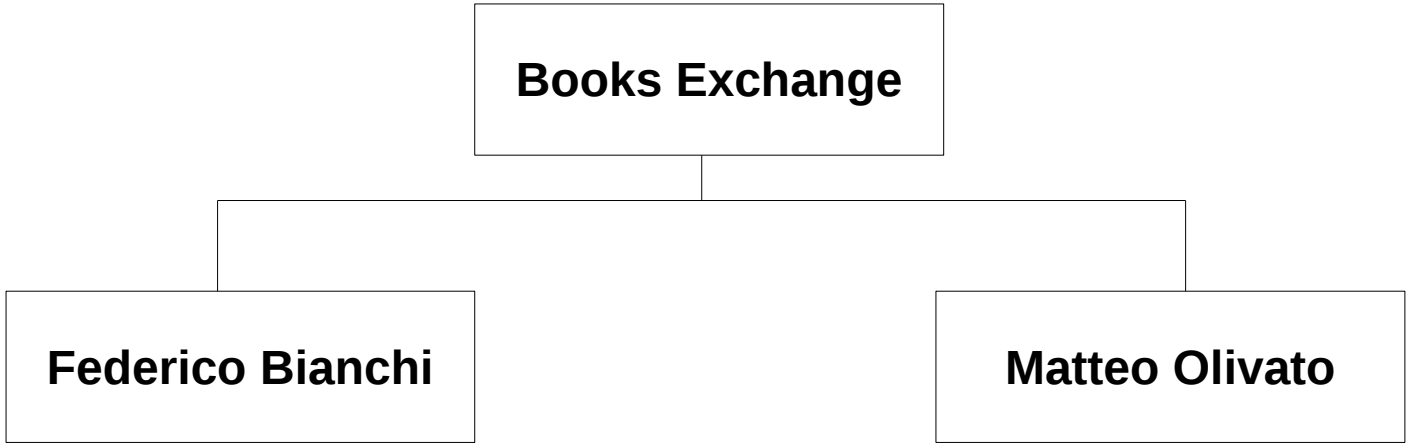
RACI (Responsible, Accountable, Consulted, Informed)

È una tabella che incrocia le attività con le strutture organizzative per l'assegnamento dei ruoli.

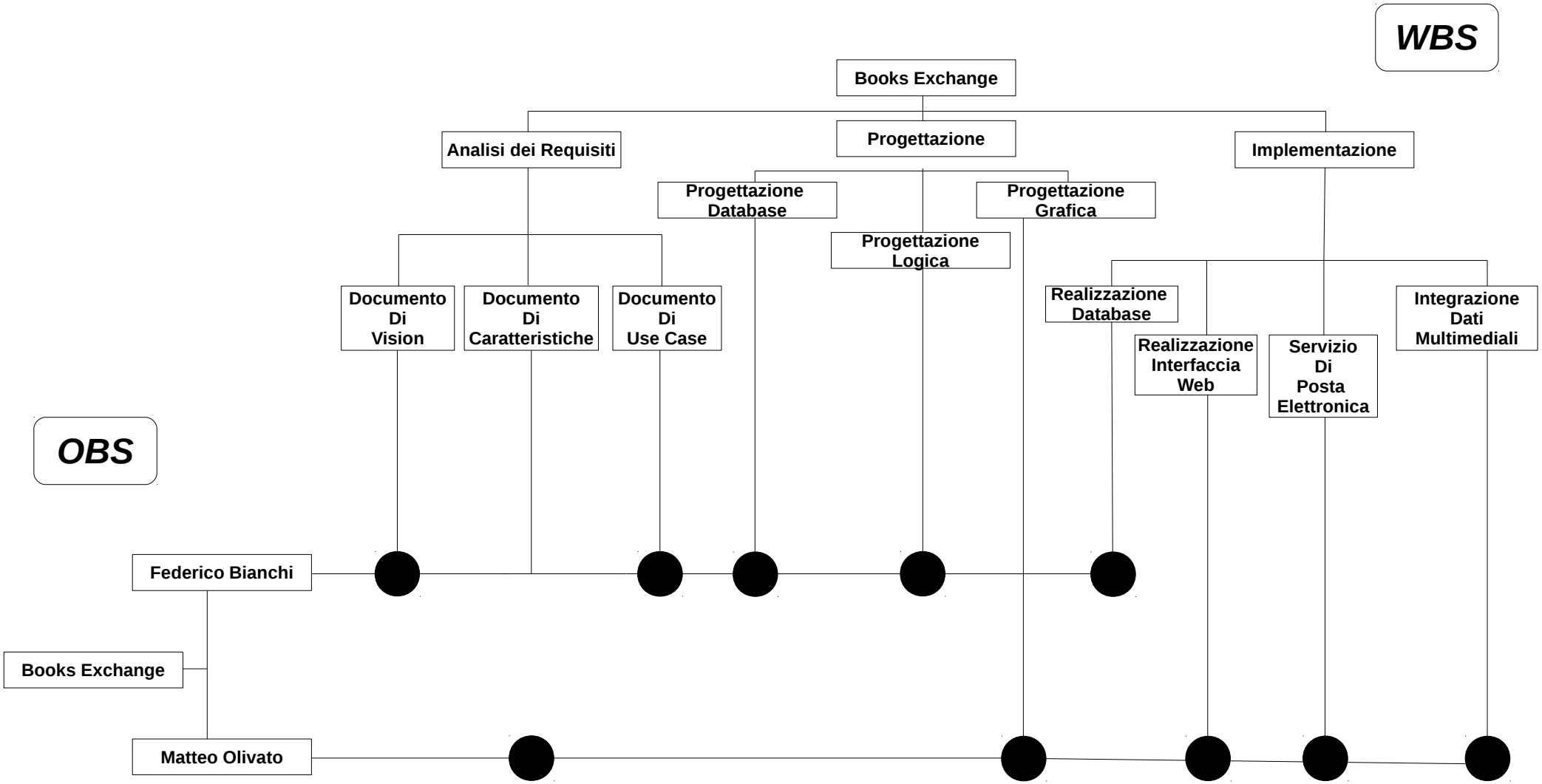
WBS(Work Breakdown Structure)



OBS(Organization Breakdown Structure)



RAM (Responsibility Assigment Matrix)



RACI (Responsible, Accountable, Consulted, Informed)

→ **R (RESPONSIBLE)** con la lettera “R” viene indicato il responsabile della realizzazione.
Colui che esegue un' attività.

→ **A (ACCOUNTABLE)** con la lettera “A” viene indicato il responsabile.
Colui che ha il potere di veto.

→ **C (CONSULTED)** con la lettera “C” viene indicato colui che è consultato prima dell'esecuzione di un'attività

→ **I (INFORMED)** con la lettera “I” viene indicato colui che è informato successivamente della decisione o dell'azione intrapresa.

RACI (Responsible, Accountable, Consulted, Informed)

RACI	Federico Bianchi	Matteo Olivato	Riccardo Pret
Documento di Vision	RCI	ACI	
Documento di Caratteristiche	ACI	RCI	
Documento di Use Case	RCI	ACI	C
Progettazione Database	RCI	ACI	
Progettazione Logica	RCI	ACI	
Progettazione Grafica	ACI	RCI	
Realizzazione Database	RCI	ACI	
Realizzazione Interfaccia web	ACI	RCI	
Servizio di Posta Elettronica	ACI	RCI	
Integrazione Dati Multimediali	ACI	RCI	
Test	ACI	RCI	C

Reticolo di Progetto - Si distinguono tre elementi fondamentali:

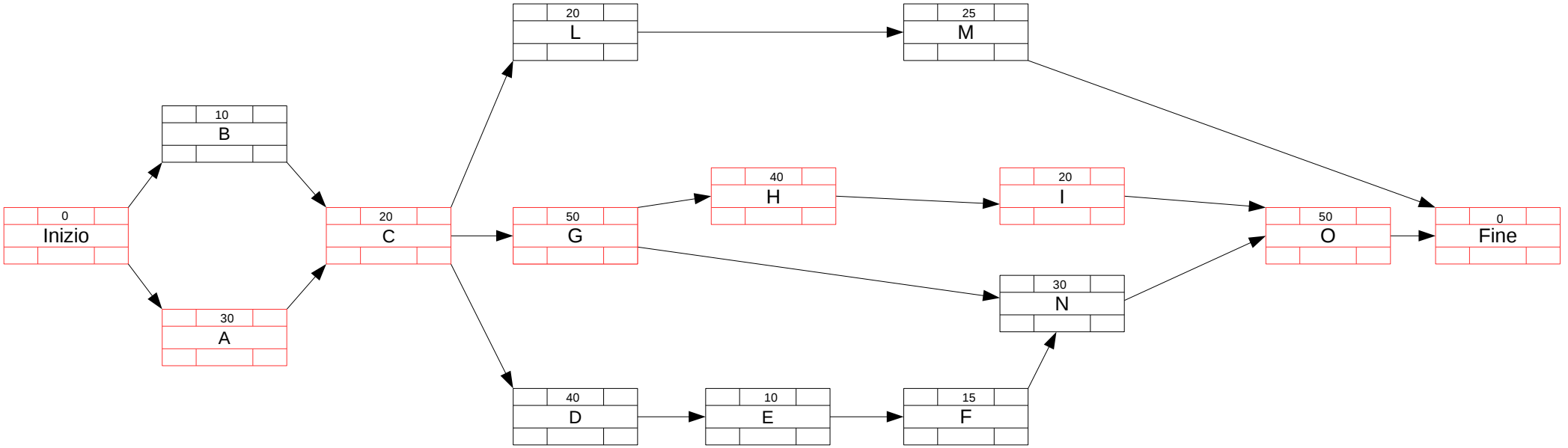
Evento → Stato rappresentativo di un progetto. Può essere l'inizio o la fine di Un'attività;
Attività → Elemento di lavoro all'interno del progetto coerente con la WBS;
Durata → Stima del tempo necessaria per il completamento di un'attività

CPM (Critical Path Method) - Ci consente di capire la durata del progetto e di stabilire quali attività sono critiche e quindi formano un percorso critico e quali attività invece possono subire degli slittamenti.
È costituito da:

→ Early Start(ES): Quanto un'attività può iniziare anticipatamente;
→ Early Finish(EF): Quanto un'attività può finire anticipatamente;
→ Late Start(LS): Quanto un'attività può iniziare in ritardo;
→ Late Finish(LF): Quanto un'attività può finire in ritardo;
→ Free Float(FF): Margine temporale che stabilisce quanto può slittare un'attività senza influenzare l'ES delle attività successive;
→ Total Float(TF): Margine temporale che stabilisce quanto può slittare un'attività senza influenzare la data di completamento del progetto.

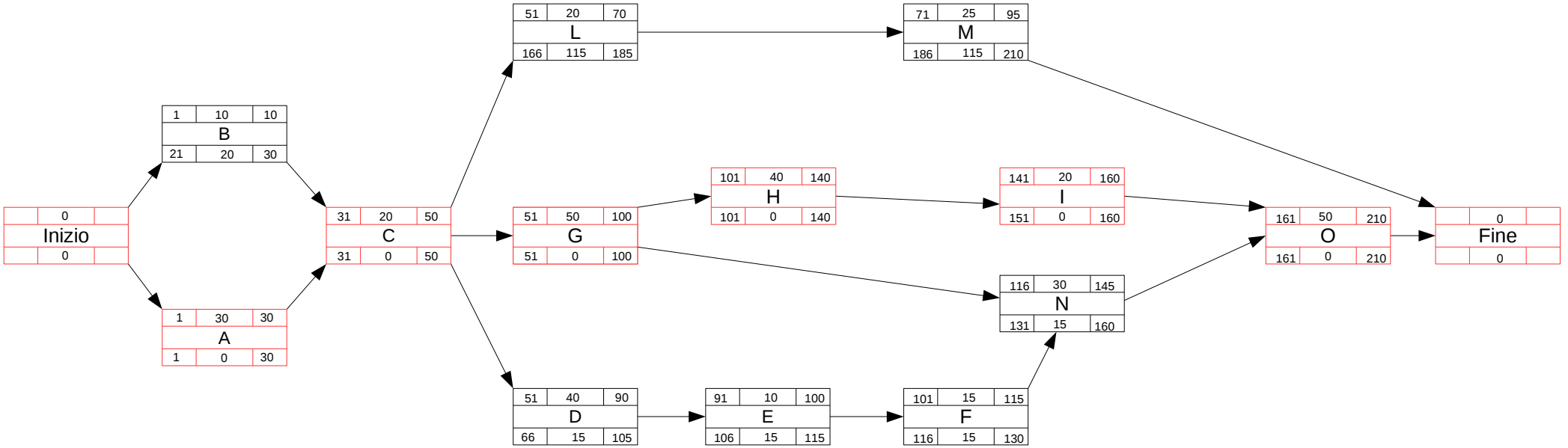
GANTT - Rappresentazione grafica di un calendario di attività, consente di capire facilmente l'avanzamento o lo slittamento delle attività.
Consente inoltre di evidenziare il percorso critico.

Reticolo di Progetto



A → Abbiamo specificato i requisiti
B → Abbiamo specitifato le caratteristiche
C → Abbiamo specificato le tecnologie
D → Progetto del Database
E → DDL
F → DML
G → Progetto della logica di controllo in Java
H → Realizzazione delle classi per la logica
I → Integrazione Gmail, Dati Multimediali remoti
L → Progetto dell'interfaccia utente tramite JSF
M → Realizzazione classi + XML
N → Interrogazioni Database
O → Logica che interagisce con il Database

CPM

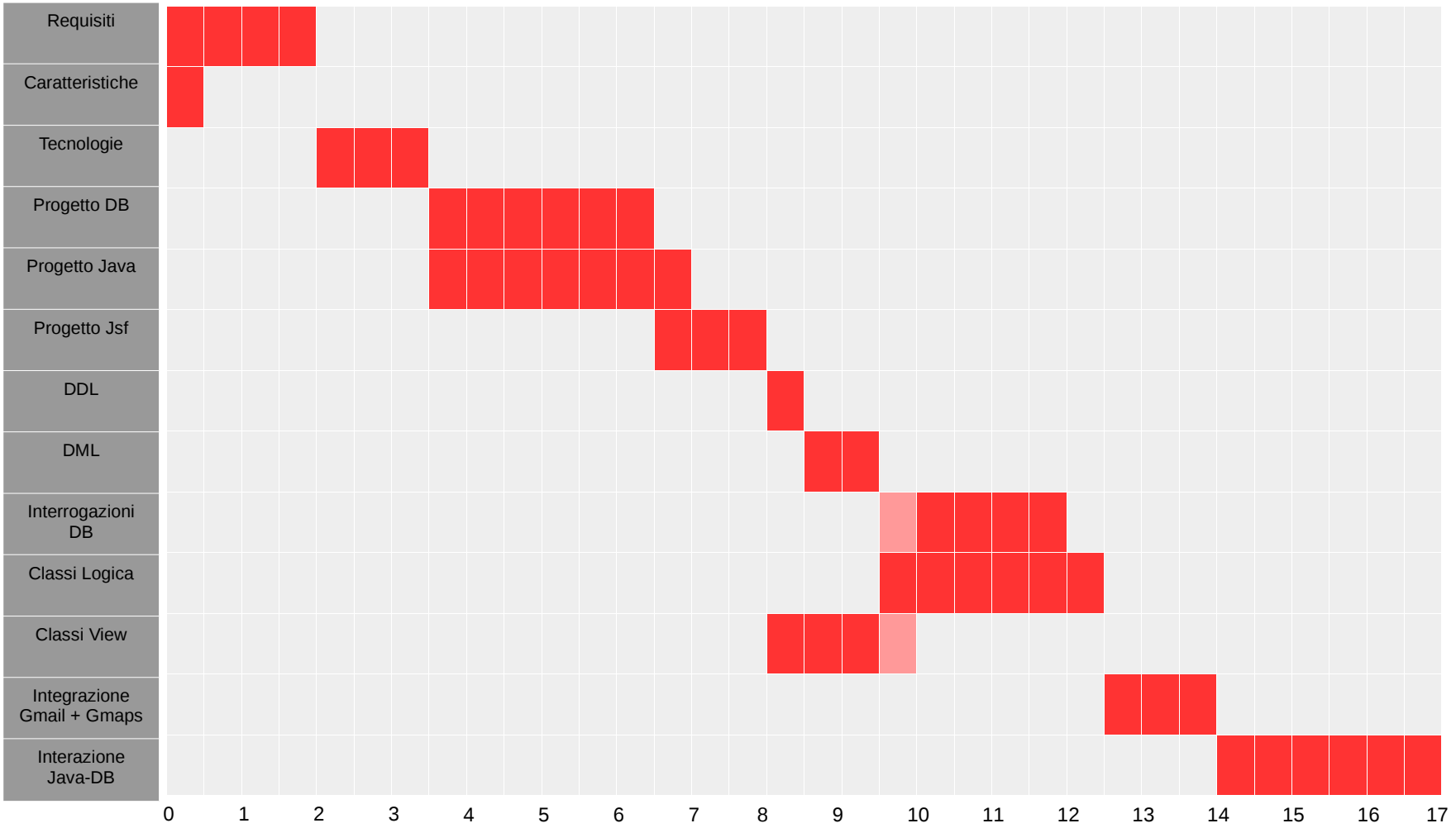


A → Abbiamo specificato i requisiti
B → Abbiamo specitifato le caratteristiche
C → Abbiamo specificato le tecnologie
D → Progetto del Database
E → DDL
F → DML
G → Progetto della logica di controllo in Java
H → Realizzazione delle classi per la logica
I → Integrazione Gmail, Dati Multimediali remoti
L → Progetto dell'interfaccia utente tramite JSF
M → Realizzazione classi + XML
N → Interrogazioni Database
O → Logica che interagisce con il Database

CPM (Critical Path Method)

Nome	Durata	ES	EF	LS	LF	TF	FF
inizio	0						
Requisiti	30	1	30	1	30	0	0
Caratteristiche	10	1	10	21	30	20	20
Tecnologie	20	31	50	31	50	0	0
Progetto Database	40	51	90	66	105	15	0
Progetto Java	50	51	100	51	100	0	0
Progetto JSF	20	51	70	166	185	115	0
DDL	10	91	100	106	115	15	0
DML	15	101	115	116	130	15	0
Interrogazioni DB	30	116	145	131	160	15	15
Classi Logica	40	101	140	101	140	0	0
Classi view	25	71	95	186	210	115	115
Integrazione Gmail, Dati Multimediali Remoti	20	141	160	151	160	0	0
Interazione Java-DB	50	161	210	161	210	0	0
Fine	0						

GANTT



L'unità di misura del Gantt sono i **Giorni**

Project Plan

COMPONENTI

- Postgresql portata da Postgresql Global Development Group Srl;
- Tecnologie JSF portate da Oracle;
- Server HTTP e FTP in Python;
- Gmail e Gmaps portato da Google.

ATTIVITÀ

Nel progetto verranno affrontate le seguenti attività:

- Analisi dei requisiti;
- Gestione tecnica del progetto;
- Realizzazione del Database con Postgresql;
- Realizzazione dell'interfaccia web con Java JSF;
- Integrazione del servizio Gmail e Gmaps.
- Integrazione della memorizzazione remota dei file multimediali;

Infrastruttura del Progetto

Per il Progetto Books Exchange è stata utilizzato:

→ Un **Workspace** in ambiente Eclipse per ogni programmatore per consentire un accesso privato ad una copia delle risorse di progetto, gestire sorgenti creare eseguibili ed eseguire dei Test;

→ Un **Repository** su Git-Hub per garantire il controllo delle versioni ed il lavoro collaborativo, precisamente:

- → Un Repository usato come ambiente di Test interno all'azienda, per Test Unitari e Test di Integrazione. Ambiente non visibile agli utenti; Link: https://github.com/ITicks/BooksExchange

- → Un Repository usato come ambiente di Collaudo per i test di Accettazione. Ambiente visibile all'utente; Link: https://github.com/ITicks/BooksExchangeCollaudo

- → Un Repository usato come ambiente di Produzione per il rilascio finale. Ambiente visibile agli utenti; Link: https://github.com/ITicks/BooksExchangeRelease

Gestione dei Rischi

Un Rischio è una variabile che può assumere dei valori tali da compromettere la riuscita del Progetto, stimabile come:

GRANDEZZA = IMPATTO X PROBABILITÀ

→ Analisi delle possibili aree di Rischio;
→ Stima di ogni Rischio;
→ Scelta della Strategia;
→ Applicazione;
→ Monitoraggio;

Risk List Books Exchange.pdf per maggiori informazioni.

In ordine di Grandezza
TOP 4 dei Rischi:
1- Problemi tecnico/funzionali al Database
2- Rilascio del progetto oltre la data di consegna
3- Servizio e-mail inutilizzabile
4- Down di rete

Design: cosa e come si è implementato il riuso?

- Design Pattern: Forniscono soluzioni efficaci a problemi ricorrenti;

- Framework: È definito da un insieme di classi astratte e dalle relazioni tra esse.
  Si differenziano dai Design Patterns per il fatto di essere astrazioni di livello più alto a livello architetturale.
  → JSF + MVC;

- Librerie:
  → JDBC;
  → Google Maps;
  → Apache Commons Mail;
  → Python Standard Libraries per i Servers in Python

- Generatori di codice: → Eclipse;

Diagramma Comportamentale: Diagramma dei casi d'uso

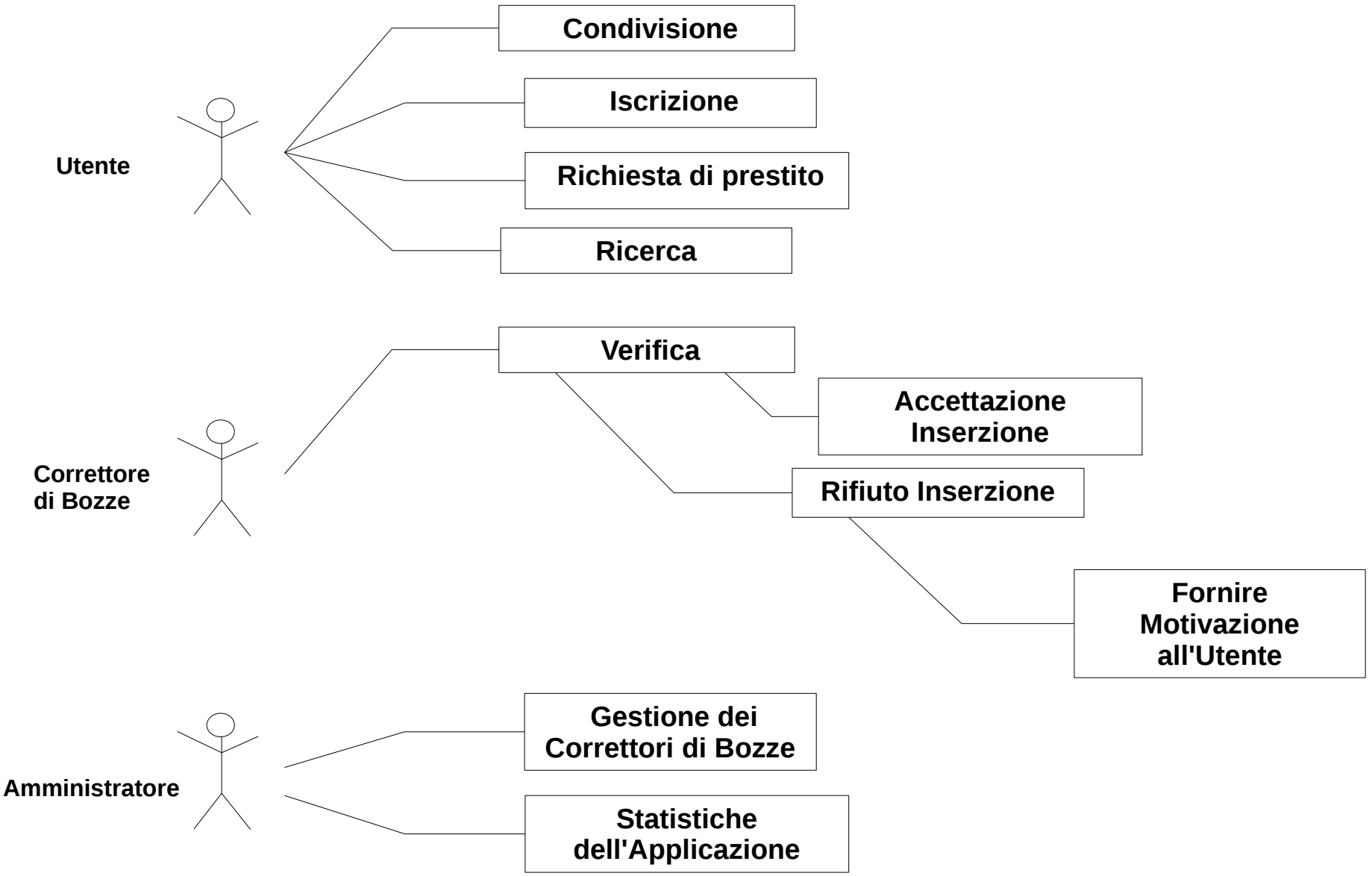
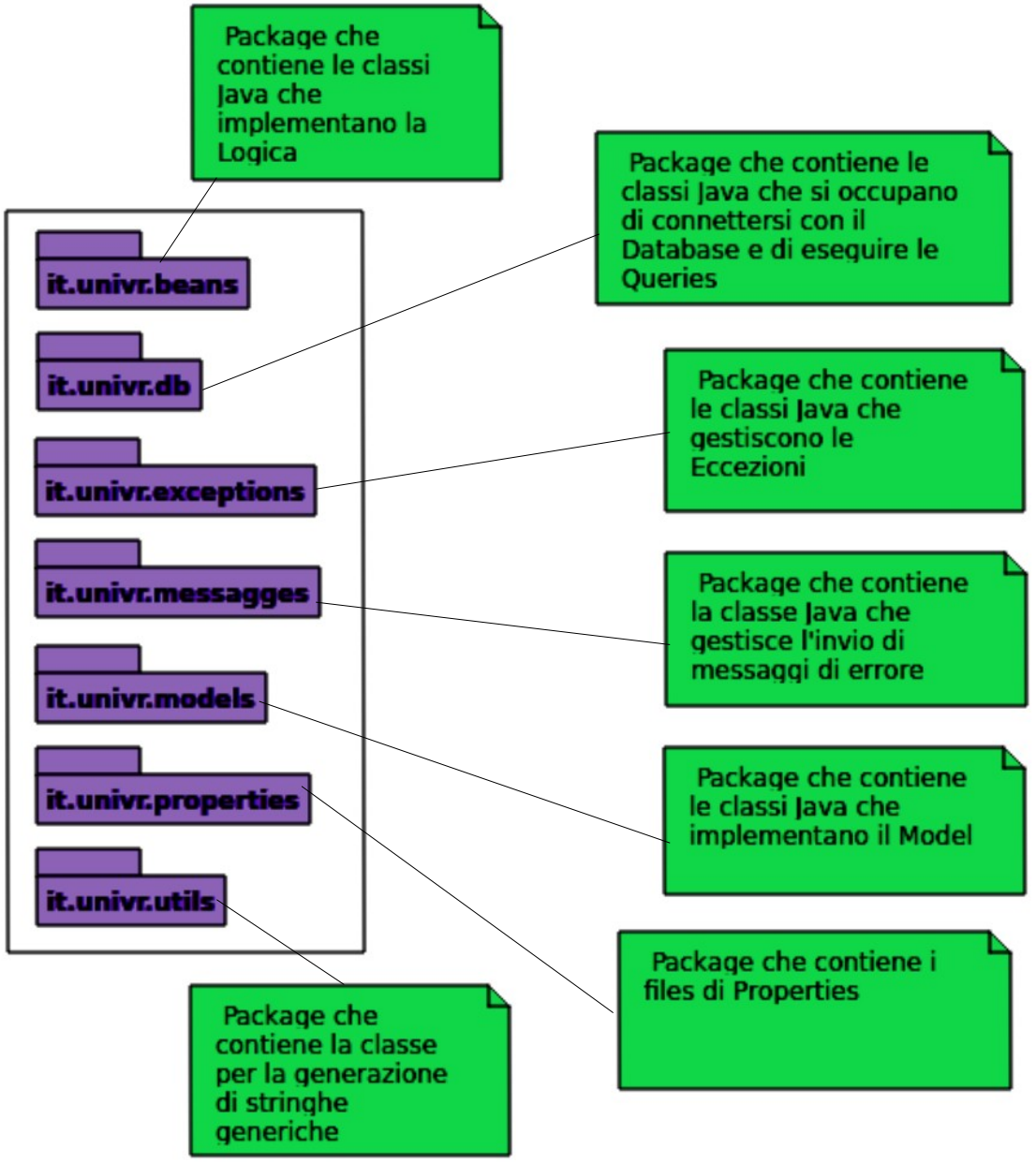


Diagramma Strutturale: Diagramma dei package



I Diagrammi sono stati generati con Umbrello

Diagramma Strutturale: Diagramma delle classi

→ file ClassDiagram_Command.png: Implementazione del Design Pattern Command per l'invio delle E-mail.

→ file ClassDiagram_Factory.png: Implementazione del Design Pattern Factory per la generazione di Eccezioni Custom.

→ file ClassDiagram_Singleton.png: Implementazione del Design Pattern Singleton per l'istanziazione del Database.

→ file ClassDiagram_Generalizzazione_AbstractUtente.png: Implementazione del Principio SOLID Open Close per la rappresentazione degli Attori del sistema.

Diagramma di Interazione: Diagramma delle sequenze
REGISTRAZIONE UTENTE

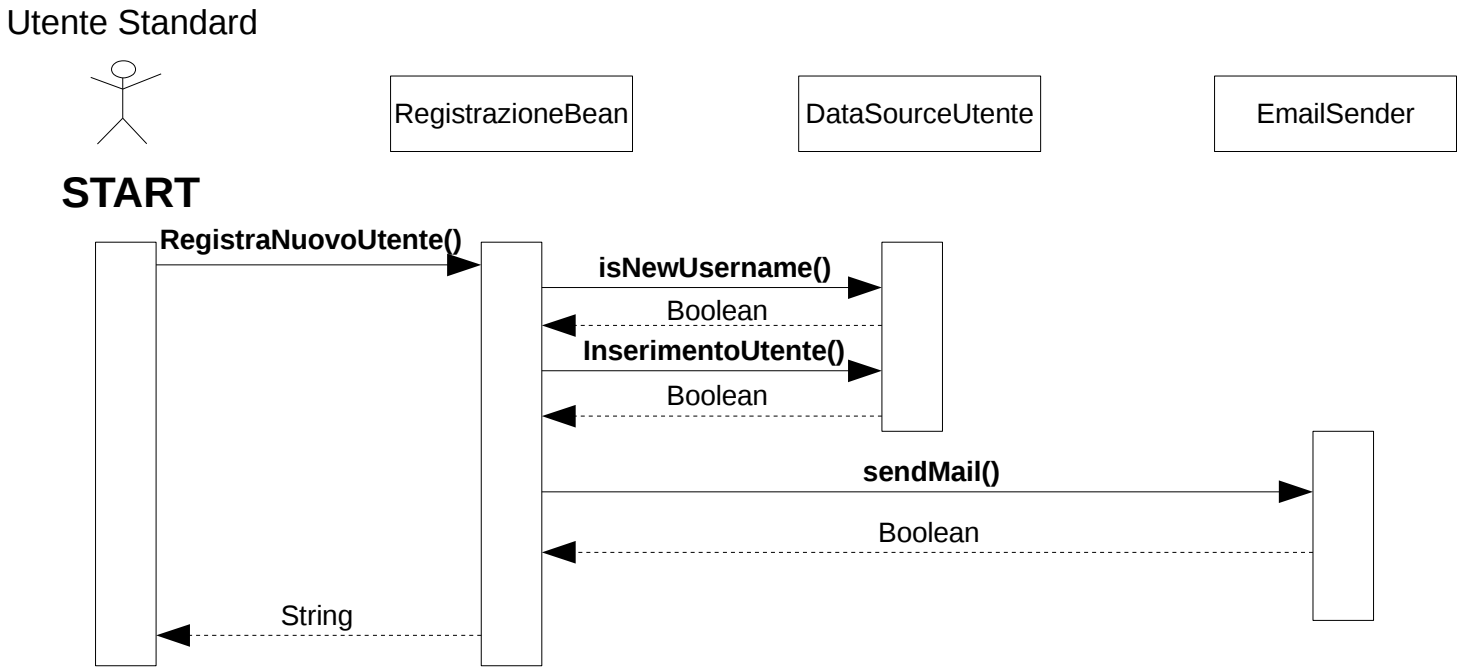


Diagramma di Interazione: Diagramma delle sequenze
INSERIMENTO INSERZIONE

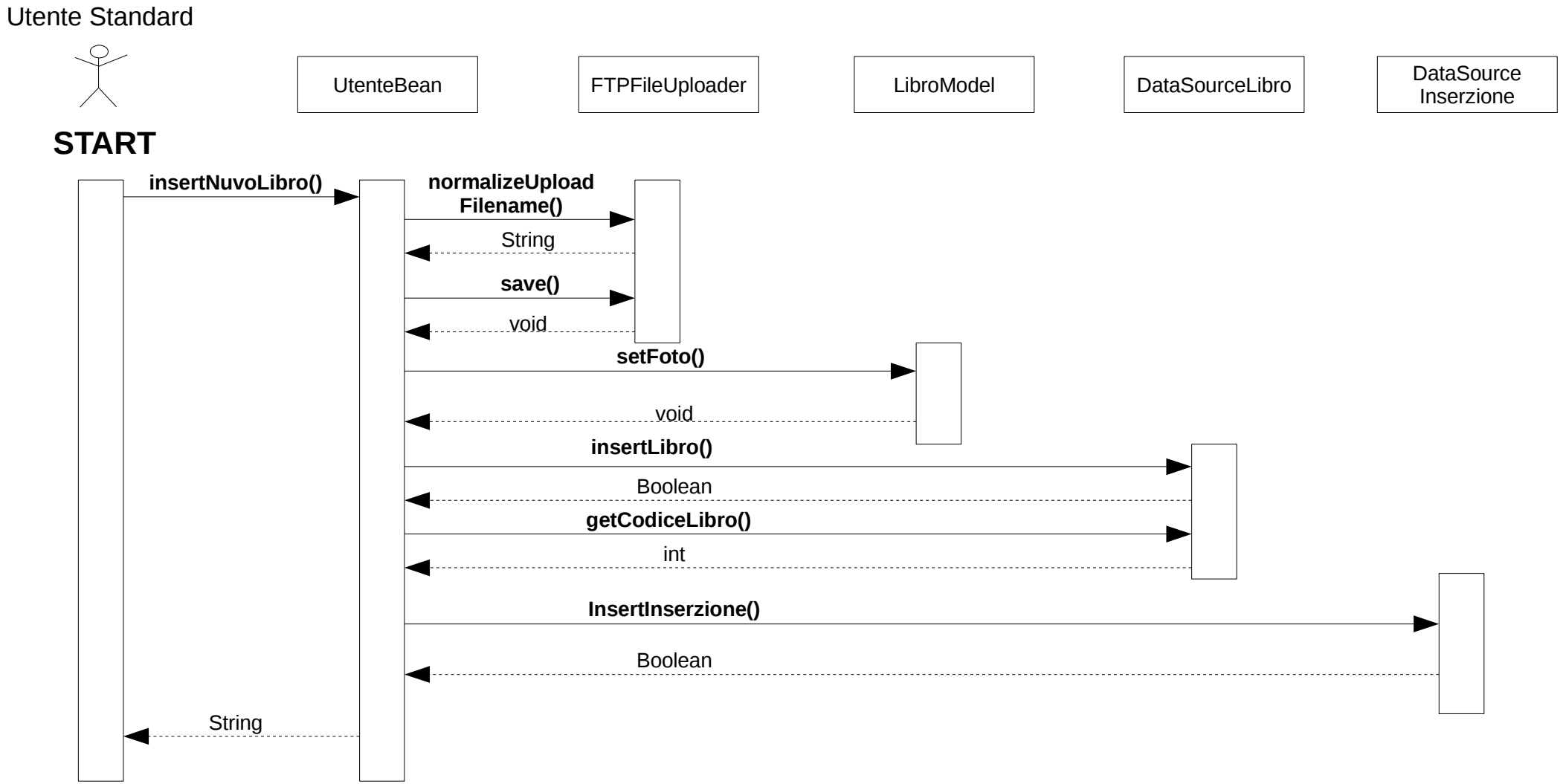


Diagramma di Interazione: Diagramma delle sequenze
RICERCA

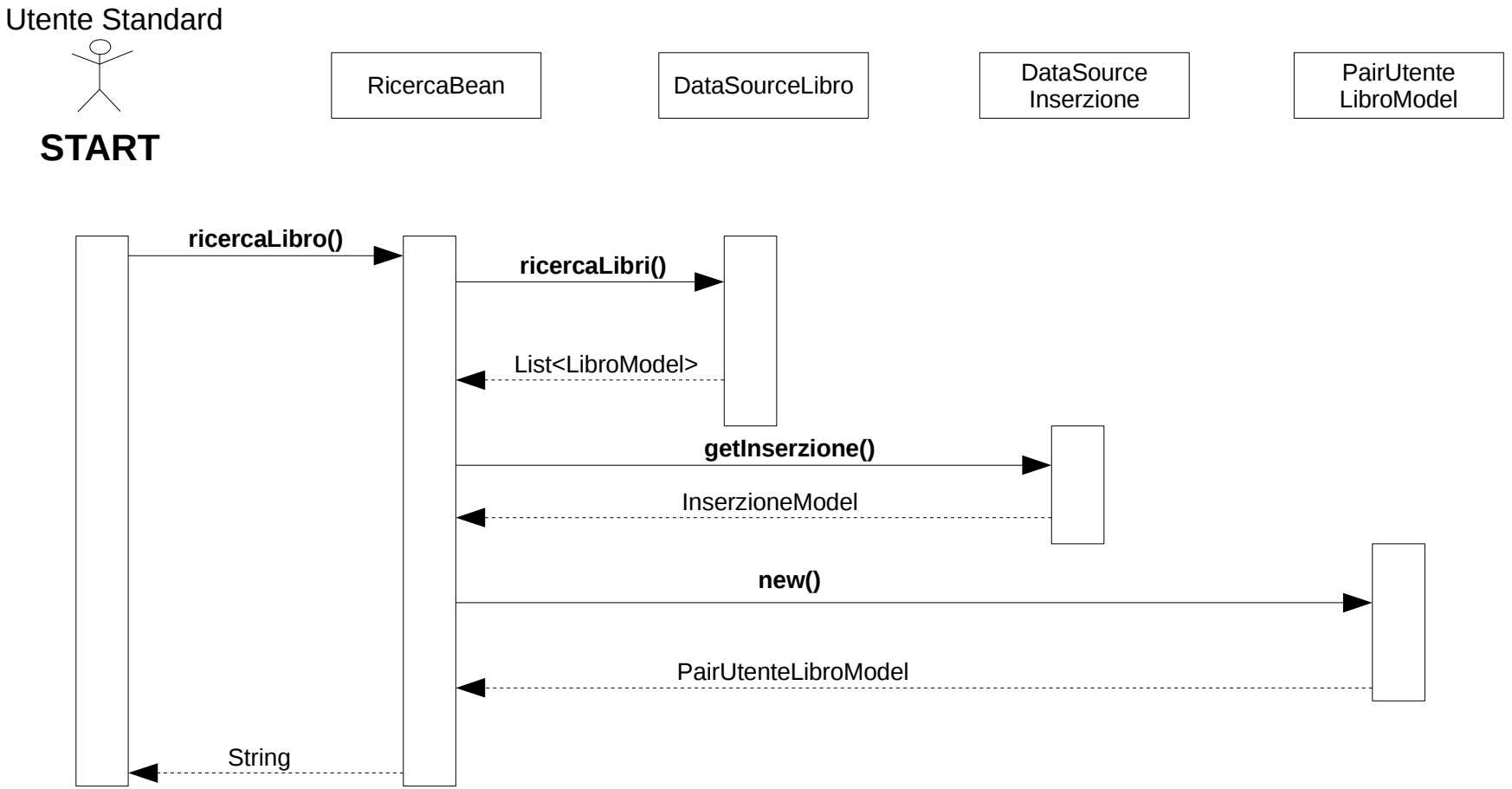


Diagramma di Interazione: Diagramma delle sequenze
VALIDA INSERZIONE

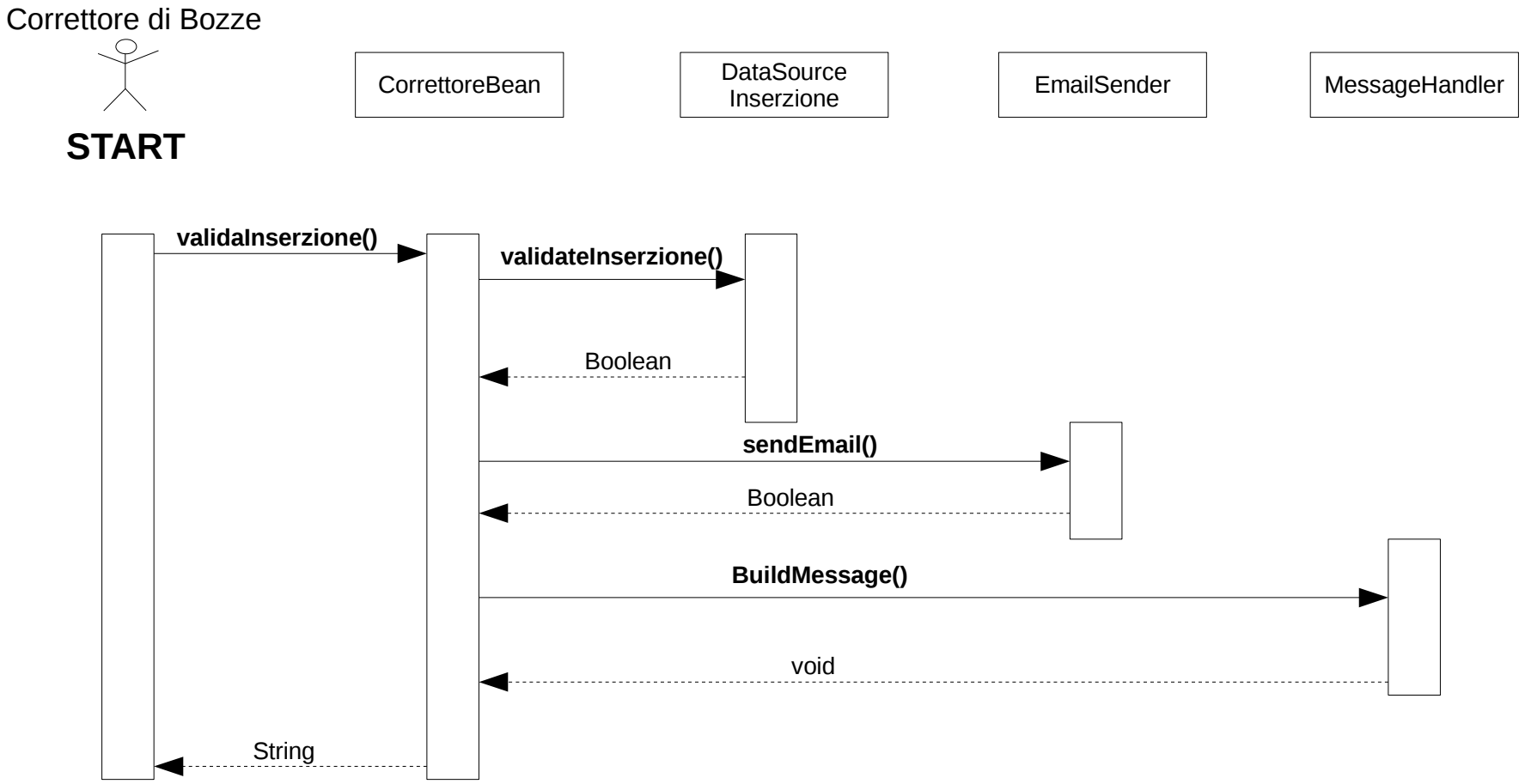
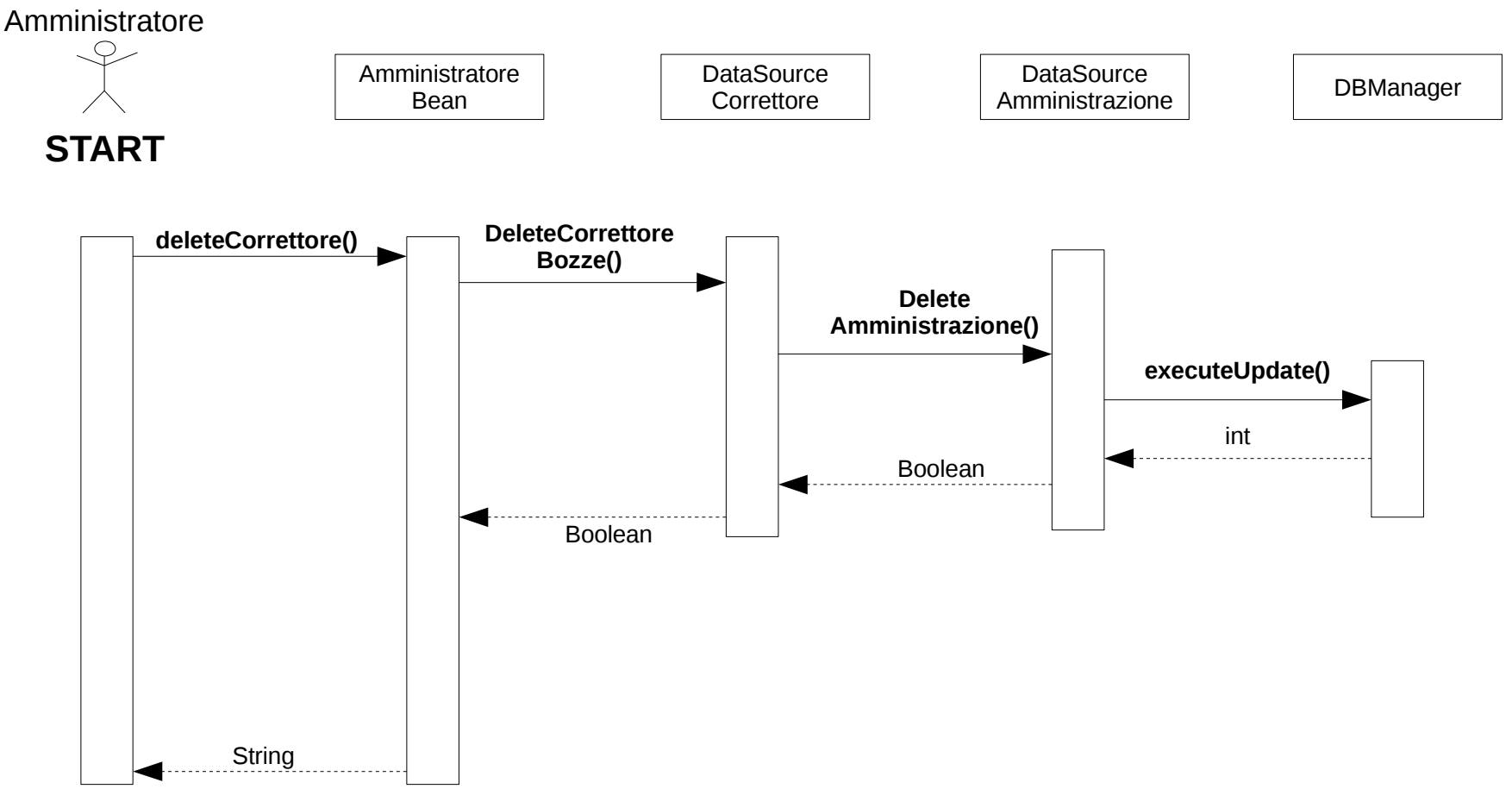
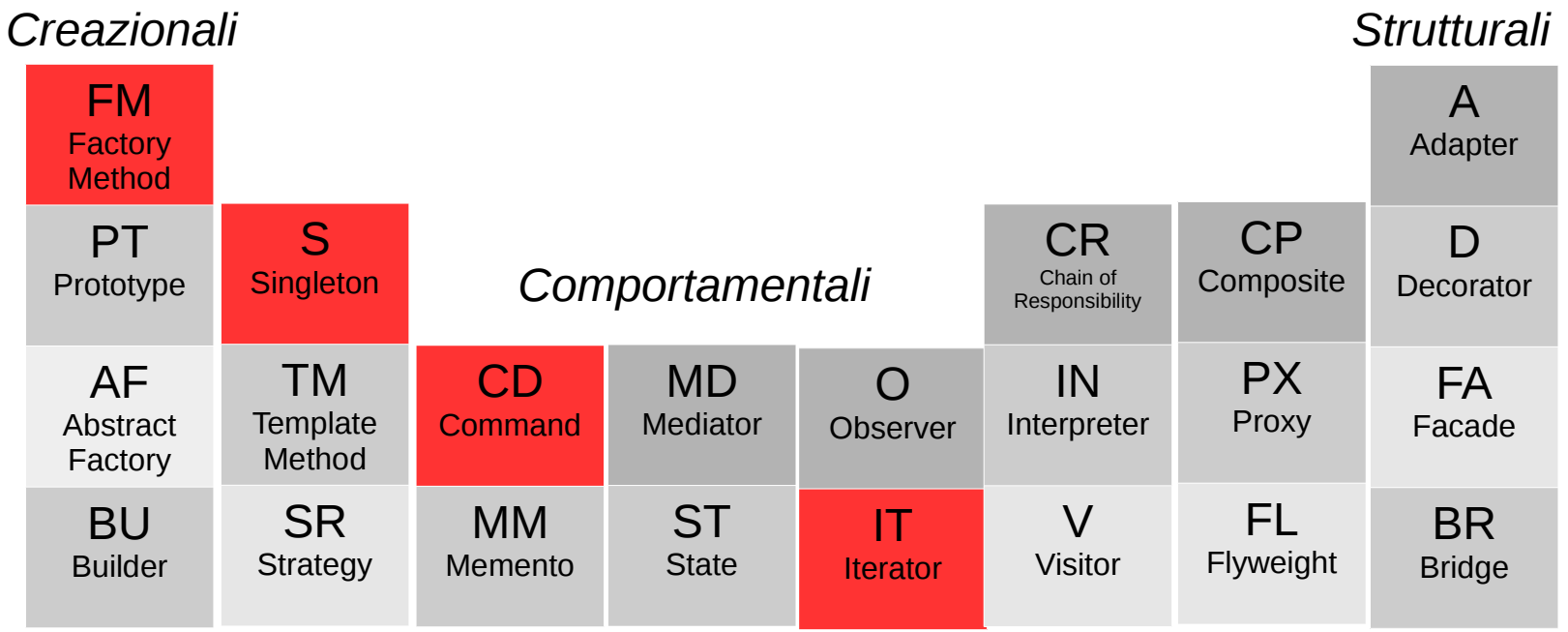


Diagramma di Interazione: Diagramma delle sequenze
ELIMINA CORRETTORE DI BOZZE



Design Pattern



Design Pattern: Quali sono stati usati e dove?

I Design Pattern utilizzati sono stati i seguenti:

Singleton → Creazione di una classe con costruttore privato, attributo statico privato se stessa. Per accedere alla Classe con i suoi attributi si utilizza il metodo getInstance()
→ Utilizzato per: DBManager
→ In: DBSingleton.java

Iterator → Utilizzato per: List, ResultSet, Hashset
→ In: DataSource, Bean

Command → Per utilizzare le operazioni come oggetti ed eliminare le dipendenze tra l'oggetto che invoca l'operazione e chi lo esegue
→ Utilizzato per: Upload FTP file, E-mail Sender
→ In: FTPFileUploader.java, EmailSender.java

Factory → Utilizzato per: Gestione errori
→ In: CustomExceptionHandlerFactory.java

Principi SOLID

Ci consentono di capire se siamo nei casi di Bad Design:

- → Software Rigidity: un singolo cambiamento coinvolge molte parti del sistema;
- → Software Fragility: un singolo cambiamento coinvolge parti inaspettate nel sistema;
- → Software Immobilty: difficoltà nel riuso;
- → Software Viscosity: facilità nel sbagliare;

I Principi SOLID utilizzati sono:

→ Single Responsibility → Ogni classe deve avere solo una responsabilità;
→ Open Close → Le classi sono aperte per estensione e chiuse per modifiche (Gerarchia di classi astratte nel progetto);
→ Principio di Liskov di Sostituzione → I sottotipi devono essere sostituibili ai loro supertipi senza alterarne le loro funzionalità semantiche;
→ Interface Segregation → Le classi non devono essere forzate a dipendere da Interfacce che non usano;

Test

Test	In carico a
Test Unitari	Sviluppatori
Test di Integrazione	Sviluppatori
Security Test	Sviluppatori
Stress Test	Sviluppatori
User Acceptance Test	Riccardo Pret

Data	Test
29.06.2015	Test Unitari
30.06.2015	Test di Integrazione
01.07.2015	Stress Test
02.07.2015	Security Test
04.07.2015	Altri System Test
05.07.2015	User Acceptance Test

Test

Test	Tipo	Descrizione	Esito
Test Unitari con JUnit	White Box	Test delle proprietà dei Model e Datasource	Passato
Test Unitari con JUnit	White Box	Test delle funzionalità dei Bean	Passato
Test di Integrazione	White Box	Test di Integrazione delle e-mail	Passato
Test di Integrazione	White Box	Test di Integrazione di GMaps	Passato
Test di Integrazione	White Box	Test di Integrazione dell'FTPUploader	Passato
Performance Test	Black Box	Test di carico	Non Passato
Stress Test	Black Box	Test di sovraccarico del Server Locale attraverso il collegamento di molti Client	Passato
Test di Sicurezza sui Form	Black Box	Sql Injection da parte di un tool specializzato su tutti i campi del form	Passato
Test di Robustezza	Black Box	Controllo se l'applicazione segnala errore inserendo dati non validi nei form	Passato
Test di Accettazione	Black Box	Prova da parte dell'utente dell'applicazione con relativa verifica di funzionalità e problemi	Passato

Lavoro collaborativo

La Pianificazione del Progetto è stata svolta da ambo i membri;

Il lavoro è stato suddiviso cercando di rispettare quanto stabilito in fase di Pianificazione, tranne su alcuni aspetti difficili dove è stato necessaria la collaborazione di entrambi i membri del team;

Il lavoro è stato svolto su due differenti computer ognuno dotato dell'ambiente di sviluppo Eclipse per avere accesso a una copia privata del Progetto;

È stata utilizzato Git-Hub come Repository remoto e come ambiente di Test interni all'azienda;