LA TAZZA

Relazione per il progetto di Ing. del Software

04.06.2019

INTRODUZIONE

In questa breve relazione documentiamo con una descrizione informale la struttura del progetto e “il perché” di alcune scelte implementative.

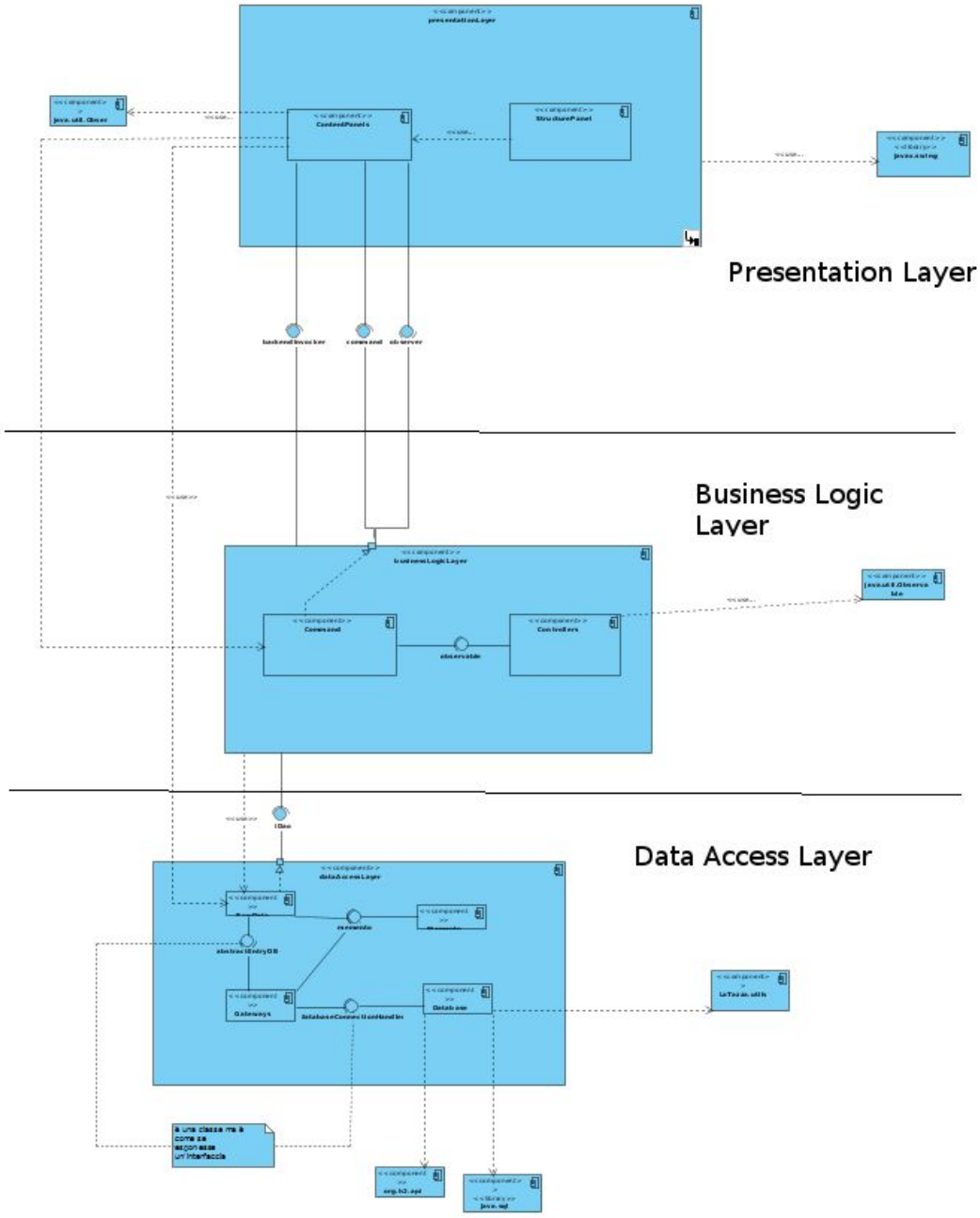
Il team è composto da 4 persone e per questo motivo ci è stato richiesto di implementare, oltre alle specifiche iniziale, la persistenza dell’applicazione sotto forma di database.

Il database utilizzato è H2.

ARCHITETTURA

Per l'applicazione abbiamo scelto di utilizzare una three-tier architecture.

L'immagine riporta il component diagram dal quale è facile vedere i vari strati dell'applicazione.



La decisione di implementare l'architettura in questo è stata presa quando abbiamo riscontrato che era necessario gestire le eccezioni di tipo *JdbcSQLIntegrityConstraintViolationException , SQLException* lanciate dal DataBase.

Queste eccezioni. infatti sono di tipo Checked e quindi obbligatorie da gestire.

Abbiamo deciso quindi di implementare un dataAccessLayer che offuscasse l'implementazione e la gestione del database.

Il dataAccessLayer è una piccola implementazione di un Object-Relational Mapping.

Infatti tramite l'interfaccia IDao, il Business Logic Layer può gestire gli oggetti in maniera molto facile, semplicemente passando l'oggetto stesso ad uno dei metodi dell'interfaccia.(save,update…)

Con uno sguardo successivo all'implementazione mi rendo conto di aver *"reinventato la ruota" ;* ma il nostro gruppo non avendo mai utilizzato i DB ORM aveva deciso di usare un database di tipo RDBMS e ho capito solo alla fine di aver creato una cosa già esistente.

Il Data Access Layer utilizza i pattern:

- Memento
- Strategy Pattern

Il pattern Memento viene utilizzato per ripristinare gli oggetti in Ram al loro stato originale se l'operazione di update del DB fallisce.

Lo Strategy Pattern viene invece utilizzato per far funzionare la piccola implementazione dell'ORM.(Le strategy sono i vari *DaoReceiver)

Per l'interazione tra Presentation Layer e Business Logic Layer abbiamo utilizzato il seguente pattern:

- Command Pattern
- Observer Pattern

Nel pkg businessLogicLayer.commandPkg sono presenti tutti i comandi che (più o meno) corrispondono ai vari use case.

Anche in questo caso come per il dataAccessLayer la classe BackEndInvoker, tramite il command pattern, offusca l'implementazione sottostante e permette al

presentationLayer di utilizzare il command Pattern senza preoccuparsi di gestione eccezioni o inconsistenze sui dati.

L'observer pattern viene invece utilizzato per aggiornare le schermate senza fare "*polling*" sul businessL.

I due strati (BusinessLayer e DataAccessLayer) in caso di errore loggano nel file *LaTazza.log* le cause dell'eccezione.

Infine vogliamo sottolineare che abbiamo tentato di rendere l'applicazione *fault-tolerant,* gestendo tutte le eccezioni , inconsistenze sui dati ed errori del database possibili.

Nel caso l'applicazione non riesca a mantenere una consistenza viene avviata una procedura di "*graceful exit*" e loggata la causa dell'errore.

AUTORI

Repo LaTazza https://github.com/Strafo/LaTazza

Non sono Bello ma Patcho Team https://github.com/non-sono-bello-ma-patcho

In particolare:

Andrea Straforini

Jacopo Dapueto

Simone Campisi

Gabriele Armanino