UNIVERSITÀ DEGLI STUDI DI SALERNO
Corso di Ingegneria Software

FATTORIA ROSABELLA
OBJECT DESIGN DOCUMENT
VERSIONE 1.1

17/01/2021

Progetto: FATTORIA ROSABELLA	Versione: 1.1
Documento: Object Design Document	Data: 17/01/2021

Coordinatore del progetto:

Nome	**Matricola**
Mignoli Pasquale Antonio	0512105826

Partecipanti:

Nome	**Matricola**
Mignoli Pasquale Antonio	0512105826
Pastore Carmine	0512106058
Laurino Nicola	0512105970

Scritto da:	Mignoli, Laurino, Pastore

Revision History

Data	**Versione**	**Descrizione**	**Autore**
22/12/2020	1.0	Prima stesura del documento	Pastore Carmine Mignoli Pasquale Laurino Nicola
17/01/2021	1.1	Aggiunta class diagram	Laurino Nicola

Sommario

1.0 Introduzione

1.1 Object Design trade-offs

Buy vs Build: con lo scopo di risparmiare il maggior tempo possibile, di presentare un prodotto valido e ridurre notevolmente i tempi di consegna , si è pensato di adottare collezioni di framework e librerie di terze parti reperibili attraverso il web. In questo caso si tende a riutilizzare componenti già pronti all'uso prestando attenzione alla scelta dei componenti che meglio si prestano allo scopo del progetto.

Memory Space vs Response Time: con lo scopo di offrire un tempo di risposta minore si è pensato di prediligerlo a sfavore dello spazio di memoria, al fine di offrire all'utente che utilizza il prodotto un sistema rapido da utilizzare. Si è scelto di spedire minor numero di dati possibili al fine di garantire un tempo di risposta minore.

1..1 Componenti off-the-shelf

Le componenti off-the-shelf sono componenti , nel nostro caso software, che vengono messe a disposizione da aziende terze, gratuitamente o a pagamento, per essere utilizzate nei propri progetti.

Le componenti utilizzate per il nostro prodotto sono le seguenti.

- Bootstrap: è una raccolta di strumenti liberi per la creazione di siti e applicazioni web. Fornisce modelli di progettazione basati su HTML, CSS, JavaScript per le componenti dell'interfaccia, come moduli, pulsanti, e layout. Bootstrap facilita il compito di progettazione del sito responsive, da desktop a mobile, riducendo notevolmente le ore di lavoro, fornendo già i moduli ottimizzati per l'operazione sopracitati. Inoltre, Bootstrap, fornisce componenti JavaScript utili per dare dinamicità all'interfaccia utente, mettendo a disposizione animazioni grafiche già pronte all'uso.
- Eclipse: IDE utilizzata per lo sviluppo del sistema ottimizzato per il linguaggio di programmazione JAVA .
- Libreria JSON di JavaScript: acronimo di JavaScript Object Notation, è un formato adatto all'interscambio dei dati fra applicazioni client-server. È basato sul linguaggio JavaScript, ma è indipendente da esso.
- Flaticon: è sito web che mette a disposizione una raccolta di icone, gratuite e non, in formato SVG e PNG.

Queste sono le componenti selezionate per lo sviluppo del progetto, in linea con la politica che punta all'ottimizzazione dei costi e del risparmio del maggior tempo possibile.

1.1.2 Design patterns

- Model-View-Controller (abbreviato spesso in MVC), che consiste nel separare i componenti software che implementano il modello delle funzionalità di business (model), dai componenti che implementano la logica di presentazione (view) e da quelli di controllo che tali funzionalità utilizzano (controller).
- Data Access Object, per la gestione della persistenza: si tratta fondamentalmente di una classe con relativi metodi che rappresenta un'entità tabellare di un RDBMS.

1.2 Linee guide per la documentazione dell'interfaccia

Per lo sviluppo ottimale del sistema sono state stipulate le seguenti linee guida per gli sviluppatori del sistema.

- Nomi delle Classi
  - Ogni classe deve avere il nome in CamelCase
  - Ogni classe deve avere un some singolare
  - Ogni classe non deve contenere numeri o caratteri speciali

    - Ogni classe che modella un'entità del database deve avere il nome composto formato dal nome dell'entità che modella seguito da "Bean"
    - Ogni classe che modella un Controller dell'entità e quindi implementa la logica di Business deve avere il nome composto formato dal nome dell'entità seguito da "Control"
    - Ogni classe che interroga il database per lo scambio dei dati, deve avere nome composto formato da nome dell'entità seguito da "Model"

- Nomi dei metodi
    - Ogni metodo deve avere nome in lowerCamelCase
    - Ogni metodo deve avere nome che inizia con un verbo in seconda persona singolare
    - I metodi di accesso devono avere nome composto formato da "get" e l'attributo che restituiscono
    - I metodi setter devono avere nome composto formato da "set" e l'attributo che modificano
    - Ogni metodo che può lanciare un'eccezione deve segnalarlo nella firma
- Nomi delle eccezioni
    - Ogni eccezione deve avere un nome che identifica l'eccezione creata
    - Ogni eccezione deve avere un nome composto formato dal problema generato seguito da "Exception"
- Nomi altre sorgenti
    - Ogni documento JSP/Html/CSS deve avere nome che riconduce al proprio contenuto.

Organizzazione delle componenti

- Tutte le componenti che realizzano l'interfaccia grafica sono collocate nella cartella WebContent.
- Tutte le risorse statiche sono organizzate in base al tipo (fogli di stile, script, icone, immagini) e raggruppate nella corrispondente sottocartella.

1.3 Definizioni, acronimi e abbreviazioni

N/A

1.4 Riferimenti

- Bootstrap
- Eclipse
- JavaScript
- JSON

2.0 Packages

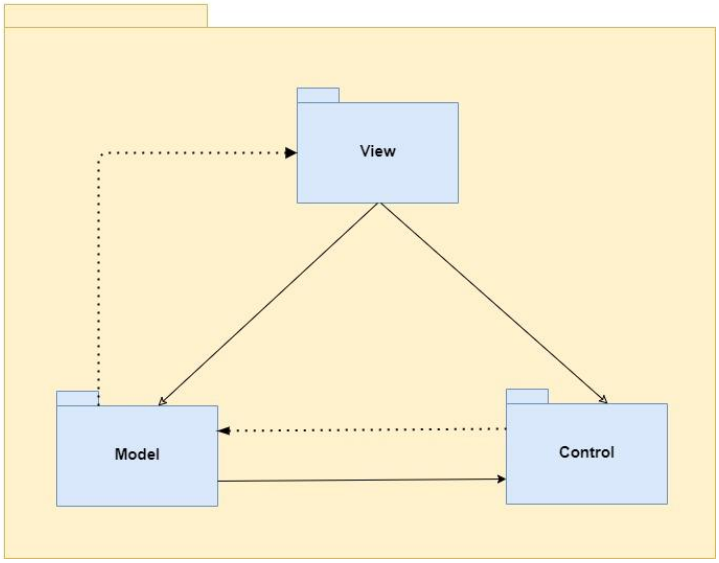
2.1 Divisione in pacchetti

Si è deciso di sviluppare il sistema seguendo un'architettura MVC, acronimo di Model View Control, che si basa sul raggruppamento di classi Java in base al loro ruolo all'interno del sistema.

Nel pacchetto Model verranno implementate tutte le classi Java incaricate di fornire i metodi per interrogare il database. In particolare, viene fornita un'interfaccia Java che espone le firme dei metodi che ciascuna entità Model dovrà implementare. I metodi esposti rappresentano le query per interrogare il database, ogni entità implementerà le query "CRUD", acronimo di create, read, update, delete).

Nel pacchetto Control verranno implementate tutte le classi Java che implementano la logica di Business per ciascuna entità.

Nel pacchetto Beans verranno implementate tutte le classi Java Bean che modellano gli oggetti del database fornendo attributi e metodi getter e setter.



Nome Pacchetto	**View**
Descrizione	All'interno di questo pacchetto sono contenute tutte le pagine jsp, incaricate di visualizzare e organizzare il contenuto delle pagine. Questo approccio ci permette di generare pagine dinamiche.
Dipendenze	Control, Model

Nome Pacchetto	**Model**
Descrizione	All'interno di questo pacchetto sono contenute le porzioni di codice incaricate di interrogare e recuperare informazioni dal database.
Dipendenze	Beans

Nome Pacchetto	**Control**
Descrizione	All'interno di questo pacchetto sono contenute le pagine di back-end, incaricate di gestire la logica di business dell'intero sistema.
Dipendenze	Model

2.2 Organizzazione dei file

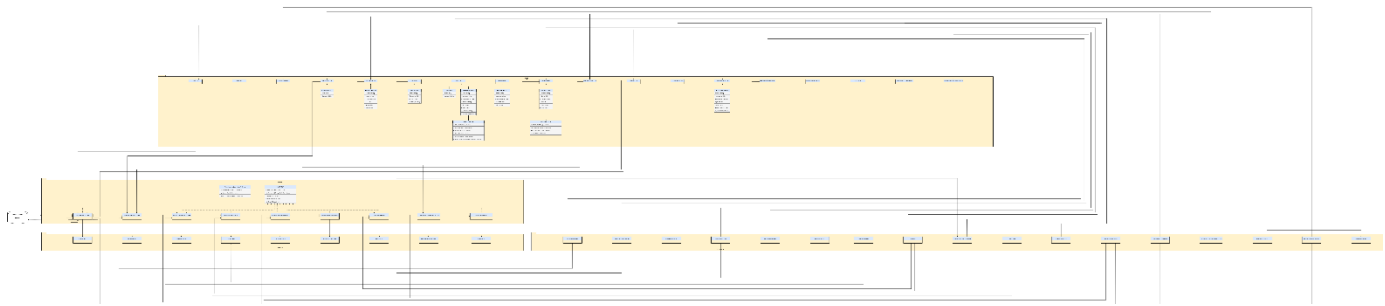
I nomi dei pacchetti saranno organizzati secondo il prefisso it.unisa.di.*nomePacchetto,* seguiranno il percorso *src/it/unisa/di/nomePacchetto.*

Il pacchetto dell'interfaccia utente seguirà il percorso *WebContent.* Le componenti statiche seguiranno il percorso *WebContent/NomeCartella.*

3.0 Interfaccia delle classi

È possibile reperire la documentazione relativa all'interfaccia pubblica delle varie classi nei file Javadoc allegati. Tali documenti includono la definizione di precondizioni, postcondizioni e invarianti per i metodi delle classi coinvolte.
La navigazione del sito è consigliabile avvenga accedendo prima tramite il file *index.jsp*.

4.0 Class diagram



Una versione PDF del class diagram è caricata tra i file di documentazione, rinominato come *ODD – Class Diagram.pdf*.