Relazione Ingegneria del Software

Ghislotti Gianluca, Esposito Andrea, Giardinetti Luca

Marzo 2022

Contents

1 Diagrammi di progetto

1.1 Diagramma dei casi d'uso

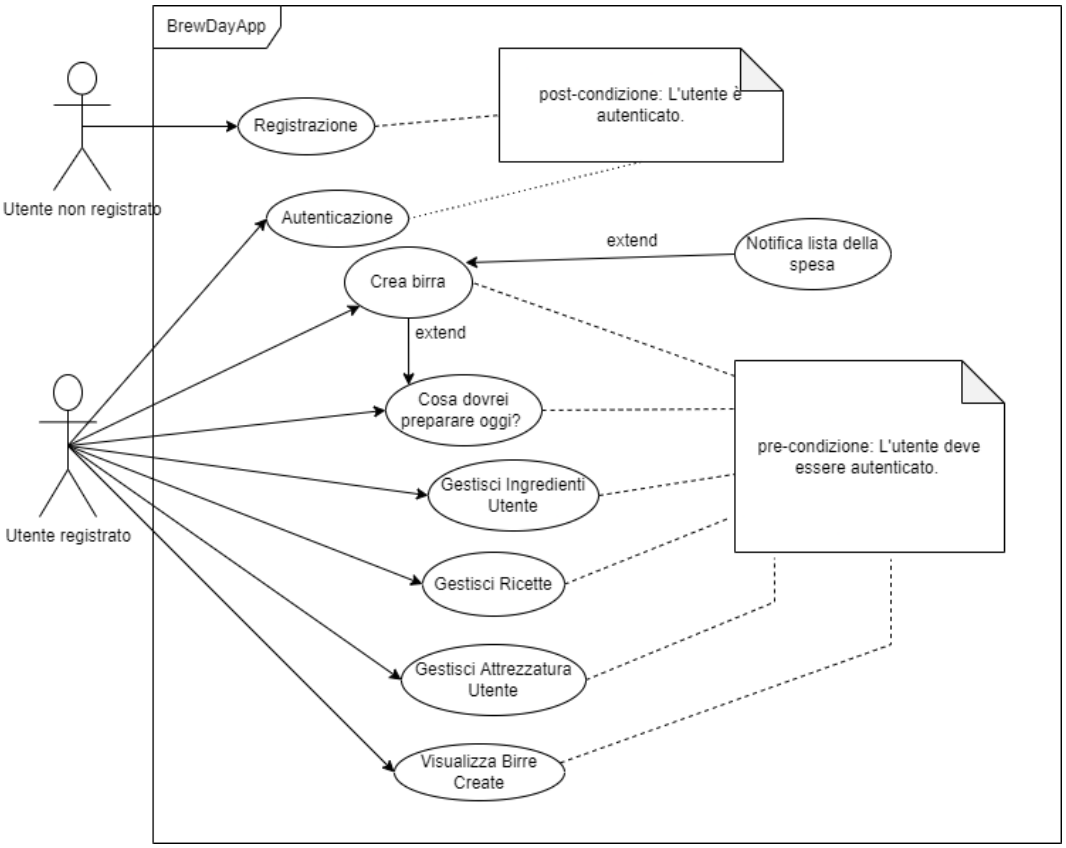

Figure 1: Diagramma dei casi d'uso

1.2 Modello di dominio

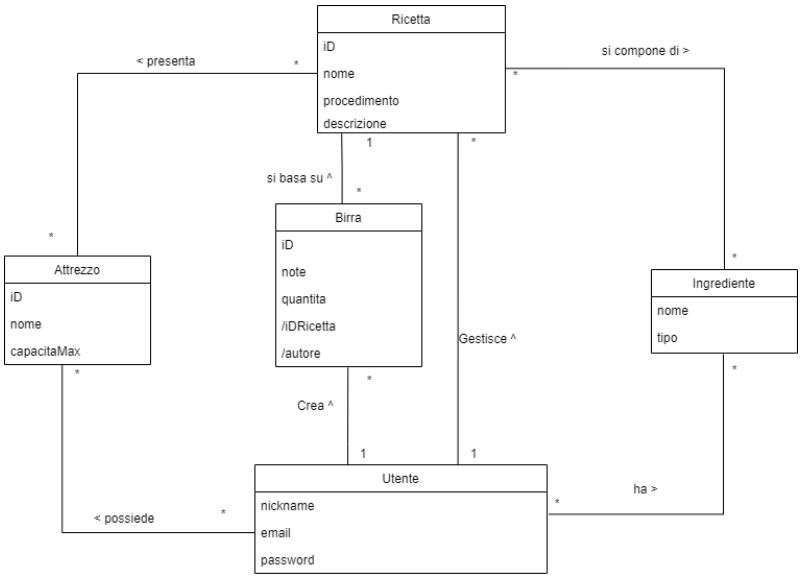

Figure 2: Modello di dominio

1.3 Diagramma SSD

Caso d'uso: *Gestisci ingredienti.*
Scenario: *Aggiunta Ingrediente.*

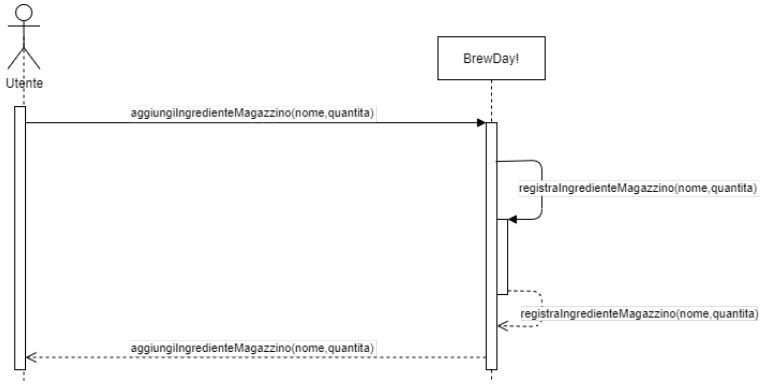

Figure 3: Diagramma SSD

1.4 Diagramma di sequenza di progettazione

Caso d'uso: *gestisci Ingredienti*.

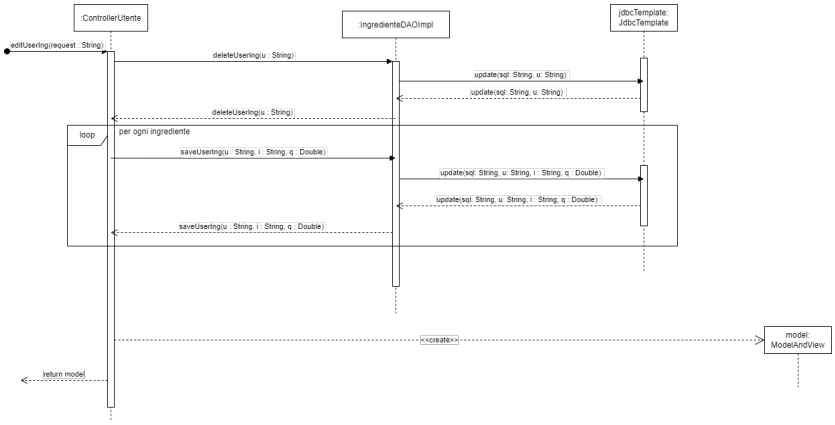

Figure 4: Diagramma di sequenza di progettazione

1.5 Diagramma architettura Software

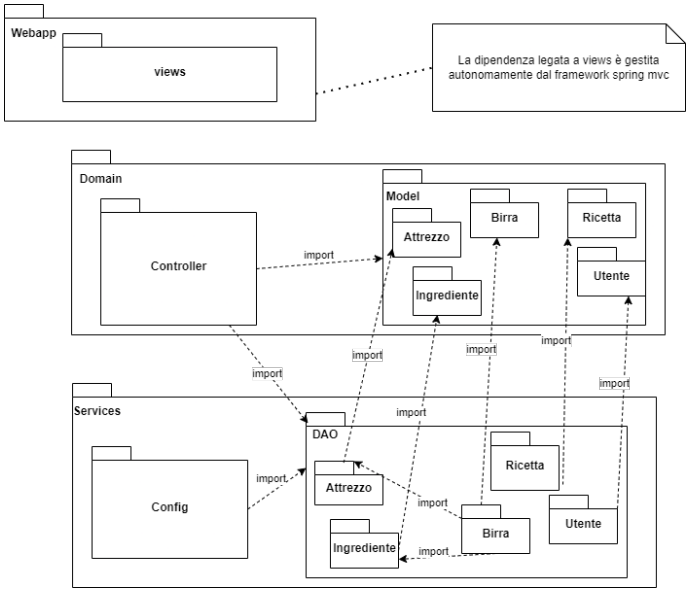

Figure 5: Diagramma Architettura Software

1.6 Diagramma degli stati

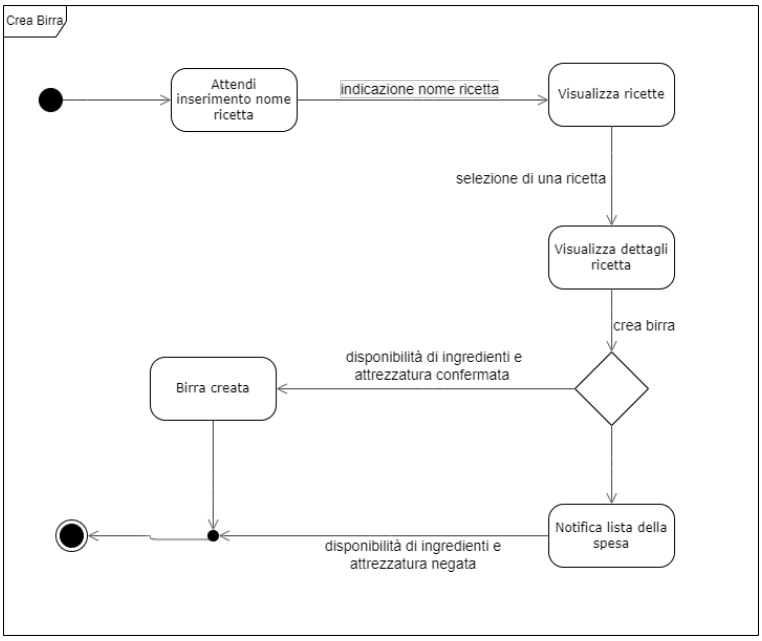

Figure 6: Diagramma a stati: Crea Birra

1.7 Diagramma delle classi

L'immagine del diagramma delle classi sarà allegata nella mail e non aggiunta direttamente nella relazione per questioni di spazio.

1.8 Diagramma di attività

Caso d'uso: *crea Birra*.

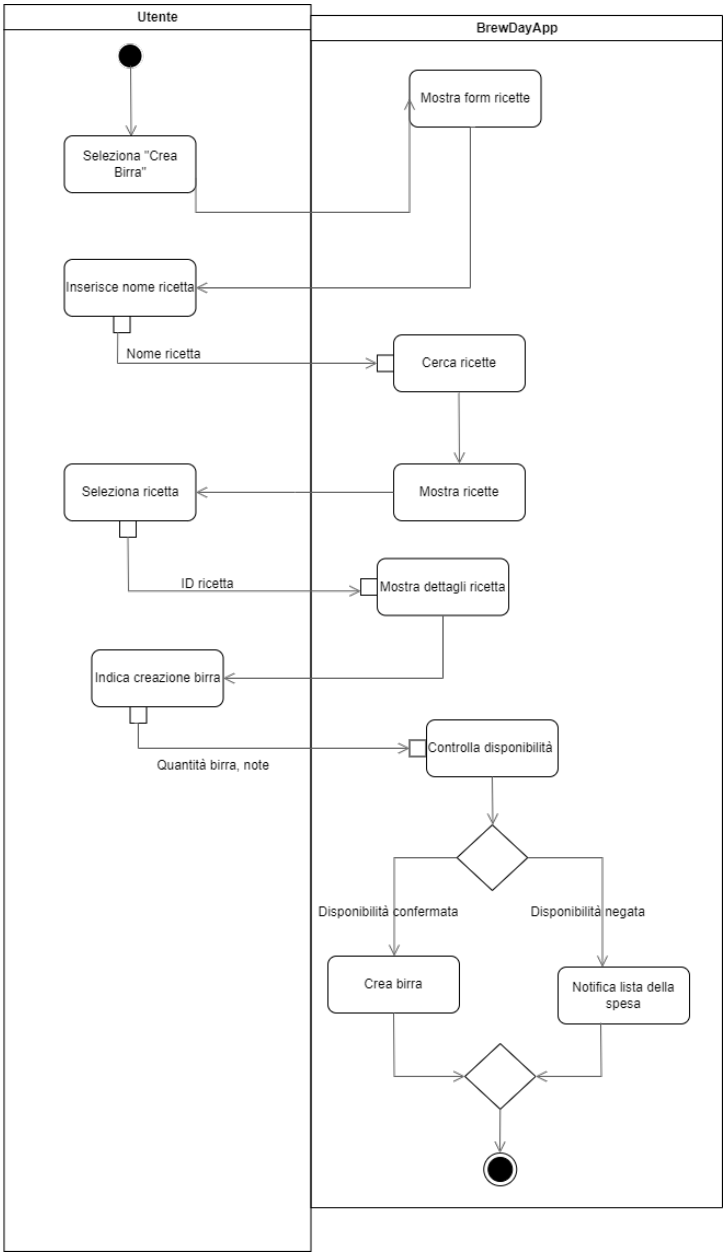

Figure 7: Diagramma di attività

2 SonarQube

2.1 Bug rilevati (Reliability)

1. Mancanza dell'attributo *lang* associato ai tag HTML in tutte le pagine jsp: *RISOLTO*

2. Mancanza dell'attributo *alt* associato a tutti i tag img all'interno di alcune pagine jsp: *RISOLTO*

3. La condizione if presente nel metodo *getNumAtt(String, double, String)* alla riga 98 di *group.brewdaytwo.services.dao.attrezzo.AttrezzoDAOImpl*, viene considerata da sostituire in quanto, secondo SonarQube, restituisce sempre false. Dopo diverse prove, ci siamo accertati del corretto funzionamento della condizione e quindi questo bug è stato erroneamente segnalato dallo strumento: *RISOLTO COME FALSO POSITIVO*.

2.2 Problema legato alla sicurezza rilevato (Hotspot Reviewed)

Le query MySQL di tipo *SELECT*, specificate all'interno dei metodi delle classi implementative DAO (xxxDAOImpl), contenevano in modo esplicito i dati da associare ad eventuali condizioni *WHERE*. Nello specifico, le query vengono trascritte e rappresentate sottoforma di una variabile String e le informazioni erano aggiunte utilizzando la concatenzazione tra stringhe, questo era ciò che veniva segnalato dallo strumento SonarQube. Per risolvere tale problema sono state apportate due modifiche al codice:

1. Le query, al posto dei dati concatenati, presentano dei punti di domanda *?*.

2. I metodi utilizzati per eseguire le query sono stati sostituiti con metodi dello stesso carattere, ma che permettono di specificare, in più, un vettore di oggetti da associare ai diversi punti di domanda presenti nella query da eseguire.

2.3 Code smell irrisolti

Nel progetto sono presenti 3 tipologie di code smell non risolti:

1. *Duplicazione di stringhe*: SonarQube rileva diversi code smell critical legati alla duplicazione di stringhe, utilizzate però o all'interno di query o per indicare nomi e informazioni relative alle view jsp.

   Abbiamo deciso di non risolvere il problema per evitare di ridurre sensibilmente la comprensione del codice.

2. *Traduzione carattere speciale*: SonarQube rileva un code smell critical legato all'utilizzo di un carattere speciale unico all'interno del codice,

questo viene utilizzato per tradurre l'inserimento del carattere *à* all'interno dei diversi form della web app.

Abbiamo deciso di non risolvere il suddetto problema in quanto ci è sembrato indispendabile gestire il seguente caso, dato che altrimenti non sarebbe stato gestito in alcun modo.

3. *Complessità cognitiva*: il problema è legato all'utilizzo, reputato eccessivo, di costrutti iterativi e condizionali nidificati.

   Abbiamo deciso di non risolvere il seguente code smell dato che avrebbe stravolto totalmente la struttura del progetto, data la mancanza di tempo dal momento della rilevazione del problema.

   Inoltre non è stato ritenuto critical dal team, data la grandezza del progetto e dal target della web app. Sarebbe stato un problema grave per applicazioni di maggiore complessità e con maggiore utenza.

2.4 Impostazioni analisi progetto

Per analizzare il progetto con SonarQube, non sono necessarie particolari impostazioni.

3 Understand

3.1 Antipattern strutturale rilevato

3.1.1 External Breakable

Rilevata la presenza di 10 dipendenze relative alle componenti DAO (interfacce e classi implementative) nel package *group.brewdaytwo.services.config.MvcConfig*.

Lo scopo di queste dipendenze consiste nello specificare quali siano le componenti che contengono la gestione (nei controller) delle varie operazioni che possono essere svolte durante l'esecuzione della webApp.

Dunque, questo antipattern non porta a problemi particolari purchè non vengano modificate i riferimenti alle componenti DAO.

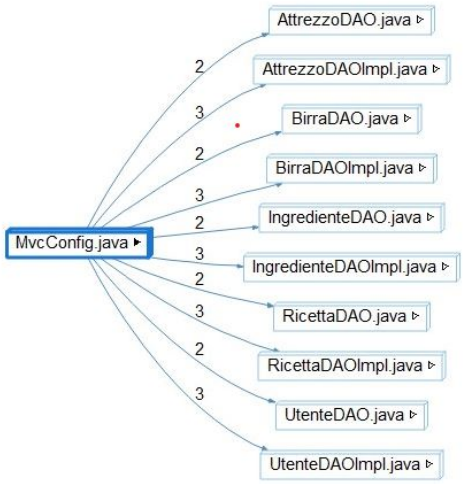

Figure 8: External Breakable

3.2 Situazioni da monitorare per futuri antipattern strutturali

3.2.1 Global Breakable

Rilevata la presenza di 7 dipendenze nel package: *group.brewdaytwo.domain.controller.ControllerUtente.*

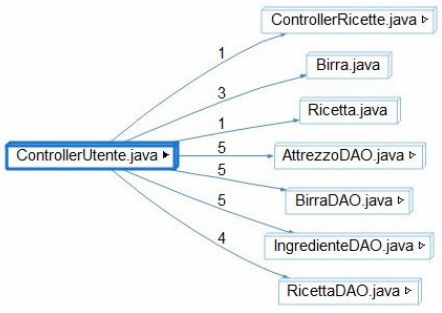

Figure 9: Global Breakable

3.2.2 Global Hub

Rilevata la presenza di 7 dipendenze/dipendenti nei seguenti package:

- *group.brewdaytwo.domain.controller.ControllerRicette*
- *group.brewdaytwo.services.dao.ingrediente.IngredienteDAO*
- *group.brewdaytwo.services.dao.attrezzo.AttrezzoDAO*

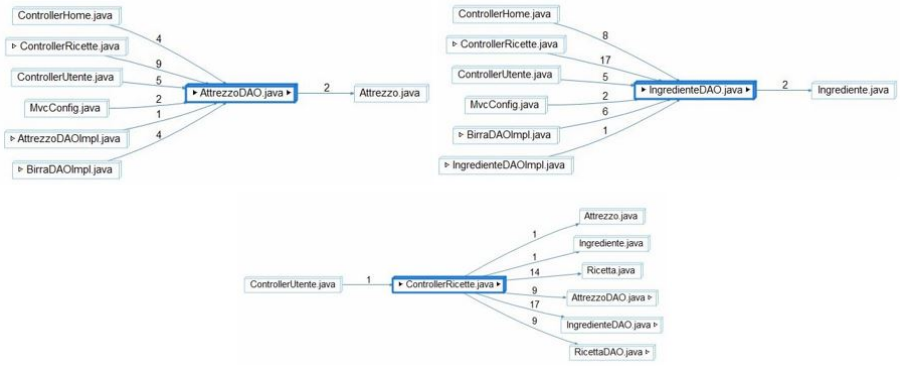

Figure 10: Global Hub

4 Pattern

4.1 Data Access Object

DAO è un pattern *Architetturale* che attraverso l'implentazione di classi, rappresenta entità tabellari di un DataBase, sostenendo il *principio della responsabilità unica*.

L'obbiettivo è quello di stratificare e isolare l'accesso ad una tabella tramite query.

I metodi del *DAO* con le rispettive query verranno richiamate dalla *Business Logic*. Inoltre utilizza la tecnica *ORM* al fine di convertire i dati del DataBase in oggetti dei linguaggi OO.

Il *DAO* presenta le seguenti componenti:

- Data Access Object Interface - Questa interfaccia definisce le operazioni standard da eseguire su uno o più oggetti del model.
- Data Access Object Class - Questa classe implementa l'interfaccia DAO, essa è responsabile per ottenere dati da una fonte che può essere un database / xml o qualsiasi altro meccanismo di archiviazione.
- Object Model - Questo oggetto contiene metodi get/set per memorizzare i dati recuperati utilizzando la classe DAO.

Motivazioni: dovendo prevedere operazioni *CRUD* del DataBase relative agli oggetti, il pattern DAO ci ha permesso di strutturare l'applicativo in modo da prevedere queste operazioni ed avere una netta separazione tra la componente Controller e la componente Model.

4.2 Identity Field

Identity Field è un *Object-Relational Structural Patterns* che permette di salvare un campo ID del database in un oggetto per mantenere l'identità tra un oggetto in memoria e una riga del database.

Motivazioni: abbiamo deciso di utilizzare questo pattern in quanto è stato ritenuto necessario legare il database al sistema di oggetti in memoria.

4.3 Service Layer

Un Service Layer definisce il confine di un'applicazione e il suo insieme di operazioni disponibili dal punto di vista dei client layer che si interfacciano.

Motivazioni: abbiamo utilizzato questo pattern per separare concettualmente lo strato di dominio legato al controllo dalle richieste web e ai dati con lo strato di configurazione dell'applicativo e di comunicazione con il DataBase.

4.4 Model View Controller (Spring)

L'**MVC** considera tre ruoli:

1. *Model*: un oggetto nonvisual, contenente tutti i dati e comportamenti diversi da quelli usati per l'interfaccia utente.
2. *View*: rappresentazione grafica del modello nella User Interface.
3. *Controller*: gestisce gli input dell'utente, manipolando il modello e causa un'appropriato aggiornamento nella View.

4.5 Singleton (Spring)

Singleton è un design pattern creazionale che ha lo scopo di garantire che di una determinata classe venga creata una e una sola istanza, e di fornirle un unico punto di accesso globale.

In spring i beans (nel progetto, classi DAO) vengono definiti nei file di configurazione come singleton per impostazione predefinita. Nonostante i beans singleton siano associabili al pattern Singleton, i due concetti sono abbastanza diversi.

4.6 Front Controller (Spring)

Front Controller è un *Web Presentation Pattern* che fornisce un controller centralizzato per la gestione delle richieste. Ogni richiesta del client e tutti i dati in entrata devono passare ed essere elaborati prima dal Front Controller, senza eccezioni. Utile per quando l'applicazione ha più punti di ingresso che si vuole centralizzare attraverso un unico punto per un'elaborazione standardizzata.

Spring implementa questo design pattern usando DispatcherServlet, per distribuire le richieste in arrivo ai controllori corretti.

4.7 View Helper (Spring)

View Helper organizza i componenti della vista per l'utente e delega l'elaborazione ad altri componenti in modo che il componente della vista non debba contenere alcuna logica di elaborazione oltre a quella di presentazione.

Spring fa uso di tag JSP personalizzati, nello specifico *JSTL*, per separare il codice dalla presentazione nelle viste.

4.8 Dependency Injection (Spring)

Il *Dependency Injection* è un Design Pattern che permette di rimuovere le dipendenze hard-coded e rendere l'applicazione loosely coupled, estendibile e mantenibile.

Può essere considerato una versione più specifica del Program Principle Inversion of Control, descritto di seguito.

4.9 Program Principle: Inversion of Control (Spring)

L'*Inversion of Control* è un Pattern per il quale è il framework ad avere il flusso di controllo del programma e non il programma scritto dal developer.

4.10 Template Method (Spring)

Template Method è un Design Pattern comportamentale; è un metodo in una superclasse, di solito una superclasse astratta, e definisce lo scheletro di un'operazione in termini di una serie di passi di alto livello.

Nel nostro progetto è legato all'utilizzo della classe JdbcTemplate, utilizzato per la connessione con il DataBase.

4.11 Factory (Spring)

Factory è un pattern creazionale che definisce un'interfaccia che permette alle sottoclassi di decidere quale oggetto istanziare. Il Factory nell'interfaccia, permette ad una classe di rinviare la creazione dell'istanza ad una o più sottoclassi concrete.

Spring lo utilizza per caricare i *beans* usando BeanFactory e il contesto dell'applicazione.

4.12 Prototype (Spring)

Prototype è conosciuto come un pattern creazionale, in quanto è usato per costruire oggetti in modo che possano essere disaccoppiati dai loro sistemi di implementazione.

Spring crea oggetti basati su un modello di un oggetto esistente attraverso la clonazione.

5 Guida all'installazione e al setup di BrewDay!

Il programma BrewDay! è disponibile all'interno del repository git, al tag *consegna* sottoforma di un file *.zip* oppure *.tar.gz*.

Una volta scaricato il file ed estratto dalla cartella compressa, è necessario installare sul proprio dispositivo una serie di programmi per supportare ed eseguire BrewDay!

5.1 Java SE development kit

Installare il Java SE Development Kit (JDK) dal link. Una volta sul sito, selezionare il sistema operativo del vostro terminale e scaricare.

5.2 Apache Tomcat v9.0

Scaricare l'installer dalla seguente pagina.

Installazione:

- Non modificare le predisposizioni già impostate dall'installer
- Solamente alla pagina "Java Virtual Machine" verrà richiesto il percorso di JDK: tipicamente è "C:/Program Files/Java/jdk-XXXX" (XXXX sostituisce la versione di JDK installata)

5.3 MYSQL

1. Scaricare ed installare *Microsoft Visual C++ 2019 Redistributable Package* dai seguenti link:

   - (64bit)
   - (32bit)

2. Scaricare l'installer **NON web** (la seconda voce), dalla seguente pagina.

3. Installazione:

   - Selezionare la versione *Developer Default* ed installare tutti i componenti associati. Alla pagina di controllo dei requisiti potrebbe essere segnalata la mancanza di alcuni componenti, ignorare tale avviso dato che riguardano funzionalità futili.
   - Selezionare la voce *Standalone MySQL Server / Classic MySQL Replication.*
   - Alla pagina *Type and Networking*: selezionare *Server Computer*, per la sezione *Connectivity*, non modificare le predisposizioni già impostate dall'installer.
   - Proseguire sino alla pagina *Accounts and Roles*, qui indicare una propria password per l'account di root (da ricordare).

- Completare il setup del programma senza modificare ciò che è stato già impostato dall'installer.

4. Una volta completato il setup aprire MySQL workbench e selezionare il delfino sulla sinistra.

5. Creare una nuova connessione cliccando sul +.

6. Indicare un nome della connessione a scelta e specificare la propria password.

7. Aprire la connessione.

8. Ora non resta che importare il database: File $->$ Open SQL Script $->$ Selezionare "progetto_brewdayDataBase" dalla cartella unzippata di BrewDay $->$ Premere il fulmine.

Il setup della componente Database è completato.

5.4 IDE Eclipse for enterprise java and web developers

1. Impostare il charset ISO-8859-1: Window $->$ Preferences $->$ Workspace $->$ "Text file encoding" $->$ Impostare e selezionare Other su ISO-8859-1 $->$Apply and close.

2. Configurare il server Tomcat:

   - Selezionare il *Pannello Server* dal menu: Window $->$ Show View $->$ Others $->$ Server $->$ Servers.
   - Fare click su *Click to create a new server*.
   - Selezionare dal menu *Tomcat v9.0*.
   - Impostare i seguenti valori:
     - Server's host name: *localhost*.
     - Server name: *Tomcat v9.0 Server at localhost*.
     - Server runtime environment: *Apache Tomcat v9.0*.
   - Su Windows è necessario indicare la directory di installazione di Tomcat che generalmente è: C:\Program Files\Apache Software\Foundation.
   - Fare doppio click sul server appena creato e aprire la sezione *Ports*.
   - Impostare al valore 0 la voce *Tomcat Admin Port*.

3. Importare la cartella unzippata di BrewDay come Maven project:

   File $->$ Import $->$ "Existing Maven Project" $->$ Indicare la cartella unzippata di BrewDay $->$ Selezionare il file *pom.xml* riconosciuto $->$ Finish.

4. Indicare i dati di accesso al database:

- Tramite l'ausilio di *Package explorer* selezionare il file MvcConfig.java: Fare doppio click su: Cartella di BrewDay − > src/main/java − > group.brewdaytwo.services.config − > MvcConfig.java.
- Alle righe indicate esplicitamente nel codice inserire tra i doppi apici presenti l'usernname (di default è root) e la password del proprio account su MySQL.
- Salvare premendo *CTRL+S*.

5. Pubblicare il progetto sul server Tomcat:
   - Selezionare nuovamente il *Pannello Server* dal menu: Window − > Show View − > Others − > Server − > Servers.
   - Fare click con il tasto destro del mouse sul server.
   - Selezionare *Add and Remove*.
   - Selezionare la cartella di BrewDay importata e cliccare su add.
   - Una volta fatto premere *Finish*.
6. Per avviare l'applicazione:
   - Selezionare il *Pannello Server* dal menu: Window − > Show View − > Others − > Server − > Servers.
   - Fare click con il tasto destro del mouse sul server
   - Selezionare *Start*.
   - Aprire un browser a propria scelta e digitare *localhost:8080/BrewDayApp* .

6 Commenti

6.1 Scelte implementative

- Leggendo le specifiche del progetto abbiamo sviluppato l'applicazione per un utenza prettamente *casalinga* o *homemade*. Di conseguenza le informazioni memorizzate e gestite dall'applicativo sono visibili solo ed esclusivamente dal proprietario, ossia da uno e un solo utente.

- *Visione degli ingredienti e dell'attrezzatura*: tra le informazioni delle ricette, gli ingredienti e l'attrezzatura sono stati indicati in modo *relativo*; gli ingredienti attraverso unità di misura *relative* (%, g/L) e l'attrezzatura senza alcuna indicazione della capacità massima.

  Nel magazzino dell'utente, gli ingredienti sono stati indicati con unità di misura *assolute* (g, L) e l'attrezzatura specificandone la massima capacità possibile (L).

- *Cosa devo produrre oggi?*: questa funzionalità prevede l'indicazione della ricetta che permette di massimizzare l'utilizzo degli ingredienti possedute dall'utente e dal batchsize dell'attrezzatura.

  Dal momento che le ricette presentano unità di misura *relative* degli ingredienti e nessuna capacità massima dell'attrezzatura richiesta, per stabilire la miglior ricetta sono state eseguite le seguenti operazioni:

  1. Trovare tutte le ricette di cui l'utente possiede gli ingredienti e l'attrezzatura necessaria per produrre un qualsiasi quantitativo di birra.
  2. Ordinare le ricette sulla base del maggior numero di ingredienti previsti nella ricetta.
  3. A parità di numero di ingredienti, le ricette vengono ordinate sulla base del maggior quantitativo totale di ingredienti richiesti.
  4. In caso di ulteriore parità la selezione avviene in modo arbitrario.
  5. Una volta individuata la "miglior" ricetta, viene calcolata e consigliata la massima quantità di birra producibile sulla base degli ingredienti e dell'attrezzatura presente in magazzino.

6.2 Problemi incontrati durante lo sviluppo

- Un problema grafico riguardante la gestione dell'attrezzatura, nello specifico l'input della quantità, ciò consiste nel caricamento errato dello stile .css importato. Questa *anomalia* appare in modo randomico navigando nell'applicazione.

- Durante varie prove dell'applicazione web, il server *MySQL* non permetteva alcuna interazione a causa di un bug MySQL secondo cui, lo spazio riservato all'avvio del server e alla memorizzazione dei dati era insufficiente. Dopo diverse ricerche, è stato scoperto essere un bug *noto* del pacchetto MySQL.

- Durante lo sviluppo delle componenti grafiche del sito, diverse volte l'IDE Eclipse non caricava correttamente le componenti mostrando una grafica errata, non in linea con i file .css importati. Questi si sono risolti inaspettatamente da soli dopo un certo periodo di utilizzo dell'applicazione web.

7 Contatti

Per qualsiasi problematica contattare uno degli sviluppatori:
- Giardinetti Luca: l.giardinetti@campus.unimib.it
- Ghislotti Gianluca: g.ghislotti@campus.unimib.it
- Esposito Andrea: a.esposito54@campus.unimib.it