

**Object Design Document**

[Versione 1]

**Sommario**

[**Informazioni sul documento** 2](#_Toc159872519)

[**Generalità** 2](#_Toc159872520)

[**Team Project** 2](#_Toc159872521)

[**Revision History** 3](#_Toc159872522)

[**1. Introduzione** 4](#_Toc159872523)

[**1.1 Object design trade-offs** 5](#_Toc159872524)

[**1.2 Linee guida per la documentazione dell’interfaccia** 6](#_Toc159872525)

[**1.3 Definizioni, acronimi e abbreviazioni** 6](#_Toc159872526)

[**1.4 Riferimenti** 6](#_Toc159872527)

[**2. Packages** 6](#_Toc159872528)

[**3. Class interfaces** 6](#_Toc159872529)

[**4. Design patterns** 6](#_Toc159872530)

[**5. Glossario** 6](#_Toc159872531)

**Informazioni sul documento**

**Generalità**

* **Progetto**: TechHeaven
* **Versione**: [Versione 1]
* **Documento**: Documento di object design
* **Data**: [26/02/2024]

**Team Project**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome Membro | Matricola | Ruolo | Contatti |
| Dorotea Serrelli | 0512113740 | Project manager | d.serrelli1@studenti.unisa.it |
| Raffaella Sabatino | 0512115114 | Team member | r.sabatino17@studenti.unisa.it |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 26/02/2024 | 0.1 | Stesura dell’introduzione al documento ODD e definizione dei trade-off e dei object design goals | Dorotea Serrelli |

**1. Introduzione**

La società TechHeavenSrl è responsabile dall’anno 2000 della gestione del negozio “TechHeaven – Il paradiso digitale”, specializzato nella vendita di prodotti elettronici, elettrodomestici, telefonia.

Tale negozio è, attualmente, un punto vendita di riferimento nella zona per lo smercio, la qualità e il prezzo dei prodotti, al punto che soddisfa un grande bacino di utenza, quasi esclusivamente residente nella provincia.

La società intende espandere i confini della propria attività ed ampliare la clientela, avvalendosi di un sistema software che consenta, sotto il profilo soggettivo, una maggiore conoscibilità della società e dell’affidabilità della stessa; sotto il profilo oggettivo, favorisca l’incremento della vendita dei prodotti.

Il sistema software verrà sviluppato per fornire alla clientela informazioni sulla società, sul punto vendita e sui prodotti trattati.

La piattaforma permetterà, infatti, al cliente di registrarsi, in modo da poter visionare i prodotti in vendita ed acquistarli, tenere traccia dello stato degli ordini effettuati presso il negozio online, creare una lista di prodotti desiderati (wishlist).

La piattaforma, inoltre, consentirà l’accesso ai seguenti dipendenti:

* Gestore degli ordini: responsabile del processo di acquisizione, registrazione ed evasione degli ordini dei clienti, nonché dell’elaborazione di richieste di approvvigionamento di prodotti da inoltrare, poi, all’ufficio acquisti.
* Gestore del catalogo: responsabile della presentazione, organizzazione e gestione del catalogo dei prodotti venduti dal negozio.

La piattaforma, quindi, consentirà al gestore degli ordini di visionare gli ordini commissionati dai clienti al negozio e gli ordini che sono stati spediti, preparare un ordine alla spedizione e fare richiesta di approvvigionamento di prodotti mancanti.

Essa, inoltre, permetterà al gestore del catalogo di visionare il catalogo e di poter inserire, cancellare e modificare un prodotto nel catalogo.

In questo documento verranno descritti i compromessi di progettazione degli oggetti effettuati, le linee guida seguite per le interfacce dei sottosistemi - riguardanti la nomenclatura, la documentazione e le convenzioni sui formati -, la decomposizione dei sottosistemi in packages e classi e le interfacce delle classi.

**1.1 Object design trade-offs**

Nella fase di progettazione degli oggetti del sistema si sono analizzati i seguenti trade-offs:

* **Spazio di memoria vs Tempo di risposta**

Maggiore spazio di memoria significa archiviare più dati, come la cronologia degli ordini, i dettagli dei prodotti e le informazioni sui clienti.

Privilegiare questo aspetto migliorerebbe l'esperienza utente e la funzionalità del software, ma può anche aumentare i costi e la complessità.

Un tempo di risposta rapido è fondamentale per un'esperienza utente fluida e per l'efficienza dell’esecuzione delle funzionalità offerte dal sistema. L'ottimizzazione del tempo di risposta può richiedere la memorizzazione nella cache di dati e l'utilizzo di algoritmi efficienti.

Visto che un tempo di risposta rapido è cruciale per la soddisfazione del cliente, la gestione fluida del processo di evasione degli ordini e la competitività del negozio online, si intende privilegiare il tempo di risposta.

* **Leggibilità vs Costi**

Un aspetto importante da prendere in considerazione nel nostro sistema è la leggibilità del codice: è da tenere conto che non necessariamente coloro che faranno manutenzione o monitoraggio del sistema saranno i creatori del sistema stesso.

Pertanto, si vorrà garantire la leggibilità del codice utilizzando commenti e documentazione dei vincoli di implementazione delle interfacce e classi coinvolte nel sistema, anche se ciò comporterà un aumento dei costi e del tempo di sviluppo.

* **Sicurezza vs Prestazioni**

Visto che il cliente sottolinea la necessità dello sviluppo di meccanismi di protezione agli attacchi informatici SQLInjection e Cross-site scripting e di integrità e riservatezza dei dati scambiati tra client e server, si intende privilegiare il requisito di sicurezza, a discapito del livello di prestazioni elevate che potenzialmente raggiungerebbe il sistema.

Tale decisione farà in modo che verrà garantito il livello di prestazioni nel tempo di risposta delineato nel primo trade-off discusso in questo paragrafo.

A seguire i trade-offs, si intende raggiungere i seguenti obiettivi:

* **Robustezza** : il sistema deve reagire correttamente a situazioni impreviste, attraverso il controllo degli errori e la gestione delle eccezioni.
* **Incapsulamento** : si vogliono nascondere i dettagli implementativi delle classi grazie all’utilizzo delle interfacce, rendendo possibile l’utilizzo di funzionalità offerte da diversi componenti o layer sottoforma di black-box.

**1.2 Linee guida per la documentazione dell’interfaccia**

**1.3 Definizioni, acronimi e abbreviazioni**

**1.4 Riferimenti**

**2. Packages**

**3. Class interfaces**

In questa sezione sono illustrate le interfacce di ciascun package, eccetto il package Storage e le classi control.

**Package Registrazione**

|  |  |
| --- | --- |
| Nome classe | RegistrazioneService |
| Descrizione | Si occupa di gestire la registrazione dei nuovi clienti del negozio. |
| Metodi | +registraCliente(username: String, password: String, nome: String, cognome: String, sesso: Sex, email: String, telefono: String, indirizzoSped: Indirizzo) : Utente |
| Invariante di classe | Nessuno |
| Nome metodo | +registraCliente(username: String, password: String, nome: String, cognome: String, sesso: Sex, email: String, telefono: String, indirizzoSped: Indirizzo) |
| Descrizione | Il metodo permette di registrare un nuovo utente nel sistema (con ruolo Cliente) con le seguenti informazioni : username, password, nome, cognome, sesso, email, numero di telefono ed indirizzo di spedizione. |
| Pre-condizione | **context** RegistrazioneService:: registraCliente(username: String, password: String, nome: String, cognome: String, sesso: Sex, email: String, telefono: String, indirizzoSped: Indirizzo) **pre**:  Utente.checkValidate(username: String, password: String, nome: String, cognome: String, sesso: Sex, email: String, telefono: String, indirizzoSped: Indirizzo) == true |
| Postcondizione | **context** RegistrazioneService::registraCliente(c:Utente) **post**:  UtenteDAO.save(c) == true |

**Package Autenticazione**

|  |  |
| --- | --- |
| Nome classe | AutenticazioneService |
| Descrizione | Si occupa di gestire l’autenticazione al sistema ad unico fattore del cliente, del gestore degli ordini e del gestore del catalogo; inoltre, si occupa delle funzionalità relative alla gestione dell’area riservata, delle informazioni personali del cliente, della reimpostazione della password e del cambio di ruolo del gestore del catalogo e del gestore degli ordini. |
| Metodi | +login(String username, String password) : Utente  +login(String username, String password, String ruoloScelto) : Utente  +logout(Utente user) : void  +cambioRuolo(Utente user, String ruoloRichiesto) : boolean  +visualizzaAreaRiservata(Utente user) : boolean  +modificaDatiPersonali(Utente user, String infoSelected, String updatedData) : Utente  +modificaIndirizzoPersonale(Utente user, String infoSelected, Indirizzo address) : Utente  +reimpostazionePassword(String username, String email) : boolean |
| Invariante di classe | Nessuno |
| Nome metodo | +login(String username, String password) : Utente |
| Descrizione | Il metodo permette di autenticare un nuovo utente nel sistema con un solo ruolo. |
| Pre-condizione | Nessuna |
| Postcondizione | **context** RegistrazioneService::registraCliente(c:Utente) **post**:  UtenteDAO.save(c) == true |
|  | +login(String username, String password, String ruoloScelto) : Utente |
|  |  |
|  |  |
|  |  |
|  | +logout(Utente user) : void |
|  |  |
|  |  |
|  |  |
|  | +cambioRuolo(Utente user, String ruoloRichiesto) : boolean |
|  |  |
|  |  |
|  |  |
|  | +visualizzaAreaRiservata(Utente user) : boolean |
|  |  |
|  |  |
|  |  |
|  | +modificaDatiPersonali(Utente user, String infoSelected, String updatedData) : Utente |
|  |  |
|  |  |
|  |  |
|  | +modificaIndirizzoPersonale(Utente user, String infoSelected, Indirizzo address) : Utente |
|  |  |
|  |  |
|  |  |
|  | +reimpostazionePassword(String username, String email) : boolean |
|  |  |
|  |  |
|  |  |

**4. Design patterns**

**5. Glossario**