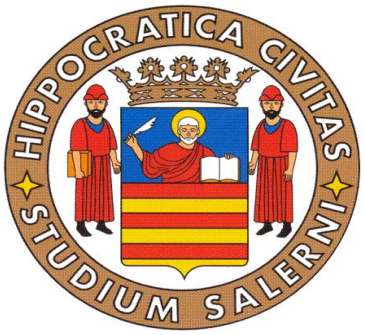
UNIVERSITÀ DEGLI STUDI DI SALERNO

CORSO DI LAUREA MAGISTRALE IN INFORMATICA



INGEGNERIA GESTIONE ED EVOLUZIONE DEL SOFTWARE
a.a 2017/2018

---

b:btech

ANALISI E MANUTENZIONE

Versione 1.1

Top Manager:

Nome
Prof. Andrea De Lucia

Partecipanti:

Nome	Matricola
UP – Umberto Picariello	0522500485
CC – Carmine Capo	0522500460
FV – Federico Vitale	0522500503

Revision History

Data	Versione	Descrizione	Autori
20/10/2017	1.0	Prima stesura	Umberto Picariello Carmine Capo Federico Vitale
25/11/2017	1.1	Revisione del documento, aggiunta della soluzione individuata e dell'Impact Set	Umberto Picariello Carmine Capo Federico Vitale

Indice

1. Scopo del documento

In questo documento saranno descritti gli obiettivi del processo di migrazione e di integrazione dei nuovi moduli all'interno del sistema esistente *BibTech*, come definito nel documento di "Identificazione e classificazione della modifica richiesta".
Verrà studiato come l'integrazione potrà impattare sugli artefatti del sistema esistente, analizzando il rapporto costi/benefici e i possibili rischi derivanti dalla fase di progettazione.
Questo documento includerà uno studio di fattibilità e verranno pianificate le fasi successive di progettazione, implementazione e testing.

2. Panoramica del sistema esistente

Il sistema attuale è un sistema di gestione di una biblioteca che permette al personale di velocizzare le comuni operazioni di amministrazione dei prestiti e rendere più immediata la loro visualizzazione.
Attualmente questo software permette di visualizzare tutte le informazioni riguardanti i libri gestiti dalla biblioteca, gli utenti e i prestiti. Inoltre, mediante un interfaccia grafica, permette al personale di caricare nuovi volumi, aggiungere nuovi utenti e gestire tutto quello che riguarda i prestiti effettuati.

Allo stato attuale il sistema è rivolto a una ristretta cerchia di persone: personale bibliotecario.

2.1 Attori e funzionalità

Il sistema, attualmente, prevede un solo attore con le relative funzionalità.

- Personale bibliotecario:
    - Gestione dei prestiti;
    - Gestione degli utenti;
    - Gestione dei libri.

2.2 Design, implementazione e testing del sistema attuale

2.2.1 Design

Non è presente una documentazione relativa all'object design.
Pertanto da una prima analisi possiamo affermare che il sistema *BibTech* non risulta seguire alcun pattern architetturale, in quanto non presenta nessuna suddivisione in sottosistemi.
Il progetto è, quindi, composto da un unico package contenente tutte le classi del sistema.
Ogni classe presenta sia la logica applicativa che l'interfaccia, gestendo anche il collegamento al database.
Inoltre, il sistema attuale presenta problemi di bassa coesione perchè alcune classi rappresentano più di un concetto. Questo ci ha portato ad individuare la Bad Smells Blob.
Secondo la change request *IS1* presente nel documento di *"Identificazione e classificazione della modifica richiesta"* il sistema BibTech risulta non decomponibile rispetto alla migrazione sul web.
Inoltre, il sistema sarà fortemente impattato dalla modifica, in quanto implementa l'interfaccia utilizzando Swing di Java.
Il passo nella fase di design è stato quello di trasformare l'architettura esistente al fine di effettuare la migrazione del sistema sul Web. Dopo un'attenta analisi, è emerso che le componenti OO esistenti non erano adatte al tipo di architettura e, per questo motivo, sono state implementate ex-novo.
In seguito, sono state aggiunte delle nuove funzionalità al sistema.
Al termine della migrazione sul web, ci si aspetta che l'interfaccia grafica sia costituita da pagine web.
Per questo motivo non sarà possibile riutilizzare le classi correnti e sarà necessario creare da zero la modularizzazione del sistema.

2.2.2 Implementazione

Non risulta una documentazione relativa all'implementazione.
Analizzando il codice, possiamo dire che il sistema BibTech è stato sviluppato con il linguaggio di programmazione object oriented Java, coniugato all'apporto delle librerie JDBC per la connessione al DBMS.

2.2.3 Testing

Non risulta una documentazione relativa al testing, così come l'implementazione delle classi di test.
Pertanto si prevede di implementare le classi di test delle nuove componenti sviluppate utilizzando il framework JUnit.

3. Analisi della modifica richiesta

In questa sezione vengono brevemente descritte le modifiche richieste analizzando gli aspetti tecnici.

3.1 Analisi Change Request IS1

Il processo di migrazione prevede che i requisiti funzionali e gli attori che interagiscono con il sistema rimangano invariati. L'obiettivo principale consiste nel migrare la graphic user interface del sistema standalone ad una nuova basata su tecnologia web.

3.2 Analisi Change Request IS2-IS3

Il processo di integrazione dei nuovi moduli prevede che le funzionalità del sistema esistente rimangano invariate. L'obiettivo principale consiste nell'integrare due nuovi moduli che permettano la ricezione di una notifica quando un libro torna disponibile per il prestito e l'accesso alla propria area riservata. I moduli in questione, saranno integrati nel sistema esistente *BibTech*.

4. Individualizzazione della soluzione progettuale

4.1 Problematiche affrontate

Issue 1: Quali funzionalità verranno migrate nel nuovo sistema?

Proposal 1.1
Migrare solo alcune delle funzionalità presenti nel sistema *BiBTech* quali gestione degli utenti e gestione dei prestiti.
Proposal 1.2
Migrare tutte le funzionalità del sistema.

Criterion 1.1
Scegliere funzionalità che acquisiscono reali vantaggi nella migrazione del sistema.
Criterion 1.2
Scegliere funzionalità che permettono il corretto utilizzo del sistema.

Argument 1.1
La funzionalità di gestione degli utenti, se migrata sul web, fornisce uno strumento mediante il quale un utente può gestire i propri dati in maniera autonoma. La migrazione sul web della funzionalità di gestione dei prestiti acquisisce il vantaggio di consentire agli utenti di poter visualizzare tutte le informazioni relative ai propri prestiti. Viene mantenuta la possibilità di visualizzazione dei prestiti da parte del personale bibliotecario.
Argument 1.2
Per un corretto funzionamento, il sistema ha necessità di disporre di tutte le funzionalità attualmente presenti.

Resolution 1.1
Analizzando le argomentazioni Argument 1.1 e Argument 1.2, l'Issue 1 è stata risolta applicando la Proposal 1.2.

Issue 2: Quali funzionalità del sistema hanno priorità maggiore?

Proposal 2.1
Gestione degli utenti.
Proposal 2.2
Gestione dei prestiti.

Criterion 2.1 - 2.2
Sviluppare le funzionalità fondamentali del sistema.

Argument 2.1
Sviluppare per prima la gestione degli utenti consente di avere immediatamente a disposizione tutti i dati personali degli utenti.
Argument 2.2
La gestione dei prestiti è la funzionalità core del sistema. Le altre funzionalità, inizialmente, potrebbero essere emulate mediante l'utilizzo di specifici driver e stub.

Resolution 2.1
Analizzando le argomentazioni Argument 2.1 e Argument 2.2, l'Issue 2 è stata risolta applicando la Proposal 2.1.

Issue 3: Sono necessarie nuove funzionalità?

Proposal 3.1
Nessuna nuova funzionalità aggiunta.
Proposal 3.2
Aggiunta di una feature che permetta all'utente di inserire una recensione riguardante un libro.

Criterion 3.1
Sviluppare funzionalità solo se strettamente necessario per ridurre i tempi di sviluppo.
Criterion 3.2
Fornire nuove funzionalità per l'inserimento di feedback.

Argument 3.1
Le funzionalità attuali consentono l'utilizzo del sistema. Nuove funzionalità non sono vitali.
Argument 3.2
Fornire la possibilità all'utente di inserire una recensione consentirebbe, a quest'ultimo, di fornire il proprio contributo agli altri utenti.

Resolution 3.1
Analizzando le argomentazioni Argument 3.1, Argument 3.2 e Argument 3.3, l'Issue 3 è stata risolta applicando la Proposal 3.1.

Issue 4: Che tipo di interfaccia utente adottare?

Proposal 4.1
Conservare la User Interface attuale.
Proposal 4.2
Riscrivere la User Interface.

Criterion 4.1
Mantenere bassi i costi della manutenzione.

Criterion 4.2
Rendere il sistema più user-friendly.

Argument 4.1
Rimpiazzare completamente l'interfaccia utente ha costi elevati.
Argument 4.2
Un'interfaccia grafica user-friendly consente all'utente di interagire con il sistema controllando oggetti grafici convenzionali.

Resolution 4.1
Analizzando le argomentazioni Argument 4.1 e Argument 4.2, l'Issue 4 è stata risolta applicando la Proposal 4.2.

Issue 5: Che modello architetturale utilizzare?

Proposal 5.1
L'architettura software del nuovo sistema viene strutturata utilizzando il modello *Three Tier* in versione *Client-Server*.
Proposal 5.2
L'architettura software del nuovo sistema viene strutturata utilizzando il modello base a due livelli.

Criterion 5.1
Utilizzare un'architettura più robusta e flessibile.
Criterion 5.2
Mantenere più bassi i costi di manutenzione.

Argument 5.1
L'architettura *Three Tier* consente la divisione netta tra la parte di presentazione e la parte che si occupa della logica applicativa. Garantisce una buona manutenibilità e consente di gestire la complessità introducendo livelli di astrazione con responsabilità ben definite.
Argument 5.2
L'architettura di base a due livelli permette di rendere l'applicazione più veloce rispetto a quella *Three Tier*, semplificando l'implementazione del software in progetti di piccole dimensioni.

Resolution 5.1
Analizzando le argomentazioni Argument 5.1 e Argument 5.2, l'Issue è stata risolta applicando la Proposal 5.1.

Issue 6: Che tipologia di linguaggi di programmazione utilizzare per lo sviluppo del nuovo sistema?

Proposal 6.1
Per l'implementazione dei moduli lato client si utilizza HTML, CSS e Javascript, mentre per l'implementazione dei moduli lato server si utilizza il linguaggio Java per applicazioni Web- based: JSP e Servlet Java.
Proposal 6.2
Per l'implementazione dei moduli lato client si utilizza HTML, CSS e Javascript, mentre per l'implementazione dei moduli lato server si utilizza il linguaggio di programmazione PHP.

Criterion 6.1
L'implementazione dei moduli client e server deve essere funzionante indipendentemente dalla piattaforma.
Criterion 6.2
L'implementazione deve essere facilitata dal riuso del codice.

Argument 6.1
Utilizzando la combinazione HTML CSS e Javascript per l'implementazione dei moduli lato client, si assicura il funzionamento su ogni browser compatibile, indipendentemente dalla piattaforma sofware/hardware. Per questi moduli non è possibile il riuso di codice preesistente poiché tali linguaggi sono stati introdotti nella progettazione del nuovo sistema. Inoltre, i moduli del server sono implementa utilizzando Java Servlet e pagine JSP, che consentono di eseguire classi Java e/o script Java su richiesta del client. Per questi moduli non è possibile il riuso di codice preesistente poiché tali linguaggi sono stati introdotti nella progettazione del nuovo sistema.
Argument 6.2
Utilizzando la combinazione HTML CSS e Javascript per l'implementazione dei moduli lato client, si assicura il funzionamento su ogni browser compatibile, indipendentemente dalla piattaforma sofware/hardware. Per questi moduli non è possibile il riuso di codice preesistente poiché tali linguaggi sono stati introdotti nella progettazione del nuovo sistema. I moduli del server sono implementati utilizzando il linguaggio PHP, che consente di eseguire classi PHP su richiesta del client. In questo caso si dovrebbe attuare il porting del codice esistente in PHP o sviluppare un wrapper.

Resolution 6.1
Analizzando le argomentazioni Argument 6.1 e Argument 6.2, l'Issue è stata risolta applicando la Proposal 6.1.

Issue 7: Aggiungere un sistema interno per l'invio di notifiche?

Proposal 7.1
Non aggiungere il sistema di notifiche.
Proposal 7.2
Aggiungere il sistema di notifiche.

Criterion 7.1
Mantenere bassi i costi e i tempi di sviluppo.
Criterion 7.2
Migliorare l'esperienza dell'utente.

Argument 7.1
Aggiungere un sistema di notifiche può risultare una delle parti più costose della manutenzione. Potrebbe esserci il bisogno di utilizzare tool esterni e il rischio che il Team non conosca il tool da utilizzare.
Argument 7.2
Le notifiche sono essenziali, l'esperienza dell'utente può essere notevolmente migliorata e semplificata grazie ad esse.

Resolution 7.1
Analizzando le argomentazioni Argument 7.1 e Argument 7.2, l'Issue è stata risolta applicando la Proposal 7.1.

Issue 8: Aggiungere un sistema di autenticazione?

Proposal 8.1
Non aggiungere l'autenticazione.
Proposal 8.2
Aggiungere l'autenticazione.

Criterion 8.1
Mantenere bassi i costi per la manutenzione.
Criterion 8.2
Garantire all'utente l'accesso riservato alla propria area.

Argument 8.1
Aggiungere un sistema di autenticazione può risultare molto costoso durante la manutenzione.
Argument 8.2
Gli utenti voglio avere un pò di privacy riguardo le proprie informazioni e lo storico dei libri di cui hanno usufruito. Aggiungere un sistema di autenticazione agevola gli utenti in questo senso.

Resolution 8.1
Analizzando le argomentazioni Argument 8.1 e Argument 8.2, l'Issue è stata risolta applicando la Proposal 8.2.

4.1.2 Soluzione individuata

La soluzione individuata consiste nell'unione di tutte le "Resolution" ottenute nel paragrafo 4.1. In dettaglio, la soluzione individuata dovrà avere le seguenti caratteristiche:

- Conservare tutte le funzionalità del sistema attuale andando ad apportare delle modifiche laddove necessario;
- Priorità alta allo sviluppo delle funzionalità relative alla gestione degli utenti;
- Non verrà aggiunta la funzionalità riguardante l'inserimento di una recensione in quanto non ritenuta vitale per il funzionamento del sistema;
- Riscrivere la User Interface rendendola user-friendly e compatibile con i vari browser e le varie piattaforme;
- L'architettura del nuovo sistema viene strutturata su tre livelli logico-funzionali (Three Tier);
- Per l'implementazione dei moduli lato client si utilizzerà HTML, CSS e Javascript, mentre per l'implementazione dei moduli lato server si utilizzerà il linguaggio Java per applicazioni Web-based: JSP e Servlet Java;
- Viene aggiunto un sistema di autenticazione.

4.1.3 Soluzioni alternative

Le soluzioni alternative sono rappresentate da tutte le proposte presentate e affrontate nel paragrafo 4.1.1, ma che non sono state selezionate per la soluzione adottata.

5. Identificazione dell'Impact Set

La soluzione individuata ha coinvolto diversi artefatti del sistema attuale. La tabella che segue descrive il Candidate Impact Set individuato, ovvero l'insieme di tutti gli artefatti che verranno modificati durante la fase di manutenzione. Inoltre, accanto ad ogni artefatto sarà associato un livello di impatto che indica il modo in cui la modifica va ad impattare sul sistema software attuale.

Per individuare le componenti del sistema impattate si utilizzerà un approccio top-down, ovvero a partire dai documenti di alto livello si andrà ad individuare gli artefatti di più basso livello che andranno ad essere modificati durante la fase di manutenzione.

Nella tabella che segue sono indicati gli artefatti del documento che saranno impattati. L'impatto della modifica verrà valutato utilizzando tre categorie:

- FORTE: se saranno necessarie pesanti modifiche nell'artefatto o se l'artefatto dovrà essere completamente sostituito.
- MEDIO: se saranno necessarie sostanziali modifiche all'artefatto, non facendo cambiare però la sua struttura in maniera eccessiva.
- DEBOLE: se saranno necessarie solo modifiche marginali.

Non avendo a disposizione alcun tipo di documentazione, andiamo a vedere le classi che verranno impattate dalla modifica:

Artefatto	Impatto	Descrizione
AggiungiAutore.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un autore.
AggiungiCopia.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire le copie riguardanti un libro.
AggiungiDizionario.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un dizionario.
AggiungiLibro.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un libro.
AggiungiManuale.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un manuale.
AggiungiPeriodico.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un periodico.
AggiungiUtente.java	FORTE	Andrà creata una pagina web attraverso la quale il personale bibliotecario potrà inserire un utente.
ControllaPrestiti.java	FORTE	Andrà creata una pagina web attraverso la quale potranno essere gestiti i prestiti.
Ricerca.java	FORTE	Andrà creata una pagina web attraverso la quale potrà essere gestita la ricerca.
Boot.java	FORTE	Il file sarà rimpiazzato.
FrameError.java	FORTE	Il file sarà rimpiazzato.
FrameMain.java	FORTE	Il file sarà rimpiazzato.
Gestione.java	FORTE	Il file sarà rimpiazzato.
MDTable.java	FORTE	Il file sarà rimpiazzato.

6. Studio di fattibilità

6.1 Identificazione, descrizione e valutazione dei costi

Identificazione	Valutazione	Motivazioni
Migrazione dei requisiti funzionali.	DEBOLE	Il processo di integrazione è effettuato in modo incrementale. Pertanto in primo momento verranno integrate le funzionalità secondo la Proposal 2.1.
Aggiunta dei nuovi requisiti funzionali.	MEDIA	Il numero di funzionalità da aggiungere non risulta elevato.
Aggiunta del sistema di autenticazione.	DEBOLE	Viene sviluppato da zero ma il team ha già esperienza con tali sistemi.
Garantire la struttura three-tier.	MEDIA	Il sistema attuale non presenta questo tipo di architettura, pertanto sarà necessario prestare molta attenzione in modo da rispettare tale architettura.
Implementazione con linguaggi di programmazione client-side HTML, CSS, JavaScript e server-side JAVA.	FORTE	Il Team non ha mai lavorato con tecnologie server-side Java, pertanto ci sarà una fase di apprendimento che andrà ad impattare sui costi.
Sostituzione interfaccia utente.	DEBOLE	Viene sviluppata da zero ma il team ha già esperienza con tali sistemi.
Velocità di accesso alle risorse.	DEBOLE	Le tecnologie attuali permettono di sorvolare gli aspetti di ottimizzazione dei tempi di risposta da parte del server.
Gestione della sicurezza.	DEBOLE	La gestione della sicurezza è realizzata tramite autenticazione per mezzo di username e password.
Organizzazione del lavoro nel team di sviluppo.	FORTE	I componenti del team di sviluppo condividono poche ore settimanali per lo sviluppo del nuovo sistema. Pertanto verranno effettuate riunioni extra il sabato e/o la domenica.
Testing funzionale.	FORTE	Il sistema attuale non risulta testato, pertanto la fase di testing dovrà essere svolta ex-novo.

Testing di accettazione	FORTE	E’ necessaria l’interazione con il cliente finale per analizzare i risultati del test.

6.2 Identificazione, descrizione e classificazione dei benefici

Identificazione	Classificazione	Motivazione
Facilità di distribuzione e manutenzione.	FORTE	L'applicazione web si trova interamente sul server, per cui la pubblicazione o l'aggiornamento di risorse sul nodo server è automaticamente reso disponibile a tutti i nodi client.
Accesso multipiattaforma.	FORTE	L'accesso all'applicazione web è indipendente dall'hardware e dal sistema operativo utilizzato dagli utenti.
Scalabilità.	MEDIA	Il nuovo sistema software può crescere insieme alle esigenza del cliente, senza particolare problemi.
Aggiunta del sistema di autenticazione.	FORTE	Garantisce la privacy degli utenti riguardo le proprie informazioni e lo storico dei libri di cui ha usufruito.

6.3 Identificazione, descrizione e classificazione dei rischi

Identificazione	Probabilità	Motivazioni
Introduzioni di nuovi fault.	MEDIA	Il testing che si intende effettuare aumenta le probabilità di riscontrare nuovi errori.
Malfunzionamenti del Server.	DEBOLE	La probabilità che accada è bassa ma le conseguenze possono essere gravi. Il sistema potrebbe essere non raggiungibile per diverso tempo.
Attacchi informatici.	DEBOLE	L'uso del web aumenta la probabilità di essere esposti ad attacchi informatici.
Carico di lavoro eccessivo per il Server.	DEBOLE	I client che comunicano con il server potrebbero sbilanciare il carico di lavoro. Tuttavia, questo è un rischio marginale visto il bacino di utenza, previsto, del software.
Gestore dello script client-side disabilitato.	MEDIA	Il client può effettuare delle elaborazioni sui dati senza coinvolgere il server. Tale elaborazione è determinata dall'utilizzo di Javascript. Se tale linguaggio di script venisse disabilitato nel browser non si potrebbero validare i dati utente e di conseguenza i dati inviati al server potrebbero essere corrotti.

6.4 Risultati previsti

La realizzazione del sistema software è definita secondo la soluzione individuata nel paragrafo 4.1.2, tenendo in considerazione costi, benefici e rischi di cui al paragrafo 6.