**System Design Document**

Sommario

[**1.** **Introduction** 2](#_Toc25827561)

[**1.1** **Purpose of the system** 2](#_Toc25827562)

[**1.2** **Design goals** 3](#_Toc25827563)

[**1.3 Definition, acronyms and abbrevations** 3](#_Toc25827564)

[**1.4 References** 3](#_Toc25827565)

[**2.** **Current software architecture** 3](#_Toc25827566)

[**3.** **Proposed software** 3](#_Toc25827567)

[**3.1 Subsystem decomposition** 3](#_Toc25827568)

[**3.2 Hardware/software mapping** 3](#_Toc25827569)

[**3.3 Persistent data management** 3](#_Toc25827570)

[**3.4 Access control and security** 3](#_Toc25827571)

[**3.5 Global software control** 3](#_Toc25827572)

[**3.6Boundary condition** 3](#_Toc25827573)

[**4.** **Subsystem services** 3](#_Toc25827574)

# **Introduction**

## **Purpose of the system**

Lo scopo del progetto Emme Shop è quello di sviluppare una piattaforma software per la creazione e gestione del proprio sito e-commerce, contemporaneamente permette all’acquirente di trovare in un unico luogo varie proposte di negozi. La piattaforma dovrebbe essere semplice e intuitivo, per utenti inesperti grazie alla guida del sistema.

Per rendere Emme shop raggiungibile dal maggior numero di utenti, deve essere disponibile su un server online, visualizzabile su tutti i browser, sia personal computer che smartphone o smart tv….

Gli utenti accederanno alla piattaforma tramite un dominio.

## **Design goals**

Gli obiettivi di progettazione rappresentano le qualità desiderate della piattaforma EmmeShop e forniscono una serie coerente di criteri che devono essere considerati quando si prendono decisioni di progettazione. Sono stati indentificati i seguenti obiettivi di progettazione.

-Il sistema si appoggia ad un DBMS esterno, sfruttando al massimo le potenzialità di ottimizzazione per il recupero dei dati.

-usando modello MVC: abbiamo 3 diversi tipi 1) Model: responsabile del dominio del problema

2) Control: responsabile della sequenza di iterazioni con l’utente e le notifiche di visualizzazione delle modifiche nel model. Determina esplicitamente il flusso di controllo.

3) View: responsabile della visualizzazione degli oggetti per l’utente del dominio dell’applicazione.

-Linguaggi di programmazioni usati:

front-end: Html per la formattazione delle modalità di paginazione o visualizzazione grafica del contenuto, testuale e non, di una pagina web attrverso tag di formattazione, e anche per la supporto di inserimento di script e oggetti esterni;

JavaScript e JQuery rende molto più semplici le operazioni di spostamento e manipolazione dei documenti HTML, gestione degli eventi, animazione e Ajax con un’API di facile utilizzo che funziona su una moltitudine di browser;

CSS per la formattazione e di documenti HTML.

back-end: Java viene scelto per interoperabilità di interfacciarsi, scambiare informazioni o servizi con altri sistemi o prodotti in maniera più o meno completa e priva di errori, con affidabilità e con ottimizzazione delle risorse.

- **Efficienza:** Emme shop risposta all’autorizzazione entro 10 secondi, nel 95% dei casi. Qualora ci siano problemi di connettività, il sistema deve comunque garantire la registrazione in locale del pagamento, in modo da permettere il normale acquisto nei negozi. Supporto di transazioni, con caratteristiche ACID per gli acquisti.

- **Interfaccia intuitiva:** l’interfaccia di Emme Shop dovrebbe essere semplificata, per permettere all’utente di velocizzare le proprie operazioni. L’interfaccia utente su basa su una finestra principale, che include una barra di campi per accedere con il proprio account (dipende se si è un venditore o cliente), oppure di registrarsi alla piattaforma.

- **Multilingua:** nel sito è disponibile una funzionalità che in base alle impostazioni di sistema traduce nella propria lingua.

-**Robustezza**: tutte le interazioni tra utente e sistema devono avvenire in modo tale che utente non possa inserire dati non validi.

-**Scalabilità**: in termini di utenti contemporaneamente gestiti dal sistema

## **1.3 Definition, acronyms and abbreviations**

## **1.4 References**

[Problem Statement](PS_EmmeShop.docx)

[Requirements Analysist Document](Requisiti%20e%20casi%20d'uso%20EmmeShop.versione%20incompleta.docx)

<http://java.sun.com>, <https://jquery.com/> , <https://www.javascript.com/>

# **Current software architecture**

Poiché il progetto EmmeShop è un progetto di ingegneria Greenfield, non abbiamo software da sostituire.

# **3. Proposed software**

## **3.1 Subsystem decomposition**

Durante la decomposizione in sottosistemi, dividiamo in sottosistemi più piccoli con una forte coerenza e basso accoppiamento.

## 

## **3.2 Hardware/software mapping**

## **3.3 Persistent data management**

I dati devono essere distribuiti, garantendo che la distribuzione sia trasparente, cioè gli utenti siano in grado di interagire con il sistema come se fosse un unico sistema logico. Ma anche dal punto di vista delle prestazioni per quanto riguarda i metodi di accesso. Inoltre le transazioni devono essere trasparenti, ogni transazione deve garantire integrità del database in tutti gli altri database.

Non sono previste estendibilità del database in immediato futuro.

La stima di frequenza di accesso al database è costante ad ogni operazione da parte di un utente, dal fatto che il sistema sviluppato è basato su letture, scritture e modifiche nel database.

Le dimensioni delle tipiche richieste effettuate sono di estrapolare in media una tupla. Nel peggiore dei casi, la query che preleva tutti i prodotti relativi ad un determinato negozio. La percentuale di questa richiesta è molto ridotta, per mezzo della piattaforma il venditore sarà motivato a suddividere i prodotti del negozio in categorie specifiche. Questo facilita anche da parte del cliente a vedere in dettaglio i prodotti in un tempo ristretto.

Identificazione di oggetti persistenti, EmmeShop si occupa di una serie di oggetti da essere memorizzati.

Dal Class Diagram sviluppato nel RAD, tutte le classi individuate devono essere rese persistenti.

Immagine che contiene testo, mappa

Descrizione generata automaticamente

## 

## **3.4 Access control and security**

EmmeShop è una piattaforma online distribuita su un server, che ha diversi utenti, specializzati in Venditore e Cliente, i quali devono avere meccanismi di autenticazione.

Lo schema di autenticazione offerto è dato da tipo di utente (venditore, cliente) e dalla coppia nome utente e password, che servono ad’ identificare l’utente

**Access Matrix**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Categoria** | **Negozio** | **Prodotto** | **Fattura** | **Carrello** | **Riferimento** |
| **Gestore** **Accounting** | view() | modifica()  view() | view() |  |  |  |
| **Venditore** | addCategoria  modifica()  view() | creaNegozio  modifica() | addProdotto  modifica()  delete() | view() |  | view() |
| **Cliente** | view() | view() | view()  addToCart() | view() | view()  modifica  acquista | view()  new() |
| **Gestore Spedizioni** |  |  |  | get() |  |  |

Quando un’utente effettua il login, tramite i campi username e password inseriti nell’area della piattaforma relativa a quel tipo di utente (venditore, cliente o gestore accounting). Il sistema valida i dati

## **3.5 Global software control**

## **3.6 Boundary condition**

# **4. Subsystem services**