Università degli Studi di Salerno

Corso di Ingegneria del Software

System Design Document

ZyphyK Sport

Docente:

Prof. Andrea De Lucia

Tutor:

Gilberto Recupito

Partecipanti:

Consiglio Luigi

D'Antuono Francesco Paolo

Vitale Ciro

Partecipanti al progetto:

nome	matricola	e-mail
Consiglio Luigi (CL)	0512110485	l.consiglio1@studenti.unisa.it
D'Antuono Francesco Paolo (DFP)	0512109798	f.dantuono10@studenti.unisa.it
Vitale Ciro (VC)	0512110719	c.vitale55@studenti.unisa.it

Revision History

data	versione	descrizione	autore
07/12/2022	1.00	inizio stesura introduction	CL
08/12/2022	1.01	stesura design goals: usability, dependability	VC
08/12/2022	1.02	stesura design goals: performance, supportability	CL
09/12/2022	1.03	stesura design goals: trade-offs	DFP
09/12/2022	1.04	revisione design goals e trade-offs	Tutto il team
10/12/2022	1.05	inizio stesura proposed software architecture	DFP,VC
11/12/2022	1.06	stesura subsystem decomposition: component diagram	VC
12/12/2022	1.07	stesura subsystem decomposition: architecture diagram	CL
12/12/2022	1.08	stesura subsystem decomposition: deployment diagram	DFP
12/12/2022	1.09	revisione e modifica component diagram e architecture diagram	Tutto il team
15/12/2022	1.10	stesura persistent data management: EER	CL,DFP
15/12/2022	1.11	revisione EER	VC
16/12/2022	1.12	stesura access control and security	Tutto il team

17/12/2022	1.13	revisione matrice di accesso	CL
18/12/2022	1.14	stesura global software control	VC,CL
18/12/2022	1.15	stesura boundary condition	DFP
19/12/2022	1.16	revisione e modifica boundary condition	DFP,CL
23/12/2022	1.17	stesura subsystem services	Tutto il team
23/12/2022	2.00	revisione del documento	Tutto il team

SDD

Introduction

Purpose of the system

Lo scopo del sistema è sviluppare una web application, più specificatamente un'e-commerce di scarpe sportive, in grado di gestire il processo di acquisto da parte dei clienti.

Inoltre, abbiamo la necessità che il servizio sia raggiungibile in qualsiasi momento, su una variegata scelta di dispositivi e accessibile da più utenti contemporaneamente.

Design goals

Usability

- NFR_USA_01 (priorità: alta)
  - il sistema non dovrà permettere input sbagliati (es. dati con formato diverso da quello atteso);
- NFR_USA_02 (priorità: bassa)
  - un cliente deve poter concludere, in qualsiasi momento, un acquisto in al più tre passi;
- NFR_USA_03 (priorità: media)
  - il sistema deve permettere l'utilizzo della piattaforma in modo immediato, anche senza nessuna consultazione di documentazione da parte dell'utente.

Dependability

- NFR_DEPEN_01 (priorità: media)
  - robustness: il sistema non deve andare in crash in caso di input sbagliati (es. login con credenziali errate);
- NFR_DEPEN_02 (priorità: alta)
  - security:
    - il sito dovrà utilizzare il protocollo di trasferimento ipertestuale sicuro;
    - le password sono memorizzate in maniera crittografata sul database;
- NFR_DEPEN_03 (priorità: alta)
  - il sistema deve garantire la netta distinzione delle operazioni sulla base degli utenti che possono effettuarle.

Performance

- NFR_PERF_01 (priorità: alta)
    - il sistema dovrà essere provvisto di un'interfaccia grafica di tipo responsive per potersi adattare ad ogni tipo di schermo.

Supportability

- NFR_SUPP_01 (priorità: media)
    - portability: l'infrastruttura deve poter permettere facilmente la compatibilità con nuove piattaforme.

Trade-offs

Performance vs Supportability

Migliorare la performance del software potrebbe richiedere l'eliminazione di alcune funzionalità di supporto, come la generazione di report o la gestione degli errori. Al contempo, mantenere un elevato livello di supportability potrebbe richiedere una leggera riduzione delle prestazioni.

La scelta è di garantire maggiore performance a discapito di alcune funzionalità di supporto.

Dependability vs Usability

Un software altamente affidabile potrebbe avere un'interfaccia utente meno intuitiva o meno attraente, mentre un software facile da usare potrebbe avere un livello di affidabilità leggermente inferiore.

La scelta è di garantire maggiore affidabilità a discapito dell'usabilità.

Definitions, acronyms, and abbreviations

Di seguito l'elenco delle definizioni che, per semplicità, sostituiremo con acronimi:

Acronimo	Definizione
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
JSP	Java Servlet Page
AJAX	Asynchronous JavaScript And XML

SQL	Structured Query Language
RAD	Requirement Analysis Document
DBMS	DataBase Management System
JDBC	Java DataBase Connectivity
HTTPS	Hypertext Transfer Protocol
EER	Enhanced Entity Relationship
FR	Functional Requirements
NFR	Non-functional requirements

References

Di seguito una lista di riferimenti agli altri documenti del progetto a cui si fa riferimento:

- Problem Statement
- RAD
- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009

Overview

Sono state così definite tutte le informazioni derivate sia dell'intervista con il committente che dalla definizione dei requisiti non funzionali nel RAD.

La restante parte del documento contiene una descrizione dettagliata e approfondita delle architetture utilizzate.

Current software architecture

Non si ha a disposizione alcuna architettura attuale, siccome stiamo procedendo per un problema di ingegnerizzazione *greenfield*.

Però, possiamo effettuare il confronto con l'architettura di un'e-commerce esistente che ha avuto successo in questo ambito: *sportsshoes.com*, il quale ha un'interfaccia intuitiva e facile da usare, con foto e descrizioni dettagliate dei prodotti, ma non offre opzioni di pagamento sicure, il sito web non risulta affidabile e performante. Anch'esso si basa su un'architettura three-tier, la problematica in comune con il nostro sistema, che andremo ad affrontare, è la separazione dei moduli che deve essere ben definita.

Proposed software architecture

Overview

ZyphyK-Sport è un e-commerce di scarpe sportive al dettaglio, che permette ai clienti di acquistare e tenere traccia dei propri ordini.

Gli ordini e il catalogo saranno gestiti dai rispettivi addetti, con ruoli diversi.
I clienti possono navigare tra le diverse categorie di prodotto, come scarpe da corsa, scarpe da calcio, scarpe da basket o scarpe da ginnastica, e scegliere i modelli che preferiscono. Sarà possibile procedere al checkout, dei prodotti contenuti nel carrello, solo se si è autenticati come cliente; quest'ultimo potrà visionare lo stato di un ordine nella pagina dedicata.

Il sistema proposto è basato su un'architettura *three-tier*.

Per quanto riguarda il *front-end* (Presentation) sono utilizzati: HTML, CSS, JS, JQUERY, AJAX e JSP.

Invece, per quanto riguarda il *back-end* (Business) sono utilizzati: Java e Java Servlet.

Infine, per quanto riguarda la persistenza dei dati (Data Layer), sono utilizzati: database MySQL e SQL.

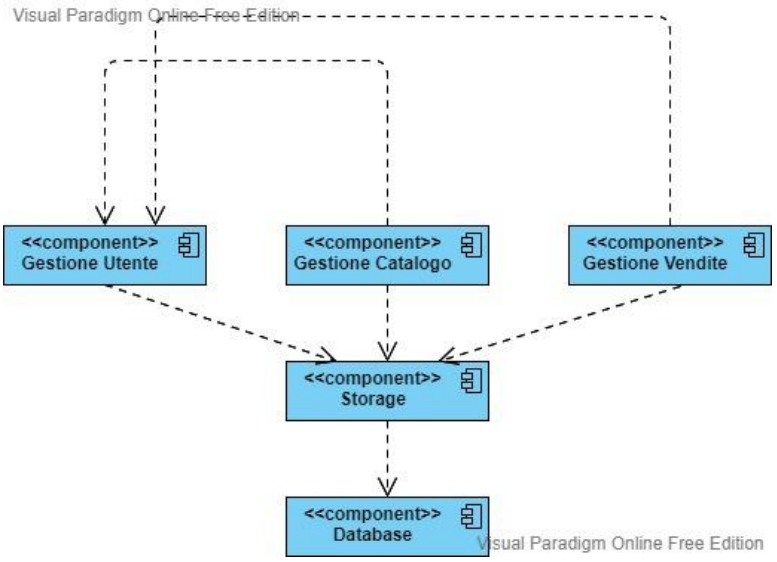
Subsystem Decomposition

Sono stati individuati i seguenti sottosistemi:

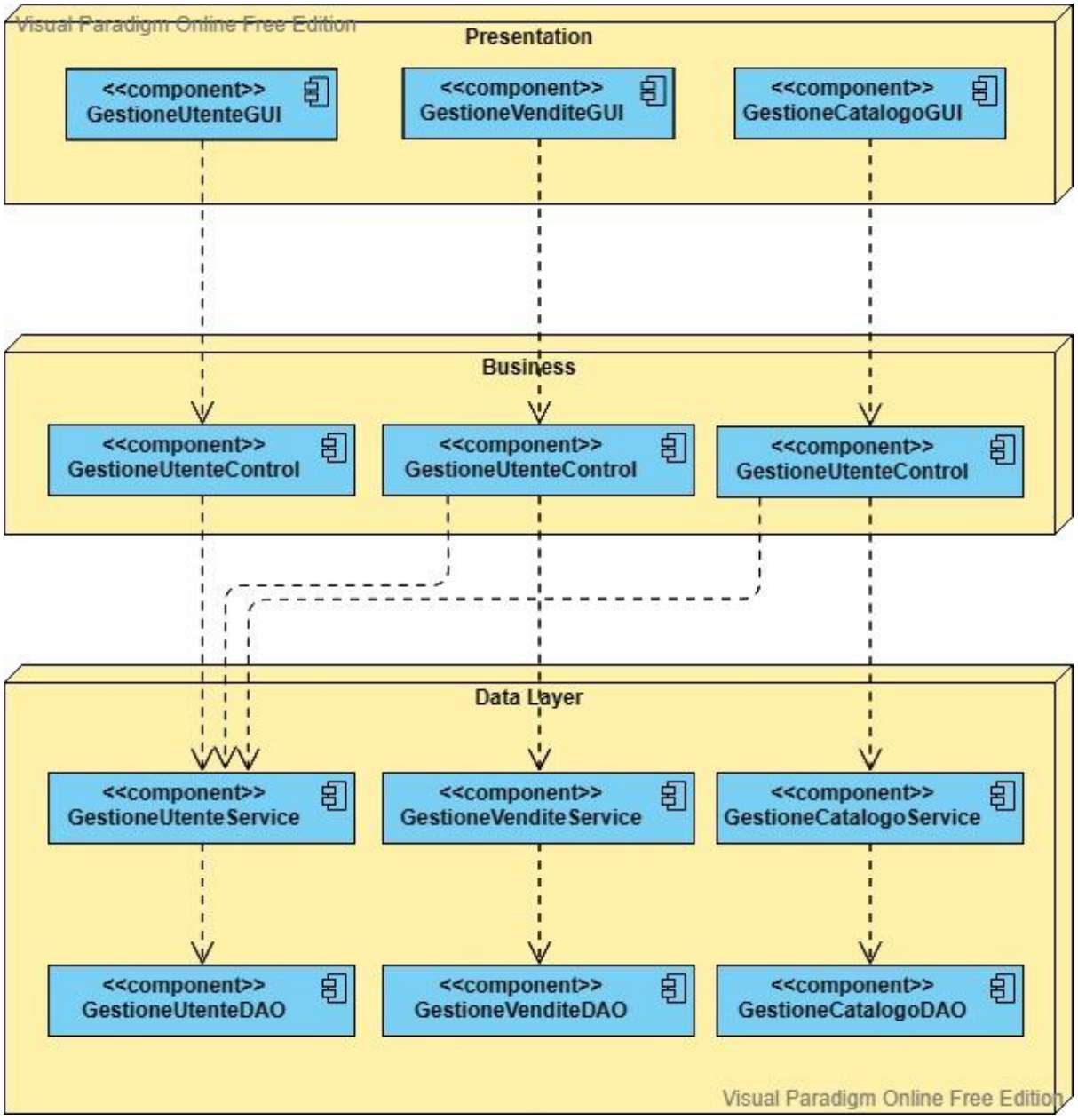
- *Gestione Utente*: gestisce l'autenticazione dei clienti e dei gestori e la registrazione dei clienti. Inoltre, si occupa anche delle funzionalità di logout.
- *Gestione Vendite*: per il gestore degli ordini si occupa della gestione degli ordini: visualizzazione di tutti gli ordini effettuati sul sistema e modifica dello stato di essi; invece, per quanto riguarda il cliente, gestisce il carrello: inserimento, modifica quantità e rimozione dei prodotti da esso, si occupa del processo di checkout, della creazione e visualizzazione degli ordini effettuati.

- *Gestione Catalogo*: per il gestore del catalogo si occupa della gestione dei prodotti: inserimento, modifica ed eliminazione; invece, per quanto riguarda il cliente e l'utente non loggato, questo sottosistema gestisce della visualizzazione del catalogo e dei prodotti.
- *Storage*: ciò che mette in collegamento i sottosistemi e il sottosistema *Database*.
- *Database*: gestisce la persistenza dei dati con un database relazionale.

Component Diagram



Architecture Diagram



Hardware/Software Mapping

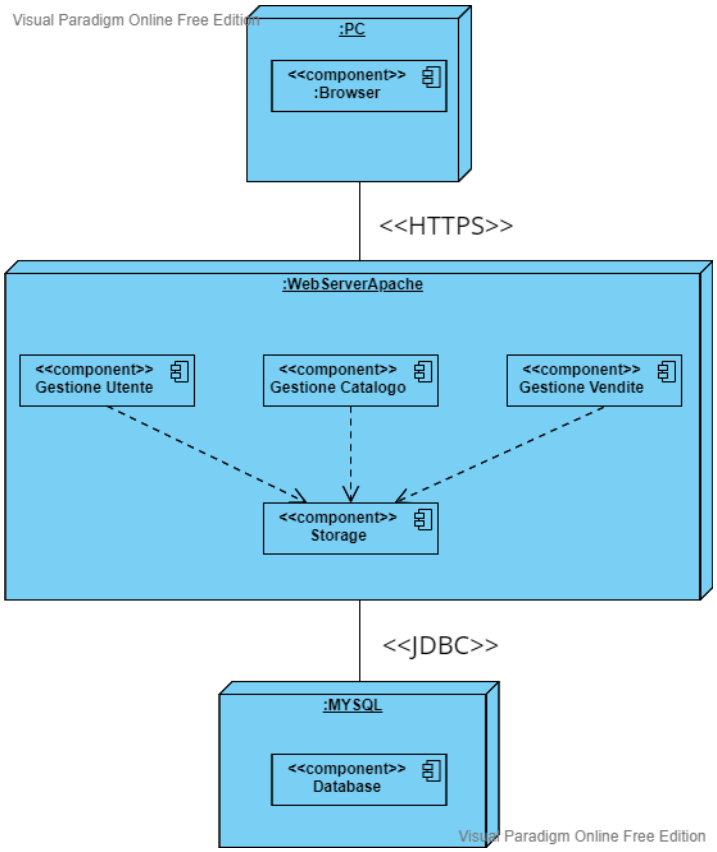
Il sistema si baserà su un'architettura three-tier, nel quale sono classificati tre tipi differenti di layer: *business layer*, *presentation layer* e *data layer*.

Il *Client* consiste di un dispositivo (mobile o fisso) che fornisce all'utente l'interfaccia grafica del sistema, utilizzando un browser, tramite il quale si inoltrano le richieste, mediante il protocollo *https*, al *Server*.

Il *Web Server Apache* riceve le richieste dal *Client* e le processa, mediante richieste, tramite *JDBC*, al DBMS *MySQL*.

È stata scelta quest'architettura poiché effettua separazione dei concetti, garantendo modifiche indipendenti e facilita la fase di testing.

Deployment Diagram



Persistent Data Management

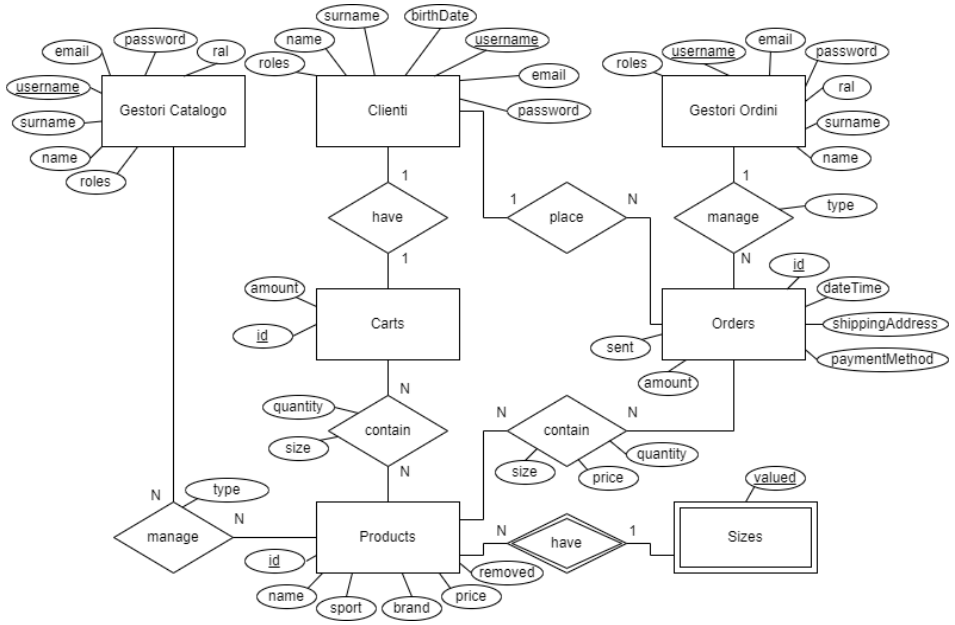
Per gestire i dati persistenti, il sistema utilizzerà un database relazionale, che garantisce accesso e utilizzo dei dati facile, sicuro e consistente, utilizzando un *DBMS*.

Il DBMS scelto è *MySQL*, per i seguenti motivi:

- *Sicurezza*: è possibile limitare l'accesso ai dati mediante l'assegnazione di autorizzazioni.
- *Elevate prestazioni*: noto per le elevate prestazioni e scalabilità, il che lo rende adatto a gestire grandi quantità di dati.
- *Amplia compatibilità*: possibilità di integrazione con varie tipologie di linguaggi di programmazione e tecnologie.
- *Transazioni*: consente di effettuare operazioni atomiche. Un'operazione atomica, consiste in un'operazione di esecuzione indivisibile dal punto di vista logico cioè o viene eseguita tutta o non viene eseguita.
- *Consistenza dei dati*: le modifiche effettuate sui dati non alterano la coerenza di questi ultimi.

Per quanto riguarda la definizione delle entità e delle relazioni del database, utilizziamo il modello EER che ci permette di costruire un modello di alto livello per un DB relazionale. L'utilizzo di un EER diagram invece di un class diagram per la definizione delle entità e delle relazioni del database consente una maggiore precisione, la possibilità di identificare e risolvere i problemi di normalizzazione e una migliore comprensione delle relazioni tra le entità e delle proprietà delle entità.

Enhanced Entity Relationship



Access Control and Security

Matrice di Accesso

Oggetti\Attori	Utente Non Loggato	Cliente	Gestore Ordini	Gestore Catalogo
GestoreCatalogo	login()			viewProfile() logout()
GestoreOrdini	login()		viewProfile() logout()	
Cliente	signUp() login()	viewProfile() logout()		
Cart		viewCart() removeFromCart() modQuantityFromCart() empty() purchase()		
Order		viewOrderDetails()	setSent()	
Product	viewProduct() selectSize()	viewProduct() selectSize() addToCart()		viewProduct() addToCatalog() removeFromCatalog() modifyFromCatalog()

Global Software Control

Il sistema prevede un controllo del software globale centralizzato di tipo event-driven, in cui c'è un componente centrale che gestisce tutte le attività del sistema in base agli eventi che si verificano. Gli eventi sono interazioni dell'utente con l'interfaccia grafica mediante, ad esempio, input da tastiera o da mouse. Utilizziamo questo tipo di controllo del software globale poiché, il componente centrale gestisce gli eventi, rendendo più semplice per gli altri componenti del sistema concentrarsi sulla loro funzionalità specifica. Inoltre, esso può gestire anche gli errori in modo centralizzato, il che può rendere più facile risolvere i problemi e minimizzare gli effetti sull'intero sistema.

Boundary Condition

Primo avvio

Per iniziare, è necessario avviare un web server che fornirà il servizio di un database MySQL per gestire i dati persistenti e eseguire il codice lato server. Dopo l'avvio, verrà visualizzata la pagina iniziale del sistema, che permetterà di accedere all'area di login utilizzando le credenziali appropriate (username e password). Una volta effettuato l'accesso, gli utenti potranno utilizzare tutte le funzionalità del sistema disponibili per il loro tipo di utente.

Avvio successivo a un fallimento

In caso di interruzione inaspettata dell'alimentazione, non sono previsti metodi per ripristinare il sistema al suo stato precedente allo spegnimento. In caso di crash del software a causa di errori commessi durante la fase di implementazione, non sono previste misure correttive; l'unico processo possibile sarà chiudere il sistema e riavviarlo.

Chiusura

Per chiudere l'applicazione, sarà sufficiente spegnere il DBMS e il server web. Verrà garantita la consistenza dei dati, annullando eventuali operazioni ancora in corso.

Subsystem services

Servizio	Descrizione
login()	consente di autenticarsi al sistema
signUp()	consente di registrarsi come Cliente
viewProfile()	consente di visualizzare il proprio profilo
logout()	consente di effettuare il logout dal sistema
viewCart()	consente di visualizzare il proprio carrello
addToCart()	consente di aggiungere un prodotto al proprio carrello
removeFromCart()	consente di rimuovere un prodotto dal proprio carrello
modQuantityFromCart()	consente di modificare la quantità di un prodotto nel carrello
empty()	consente di svuotare il carrello
purchase()	consente di effettuare l'acquisto dei prodotti contenuti nel carrello
viewOrderDetails()	consente di visualizzare i dettagli di un ordine
setSent()	consente di modificare lo stato di spedizione di un ordine
viewProduct()	consente di visualizzare i prodotti del catalogo
selectSize()	consente di selezionare la taglia del prodotto
addToCatalog()	consente di aggiungere un prodotto al catalogo
removeFromCatalog()	consente di rimuovere un prodotto dal catalogo
modifyFromCatalog()	consente di modificare un prodotto del catalogo