UNIVERSITÀ DI PISA

Progetto di Programmazione Avanzata

Corso di Laurea in Ingegneria Informatica

Anno Accademico 2023-2024

Valerio Cannalire

Sommario

1. Introduzione

1.1 Caso d'uso

Gli ultimi anni sono stati segnati dall'inflazione che non ha risparmiato alcun settore. I consumatori possono fare ben poco di fronte a questo fenomeno se non misurare più attentamente le loro spese. SmartFreezer è stato pensato per aiutare i consumatori negli acquisti al supermercato, permettendo loro di approfittare al meglio delle offerte proposte sfruttando il congelatore quando possibile.

Generalmente il consumatore quando si reca al supermercato ha una lista della spesa che riporta i beni che ha intenzione di acquistare, ma spesso non ha un'idea ben precisa delle offerte che troverà sugli scaffali e quindi non ne avrà programmato preventivamente l'acquisto. Inaspettatamente trova i polpettoni scontati del 40%: sono buoni ma solitamente costosi, e il supermercato difficilmente li mette in offerta. Quanto spazio ha in congelatore? Ha alcuni polpettoni ancora congelati? Può liberare dello spazio in congelatore rimuovendo qualcosa che sta per scadere?

SmartFreezer è un'applicazione che permette di tenere un elenco preciso del contenuto interno al congelatore, etichettando ogni alimento con una categoria e specificando per ognuno quale sia la sua data di scadenza e quante porzioni ne siano presenti. Si sottolinea che il numero di porzioni rappresenta anche l'unità di misura del volume interna al congelatore. Infine si specifica che il criterio col quale assegnare la data di scadenza per la conservazione all'interno del congelatore a partire da quella consigliata per la conservazione nel frigorifero è a completa discrezione del consumatore.

1.2 Struttura e avvio dell'applicazione

SmartFreezer è un'applicazione distribuita, composta da un server che interagisce con un database Sql e da un client che dialoga unicamente col server. Il client inoltra le richieste al server mediante il protocollo HTTP e il server risponde inviando i dati codificati con JSON.
Per avviare l'applicazione è necessario avviare prima il server, che si occuperà di creare e popolare con alimenti generici il database nel caso questo non sia già esistente, e solo in seguito sarà possibile avviare l'applicazione client.

2. Lato Client

2.1 Schermate

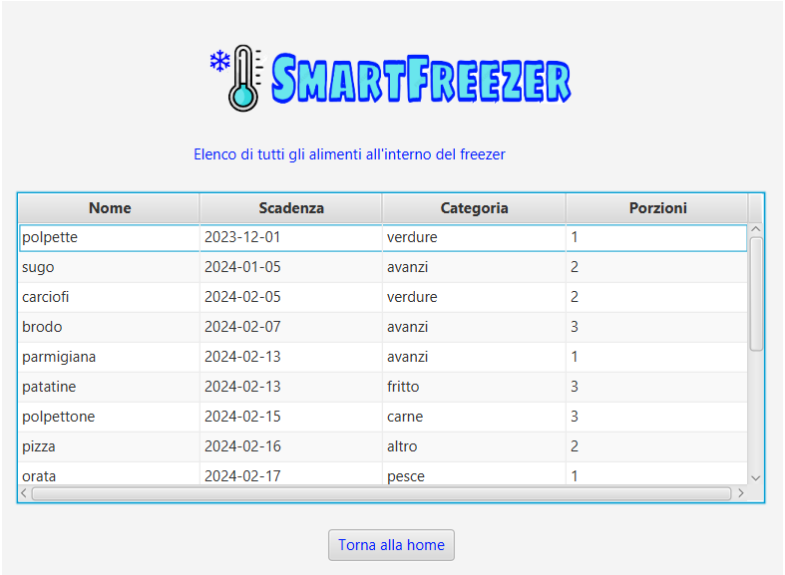
Di seguito verranno illustrate le principali caratteristiche delle schermate con cui l'utente può interagire. Si specifica la presenza in tutte le schermate, eccetto quella Home, di un tasto che permette di tornare a questa pagina. Inoltre si aggiunge che in ogni tabella sarà possibile eliminare

una singola porzione o tutte le porzioni di un alimento attraverso un menù che comparirà cliccando sulla riga con il tasto destro del mouse.

2.1.1 Schermata Home

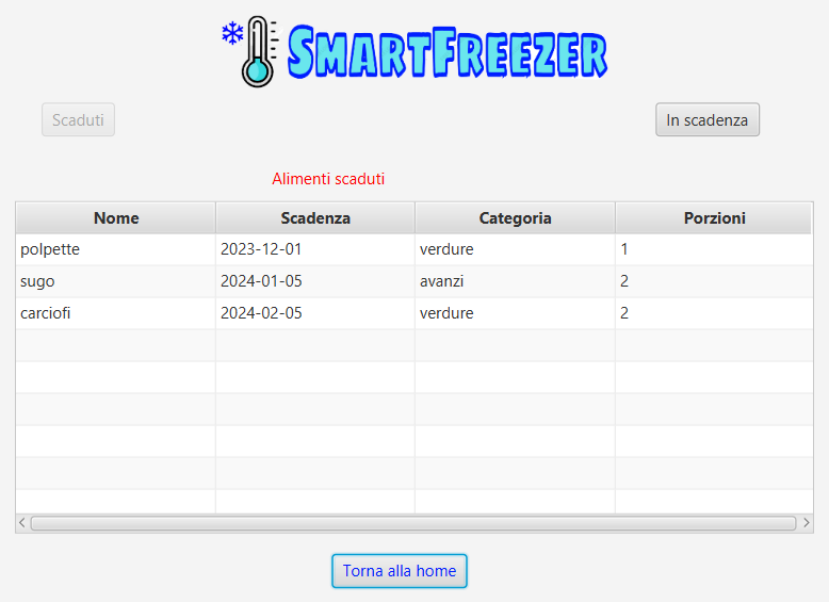
La schermata Home presenta i tasti per passare alle altre schermate e i numeri che indicano lo spazio rimanente, la quantità delle porzioni scadute e il numero di quelle che scadranno nella data odierna e nei successivi 7 giorni.

2.1.2 Schermata Tutti



La schermata Tutti offre l'elenco completo degli alimenti ordinati in base alla loro scadenza.

2.1.3 Schermata Scadenza



In questa schermata è possibile generare una tabella che contenga l'elenco degli alimenti scaduti oppure l'elenco degli alimenti che scadranno nella data corrente e nei successivi 7 giorni.

2.1.4 Schermata Ricerca

Questa pagina offre la possibilità di effettuare una ricerca personalizzata inserendo il nome di un alimento e la sua categoria (si pensi alle polpette di verdure) oppure solo uno di questi due parametri. Per l'inserimento dei parametri nella schermata sono presenti un TextField per il nome e una ChoiceBox per la categoria.

La gestione del caso con entrambi i campi assenti viene gestita dal lato client che lancia una RicercaException, non inoltrando la richiesta al server. La ricerca viene effettuata ignorando la presenza o meno delle lettere maiuscole. Una volta effettuata la ricerca i campi di inserimento non saranno più interagibili e per effettuare una nuova ricerca sarà necessario resettare la pagina attraverso l'apposito tasto.

2.1.5 Schermata inserimento

Questa schermata permette l'inserimento di un nuovo alimento all'interno del freezer. Per l'inserimento dei dati sono presenti gli stessi oggetti JavaFX introdotti nella schermata precedente per il nome e la categoria, mentre la scadenza viene gestita da un DatePicker e le porzioni da uno Slider.

Il nome e la scadenza sono necessari, senza di essi la richiesta di inserimento non viene inoltrata al server lanciando una InserimentoException. La categoria e il numero di porzioni invece sono opzionali, di default la categoria assegnata è "altro" e il numero di porzioni inserite, se non specificato, è 1.

Il client gestisce anche la mancanza di spazio, lanciando una SpazioInsufficienteException nel caso il numero di porzioni che si cercano di inserire sia maggiore dello spazio rimanente. Qualora all'interno del congelatore sia presente un alimento con lo stesso nome, la stessa scadenza e appartenente alla stessa categoria, il server sommerà le nuove porzioni a quelle esistenti. Anche in questo caso viene ignorata la presenza o meno di lettere maiuscole nel nome dell'alimento. Come nel caso della schermata Ricerca l'interfaccia d'inserimento non è più interagibile una volta che la richiesta è stata inviata al server, quindi per procedere con un nuovo inserimento sarà necessario cliccare l'apposito tasto.

2.2 Note tecniche sul lato client

Al fine di ridurre le ridondanze di codice è stata creata una classe SchemaController che mette a comune alcuni aspetti delle schermate che mostrano i risultati: si occupa della creazione di una TableView contenente le colonne correlate agli attributi di un alimento e provvede con opportuni metodi al popolamento di essa e alla rimozione di porzioni o alimenti. Siccome le attese di risposte dal server potrebbero non essere trascurabili, tutto il codice che concerne le richieste ad esso è stato opportunamente inserito all'interno di Task e Platform.runLater in modo che le interfacce rimangano bloccate durante le attese.

2.3 Unit test

Il lato client è dotato di una Unit test che durante la fase di compilazione si preoccupa di controllare se il server sia attivo, inoltrando a esso una richiesta e verificando che la risposta sia diversa da null.

3. Lato server

3.1 Lista delle API

Di seguito è riportata la lista delle API offerte al Client.

- **/all GET** ritorna l'elenco completo degli alimenti all'interno del congelatore ordinando essi a partire da quello con data di scadenza minore.
- **/scaduti GET** ritorna l'elenco degli alimenti scaduti a partire da quello con data di scadenza minore.
- **/scadenza GET** ritorna l'elenco degli alimenti in scadenza nella data odierna e nei successivi 7 giorni, a partire da quello con data di scadenza minore.
- **/numeri GET** ritorna in un array di interi codificato lo spazio rimanente all'indice [0], il numero di alimenti scaduti all'indice [1] e il numero di prodotti in scadenza nella data odierna e nei successivi 7 giorni all'indice [2].
- **/cerca POST** ritorna l'elenco degli alimenti che rispondono ai parametri d'ingresso, il nome e la categoria. È sufficiente che il client fornisca solo uno di essi.
- **/add POST** inserisce all'interno del database un nuovo alimento. Se all'interno del congelatore era già presente un alimento con la stessa chiave, aggiunge le nuove porzioni alle esistenti.
- **/remove/singolo DELETE** rimuove una singola porzione dell'alimento dato in input effettuando un update. Qualora la porzione dell'alimento sia una sola questo allora verrà completamente eliminato.
- **/remove/tutto DELETE** rimuove interamente dal database l'alimento dato in input.

3.2 Note tecniche sul lato server

Prima che venga lanciata l'applicazione Spring il server si accerta dell'esistenza del database stabilendo una connessione con MySql e andando a cercare nei metadata il nome del database. Qualora questo non sia presente il server procederebbe con l'esecuzione del file "script.sql" che contiene il codice per la creazione e il popolamento del database. L'esecuzione del file viene effettuata attraverso l'impiego della classe ScriptRunner.

Un alimento all'interno del database è rappresentato attraverso 4 attributi: nome, scadenza, categoria, porzioni. La chiave è composta da nome, scadenza e categoria (categoria è stata inserita all'interno della chiave perché alimenti di categorie diverse potrebbero avere lo stesso nome: Es. polpette di carne e polpette di verdure).

Il server interroga il database attraverso JPA. Le query sono state definite all'interno del repository correlato agli alimenti sfruttando, quando è stato possibile, il meccanismo delle keywords, e quando non è stato possibile le query sono state generate per mezzo del tag @Query. A causa della chiave composta è stato necessario definire una @IDClass.