*System*

*Design Document*

*Progetto*

*OctoPlus*



**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 22/11/2023 | **0.1** | **Prima stesura** | **Tutto il team** |
| 25/11/2023 | **0.2** | **Aggiunte Referenze a RNF** | **Donnarumma Salvatore** |

# Introduzione

# 1.1 Scopo del sistema

OctoPlus ha come obbiettivo primario di offrire una piattaforma e-commerce per l’acquisto di attrezzatura da scuba-diving e immersioni che sia all’altezza dei competitor esistenti.

* 1. Obbiettivi di progettazione

|  |  |  |  |
| --- | --- | --- | --- |
| Categoria | ID | Descrizione | Reference |
| Perfomarce | OP\_1Persistenza | Per garantire la persistenza dei dati relativi a utenti, prodotti e ordini effettuati, è necessario l’impiego di un database relazionale in verranno memorizzate queste informazioni. | RNF4.7 |
| Performance | OP\_2Capacity | Il sistema deve essere in grado di gestire picchi di utenza e deve essere in grado di lavorare correttamente anche con un afflusso medio-alto di utenti. | RNF4.8 |
| Performance | OP\_3LoadingTime | Il sistema non deve avere soffrire di caricamenti troppo lunghi per offrire una navigazione fluida e piacevole all’utente. Una pagina deve caricarsi entro un tempo massimo di 5 secondi. | RNF4.9 |
| Performance | OP\_4ResponseTime | Il sistema deve garantire un tempo di risposta agli input degli utenti di massimo 10 secondi. | RNF4.9 |
| Affidabilità | OP\_5Robustezza | Il sistema non deve permettere utilizzi pericolosi e impropri. | RNF4.10 |
| Affidabilità | OP\_6SistemSecurity | Il sistema deve disporre di meccanismi di protezione che non permettono ai comuni utenti di accedere alle aree personali dei gestori. | RNF4.6 |
| Affidabilità | OP\_7ErrorManagement | Il sistema deve fornire messaggi di errore chiari e suggerimento per aiutare gli utenti a risolvere eventuali problemi. | RNF4.5 |
| Affidabilità | OP\_8DataSecurity | Il sistema implementa meccanismi di crittografia dei dati sensibili degli utenti per la loro protezione. | RNF4.1 |
| End User | OP\_9FriendlyUsage | Il sistema offre un’interfaccia intuitiva con struttura di navigazione chiara. Consente agli utenti di trovare rapidamente i prodotti desiderati offrendo una ricerca mirata e filtrata e il carrello è accessibile in qualunque momento. | RNF4.2 |
| End User | OP\_10Accessibility | Il sistema è supportato dai dispositivi mobile e desktop garantendo la responsività del layout. | RNF4.3 |
| End User | OP\_11Compatibily | Il sistema deve garantire la compatibilità con diversi browser e sistemi operativi. | RNF4.4 |
| Mantenimento | OP\_12MVCModel | L’architettura del sistema deve essere ispirata al modello MVC per favorire la modularità, manutenibilità e estendibilità. Il sistema permetterà l’aggiunta di nuove funzionalità per mantenere il sito aggiornato e permettere un’esperienza migliore agli utenti. | RNF.11 |

* + 1. Trade-off

|  |  |
| --- | --- |
| Trade-off | Razionale |
| Tempo di distribuzione vs Funzionalità | Per ridurre al minimo i tempi di sviluppo ci limiteremo ad implementare solo le funzionalità a priorità elevata come definito nelle specifiche iniziali ai fini di rientrare nei tempi prestabiliti per il rilascio e con meno bug. |
| Costi di sviluppo vs Velocità | Si è disposti ad utilizzare API base di Java come JDBC impiegando l’uso delle PreparedStatement invece delle Statement standard per migliorare la velocità esecutiva delle query. Tuttavia la scrittura del codice JDBC è più verbosa rispetto ad API o framework più astratti, portando potenzialmente a costi di sviluppo maggiori. |
| Velocità vs Spazio | Nelle operazioni che non dipendono dalle API esterne, per rispettare il vincolo dei tempi di risposta si è disposti ad occupare più spazio di memoria per aumentare la velocità. |

* 1. Riferimenti
* Requirements Analysis Document (RAD)

1. Architettura del sistema corrente
2. Architettura Proposta

3.1 Overview

1. Il sistema si basa sul modello MVC, presentato come una architettura suddivisa in tre layer principali inerenti a data, presentation e business logic. Il modello MVC è un caso speciale di Architettura Repository: il sottosistema Model implementa la struttura dati centrale, il sottosistema Controller gestisce esplicitamente il flusso di controllo e la View visualizza il modello. L'uso di un'architettura MVC è particolarmente vantaggioso per applicazioni complesse come i siti e-commerce, in quanto offre una struttura organizzata e scalabile per gestire la complessità crescente dell'applicazione nel tempo. L’uso di questa architettura facilita il lavoro del team di sviluppatori, consentendo loro di concentrarsi su aree specifiche dell’applicazione senza interferire troppo con altre parti del sistema.

3.2 Scomposizione in sottosistemi

Di seguito tutti i sottosistemi individuati con relative funzionalità offerte:

* **View**: sottosistema che si occupa della gestione delle interfacce utente. Rappresentano le componenti che permettono l’interazione degli utenti con il sistema.
* **GestioneUtenti**: fornisce le funzionalità di registrazione e autenticazione di un Utente/Admin e modifica dei dati personali.
* **GestioneProdotti**: fornisce le funzionalità per la gestione dei prodotti del catalogo.
* **GestioneAdmin**: fornisce le funzionalità per la gestione dei profili admin.
* **GestioneOrdini**: fornisce la funzionalità per la gestione degli ordini.
* **Storage**: sottosistema che si occupa della gestione dei dati persistenti tramite l’uso di un database.

3.3 Mapping Hardware e Software

Il sistema che sarà sviluppato, trattandosi di un applicativo web, possiamo rappresentarlo

attraverso tre nodi:

* **PC**: dove sarà installato un web browser con il quale sarà possibile inoltrare richieste http al server su cui sarà installato l’applicativo.
* **WebServer Tomcat**: dove avverrà il deployment dell’applicativo. Qui il sistema interagisce con un DBMS mediante l’utilizzo di driver JDBC.
* **MySQL Database**: ambiente che ospiterà il database del sistema.

3.4 Gestione della persistenza

La gestione dei dati persistenti viene effettuata tramite un Database MySQL. I dati da rendere persistenti sono:

* Informazioni relative agli utenti che si registrano al sito [nome, cognome, email, password, numero telefono, tipo di utente].
* Informazioni relative ai prodotti che sono presentati nel catalogo [nome, categoria, prezzo, descrizione, foto, statistiche].
* Informazioni relative agli ordini effettuati dagli utenti [utente che ha effettuato l’ordine, data, stato, prezzo totale, indirizzo, data acquisto, data consegna]
* Informazioni relative alle modifiche effettuate dagli amministratori [timestamp della modifica, tipo di operazione].

3.5 Controllo degli accessi e sicurezza

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Oggetti/Attori | Utente | Gestore Admin | Gestore Ordini | Gestore Catalogo |
| Gestione Utenti | RegistrazioneUtente  AutenticazioneUtente  CambioPassword | AutenticazioneUtente  CambioPassword | AutenticazioneUtente | AutenticazioneUtente |
| Gestione Prodotti | AggiungereProdottoCarrello  EliminareProdottoCarrello  AcquistoProdotto  AumentareQntProdottoCarrello  DiminuireQntProdottoCarrello  RicercaProdotto |  |  | RimuoviProdottoCatalogo  AggiungiProdottoCatalogo  ModificaProdottoCatalogo  OrdinamentoProdotti |
| Gestione Admin |  | AggiuntaNuovoAdmin  CambioPasswordAdmin  RimozioneAdmin |  |  |
| Gestione Ordini | CreazioneOrdine |  | RimozioneOrdine  RicercaOrdine |  |
| View | VisualizzaOrdiniEffettuati  VisualizzaDatiPersonali  VisualizzaPaginaProdotto  VisualizzaCarrello  VisualizzaListaProdotti  VisualizzaDettagliOrdine  VisualizzaPaginaCheckout | VisualizzaUtenti | VisualizzaStoricoOrdine | VisualizzaCatalogo |
| Database |  |  |  |  |

3.6 Controllo global del software

Il nostro sistema avrà un controllo del flusso globale di tipo Event Driven. Essendo un applicazione web, sarà il Web Server ad occuparsi dello smistamento delle varie richieste HTTP verso delle apposite Servlet che si occuperanno di gestire la richiesta, interagire con le altre componenti del sistema ed elaborare una risposta.

3.7 Boundary Condition

Boundary condition analizzate:

* **Avvio del sistema**: il sistema viene avviato accedendo al Server Tomcat e al server MySQL. Viene successivamente stabilita una connessione al database mediante driver JDBC. Verifica la sanità dei dati persistenti e rende infine disponibili le sue funzionalità e servizi agli utenti.
* **Spegnimento del sistema**: il sistema verrà reso offline per permettere le operazioni di manutenzione. Il sistema verifica che non ci siano connessioni aperte verso l’esterno e in caso affermativo termina l’esecuzione dell’applicativo e verrà chiusa la connessione al database. L’orario designato per permettere la manutenzione sarà una fascia orario in cui si registreranno il minor numero di accessi al sito.