Università degli studi di Salerno

Biblionet

Main Modification Report

Data	02/05/2023
Destinatario	Prof. Andrea De Lucia, Dr. Stefano Lambiase
Presentato da	Giuseppe De Filippo, Cosimo Stabile
Approvato da	

Composizione Team

Nome	Matricola	Acronimo
Cosimo Stabile	0522500769	CS
Giuseppe De Filippo	0522500992	DF

Revision History

Data	Versione	Descrizione	Autori
05/05/23	1.0	Stesura Change Request	CS, DF

Descrizione del sistema corrente

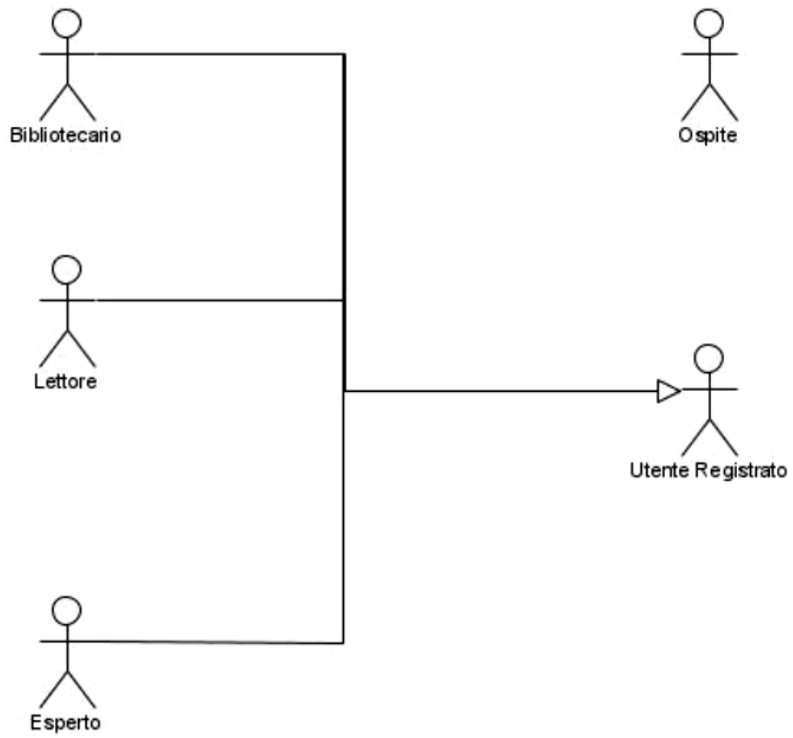
Biblionet semplifica l'accesso al mondo della lettura ai suoi utenti. Attraverso una piattaforma online, viene fornito al lettore la possibilità di cercare le biblioteche a egli più vicine e di visionare in modo semplice e veloce la lista dei libri disponibili nelle stesse.

Il sistema ha un duplice compito, ottimizzare il lavoro delle biblioteche che si interfacciano con i loro clienti lettori e, allo stesso tempo, aiutare i lettori meno esperti a coltivare l'hobby della lettura. In particolare, esso permette di:

- Conoscere le biblioteche della zona in modo facile e veloce
- Prenotare un libro a distanza
- Fornire un canale di contatto rapido con le biblioteche

Inoltre, il sistema offre la possibilità di gestione online di vari "club del libro", attraverso cui organizzare eventi, gestiti da esperti, con l'obiettivo di stimolare l'utenza a partecipare ad attività di lettura.

Attori del sistema



- **Ospite**: utente non registrato al sistema, può accedere solo ad operazioni di visualizzazione dei dati.
- **Esperto**: rappresenta un esperto in generi letterari impiegato in una biblioteca.
- **Lettore**: rappresenta un lettore.
- **Biblioteca**: rappresenta la biblioteca e si occupa della gestione della stessa all'interno dell sistema.
- **Utente registrato**: generalizzazione di Esperto, Lettore, Biblioteca.

Attori esterni

- **ISBN API**: API esterna che consente l'individuazione di informazioni sui libri tramite il loro ISBN.

Requisiti funzionali

Nella presente sezione saranno riassunti i requisiti funzionali del sistema Biblionet. I requisiti sono raggruppati nei seguenti gruppi:

1. Gestione Utente (GU)
2. Gestione Club del Libro (GCL)
3. Gestione Prestito Libri (GPL)
4. Gestione Preferenze di Lettura (GDPL)
5. Gestione Collaborazione Scolastica (GS) (questo gruppo non è stato più implementato).

RF_GU: Gestione utente

ID	Nome	Descrizione	Attori
RF_GU_1	Registrazione lettore	Il sistema permette ai lettori di registrarsi sulla piattaforma.	Ospite
RF_GU_2	Registrazione esperto	Il sistema permette ad un esperto di registrarsi alla piattaforma, specificando la biblioteca in cui lavora e alcune delle sue competenze.	Ospite
RF_GU_3	Registrazione account	Il sistema permette ad un bibliotecario di registrare la	Ospite

	biblioteca	propria biblioteca per organizzare dei Club del Libro, offrire prenotazioni, e organizzare collaborazioni con le scuole.
RF_GU_4	Login	Il sistema permette agli utenti di effettuare l'accesso.
RF_GU_5	Logout	Il sistema permette agli utenti loggati di disconnettersi dal sistema.
RF_GU_6	Visualizzazione area utente	Il sistema permette ad un utente loggato di visualizzare la propria area utente.
RF_GU_7	Modifica dati account	Il sistema permette agli utenti di modificare i propri dati personali attraverso la propria area utente.
RF_GU_8	Cancellazione account	Il sistema permette ad un utente che possiede un account di eliminarlo.

RF_GCL: Gestione Club del Libro

ID	Nome	Descrizione	Attori
RF_GCL_1	Creazione club del libro	Il sistema permette ad un esperto di registrare un club del libro per la biblioteca in cui lavora.	Esperto
RF_GCL_2	Modifica dati club del libro	Il sistema permette ad un esperto di modificare le informazioni di un club del libro	Esperto
RF_GCL_3	Eliminazione club del libro	Il sistema permette ad un esperto di eliminare un club del libro.	Esperto
RF_GCL_4	Visualizzazione membri club del libro	Il sistema permette ad un esperto di visualizzare i membri iscritti ad un club del libro.	Esperto
RF_GCL_5	Organizzazione	Il sistema permette ad un esperto di	Esperto

	evento	organizzare un evento relativo agli interessi del club del libro, indicando opzionalmente un libro e un genere di riferimento, inviando una notifica ai lettori iscritti.	
RF_GCL_6	Visualizzazione e Modifica evento	Il sistema permette ad un esperto di modificare i dettagli di un evento organizzato.	Esperto
RF_GCL_7	Eliminazione evento	Il sistema permette ad un esperto di eliminare un evento organizzato.	Esperto
RF_GCL_8	Comunicazione con i membri del club	Il sistema permette ai lettori ed agli esperti di creare canali di comunicazioni con altri membri di un club del libro	Esperto, Lettore
RF_GCL_9	Visualizzazione Clubs del libro	Il sistema permette ai lettori di cercare dei clubs in base ad alcuni filtri.	Lettore
RF_GCL_10	Iscrizione ad un club del libro	Il sistema permette ad un lettore di iscriversi ad un club del libro.	Lettore
RF_GCL_11	Abbandono club del libro	Il sistema permette ad un lettore di abbandonare un club del libro.	Lettore
RF_GCL_12	Iscrizione ad un evento	Il sistema permette ad un lettore appartenente ad un club del libro di iscriversi e partecipare ad un evento relativo al club.	Lettore
RF_GCL_13	Abbandono evento	Il sistema permette ad un lettore iscritto ad un evento di poter annullare la propria iscrizione all'evento.	Lettore

RF_GPL: Gestione Prestiti Libri

ID	Nome	Descrizione	Attori

RF_GPL_1	Inserimento nuovi libri	Il sistema consente a un bibliotecario di inserire nuovi libri all'interno della lista dei libri prenotabili presente sul sito.	Bibliotecario
RF_GPL_2	Rimozione libri prenotabili	Il sistema consente a un bibliotecario di poter eliminare uno o più libri dalla lista dei libri prenotabili presente sul sito.	Bibliotecario
RF_GPL_3	Visualizzazione libri prenotabili	Il sistema permette a un lettore di visualizzare tutta la lista dei libri prenotabili, in base a dei filtri come genere o autore, e successivamente le biblioteche nelle quali il libro è prenotabile.	Lettore
RF_GPL_4	Richiesta prestito libro	Il sistema permette a un lettore di aprire una richiesta di prenotazione di un libro da una biblioteca.	Lettore
RF_GPL_5	Visualizzazione ticket prestito	Il sistema permette al bibliotecario di visualizzare la lista dei ticket per i prestiti.	Bibliotecario
RF_GPL_6	Accettazione ticket prestito	Il sistema permette al bibliotecario di accettare un ticket per un prestito libro.	Bibliotecario
RF_GPL_7	Restituzione prestito libro	Il sistema permette il reinserimento di un libro nella lista dei libri prenotabili, dopo la restituzione in biblioteca.	Bibliotecario
RF_GPL_8	Visualizzazione libri in prestito	Il sistema permette ad un gestore della biblioteca di visualizzare i libri attualmente in prestito a degli utenti.	Bibliotecario
RF_GPL_9	Visualizzazione richiesta libri in prestito	Il sistema permette ad un lettore di visualizzare la lista dei libri presi in prestito dalle biblioteche, con il relativo tempo mancante alla restituzione per non incorrere in mora.	Lettore

RF_GPDL: Gestione Preferenze di lettura

ID	Nome	Descrizione	Attori
RF_GPDL_1	Inserimento conoscenze	Il sistema permette ad un Esperto registrato di inserire i generi di lettura in cui spicca.	Esperto
RF_GPDL_2	Inserimento generi letterari preferiti	Il sistema permette ad un utente lettore di inserire i propri generi preferiti.	Lettore
RF_GPDL_3	Aggiornamento generi letterari	Il sistema permette a un lettore oppure ad un esperto di aggiornare i suoi interessi di lettura inserendone dei nuovi.	Lettore, Esperto
RF_GPDL_4	Sottomissione questionario di supporto	Il sistema permette a un lettore di partecipare ad un quiz che lo aiuterà nella scelta delle abitudini di lettura.	Lettore
RF_GPDL_5	Visualizza lista Esperti ordinati in base a preferenze di lettura	Il sistema mostra a un lettore una lista di esperti competenti su una certa tipologia di libri interessanti per il lettore.	Lettore
RF_GPDL_6	Comunicazione con Esperto	Il sistema permette la comunicazione fra un lettore ed un esperto che non appartengono al medesimo club del libro ma che hanno delle preferenze di lettura simili.	Esperto, Lettore

Nota sui requisiti funzionali

Dalla documentazione originaria del progetto Biblionet sono stati raccolti requisiti funzionali relativi e individuati attori relativi alla “Gestione della collaborazione scolastica”.

In precedenza era stato proposto di aggiungere a BiblioNet delle funzionalità che ne permettessero l'adozione anche in contesti scolastici. Tale obiettivo di business è stato abbandonato. Le informazioni riportate nel presente documento sono finalizzate a rappresentare lo stato attuale del sistema, per cui si è deciso di non riportare informazioni relative alle funzionalità non attualmente presenti nel sistema.

Design

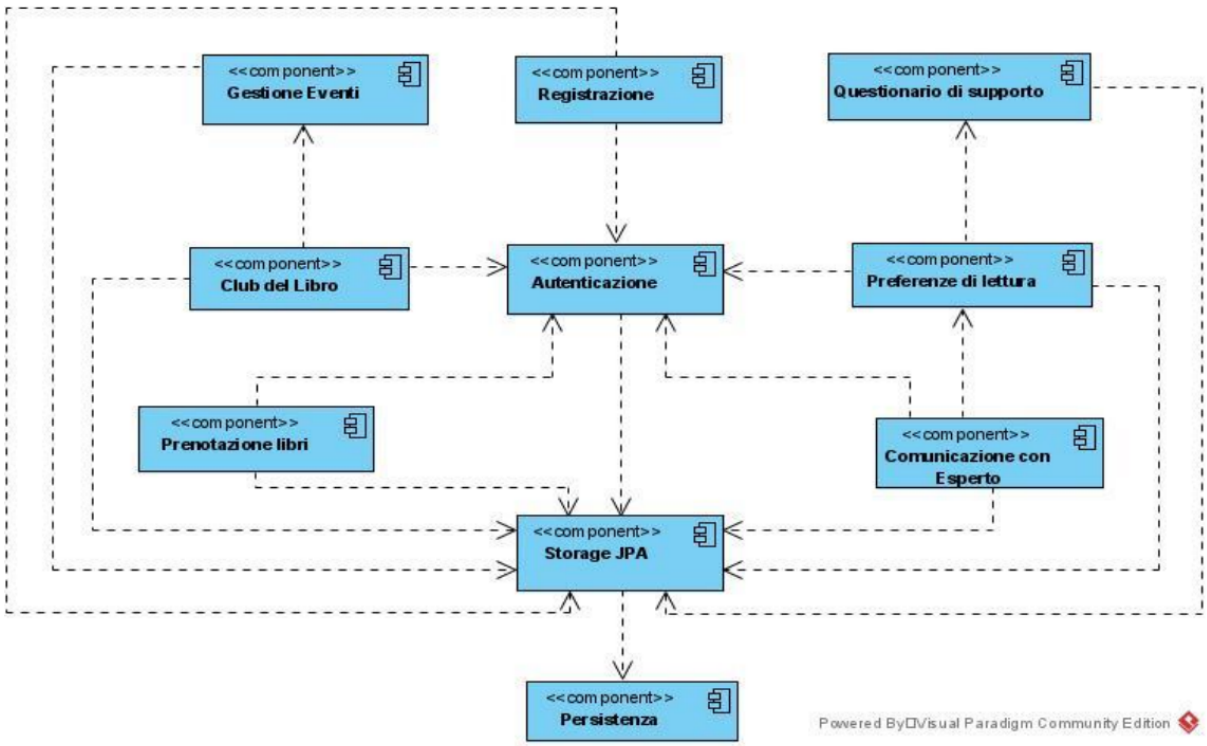
Il sistema proposto è basato sullo stile architetturale Three Tier, implementato utilizzando Spring MVC.

Decomposizione in sottosistemi

I sottosistemi individuati sono:

- **Registrazione**: si occupa di gestire la registrazione dei vari tipi di utente: biblioteca, lettore, esperto e scuola.
- **Autenticazione**: è responsabile delle funzionalità di Login, Logout, visualizzazione area utente e la modifica dati account.
- **Club del Libro**: è responsabile della visualizzazione dei vari club a cui iscriversi, della gestione della partecipazione a questi club e della creazione di un club da parte di un Esperto.
- **Gestione eventi**: si occupa delle funzioni riguardanti la creazione da parte degli Esperti, la visualizzazione e la partecipazione agli eventi per i membri del club del libro.
- **Prenotazione libri**: si occupa delle funzioni riguardanti la visualizzazione dei libri prenotabili, la loro prenotazione e il controllo dei prestiti in corso.
- **Preferenze di Lettura**: si occupa dell'inserimento e della gestione delle preferenze di lettura di un Lettore e delle conoscenze di un Esperto.
- **Questionario di supporto**: si occupa di gestire il questionario di supporto, utile per aiutare nuovi utenti nella scelta di un genere letterario con il quale cominciare nel mondo della lettura.
- **Comunicazione con esperto**: si occupa della gestione di canali di comunicazione per mettere in contatto un lettore con un esperto.
- **Persistenza**: si occupa di gestire la persistenza dei dati con un database.
- **Storage JPA**: si interpone tra i vari sottosistemi e il sottosistema di Persistenza.

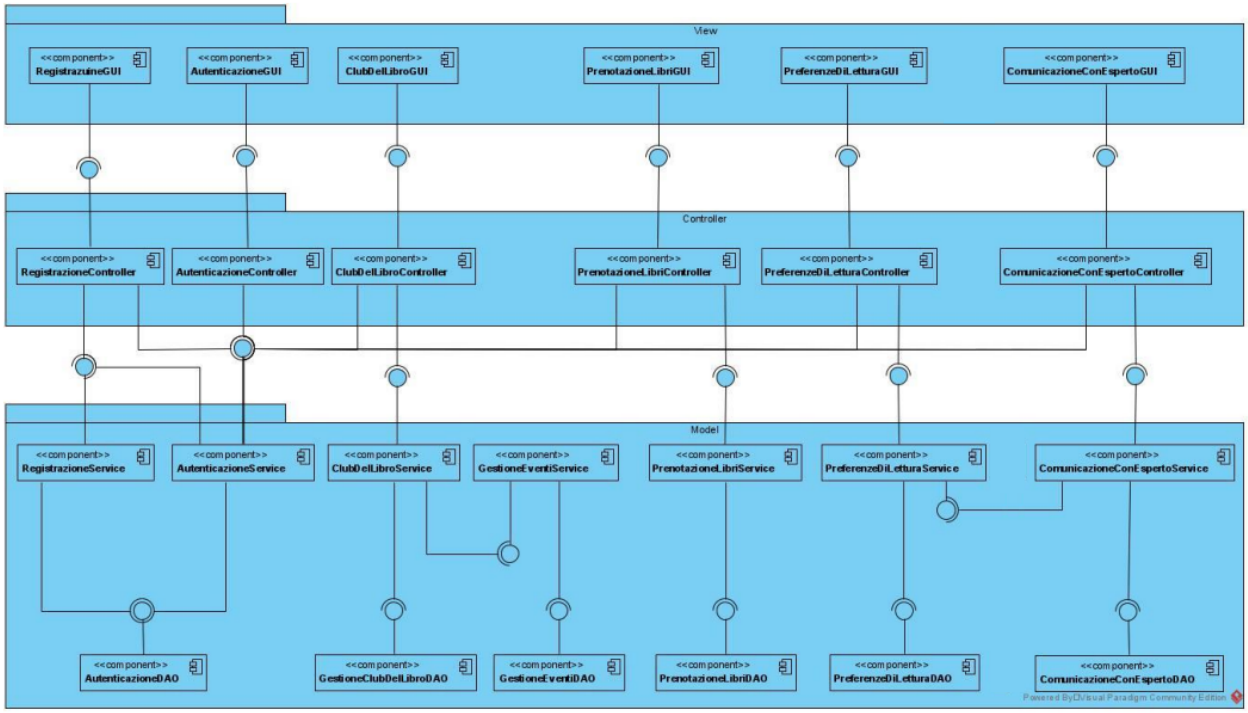
Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un component diagram UML.



Alcuni sottosistemi sono gestiti da componenti COTS (Commercial off the shelf), di seguito un elenco:

- Storage JPA verrà gestito da Spring Data JPA
- Persistenza sarà gestita attraverso un DBMS relazionale

Diagramma Architetturale

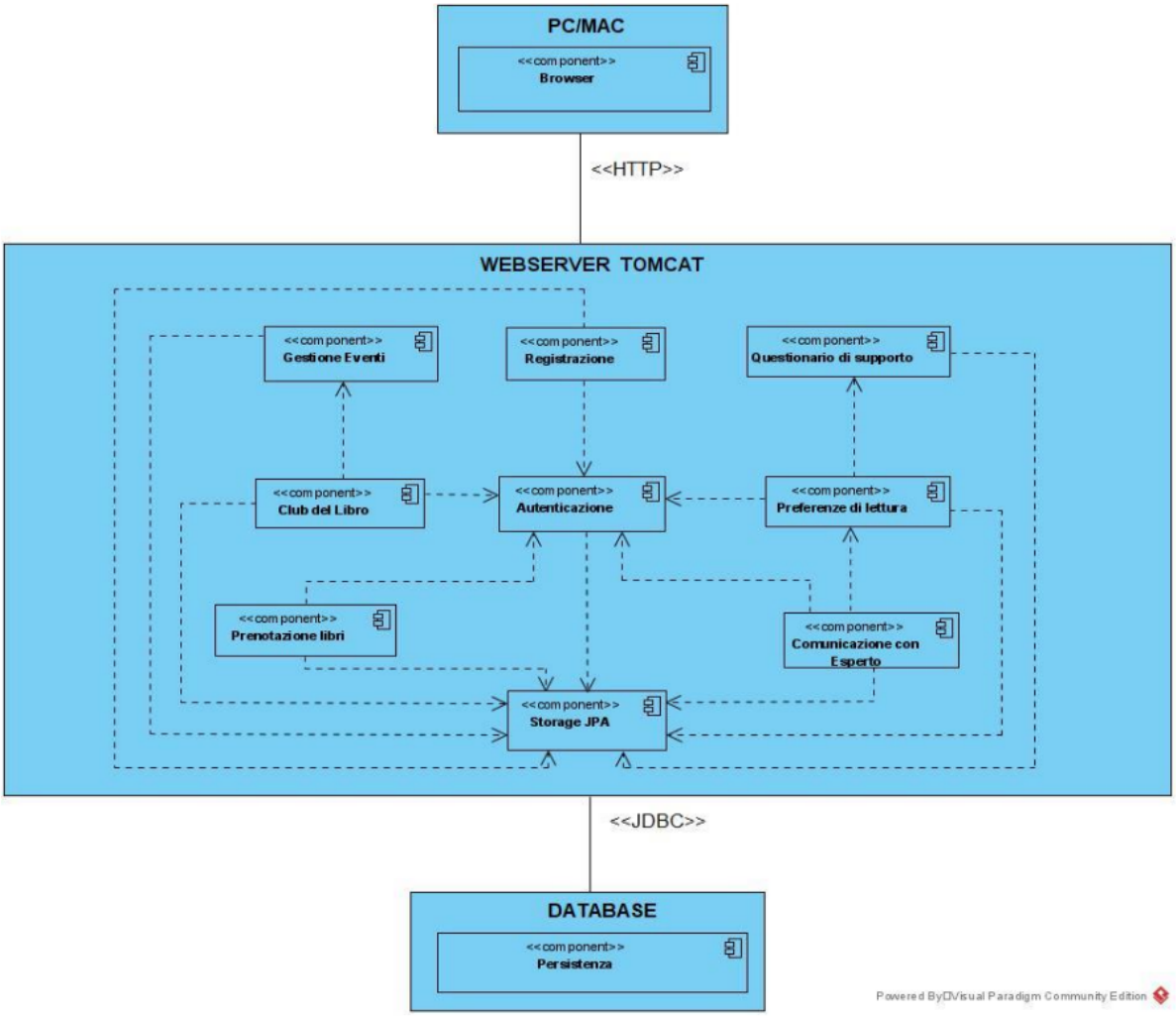


Mapping hardware/software

L' applicazione web si basa su una piattaforma hardware costituita da un server che risponde alle richieste effettuate dai clienti da una qualsiasi macchina con un browser ed una connessione ad Internet.

Biblionet è una web application che risiede su un web server, e che si basa su un'architettura non distribuita: risiede su un solo nodo.

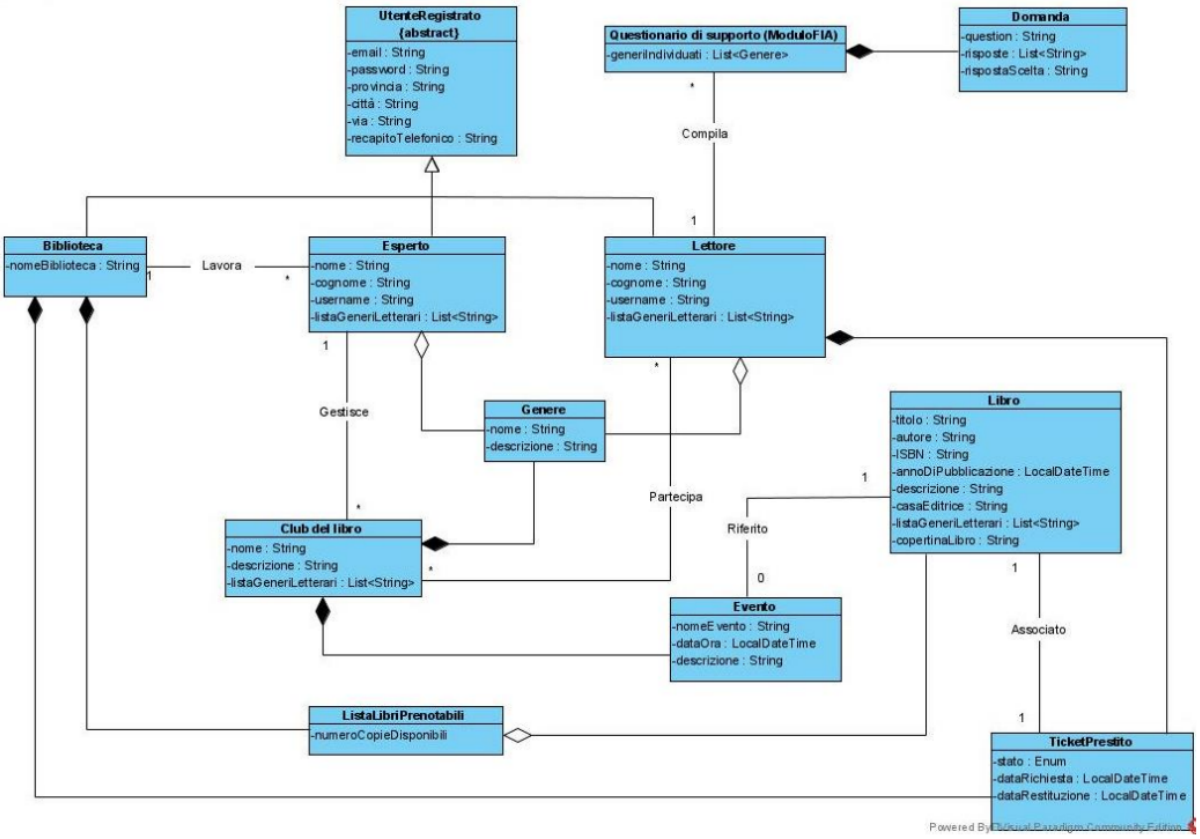
Di seguito un UML deployment diagram che descrive il mapping hardware/software.



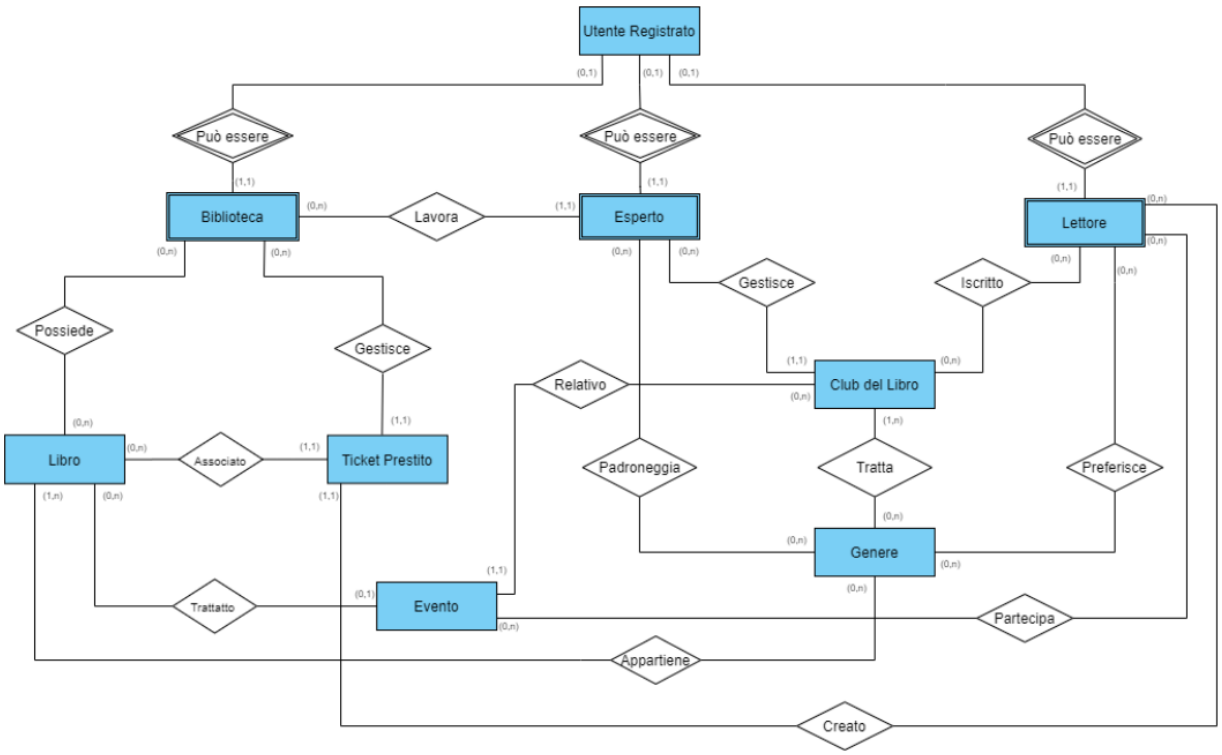
Gestione dei dati persistenti

Per la gestione del salvataggio dei dati persistenti del sistema Biblionet utilizza un database relazionale, così da gestire agevolmente l'accesso concorrente ai dati e contemporaneamente garantire la consistenza dei dati tramite l'utilizzo di un DBMS.

Di seguito è riportato l'Entity Class Diagram di Biblionet:

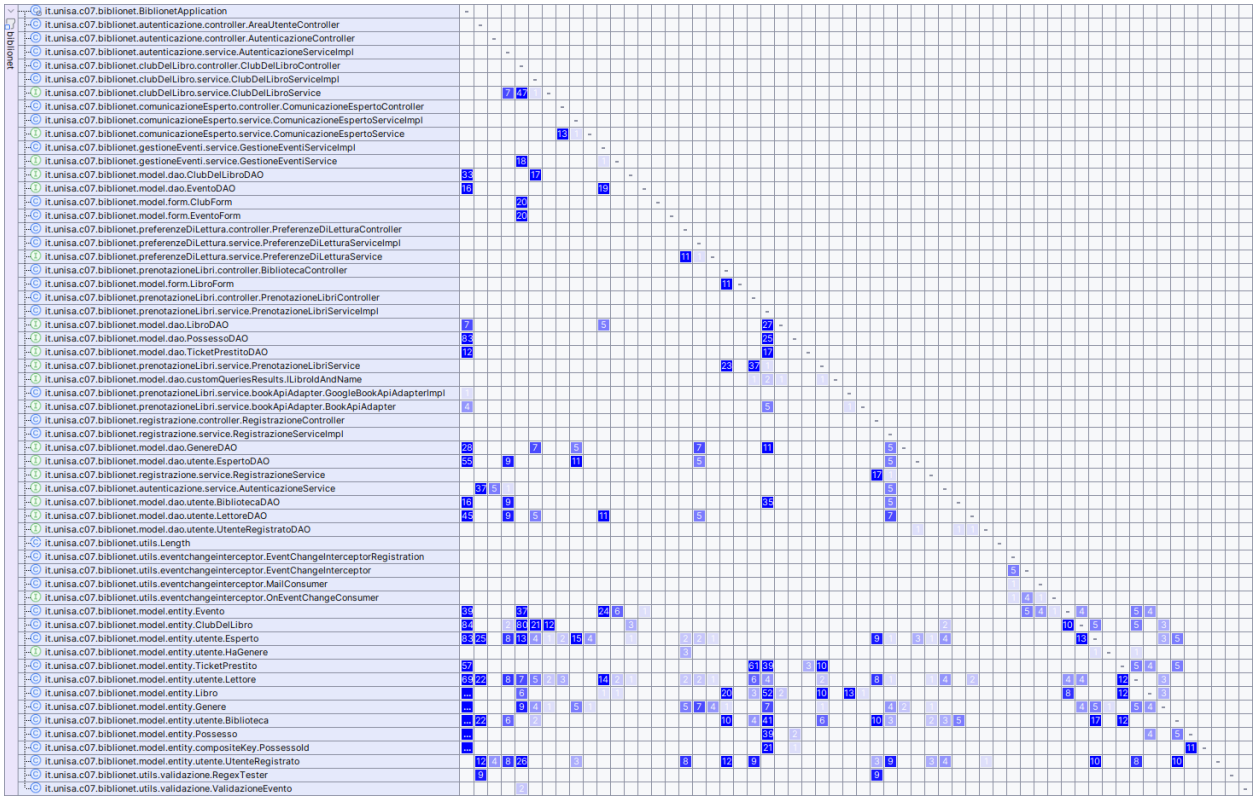


È altresì riportato lo schema ER del database di Biblionet:


Utente Registrato
Può essere
Può essere
Può essere
Biblioteca
Lavora
Esperto
Lettore
Possiede
Gestisce
Gestisce
Iscritto
Relativo
Club del Libro
Libro
Associato
Ticket Prestito
Padroneggia
Tratta
Preferisce
Genere
Trattato
Evento
Partecipa
Appartiene
Creato

Matrice delle dipendenze

Di seguito è riportata la matrice delle dipendenze del sistema biblionet, limitata alla sola cartella /src (quindi escludendo il codice di test).



Modifiche Proposte

Di seguito sono riportate, in ordine di priorità decrescente, le Change Request per l'applicazione in questione.

CR01: Migrazione a microservizi

ID	Data Richiesta	Richiesta da
CR01	23/3/2023	CS, DF

Descrizione
Ristrutturazione secondo un'architettura a microservizi del sistema.
Scopo
Migliorare l'architettura del sistema corrente in modo da migliorare manutenibilità, scalabilità, flessibilità e disponibilità.
Priorità
Alta[X] Media[ ] Bassa[ ]
Conseguenze, se non accetto
Il sistema permane nella situazione corrente.

CR02: Integrazione post

ID	Data Richiesta	Richiesta da
CR02	23/3/23	CS, DF

Descrizione
Integrare una bacheca contenente post nella sezione “Club del Libro” dell’applicazione.
Scopo
Permettere agli utenti di scambiarsi informazioni riguardanti il club del libro e consentire agli stessi di commentare i libri del mese del club del libro.
Priorità
Alta[X] Media[ ] Bassa[ ]
Conseguenze, se non accetto
Il sistema non fornirebbe funzionalità atte a ospitare le discussioni tra gli utenti dello stesso. Tali conversazioni verrebbero condotte su altre piattaforme (es: di messaggistica istantanea) oppure dal vivo. È difficile

mantenere una traccia di informazioni scambiate in questo modo e quindi rischiano di essere disperse/non adeguatamente conservate.

CR03: Ristrutturazione interfaccia grafica

ID	Data Richiesta	Richiesta da
CR03	23/3/23	CS, DF

Descrizione
Ristrutturazione interfaccia grafica in modo che si adatti al sistema a microservizi
Scopo
Dato il cambio architetturale descritto dalla CR 01, si impone il remake dell'interfaccia grafica affinché si integri con un sistema a microservizi.
Priorità
Alta[X] Media[ ] Bassa[]
Conseguenze, se non accetto

L’applicazione risulterebbe priva di un’interfaccia grafica funzionante col sistema a microservizi ottenuto in output alla CR 01.

Rimarrebbero, inoltre, i problemi di usabilità succitati.

CR04: Miglioramento qualità codice

ID	Data Richiesta	Richiesta da
CR04	23/3/23	CS, II

Descrizione
Refactoring del codice con tool di code smell
Scopo
Fare attività di clean code per rendere il codice più leggibile e prevenire bug.
Priorità
Alta[ ] Media[X] Bassa[ ]
Conseguenze, se non accetto

Mantenendo la situazione attuale, si aumenta la probabilità di riscontrare bug nel tempo, soprattutto considerando le modifiche specificate nelle CR precedenti.

Impact Analysis e Studio di fattibilità

Abbiamo condotto un'analisi sugli effetti di ciascuna modifica poiché era necessario valutare se le tali modifiche potessero avere un impatto negativo sul resto del sistema. Sono stati identificati i moduli impattati da ciascuna modifica. Tale analisi è stata condotta utilizzando la matrice delle dipendenze (ottenuta attraverso il tool DSM di IntelliJ) e del codice sorgente.

Impact analysis CR01, CR03: Migrazione a microservizi e ristrutturazione interfaccia grafica

Con l'implementazione contestuale delle change request CR01 e CR03 si mira a ottenere un sistema a microservizi in cui il front-end è un servizio a sé stante: la comunicazione tra i servizi avviene tramite una API RESTful.

Il sistema di partenza, come descritto nelle precedenti sezioni, è basato su un'architettura three-tier. Di seguito è riportata una lista delle attività che si è deciso di eseguire per raggiungere gli obiettivi fissati dalle CR:

1. Rimozione del Presentation layer dal sistema three-tier di partenza
2. Modifica dell'interfaccia tra Presentation Layer e Logic Layer (controller) al fine di ottenere controller RESTful
3. Decomposizione del sistema in moduli (al fine di ottenere un monolite modulare) effettuata in base all'organizzazione delle funzionalità espressa a livello dei requisiti funzionali. I moduli ottenuti a valle di questa fase saranno i seguenti:
    - Gestione Club del Libro + Gestione Preferenze di Lettura
    - Gestione Prestiti Libro
    - Gestione Autenticazione

    A questo stadio vanno assegnate le responsabilità sui dati ai singoli moduli.
4. Realizzazione sistema di autenticazione e autorizzazione basato su JWT
5. Suddivisione dei moduli in servizi indipendenti
6. Sviluppo del "servizio" front-end

Lo sviluppo del “servizio” front-end, che verrà implementato con l’ausilio del framework ReactJS, può essere eseguito in parallelo alle attività elencate in precedenza.

Il SIS per le modifiche alla CR consta dell’intero contenuto di:

- /src/resources/templates
- /src/resources/static/

Inoltre il SIS è composto anche da i file:

- AreaUtenteController.java
- AutenticazioneController.java
- ClubDelLibroController.java
- ComunicazioneEspertoController.java
- PreferenzeDiLetturaController.java
- BibliotecaController.java
- PrenotazioneLibriController.java
- RegistrazioneController.java
- Libro.java
- LibroDAO.java
- Esperto.java
- Biblioteca.java
- Lettore.java
- LibroForm.java
- EventoForm.java
- ClubForm.java
- AutenticazioneServiceImpl.java
- RegistrazioneServiceImpl.java
- RegexTester.java
- ValidazioneEvento.java

A partire dal SIS, utilizzando la matrice delle dipendenze in figura, è stato determinato il seguente CIS:

- AreaUtenteController.java
- AutenticazioneController.java
- ClubDelLibroController.java
- ComunicazioneEspertoController.java

- PreferenzeDiLetturaController.java
- BibliotecaController.java
- PrenotazioneLibriController.java
- RegistrazioneController.java
- Libro.java
- LibroDAO.java
- Esperto.java
- Biblioteca.java
- Lettore.java
- Genere.java
- LibroForm.java
- EventoForm.java
- ClubForm.java
- AutenticazioneServiceImpl.java
- RegistrazioneServiceImpl.java
- RegexTester.java
- ValidazioneEvento.java
- GestioneEventiService.java
- GestioneEventiServiceImpl.java
- PrenotazioneLIbriServiceImpl.java
- PrenotazioneLibriService.java
- GoogleBookAPIAdapterImpl.java
- BookAPIAdapter.java
- Evento.java
- TicketPrestito.java
- ClubDelLibroServiceImpl.java
- ClubDelLibroService.java
- ClubDelLibroDAO.java
- ComunicazioneEspertoService.java
- ComunicazioneEspertoServiceImpl.java
- EspertoDAO.java
- PreferenzeDiLetturaService.java
- PreferenzeDiLetturaServiceImpl.java
- RegistrazioneService.java
- RegistrazioneServiceImpl.java

- AutenticazioneService.java
- BibliotecaDAO.java
- ClubDelLibroDAO.java
- LettoreDAO.java
- GenereDAO.java
- HaGenere.java

Studio di fattibilità

La rimozione del Presentation layer dal sistema three-tier di partenza e la modifica dell'interfaccia tra Presentation Layer e Logic Layer (controller) al fine di ottenere controller RESTful non presentano particolari problemi di fattibilità

Monolite modulare

Dato che l'applicazione di partenza è stata sviluppata mediante Spring, nella migrazione del monolite verso un'architettura di tipo monolite modulare si è deciso di ricorrere allo strumento Spring Modulith, messo a disposizione dalla stessa Spring.

Spring Modulith è un insieme di librerie di supporto nello sviluppo di applicazioni Spring Boot Modulari; esso consente, tra le altre cose, di verificare in maniera automatica se il sistema rispetta alcune delle caratteristiche di un monolite modulare.

Un modulo di un modulith ben formato costituisce un package a sé stante composto di due parti:

- Una parte esterna, che contiene elementi per i quali è consentita l'interazione con elementi all'esterno del package (Service, DTO)
- Una parte interna, che incapsula elementi che non godono di tale proprietà (Repository, Entità)

Per quanto concerne, invece, le responsabilità sui dati sono state prese le seguenti decisioni.

La responsabilità dell'entità "Libro" è contesa tra i moduli Biblioteca e ClubDelLibro. L'entità "Libro", sarà provvisoriamente inserita quindi in un package pubblico a sé stante, in modo da poter essere acceduta dai moduli Biblioteca e ClubDelLibro.

Anche l'entità Genere è contesa tra più moduli: in questo caso si è deciso di intervenire sulle singole relazioni che coinvolgono l'entità Genere, sostituendo tale entità con la sola chiave (il nome del Genere). Si è deciso di attribuire la responsabilità dell'Entità Genere al modulo GestioneClubDelLibro+GestionePreferenzeDiLettura: informazioni aggiuntive riguardanti un Genere possono essere ottenute interagendo con tale modulo.

- Il modulo GestioneBiblioteca avrà la responsabilità delle entità Biblioteca, Possesso, PossessoId, TicketPrestito
- Il modulo GestioneClubDelLibro+GestionePreferenzeDiLettura avrà la responsabilità delle entità ClubdelLibro, Esperto, Lettore, Evento e Genere
- GestioneAutenticazione offrirà le funzionalità di login agli utenti.
- L'entità Libro è condivisa tra i vari moduli.

Le funzionalità di registrazione e modifica dei dati degli utenti vengono implementate nei servizi che hanno le responsabilità per tali utenti.

Si è deciso di modificare gli aspetti relativi ad autenticazione e autorizzazione del sistema per gestirli attraverso JWT (JSON Web Token), una tecnologia ampiamente utilizzata per sistemi a microservizi e adatta anche per sistemi con un'architettura di tipo monolite modulare.

Microservizi

A questo punto il sistema dovrebbe essere ulteriormente trasformato in un sistema a microservizi. Un sistema a microservizi puro prevede che ciascun microservizio sia dotato di un database separato, ma data la stretta relazione tra le entità del database si è deciso di scartare una soluzione di questo tipo. È pratica piuttosto comune, infatti, adottare uno stesso database per più microservizi di uno stesso sistema.

Anche se il database è condiviso, però, un'architettura basata su microservizi prescrive che ciascun microservizio debba avere responsabilità esclusiva su una porzione dei dati del database.

L'obiettivo, quindi, è creare dei microservizi che usano lo stesso database, ma tali che ciascun microservizio gestisca in maniera esclusiva i dati di cui è responsabile.

La separazione più immediata consiste nell'ottenere un servizio indipendente a partire da ciascun modulo individuato nelle fasi precedenti (GestioneBiblioteca, GestioneClubDelLibro+GestionePreferenzeDiLettura, GestioneAutenticazione).

La comunicazione del client coi microservizi verrà mediata da un gateway.

Attribuire la responsabilità esclusiva sui dati, nel nostro caso, significa fare in modo che ciascun microservizio si trovi a gestire delle entità e che tali entità siano gestite solo da quel microservizio. Tuttavia ci si è resi conto che effettuare un'operazione del genere nel sistema in questione è arduo, e che lo sforzo implementativo finirebbe per superare i benefici attesi.

In base a quanto detto in precedenza, la seguente sembra essere la suddivisione migliore del sistema in microservizi.

- BibliotecaService: **Biblioteca**, **Libro**, Possesso, PossessoId, TicketPrestito, **Lettore**, **Genere**
- ClubDelLibroService: Esperto, **Biblioteca**, **Lettore**, Evento, **Libro**, **Genere**

In precedente elenco è corredato anche di informazioni relative alle entità sotto la responsabilità dei microservizi BibliotecaService e ClubDelLibroService (per agilità nella trattazione si è scelto di non considerare il microservizio AutenticazioneService). Sono evidenziate in grassetto le entità di cui entrambi i microservizi necessitano per funzionare al meglio.

- Un Libro può appartenere a una Biblioteca, ma anche essere associato a un Evento;
- Genere è una proprietà comune alle entità Libro, Esperto, Lettore, ClubDelLibro;
- Biblioteca è, ovviamente, la principale entità del microservizio BibliotecaService, tuttavia un Esperto è associato a una Biblioteca;
- Un Lettore può iscriversi a un ClubDelLibro, ma può anche richiedere un prestito in una biblioteca (TicketPrestito);

Separare questa rete di relazioni in maniera coerente è compito arduo e gravoso in termini di tempo, per cui a nostro avviso le soluzioni possibili sono tre:

1. Duplicare le entità e relativi DAO su più microservizi;
2. Modificare le relazioni tra le entità del Database sostituendo, nel minor numero possibile di casi, un riferimento a un'entità con la chiave a tale entità;

3. Creare un microservizio che si occupa di parte della persistenza delle entità comuni a più microservizi;

A nostro avviso ciascuna di queste soluzioni presenta difetti:

- La soluzione discussa al punto 1 viola i vincoli tipici di un sistema a microservizi, in particolare viola il principio secondo cui ciascun microservizio è responsabile in maniera esclusiva dei dati che gestisce
- La soluzione discussa al punto 2 costringe a gestire la consistenza del database (vincoli di chiave esterna, ...) direttamente nel codice, senza supporto del DBMS
- Con la soluzione 3 nessun microservizio funzionerebbe nel caso di mancato funzionamento del microservizio responsabile della persistenza

Nessuna delle soluzioni proposte, quindi, sembra essere ottimale.

A nostro avviso, non ci sarebbero vantaggi significativi nel trasformare il sistema a microservizi una volta ottenuto un monolite modulare. Si è deciso, quindi, di trasformare l'architettura in una di tipo "monolite modulare" e di non proseguire oltre con la realizzazione di un sistema a microservizi.

Tra le motivazioni che hanno determinato la scelta di rinunciare allo sviluppo dell'architettura a microservizi vi è anche la riduzione in itinere del team di manutenzione, passato da tre componenti a due.

Impact analysis CR01* (ridotta), CR03: Migrazione a monolite modulare e ristrutturazione interfaccia grafica

Nei precedenti paragrafi si è discusso della decisione di ridurre l'entità della modifica richiesta dalla CR01 e, quindi, a limitarsi a trasformare definitivamente l'architettura del sistema in un'architettura di tipo monolite modulare anziché procedere con un'architettura a microservizi.

È plausibile, quindi, che le componenti impattate da questa versione ridotta della CR01 si discostino da quelle inizialmente discusse nella precedente impact analysis.

Così non è; si è proceduto a calcolare gli Impact Set ex-novo e si è preso coscienza del fatto che tali insiemi risultano immutati rispetto a quanto precedentemente riportato.

CR02: Integrazione post

Il sistema attuale non fornisce funzionalità che permettano la comunicazione tra i lettori e gli esperti di un club del libro, per cui con questa CR si mira a fornire agli esperti la possibilità di pubblicare dei post e ai lettori o ad altri esperti di rispondere agli stessi così da poter favorire lo sviluppo di una community online.

Dal momento che queste funzionalità riguardano specificamente il sottosistema relativo al club del libro ci si attende che vengano impattati in maniera diretta i seguenti componenti, che, di conseguenza, compongono lo starting impact set per la CR:

- ClubDelLibroService.java
- ClubDelLibroServiceImpl.java
- ClubDelLibroController.java

Data la natura delle modifiche da apportare si ritiene verosimile che il SIS coincida col CIS.

L'aggiunta di queste funzionalità prevede la creazione di un'entità Post. Tale entità conterrà informazioni relative alla data di creazione del post e il suo contenuto. Post sarà coinvolto in relazioni con l'entità Club del Libro e con l'entità Lettore.

Gli utenti potranno lasciare commenti ai post appartenenti ai Club del libro ai quali sono iscritti. L'entità Commento avrà relazioni con l'entità Lettore/Esperto.

È prevista, quindi, la creazione di un repository in grado di effettuare operazioni CRUD sui post e sui commenti e lo sviluppo di service e controller ad hoc.

È necessario intervenire anche sul front-end per rendere fruibili tali funzionalità agli utenti sviluppando le pagine che supportino la creazione di post e l'inserimento di commenti.

CR04: Miglioramento qualità del codice

Si è utilizzato il tool SonarLint per analizzare il codice sorgente alla ricerca di code smell e vulnerabilità. Attraverso il tool si è scoperto che i seguenti file (che costituiscono, di conseguenza lo SIS) presentano problematiche da risolvere:

- AreaUtenteController.java
- AutenticazioneController.java

- AutenticazioneServiceImpl.java
- BiblionetApplication.java
- ClubDelLibroContoller.java
- ClubDelLibroService.java
- ClubDelLibroServiceImple.java
- Evento.java
- EventChangeInterceptor.java
- Esperto.java
- Genere.java
- ILibroIDAndName.java
- Length.java
- Lettore.java
- MailConsumer.java
- PossessoId.java
- RefreshConfig.java
- RegexTester.java
- RegistrazioneController.java
- UtenteRegistrato.java
- VisualizzaListClubResponse.java
- VisualizzaCreaClubLibroResponse.java

Data la natura dei problemi rilevati attraverso SonarLint si ritiene che l'insieme di file elencato costituisca il CIS per questa CR.

A causa della riduzione del team di manutenzione si è deciso di non implementare tale modifica.

Analisi post-implementazione

In questa sezione vengono analizzati i risultati relativi alle implementazioni delle CR proposte.

CR01* (ridotta), CR03: Migrazione a monolite modulare e ristrutturazione interfaccia grafica

L'implementazione di questa prima modifica è stata un successo, poiché tutto funziona come previsto. Il sistema ottenuto a valle di questa modifica è in condizioni tali da permettere di eseguire le future richieste di modifica. A supporto di questa affermazione vi sono i risultati ottenuti in seguito al testing di non regressione (100% fallimento).

Dall'analisi della modifica è emerso che nessun componente non appartenente al CIS è stato impattato dalla modifica, mentre un solo componente appartenente al CIS non è stato modificato.

Per cui si evince che:

- DIS = ∅
- AIS = CIS + DIS - {GenereDAO.java}

È possibile, a questo punto, estrarre delle metriche relative alla modifica in questione:

$$Recall = \frac{|CIS \cap AIS|}{|AIS|} = \frac{44}{44} = 100.0\%$$

$$Precision = \frac{|CIS \cap AIS|}{|CIS|} = \frac{44}{45} = 97.8\%$$

CR02: Integrazione post

È stata realizzata con successo anche la modifica richiesta dalla CR02, come si evince dai risultati dell'esecuzione dei test di non regressione (100% fallimento). Per tale modifica si è ricavato che:

- DIS = ∅
- AIS = CIS

È possibile, a questo punto, estrarre delle metriche relative alla modifica in questione:

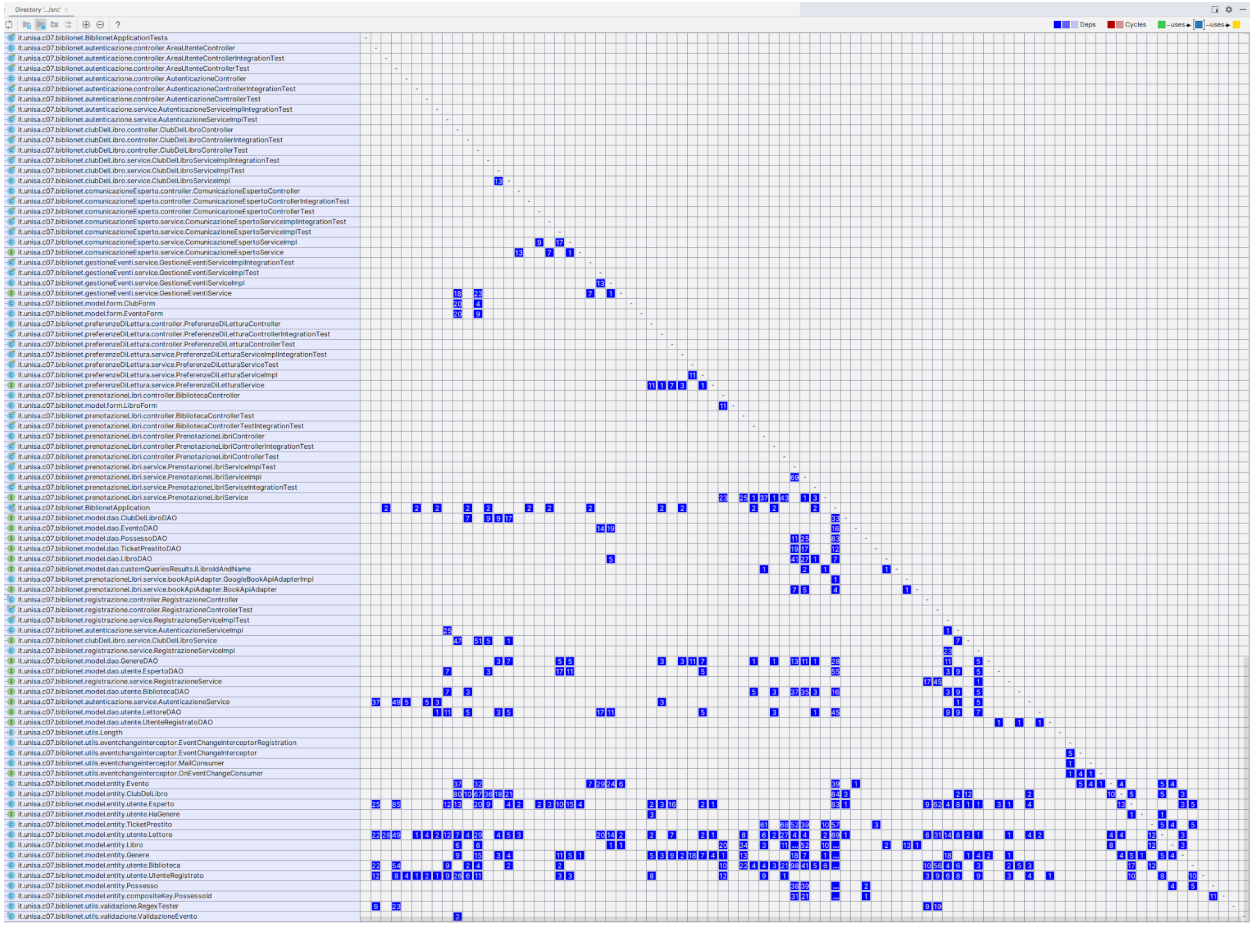
$$Recall = \frac{|CIS \cap AIS|}{|AIS|} = \frac{3}{3} = 100.0\%$$

$$Precision = \frac{|CIS \cap AIS|}{|CIS|} = \frac{3}{3} = 100.0\%$$

Impatto sui file di test

Di seguito è discusso l'impatto sui file di test per ciascuna delle Change Request riportate in precedenza.

A tale scopo si è ottenuta un'altra matrice delle dipendenze che comprende anche i file di test per il sistema Biblionet.



CR01, CR03: Migrazione a microservizi e ristrutturazione interfaccia grafica

Si è calcolato, in un paragrafo precedente, il Candidate Impact Set per le Change Request in esame.

Tale set costituirà lo Starting Impact Set per determinare l'impatto delle modifiche sui file di test.

Dall'analisi della matrice delle dipendenze si è determinato che il Candidate Impact Set comprendente anche i test per tali CR è il seguente:

- AutenticazioneController.java
- AutenticazioneService.java
- AutenticazioneServiceImpl.java
- AreaUtenteController.java
- Biblioteca.java
- BibliotecaController.java
- BibliotecaDAO.java
- BookAPIAdapter.java
- ClubDelLibroController.java
- ClubDelLibroDAO.java
- ClubDelLibroService.java
- ClubDelLibroServiceImpl.java
- ClubForm.java
- ComunicazioneEspertoController.java
- ComunicazioneEspertoService.java
- ComunicazioneEspertoServiceImpl.java
- Esperto.java
- EspertoDAO.java
- Evento.java
- EventoForm.java
- Genere.java
- GenereDAO.java
- GestioneEventiService.java
- GestioneEventiServiceImpl.java
- GoogleBookAPIAdapterImpl.java
- HaGenere.java
- Lettore.java

- LettoreDAO.java
- Libro.java
- LibroDAO.java
- LibroForm.java
- PrenotazioneLibriController.java
- PrenotazioneLibriService.java
- PrenotazioneLIbriServiceImpl.java
- PreferenzeDiLetturaController.java
- PreferenzeDiLetturaService.java
- PreferenzeDiLetturaServiceImpl.java
- RegexTester.java
- RegistrazioneController.java
- RegistrazioneService.java
- RegistrazioneServiceImpl.java
- TicketPrestito.java
- ValidazioneEvento.java

Il CIS così ottenuto è il seguente:

- AutenticazioneController.java
- AutenticazioneService.java
- AutenticazioneServiceImpl.java
- AreaUtenteController.java
- Biblioteca.java
- BibliotecaController.java
- BibliotecaDAO.java
- BookAPIAdapter.java
- ClubDelLibroController.java
- ClubDelLibroDAO.java
- ClubDelLibroService.java
- ClubDelLibroServiceImpl.java
- ClubForm.java
- ComunicazioneEspertoController.java
- ComunicazioneEspertoService.java
- ComunicazioneEspertoServiceImpl.java
- Esperto.java

- EspertoDAO.java
- Evento.java
- EventoForm.java
- Genere.java
- GenereDAO.java
- GestioneEventiService.java
- GestioneEventiServiceImpl.java
- GoogleBookAPIAdapterImpl.java
- HaGenere.java
- Lettore.java
- LettoreDAO.java
- Libro.java
- LibroDAO.java
- LibroForm.java
- PrenotazioneLibriController.java
- PrenotazioneLibriService.java
- PrenotazioneLIbriServiceImpl.java
- PreferenzeDiLetturaController.java
- PreferenzeDiLetturaService.java
- PreferenzeDiLetturaServiceImpl.java
- RegexTester.java
- RegistrazioneController.java
- RegistrazioneService.java
- RegistrazioneServiceImpl.java
- TicketPrestito.java
- ValidazioneEvento.java
- AreaUtenteControllerTest.java
- AutenticazioneControllerTest.java
- AutenticazioneServiceImplIntegrationTest.java
- PreferenzeDiLetturaControllerIntegrationTest.java
- AutenticazioneServiceImplTest.java
- RegistrazioneServiceImplTest.java
- ClubDelLibroControllerIntegrationTest.java
- CludDelLibroControllerTest.java
- ComunicazioneEspertoServiceImplTest.java

- BibliotecaControllerTest.java
- BibliotecaControllerTestIntegrationTest.java
- PrenotazioneLibriControllerIntegrationTest.java
- PrenotazioneLibriControllerTest.java
- PrenotazioneLibriServiceImplTest.java
- PrenotazioneLibriServiceIntegrationTest.java
- RegistrazioneControllerTest.java
- RegistrazioneServiceImplTest.java
- ClubDelLibroServiceImplIntegrationTest.java
- ClubDelLibroServiceImplTest.java
- ComunicazioneEspertoServiceImplIntegrationTest.java
- ComunicazioneEspertoControllerTest.java
- PreferenzeDiLetturaControllerTest.java
- GestioneEventiServiceImplIntegrationTest.java
- GestioneEventiServiceImplTest.java
- EventChangeInterceptor.java
- MailConsumer.java
- OnEventChangeConsumer.java
- PreferenzeDiLetturaServiceImplIntegrationTest.java
- PreferenzeDiLetturaServiceTest.java