

Progetto Programmazione Avanzata

Lorenzo Monaci

Ingegneria Informatica UniPi, A.A. 22-23

Indice

1.Panoramica.....	1
1.1 L'idea alla base.....	1
1.2 Inizializzazione del DataBase.....	1
2.Applicazione.....	2
2.1 Schermate dell'applicazione.....	2
2.1.1 Pagina di Login.....	2
2.1.2 Pagina di Registrazione.....	2
2.1.3 Menu.....	4
2.1.4 Menu dei Piatti.....	5
2.1.5 Schermata di Conferma.....	6
2.1.6 Schermata conclusiva della transazione.....	7
2.2 Dialogo con il Server.....	7
2.3 Scelte Progettuali e Meccanismi.....	8

2.3.1 Punti.....	8
2.3.2 Menu Pranzo e Cena.....	9
2.3.3 Orario per la Consegna.....	9
2.3.4 HTTPConnector.java.....	9
2.3.5 JSONResponse.java.....	9
2.3.6 JSONReader.java.....	10
2.3.7 Database Attivo.....	10
2.3.8 Unit Tests.....	10
2.3.9 Cambio Lingua.....	10
2.3.10 Arrotondamento.....	11
2.3.11 ParameterException.java.....	11

3.Server.....11

3.1 Avvio.....	11
3.2 Dialogo.....	12
3.3 Unit Test.....	12

1. Panoramica

1.1 L'idea alla Base

Il progetto SushiDelivery è pensato come un'applicazione Client-Server che previa registrazione e autenticazione dell'utente, permette a quest'ultimo di piazzare la sua ordinazione di piatti orientali, principalmente a base di sushi, e ricevere il pasto direttamente all'indirizzo specificato.

L'applicazione installa i dati necessari al primo avvio tramite un apposito tasto di configurazione presente nella prima schermata, che esegue uno script SQL per la creazione dello schema di base di dati e l'inserimento dei dati.

1.2 Inizializzazione del Database

Come descritto nella sezione precedente, il Database è inizializzato accedendo a un tasto presente nel menu 'Configurazione' nella prima schermata (*ovvero quella di Login*) alla voce Installa Database.

La pressione del tasto avvia una funzione che legge il file 'buildDB.sql' e tramite un'opportuna¹ conversione in stringa ne viene eseguito il contenuto.

Il client si connette direttamente al DBMS con i parametri:

IP:	localhost
Port:	3306
Database:	620826
Username:	root
Password:	root

Coinvolgendo i file

```
'src/main/java/it/unipi/mainproject/sushideliveryservice/LoginController.java'  
'src/main/resources/buildDB.sql'
```

¹ Il contenuto del file viene prima convertito in String poi tramite l'utilizzo delle funzioni *reduce* e *split* viene creato un array di stringhe, ognuna di esse rappresenta una singola query da eseguire.

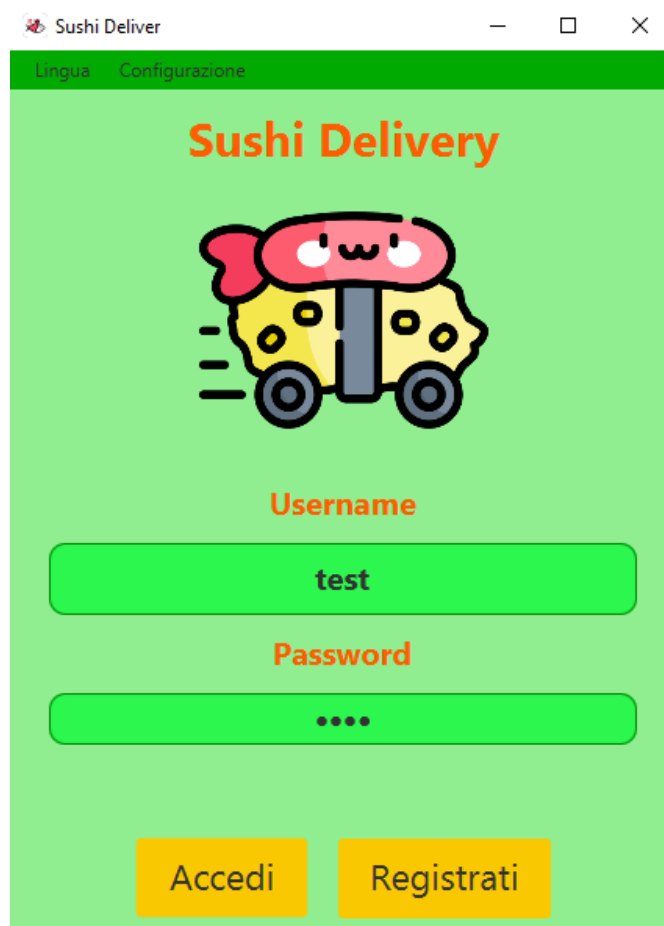
2. Applicazione

2.1 Schermate dell'Applicazione

2.1.1 Pagina di Login

All'avvio dell'applicazione è chiesto all'utente di autenticarsi o, qualora non avesse un account, di registrarsi tramite un opportuno tasto².

Se l'utente dovesse inserire accidentalmente delle credenziali errate, un messaggio di errore apposito apparirà a schermo per notificare l'accaduto.



2.1.2 Pagina di registrazione

La pagina di registrazione richiede all'utente i dati necessari alla creazione di un account, i quali *username*, *email*, *password*, *ripeti-password* e la *data di nascita*.

² Vedi sezione 2.1.2

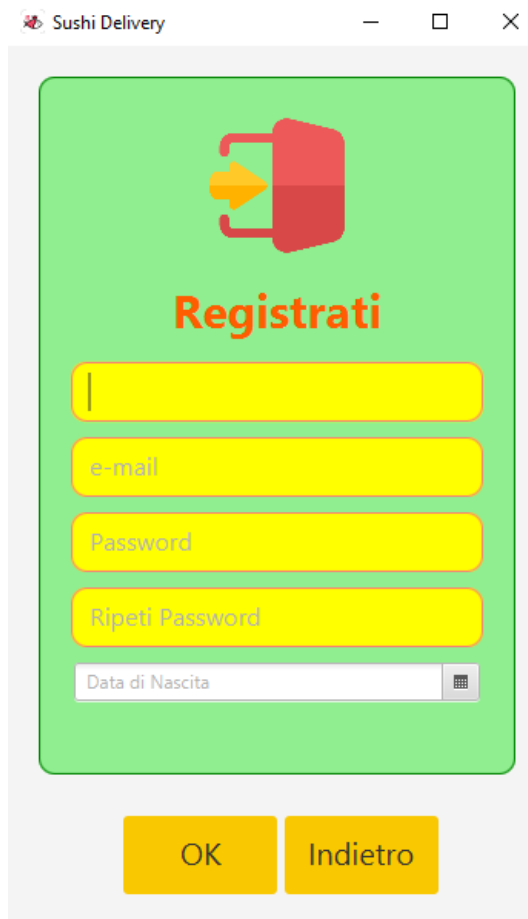
Sui campi username, password e ripeti-password valgono i seguenti vincoli:

- Username: da 4 a 10 caratteri alfabetici
- Password/Ripeti-Password: almeno 8 caratteri tra cui, un numero, una maiuscola e una minuscola


Si controlla che il campo mail che effettivamente sia inserita un testo del formato corretto.

La password una volta verificato che sia del giusto formato, è criptata utilizzando `BCrypt.hash()` di `org.springframework`.


La decriptazione avverrà poi lato Server tramite la funzione `Bcrypt.checkpw()`.



Sushi Delivery



Registrati



2.1.3 Menu

Il menu principale dell'applicazione saluta l'utente ricordandogli i punti accumulati fino a quel momento³ e mostra 3 tasti:

- Pranzo: l'utente sceglie di prenotare e scegliere i piatti dal menu per il pranzo
- Cena: come sopra «Mutatis mutandis»
- Esci: permette di effettuare il logout



³ Vedi sezione 2.3.1

2.1.4 Menu dei Piatti

All'utente viene mostrata una schermata che visualizza il menu coerente con la sua scelta nella schermata precedente⁴ le cui voci possono essere aggiunte a una lista sottostante che rappresenta il carrello, cioè i piatti che l'utente intende ordinare.

Insieme al carrello è visibile il totale dell'ordine e gli eventuali punti acquisiti per l'ordine corrente, i quali si aggiornano quando vengono inseriti/rimossi piatti dal carrello.

L'inserimento/rimozione avviene tramite un menu a scomparsa.

Menu

Codice	Nome	Ingredienti	Prezzo
1	Edamame	Fagioli giapponesi lessi	3.85
2	Wakame	Alghe in salsa di sesamo	4.4
3	Involtni Primavera - 3 pezzi	Involtni di verdura in pasta frita con salsa agrodolce	5.5
4	Involtni Ebi - 3 pezzi	Involtni di gamberi in pasta frita con salsa agrodolce	5.5
5	Misto Mare	Misto di pesce in salsa agrodolce	7.5
6	Tako seaweed	Polpo condito con salsa di soya e alghe	6.6
7	Gyoza al vapore - 3 pezzi	Ravioli di carne	3.3
8	Gyoza alla piastra - 3 pezzi	Ravioli di carne	3.3
9	Gyoza di gamberi alla griglia - 3 pezzi	Ravioli di gamberi	3.3
10	Tartare sake avocado	Salmon e avocado	7.3

Carrello | Totale: [0.0 €] | Punti acquisiti: {0}

Nome	Prezzo €
------	----------

Il Carrello è Vuoto!

Preferiti

Nome	Prezzo €
Involtni Ebi - 3 pezzi	5.5
Misto Mare	7.5
Tako seaweed	6.6
Gyoza al vapore - 3 pezzi	3.3
Gyoza di gamberi alla g...	3.3
Onigiri Alga	3.2
Onigiri Ebi	3.6
Chirashi Tuna	9.2
Sushi Misto - 20 pezzi	20.2
Tris di Tartare	13.85

Indietro

Ordina

Oltre alla possibilità di tornare indietro (qualora l'utente si fosse confuso con il menu, ad esempio), quando l'utente è convinto del suo ordine deve solo premere il tasto ordina per visualizzare la schermata di conferma.

⁴ Per le differenze tra i menu si veda la sezione 2.3.2

Una funzione aggiuntiva è quella di memorizzare per ogni utente fino a un massimo di 10 piatti preferiti, che vengono visualizzati in una tabella per poter essere aggiunti più facilmente nel carrello.

2.1.5 Schermata di Conferma

Nella schermata di conferma viene visualizzato il riepilogo dell'ordine e in un menu a tendina è possibile selezionare l'orario per la consegna⁵, dopodiché inserire indirizzo e numero di telefono, per poi scegliere il metodo di pagamento e infine confermare l'ordine, il quale verrà registrato nel DBMS

Name	Price
Chirashi Tuna	9.2

Choose a time ▼

Shipping address

Phone nr.

Back

How do you want to pay?

Card Number CVV

Credit Card

On Delivery

Subtotal: 9.2 €

Discount:

Total: 9.2 €

Confirm Order

0 Points Available

Insert points to use

0

⁵ Vedere sezione 2.3.3

2.1.6 Schermata di conclusione della transazione

L'utente viene ringraziato per aver completato l'ordine e si informa che quest'ultimo verrà spedito a breve, dopodiché può uscire dall'applicazione con un apposito tasto.

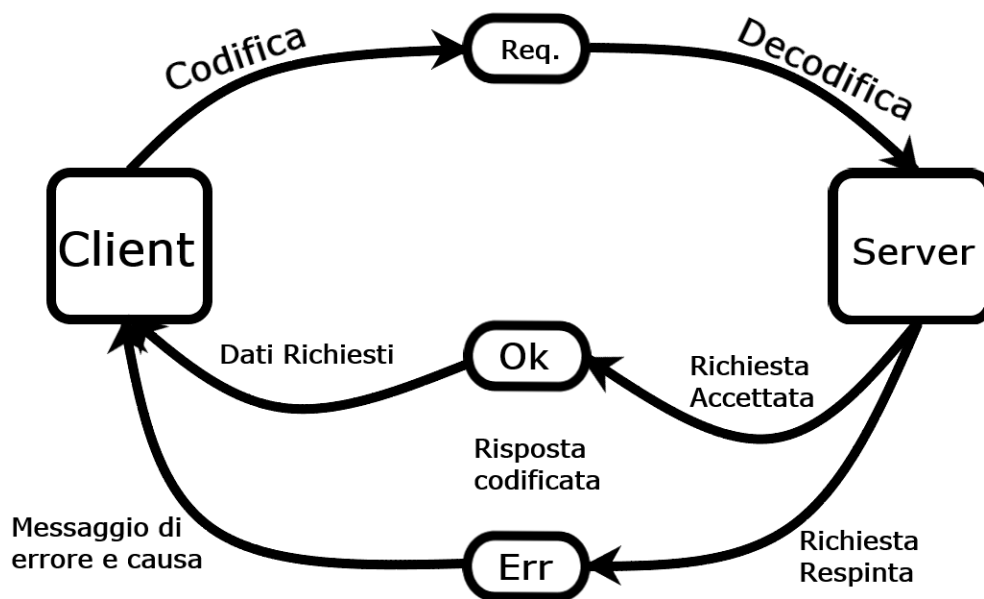


2.2 Dialogo con il Server

Il client interagisce con il server tramite richieste HTTP principalmente di tipo POST, utilizzando la classe `HTTPConnector`⁶.

Le richieste e le relative risposte sono esclusivamente codificate in formato Json e seguono il seguente schema di base:

⁶ Spiegazione nella sezione 2.3.4



Le risposte generate dal server utilizzano la classe `JSONResponse.java`⁷ e le rispettive discendenti:

- `JSONResponseUser.java`
- `JSONResponseMenu.java`

Alla ricezione lato Client la risposta viene deserializzata in un oggetto di tipo `JsonObject` e a seconda del valore del campo “ok” si agisce di conseguenza.

2.3 Scelte Progettuali e Meccanismi

2.3.1 Punti

L'utente accumula punti da utilizzare negli ordini successivi piazzando ordini da almeno 15€ (*1 pt. = 15€*), questi punti vengono spesi alla schermata finale, facendo scegliere all'utente l'importo e valgono uno sconto di 0.75€ ognuno.

⁷ Spiegazione nella sezione 2.3.5

2.3.2 Menu Pranzo e Cena

I due menu si differenziano per la quantità di piatti in essi contenuti, quello per la cena ne ha una quantità maggiore.

2.3.3 Orario per la consegna

Alla schermata finale si sceglie la fascia oraria per la consegna dell'ordine, le fasce orarie sono scandite a intervalli di 15' e si possono effettuare un massimo di 4 ordini per fascia oraria, qualora non ci fosse spazio, l'interfaccia informa l'utente tramite un messaggio a schermo.

2.3.4 HTTPConnector.java

La classe HTTPConnector.java è stata creata con lo scopo di migliorare la leggibilità del codice e ottenere una programmazione modulare.

La suddetta classe contiene 3 metodi per effettuare la connessione al Server mediante richieste HTTP, rispettivamente sfruttando i protocolli **POST** e **DELETE**, e ricevendo come parametro il path della API corrispondente all'interno del Server.

2.3.5 JSONResponse.java

La classe JSONResponse.java ha lo scopo di costruire una risposta lato server, strutturata nel seguente modo:

- Flag ok: indica se la richiesta è andata a buon fine
- msg: una Stringa che contiene l'esito della richiesta
- err: un valore numerico per differenziare i vari tipi di errore, valori diversi da 0 indicano che qualcosa non ha funzionato correttamente.

Per elasticità sono state realizzate delle sottoclassi di JSONResponse che seguono

lo stesso schema, con la differenza che hanno un campo extra che contiene un Java Bean il quale rappresenta il *'payload'* della risposta.

2.3.6 JSONReader.java

La classe JSONReader si occupa di ricevere e deserializzare la risposta del server, restituendo un `JsonObject`.

2.3.7 Database Attivo

Per permettere il funzionamento dell'applicazione, quando si inizializza il DB viene creato un *event*⁸ che scatta ogni giorno a *mezzanotte*, il quale si occupa di effettuare la *truncate* della tabella relativa agli ordini, cosicché il giorno seguente si possano conteggiare gli ordini relativi a una certa fascia oraria in maniera consistente.

2.3.8 Unit Tests

Il client testa la connessione al DB e al server con i metodi POST e DELETE, ovviamente nel caso in cui il server sia spento falliscono a prescindere.

2.3.9 Cambio lingua

L'utente può scegliere dalla schermata di login se utilizzare l'applicazione in italiano o in inglese.

Il meccanismo si avvale della classe `XStream` per leggere le etichette di ciascun `Button`, `Label`, ecc. da un file XML (uno per lingua) attraverso una classe serializzabile (implementa `Serializable.java`) `Linguaggio.java`

⁸ File: `'src\main\resources\buildDB.sql'`

2.3.10 Arrotondamento

Nei calcoli del *prezzo totale* viene adoperato un arrotondamento alla prima cifra decimale per via degli eventuali errori commessi dai calcoli con le operazioni aritmetiche che coinvolgono numeri in virgola mobile.

Il calcolo per l'arrotondamento è il seguente: $\tau_{arr} = \frac{[(\tau \times 100)]}{100}$

dove τ indica il prezzo possibilmente affetto da errori di calcolo.

Nel codice si fa uso della funzione `Math.round()` e di un cast a `float`

2.3.11 ParameterException.java

La classe `ParameterException.java` estende `Exception.java` e contiene un campo che rappresenta il codice di errore dell'eccezione sollevata.

Viene lanciata dal server nelle API `'user/login'` e `'user/sign'` per segnalare errori di *nome utente non disponibile* o *credenziali errate*.

3 Server

3.1 Avvio

Il server non può avviarsi se non esiste ancora il DB.

Le API sono esposte con una logica relativa alla richiesta:

- `/user:`
 - `/sign` POST
 - `/login` POST
- `/dish:`
 - `/list` POST
 - `/starred` POST
 - `/makestarred` POST
 - `/removestarred` DELETE

- /order
 - /confirm POST
 - /choose POST

3.2 Dialogo con il DB

Per generare le risposte, il Server utilizza le interfacce fornite da `CrudRepository` per le query semplici, mentre vengono utilizzati `Statement`, `PreparedStatement` e `Connection` per le query complesse.

3.3 Unit Test

Il server testa la connessione al DB