**FOODART**

UNIVERSITÀ DEGLI STUDI DI SALERNO  
CORSO DI LAUREA TRIENNALE IN INFORMATICA  
CORSO DI INGEGNERIA DEL SOFTWARE  
ANNO ACCADEMICO 2021/2022





Immagine che contiene cibo, verdura, tavolo, frutta

Descrizione generata automaticamente

Immagine che contiene erba, esterni, pianta, albero

Descrizione generata automaticamente

OBJECT DESIGN   
DOCUMENT:  
versione 1.0

**TOP MANAGER**

|  |
| --- |
| **PROFESSORE** |
| Prof. De Lucia Andrea |
| **TUTOR** |
| Iannone Emanuele |

# **PARTECIPANTI**

|  |  |
| --- | --- |
| **NOME E COGNOME** | **MATRICOLA** |
| Davide Amitrano | 0512106034 |
| Donato Miranda | 0512106148 |
| Alfonso Zappia | 0512106076 |

# **STORICO DELLE VERSIONI**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA** | **VERSIONE** | **DESCRIZIONE** | **AUTORI** |
| 13/10/2021 | 1.0 | Creazione del documento, stesura di una piccola introduzione, descrizione del modello di sistema, con scenari e casi d’uso. | Tutti |
| 14/10/2021 | 1.1 | Creazione del modello dinamico e di sistema, con una prima stesura dei sequence diagrams, statechart diagrams e tabelle degli oggetti | Tutti |
| 15/10/2021 | 1.2 | Creazione dei percorsi navigazionali e degli screen-mockups | Tutti |

# **INDICE**

1. Introduzione 4  
   1.1 Object Design Trade-Offs   
   1.2 Linee guida per l’interfaccia   
   1.3 Definizioni, acronimi e abbreviazioni  
   1.4 Riferimenti
2. Packages 6  
   2.1 control  
   2.2 model  
   2.3 service  
   2.4 view
3. Interfaccia delle classi 9

**1. INTRODUZIONE  
1.1 OBJECT DESIGN TRADE-OFFS**

**Funzionalità vs Usabilità**Si è deciso di dare priorità all’usabilità, dato che la nostra web-application dovrà essere quanto più immediata possibile, dato che abbraccerà molte tipologie diverse di utenti. Le funzionalità che non hanno una priorità alta potranno essere implementate in un secondo momento.

**Prestazioni vs Costi**Abbiamo realizzato il nostro sistema utilizzando materiale *open-source*, minimizzando i nostri costi e realizzando il massimo delle prestazioni possibili.

**Interfaccia vs Tempo di risposta**Il tempo di risposta tra server e interfaccia utente è abbastanza rapido da soddisfare le esigenze dei vari utente del nostro sito. C’è da dire che, però, maggiore saranno le dimensioni del nostro database, maggiore sarà il tempo di risposta.

**1.2 LINEE GUIDA PER L’INTERFACCIA**

Per realizzare un codice comprensibile, che sia fedele ai dettami della programmazione ad oggetti e che possa essere riutilizzato in futuro, rispettando la trasparenza sulla logica delle operazioni.

**1.2.1 Packages**  
Il codice verrà opportunamente diviso in packages, modellando efficacemente i sottosistemi. L’uso dei package servirà a minimizzare l’accoppiamento tra le classi, evidenziando però la coesione tra quelle che hanno comportamento simile. Ogni package verrà identificato dal suo nome, scritto secondo la seguente notazione, cioè *type\_name*, cioè riconoscendo la tipologia di pacchetto che accoppierà classi dal comportamento simile e, infine, il suo nome.

**1.2.2 Naming Convention**Per il nostro sistema abbiamo optato per utilizzare una naming convention precisa, che identifichino univocamente gli elementi al suo interno. Verranno usati nomi descrittivi, ma allo stesso tempo non particolarmente lunghi, che seguano la notazione CamelCase, cioè senza spazi e con le iniziali maiuscole. Questo varrà per i nomi delle classi e delle variabili.

**1.2.3 Costanti**Per il nostro sistema è previsto anche l’uso di costanti, che verranno scritte secondo la notazione UPPER\_CASE, cioè tutte lettere maiuscole e l’underscore a separare costanti composte da più parole.

**1.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI**

|  |  |
| --- | --- |
| **DENOMINAZIONE** | **DESCRIZIONE** |
| **FoodArt** | Nome del sistema. |
| **Utente generico** | Dicitura che viene utilizzata per definire diverse tipologie di utenti. |
| **Utente ospite** | Un qualsiasi utente che utilizza il sistema, senza essere però loggato. |
| **Cliente** | Un utente loggato che ha la possibilità di acquistare prodotti all’interno del sito |
| **Prodotto** | L’insieme dei dati relativi ad un prodotto alimentare artigianale che viene inserito all’interno del sito. |
| **Rivenditore** | Un utente loggato che ha la possibilità di vendere i propri prodotti all’interno del sito. |
| **Amministratore** | Un utente loggato che ha la responsabilità di gestire le approvazioni e di controllare l’andamento del sito. |
| **DBMS** | *Database Management System*, un sistema software per la gestione di database. |
| **MySQL** | Un DBMS per la gestione di database relazionali. |
| **HTML** | *HyperText Markup Language*, linguaggio di markup per realizzare pagine web. |
| **JSP** | *JavaServer Pages*, insieme di tag all’interno di una pagina HTML in cui può essere integrato del codice Java. |
| **JS** | *JavaScript*, linguaggio di programmazione utilizzato per la creazione di effetti dinamici e interattivi all’interno delle pagine web. |
| **CSS** | *Cascading Style Sheets*, insieme di direttive che permettono di definire la formattazione di pagine web. |
| **HTTPS** | *HyperText Transfer Protocol over Secure Socket Layer*, protocollo per la comunicazione sicura attraverso la rete. |
| **MVC** | *Model-view-controller*, pattern architetturale in grado di separare la logica di presentazione dei dati dalla logica di business. |
| **Apache TomCat** | Un server web che fornisce una piattaforma per l’esecuzione di applicazioni web in Java. |
| **RAD** | *Requirements Analysis Document*, documento redatto per la descrizione dei requisiti di un progetto. |
| **SDD** | *System Design Document*, documento redatto per la descrizione del nostro sistema. |
| **PS** | *Problem Statement*, documento redatto per delineare punti negativi della situazione attuale e fornire una prima panoramica del sistema proposto. |
| **Servlet** | Oggetti scritti in linguaggio Java che operano all’interno di un server web. |
| **API** | *Application Programming Interface*, cioè un insieme di librerie software di un determinato linguaggio. |
| **JDBC** | *Java DataBase Connectivity*, un connettore per database che consente l’accesso ai dati persistenti da qualsiasi programma scritto in Java. |
| **Failure** | Fallimento, che può essere legato a problemi hardware o software. |

**1.4 RIFERIMENTI**

* PS
* RAD
* SDD
* Object-Oriented Software Engineering Using UML, Patterns, and Java by Bernd Bruegge and Allen H. Dutoit.
* Corso di Ingegneria del Software 🡪 <http://elearning.informatica.unisa.it/>

**2. PACKAGES**

**2.1 control**

|  |  |  |
| --- | --- | --- |
| **DENOMINAZIONE** | **SUBPACKAGE** | **DESCRIZIONE** |
| **LogoutControl** | *control.auth* | Gestisce il logout |
| **CartControl** | *control.cart* | Gestisce il carrello |
| **ShoppingCart** | *control.cart* | Gestisce la pagina del carrello |
| **CategoryControl** | *control.category* | Gestisce le categorie |
| **FeedbackControl** | *control.feedback* | Gestisce le recensioni |
| **ImageCategoryControl** | *control.image* | Gestisce le immagini delle categorie |
| **ImageProductControl** | *control.image* | Gestisce le immagini dei prodotti |
| **IndexControl** | *control.index* | Gestisce la homepage |
| **LoginControl** | *control.login* | Gestisce il login |
| **CompleteOrderControl** | *control.order* | Gestisce il completamento dell’ordine |
| **RecapOrderControl** | *control.order* | Gestisce il checkout |
| **SingleProductControl** | *control.product* | Gestisce la pagina del singolo prodotto |
| **RegisterControl** | *control.register* | Gestisce la registrazione |
| **SearchbarControl** | *control.searchbar* | Gestisce la barra di ricerca |
| **DashboardControl** | *control.dashboard* | Gestisce la dashboard |

**2.2 model**

|  |  |  |
| --- | --- | --- |
| **DENOMINAZIONE** | **SUBPACKAGE** | **DESCRIZIONE** |
| **AmministratoreBean** | *model.amministratore* | Memorizza i dati dell’amministratore |
| **AmministratoreDAO** | *model.amministratore* | Interfaccia per l’accesso ai dati dell’amministratore |
| **AmministratoreDAOImp** | *model.amministratore* | Permette l’accesso ai dati dell’amministratore |
| **CategoriaBean** | *model.categoria* | Memorizza i dati delle categorie |
| **CategoriaDAO** | *model.categoria* | Interfaccia per l’accesso ai dati della categoria |
| **CategoriaDAOImp** | *model.categoria* | Permette l’accesso ai dati della categoria |
| **FeedbackBean** | *model.feedback* | Memorizza i dati di una recensione |
| **FeedbackDAO** | *model.feedback* | Interfaccia per l’accesso ai dati di una recensione |
| **FeedbackDAOImp** | *model.feedback* | Permette l’accesso ai dati di una recensione |
| **ImmagineBean** | *model.immagine* | Memorizza i dati di un’immagine |
| **ImmagineDAO** | *model.immagine* | Interfaccia per l’accesso ai dati di un’immagine |
| **ImmagineDAOImp** | *model.immagine* | Permette l’accesso ai dati di un’immagine |
| **IndirizzoConsegnaBean** | *control.indirizzoConsegna* | Memorizza i dati di un indirizzo |
| **IndirizzoConsegnaDAO** | *control.indirizzoConsegna* | Interfaccia per l’accesso ai dati di un indirizzo |
| **IndirizzoConsegnaDAOImp** | *control.indirizzoConsegna* | Permette l’accesso ai dati di un’immagine |
| **MetodoPagamentoBean** | *model.metodoPagamento* | Memorizza i dati di una carta |
| **MetodoPagamentoDAO** | *model.metodoPagamento* | Interfaccia per l’accesso ai dati di una carta |
| **MetodoPagamentoDAOImp** | *model.metodoPagamento* | Permette l’accesso ai dati di una carta |
| **OrdineBean** | *model.ordine* | Memorizza i dati dell’ordine |
| **OrdineDAO** | *model.ordine* | Interfaccia per l’accesso ai dati dell’ordine |
| **OrdineDAOImp** | *model.ordine* | Permette l’accesso ai dati dell’ordine |
| **ProdottoBean** | *model.prodotto* | Memorizza i dati del prodotto |
| **ProdottoDAO** | *model.prodotto* | Interfaccia per l’accesso ai dati del prodotto |
| **ProdottoDAOImp** | *model.prodotto* | Permette l’accesso ai dati del prodotto |
| **RivenditoreBean** | *model.rivenditore* | Memorizza i dati del rivenditore |
| **RivenditoreDAO** | *model.rivenditore* | Interfaccia per l’accesso ai dati del rivenditore |
| **RivenditoreDAOImp** | *model.rivenditore* | Permette l’accesso ai dati del prodotto |
| **UtenteBean** | *model.utente* | Memorizza i dati di un utente |
| **UtenteDAO** | *model.utente* | Interfaccia per l’accesso ai dati dell’utente |
| **UtenteDAOImp** | *model.utente* | Permette l’accesso ai dati dell’utente |
| **VoceBean** | *model.voce* | Memorizza i dati di una voce |
| **VoceDAO** | *model.voce* | Interfaccia per l’accesso ai dati della voce |
| **VoceDAOImp** | *model.voce* | Permette l’accesso ai dati della voce |

**2.3 service**

|  |  |  |
| --- | --- | --- |
| **DENOMINAZIONE** | **SUBPACKAGE** | **DESCRIZIONE** |
| **ProductItem** |  | Gestisce i prodotti nel carrello |
| **ShoppingCart** |  | Gestisce i metodi del carrello |

**2.4 view**

|  |  |
| --- | --- |
| **DENOMINAZIONE** | **DESCRIZIONE** |
| **index.jsp** | Pagina che contiene l’homepage del sito |
| **all\_category.jsp** | Pagina che mostra tutte le categorie |
| **category.jsp** | Pagina che mostra una singola categoria |
| **complete\_order.jsp** | Pagina che mostra un messaggio di ringraziamento dopo aver completato l’ordine |
| **login.jsp** | Pagina che mostra il form del login |
| **recap\_order.jsp** | Pagina che mostra il checkout dell’ordine |
| **register.jsp** | Pagina che mostra il form della registrazione |
| **search\_page.jsp** | Pagina che mostra i risultati della ricerca |
| **shopping\_cart.jsp** | Pagina che mostra il carrello |
| **single\_product.jsp** | Pagina che mostra il singolo prodotto |

**3. INTERFACCIA DELLE CLASSI**

|  |  |
| --- | --- |
| **Nome** | AmministratoreBean |
| **Descrizione** | Rappresenta un amministratore all’interno del sistema |
| **Attributi** | -idUtente: int -ruolo: String |
| **Signature dei metodi** | +getIdUtente()  +setIdUtente(int idUtente) +getRuolo()  +setRuolo(String ruolo) |
| **Pre-condizioni** | **context** AmministratoreBean::setRuolo(r) **pre**: p.*matches*(‘/^[A-Za-z]{0,50}$/’) |
| **Post-condizioni** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome** | CategoriaBean |
| **Descrizione** | Rappresenta una categoria all’interno del sistema |
| **Attributi** | -idCategoria: int -nome: String -pathName: byte[] |
| **Signature dei metodi** | +getIdCategoria()  +setIdCategoria(int idCategoria) +getNome()  +setNome(String nome) +getPathName() +setPathName(byte[] pathName) |
| **Pre-condizioni** | **context** CategoriaBean::setNome(n) **pre**: n.*matches*(‘/^[A-Za-z]{0,45}$/’) |
| **Post-condizioni** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome** | FeedbackBean |
| **Descrizione** | Rappresenta una recensione all’interno del sistema |
| **Attributi** | -idFeedback: int -titolo: String -commento: String -valutazione: float -idCommentatore: int -idProdotto: int -idRivenditore: int |
| **Signature dei metodi** | +getIdFeedback() +setIdFeedback(int idFeedback)  +getTitolo() +setTitolo(String titolo)  +getCommento() +setCommento(String commento)  +getValutazione()  +setValutazione(float valutazione)  +getIdCommentatore() +setIdCommentatore(int idCommentatore)  +getIdProdotto()  +setIdProdotto(int idProdotto)  +getIdRivenditore()  +setIdRivenditore(int idRivenditore) |
| **Pre-condizioni** | **context** FeedbackBean::setTitolo(t) **pre**: t.*matches*(‘/^[A-Za-z]{0,50}$/’)  **context** FeedbackBean::setCommento(c) **pre**: c.*matches* /[^A-Za-z0-9 .'?!,@$#-\_\n\r]{1, 65,535/’) **context** FeedbackBean::setValutazione(v) **pre**: v.*matches* (‘^[1-9][0-9]?$|^5$’) |
| **Post-condizioni** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome** | ImmagineBean |
| **Descrizione** | Rappresenta una recensione all’interno del sistema |
| **Attributi** | -idImmagine: int -pathName: byte[] -idProdotto: int |
| **Signature dei metodi** | +getIdImmagine() +setIdFeedback(int idFeedback)  +getPathName() +setPathName(byte[] pathName)  +getIdProdotto() +setIdProdotto(int idProdotto) |
| **Pre-condizioni** |  |
| **Post-condizioni** |  |
| **Invarianti** |  |