**Università degli Studi di Salerno**

**Corso di Ingegneria del Software**

**BookPoint**

**SDD**

**Versione 1.0**

Partecipanti:

|  |  |
| --- | --- |
| **Nome Cognome** | **Matricola** |
| **Massimo Giordano** | **0512104480** |
| **Giovanni Buonocore** | **0512104612** |
| **Gennaro Teodoro** | **0512104876** |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
|  |  |  |  |

Sommario

[1 Introduzione 2](#_Toc533693297)

[1.1 Purpose of the system 2](#_Toc533693298)

[1.2 Design Goals 2](#_Toc533693299)

[1.3 Definition, acronyms, and abbreviations 2](#_Toc533693300)

[1.4 References 2](#_Toc533693301)

[1.5 Overview 2](#_Toc533693302)

[2 Current software architecture 3](#_Toc533693303)

[3 Proposed software architecture 3](#_Toc533693304)

[3.1 Overview 3](#_Toc533693305)

[3.2 Subsystem decomposition 3](#_Toc533693306)

[3.3 Hardware/software mapping 3](#_Toc533693307)

[3.4 Persistent data management 3](#_Toc533693308)

[3.5 Access control and security 3](#_Toc533693309)

[3.6 Global software control 3](#_Toc533693310)

[3.7 Boundary conditions 3](#_Toc533693311)

[4 Subsystem services 3](#_Toc533693312)

# Introduzione

## Purpose of the system

Il titolare di una libreria in via Mazzini vuole espandere il range di vendita dei libri nel suo negozio e per fare questo ha deciso di utilizzare un e-commerce online.  
Il sistema dovrà quindi permettere la ricerca e l’acquisto di libri disponibili nel negozio, inoltre il sistema offrirà la possibilità di lasciare commenti e di tenere traccia dei libri preferiti.

## Design Goals

**Tempi di risposta:** Il sistema dovrà essere performante e reattivo in ogni situazione e garantire tempi di risposta ragionevoli per l'utente, in modo che egli non percepisca il sistema come bloccato e non provi frustrazione durante il suo utilizzo.

**Validazione input:** il sistema deve essere robusto, ovvero ogni volta che l’utente compila degli input di testo, ad esempio il login/registrazione, l’input sarà controllato in modo da non permettere la sottomissione di input non ammesso.

**Sicurezza:** quando l’utente si registra o cambia la password, quest’ultima viene criptata e salvata all’interno del database, così da non permettere, in nessun modo di risalire alla password se non attraverso il codice di decifratura.

**Criteri di costi (Tempo):** il sistema deve essere consegnato entro il secondo appello di is.

**Criteri di mantenimento:** Il codice deve essere comprensibile e facile da modificare in caso di futuri cambiamenti. Per far questo si userà Javadoc per commentare i metodi, il che aiuterà l’individuazione del metodo da modificare.

**Usabilità:** Il sito deve essere semplice dal punto di vista dell’usabilità. Con semplice si intende che il sito deve guidare l’utente in tutte le azioni che fa, aiutandolo attraverso un’interfaccia semplice con immagini, testi e bottoni, in modo da minimizzare il numero di errori che un utente può commettere.

## Definition, acronyms, and abbreviations

## References

Vedere RAD

## Overview

Il seguente documento di System Design (SDD) mostra i dettagli tecnici del sistema BookPoint.

Il documento si compone di una prima parte in cui vengono introdotti gli obiettivi di design. Verrà poi spiegata l’architettura del sistema proposto. Verranno brevemente illustrate le parti di cui si compone il documenti:

* Nel capitolo 2 viene mostrata l’architettura del sistema corrente;
* Nel capitolo 3 viene mostrata l’architettura del sistema proposto, in dettaglio:
  + Decomposizione del sistema in sottosistemi di funzionalità e operazioni correlate
  + Mapping Hardware/Software;
  + Gestione dei dati persistenti descriver, i dati persistenti memorizzati dal sistema;
  + Controllo degli accessi e della sicurezza descrive, tramite una matrice degli accessi, le operazioni effettuabili da ogni tipologia di utente;
  + Controllo del flusso globale descrive, quali operazioni eseguire ed in che ordine;
  + Condizioni boundary che includono l’avvio e lo spegnimento del sistema.
* Nel capitolo 4 vengono mostrati i servizi forniti da ogni sottosistema.

# Current software architecture

# Proposed software architecture

## Overview

In seguito verrà mostrata la suddivisione del sistema in sottosistemi con i relativi compiti che ogni sottosistema dovrà offrire e il mapping hardware/software del sistema.

Successivamente vengono descritti i dati persistenti con il relativo schema relazionale, verrà discusso il controllo degli accessi con la descrizione degli utenti del nostro sistema, il controllo generale del software e le boundary contitions.

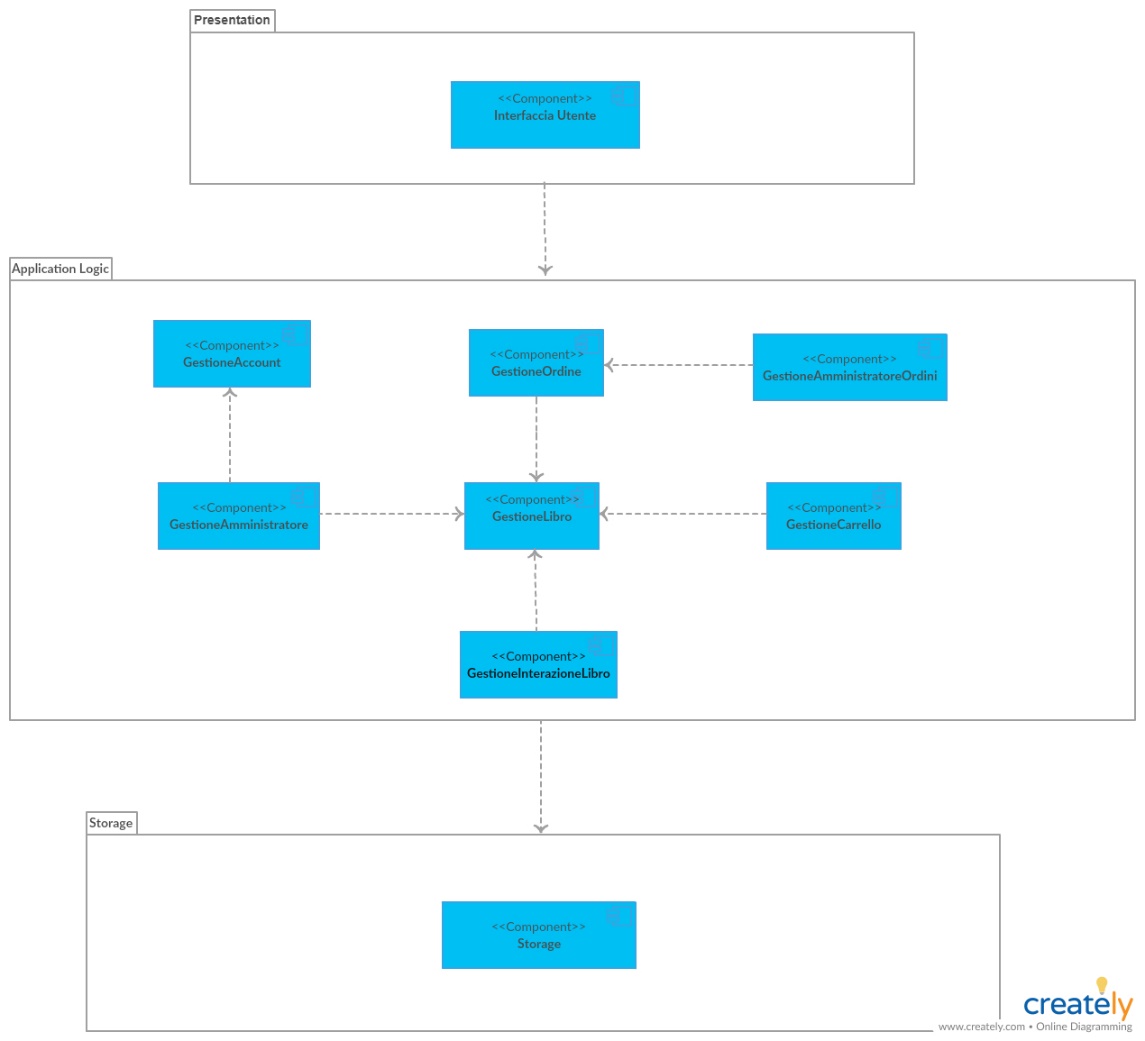
## Subsystem decomposition

L’architettura scelta per questo sistema è la Model-View-Controller (MVC).

Essa si struttura in tre componenti, ognuno con un compito diverso all’interno del sistema.

* Il Model si occupa di gestire i dati persistenti. Contieni infatti metodi di lettura e scrittura su Database.
* La View si occupa di far visualizzare i dati all’utente e si occupa dell’interazione tra quest’ultimo ed il sistema.
* Il Controller riceve i comandi dell’utente attraverso le view e si occupa della logica di controllo dell’applicazione interagendo con gli altri due componenti.

L’architettura MVC verrà implementata con la tecnologia Servlet-JSP



**Interfaccia Utente**: Fornisce all’utente l’interfaccia per interagire col sistema.

**Gestione Account**: Fornisce i servizi per gestire il proprio account, quali: gestione delle carte di credito, gestione degli indirizzi e cambio password.

**Gestione Libro**: Fornisce i servizi per ricercare, modificare, aggiungere e rimuovere dei libri.

**Gestione Carrello**: Gestisce il carrello, permettendone la modifica e la persistenza.

**Gestione Ordine**: Permette di consultare lo storico degli ordini, visualizzare la fattura, ricercare un ordine e di modificarne lo stato e la data di consegna.

**Gestione Amministratore**: Fornisce all’amministratore gli strumenti necessari per aggiungere, modificare e rimuovere un libro, per ricercare un account e modificarne il tipo.

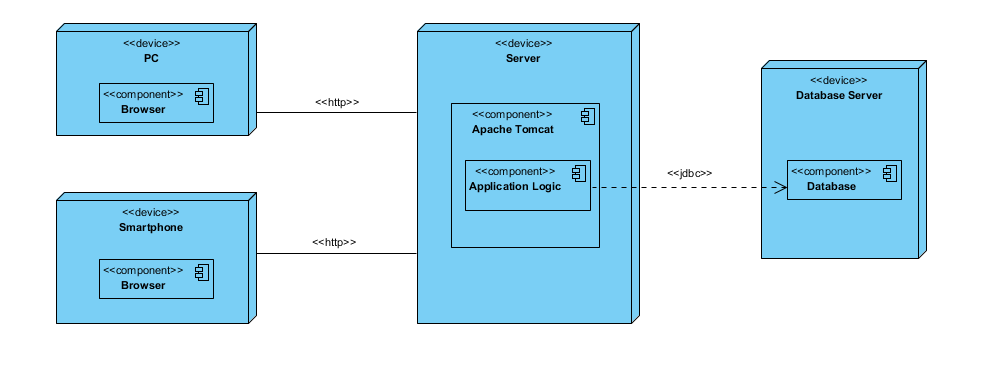
**Gestione Amministratore Ordini**: Fornisce all’amministratore gli strumenti necessari per ricercare un ordine e modificarne lo stato e la data di consegna.

**Gestione Interazione Libro**: Fornisce i servizi per aggiungere o rimuovere una recensione e per aggiungere o rimuovere un libro dai preferiti.

**Storage**: gestisce il salvataggio, la modifica e il recupero dei dati dal database.

## Hardware/software mapping

Il sistema utilizza un’architettura Client/Server.   
Il WebServer è rappresentato da Apache Tomcat 9 ed è situato su una singola macchina, la logica del sistema è costituita da Java Servlet mentre l’interfaccia utente è realizzata utilizzando pagine JSP (Java Servlet Page). Il Client è rappresentato dal Web Browser utilizzato dall’utente.  
La comunicazione tra i nodi è rappresentata da richieste e risposte http tra client e server, e da query in JDBC tra server e database.



## Persistent data management

## Access control and security

## Global software control

Il Web Server si occupa di gestire le varie richieste effettuate dal client.  
Il server smista le richieste alle classi Java Servlet opportune che si occuperanno di gestire la richiesta, eventualmente interagire con il model, e dare una risposta.  
Dopodiché il server crea la pagina jsp che verrà poi convertita in pagina html e visualizzata dall’utente.

## Boundary conditions

Le condizioni limite riguardano l’accensione e lo spegnimento del sistema per quanto riguarda il lato Server. Dal lato Client si riferiscono agli errori di connessione al server.

**Avvio del sistema**

Il sistema dopo essersi avviato presenta un’interfaccia ai client. Dopo aver effettuato l’autenticazione ogni utente può accedere alle funzionalità disponibili.

|  |  |
| --- | --- |
| **Nome** | Startup Server |
| **Partecipanti** | Amministratore |
| **Flusso di eventi** | |
| **Attore:** | Decide di avviare il server, accede al pc dove risiede il server e clicca “Avvia” |
| **Sistema:** | Ricevuta la richiesta, attiva il server e tutti i servizi in remoto, rendendosi disponibile ad eventuali richieste da parte degli utenti. |

**Terminazione del sistema**

È possibile terminare il sistema se e solo se tutti i sottosistemi sono stati disattivati in precedenza. Prima della disattivazione totale del database-server e dell’application-server verranno disconnessi tutti i client connessi al sistema

|  |  |
| --- | --- |
| **Nome** | Shutdown Server |
| **Partecipanti** | Amministratore |
| **Flusso di eventi** | |
| **Attore:** | Decide di arrestare il server, accede al pc dove risiede il server e clicca “Arresta” |
| **Sistema:** | Ricevuta la richiesta, termina tutte le connessioni attive. Successivamente termina tutte le operazioni in corso e il server. |

**Fallimento del sistema**

Nel caso si verifichi un errore dovuto all’hardware o al software si cercherà di ripristinare una configurazione del sistema precedente allo stato d’errore. Poiché i dati sono gestiti dal DBMS non c’è alcun rischio di perderli. Tuttavia, non è da escludere la perdita dei dati, se si verifica un guasto al supporto di memorizzazione dei dati nel database-server. Per minimizzare questo rischio, si eseguiranno periodicamente dei backup del database del sistema e periodicamente il sistema hardware verrà sottoposto a controlli.

# Subsystem services