Mario Tortorici

MATRICOLA: 0737892

EMAIL: mario.tortorici@community.unipa.it

SMART VENDING MACHINE

GENERAL DESCRIPTION

La web-app permette il funzionamento di un sistema di distributori automatici con cui è possibile interagire online.

Tale applicazione consiste in una SPA (Single-Page-Application), dotata di un form di **login** per consentire l'accesso ai vari utenti e un form di **registrazione** per permettere ad un nuovo cliente di registrarsi alla piattaforma.

Una volta effettuata l'autenticazione, l'utente sarà in grado di accedere alla propria **area privata** dedicata, distinta in base alla propria tipologia.

Il sistema prevede **due** tipologie di utente: **Cliente** e **Distributore**.

Con **Cliente** si intende quel particolare tipo di utente che, previa registrazione, può stabilire un'**associazione** con un Distributore e procedere all'acquisto dei prodotti.

Per **Distributore**, invece, si intende quell'utente che rappresenta il dispositivo attraverso cui i Clienti acquistano i prodotti.

Tecnologie utilizzate

1. Classi **Java Servlet**, per l'accettazione di richieste http, per la costruzione delle relative pagina web di risposta e per il processo di validazione ed elaborazione dei dati inseriti dal client
2. **HTML**, **CSS**, **JavaScript** (**jQuery** e **AJAX**), per la gestione dell'interfaccia utente e l'invio delle richieste http
3. **Apache Tomcat 10**, quale middleware per il deployment della web application

4. DB relazionale **MySQL** per la memorizzazione dei dati degli utenti
5. Ambiente di sviluppo integrato (IDE) **IntelliJ IDEA** per la stesura, il debugging del codice e la gestione delle risorse.

FUNCTIONAL REQUIREMENTS

Generali

1. Dovrà presentare in primis un'interfaccia per il login dell'utente
2. In alternativa, dovrà permettere una procedura per la registrazione del Cliente
3. Una volta effettuato con successo il login, dovrà creare una interfaccia personalizzata in base al tipo di utente:
    a. **Cliente**: essa sarà caratterizzata dalla presenza di un container contenente vari button che consentiranno di:
        i. associare/dissociare un distributore
        ii. ricaricare il saldo del wallet
        iii. modificare la password di accesso
        iv. visualizzare la lista degli ordini effettuati
        v. effettuare il logout
    b. **Distributore**: essa sarà caratterizzata dalla presenza di una navbar in cui verranno visualizzati lo **stato** e la **posizione** del distributore e di un container contenente vari button che consentiranno di:
        i. Scegliere il metodo di acquisto (**Contanti** o tramite **App**). La prima tipologia è stata pensata per adattare al meglio l'interfaccia ad un contesto reale, non richiede l'associazione del cliente al distributore (quindi anche chi non è registrato alla piattaforma potrà comunque acquistare i prodotti) e richiede che lo stato del distributore sia **libero** (**true**). L'acquisto tramite App, invece, prevede che il pagamento avvenga tramite il wallet virtuale del cliente e richiede che egli sia registrato alla piattaforma ed associato al distributore.
        ii. Selezionare la tipologia del prodotto (**Snack** o **Bevanda**)
        iii. Selezionare il prodotto desiderato ed acquistarlo. In ogni step precedente all'erogazione sarà sempre possibile annullare l'operazione o cambiare la scelta del prodotto. All'erogazione, il cliente verrà reindirizzato alla pagina iniziale, dove potrà scegliere di acquistare altri prodotti e, nel caso in cui egli sia associato, di dissociarsi dal distributore.

4. La validazione dei dati immessi avverrà sia lato Client attraverso gli attributi HTML, sia lato Server all'interno delle Servlet. Questa duplice validazione servirà per fornire usabilità e sicurezza all'applicazione e fornirà i parametri da passare a dei Prepared Statement per effettuare operazioni sul DB.

Client Side

1. Tecnologie utilizzate per lo sviluppo **lato Client**:
    a. HTML & CSS
    b. JavaScript
    c. AJAX
    d. jQuery
    e. Bootstrap, quale framework per gli stili dell'interfaccia.
2. Viste **Utente**:
    a. **loginForm**: contiene un form per l'inserimento delle credenziali di accesso (**email** e **password**), due button per effettuare il submit o il reset di esso e una nav contenente il link per richiedere il form di registrazione
    b. **registerForm**: contiene un form per l'inserimento dei parametri di registrazione (**nome**, **cognome**, **data di nascita**, **email** e **password**), due button per effettuare il submit o il reset di esso e una nav contenente il link per richiedere il form di accesso.
3. Viste **Cliente**:
    a. **homeCliente**: contiene un messaggio di benvenuto, un container in cui verrà visualizzato il saldo disponibile sul wallet ed un insieme variabile di button ("**Associa Distributore**", "**Account**" e "**Logout**" se lo stato del Cliente è disponibile, "**Associa Distributore**" e "**Account**" altrimenti) per effettuare le operazioni descritte in precedenza
    b. **associaDistributore**: contiene un form in cui inserire l'id del distributore a cui il cliente intende associarsi e due button ("**Associa**" e "**Annulla**") che consentono, rispettivamente di inviare la richiesta alla servlet corrispondente e di annullare l'operazione
    c. **modificaPassword**: contiene un form per la modifica della password di accesso, caratterizzato dalla presenza di due campi in cui inserire la password **precedente** e quella **nuova** e da due button ("**Modifica**" e "**Annulla**") che consentono, rispettivamente, di inviare la richiesta alla servlet corrispondente e di annullare l'operazione
    d. **ordiniEffettuati**: contiene una tabella in cui verrà visualizzata la lista di tutti gli ordini effettuati dal cliente

    e. **ricaricaWallet**: contiene un form per la ricarica del Wallet del cliente, caratterizzato dalla presenza di campi in cui inserire il numero della carta di credito, l'intestatario, la data di scadenza (mese e anno), il cvv e l'importo da ricaricare. Saranno inoltre presenti due button ("**Ricarica**" e "**Annulla**") che consentono, rispettivamente, di inviare la richiesta alla servlet corrispondente e di annullare l'operazione.
4. Viste **Distributore**:
    a. **homeDistributore**: è una vista che varia in funzione dello stato del distributore: se esso è **disponibile** (**true**), verranno visualizzati due button ("**Acquista in Contanti**" e "**Acquista tramite App**") che consentono, rispettivamente, di acquistare in contanti (senza effettuare l'associazione al distributore) o con il wallet (effettuando l'associazione al distributore). Altrimenti verranno visualizzati due button ("**Acquista**" e "**Dissocia**") che consentono, rispettivamente, di acquistare i prodotti usando il saldo del wallet e di dissociare il cliente dal distributore
    b. **sceltaProdotto**: contiene due button ("**Snack**" e "**Bevanda**") che consentono, rispettivamente, di visualizzare i prodotti rientranti nella categoria "snack" e quelli rientranti nella categoria "bevande". Inoltre sono presenti altri due button ("**Indietro**" e "**Annulla**") che facilitano la navigazione e l'annullamento dell'operazione
    c. **elencoBevande**: contiene tanti button quante sono le bevande che è possibile ordinare e due button ("**Indietro**" e "**Annulla**") che facilitano la navigazione e l'annullamento dell'operazione
    d. **elencoSnacks**: contiene tanti button quanti sono gli snack che è possibile ordinare e due button ("**Indietro**" e "**Annulla**") che facilitano la navigazione e l'annullamento dell'operazione
    e. **erogazioneProdotto**: è costituita da un container con una scritta per comunicare al Cliente che il prodotto è in erogazione
    f. **prodottoErogato**: è costituita da un container con una scritta per comunicare al Cliente che il prodotto è stato erogato
5. Componente **footer**: componente posto in fondo alla pagina in cui vengono riportati i credits dell'applicazione

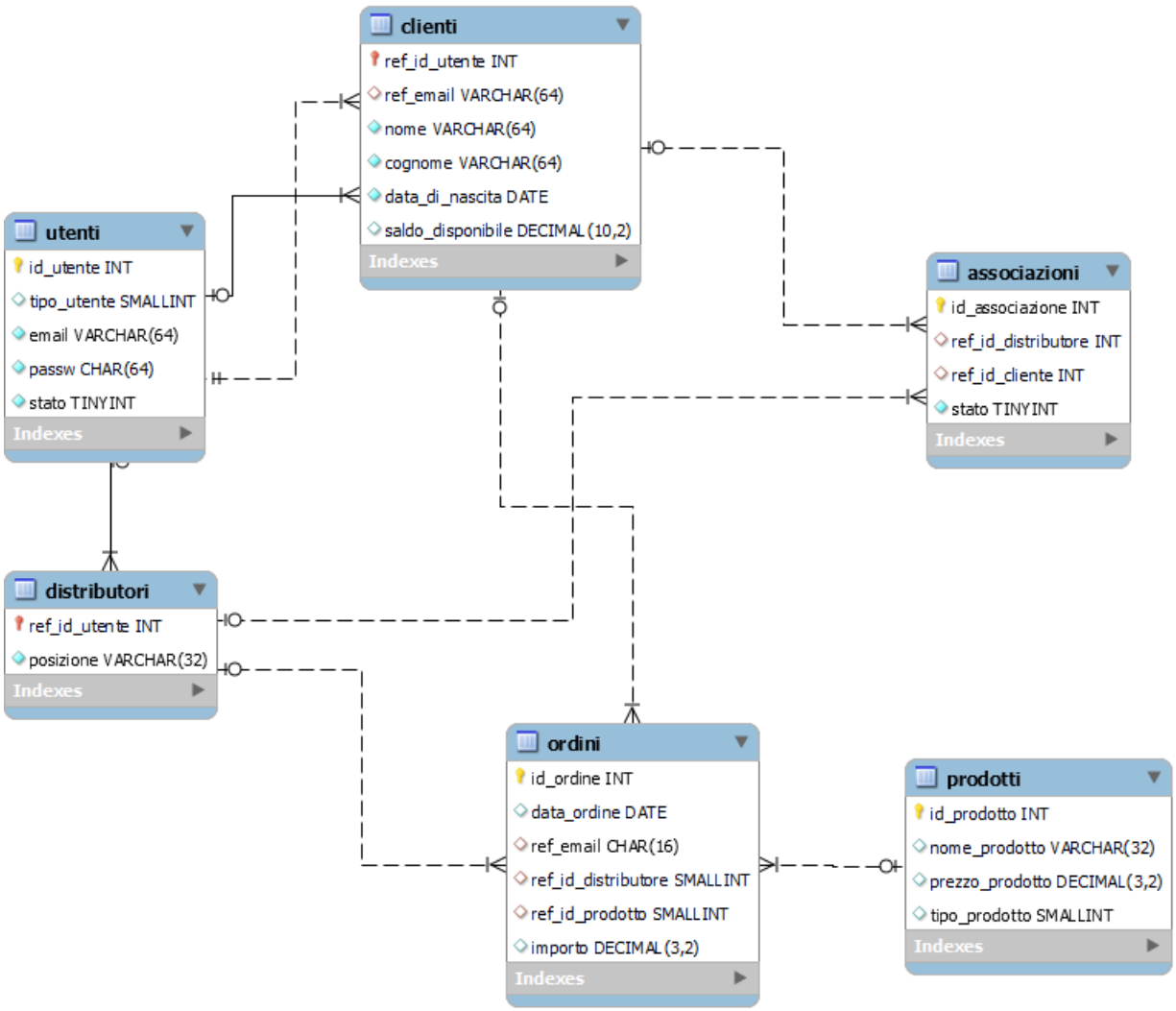
Server Side

1. Tecnologie utilizzate per lo sviluppo **lato Server**:
    a. Java Servlet
    b. JSP.
2. Il package "**controllers**" è costituito dalle classi **Java Servlet** che si occuperanno della gestione delle viste jsp, del reindirizzamento, del controllo e del confronto dei dati presenti nel DB, delle sessioni (inserimento, modifica e rimozione) e dei messaggi di errore. Esso è a sua volta suddiviso nei package
    a. "**cliente**": contiene tutte le classi **Java Servlet** utili al **Cliente**:
        i. **AssociaDistributoreServlet**: accetta richieste POST e permette di instaurare una associazione tra il Cliente che la richiede e un determinato Distributore
        ii. **ModificaPasswordServlet**: accetta richieste POST e permette di controllare che i parametri in input soddisfino le condizioni per effettuare la modifica della password del Cliente che la richiede
        iii. **RegistraClienteServlet**: accetta richieste POST e permette di controllare che i parametri in input soddisfino le condizioni per effettuare la registrazione di un Cliente
        iv. **RicaricaWalletServlet**: accetta richieste POST e permette di ricaricare il wallet del Cliente che la richiede
    b. "**distributore**": contiene tutte le classi **Java Servlet** utili al **Distributore**:
        i. **EffettuaPagamentoServlet**: accetta richieste POST e permette di controllare che i parametri in input soddisfino le condizioni per acquistare un prodotto
    c. "**utente**": contiene tutte le classi **Java Servlet** utili sia al **Cliente** che al **Distributore**:
        i. **DissociaUtentiServlet**: accetta richieste GET e termina l'associazione tra un Cliente e un Distributore
        ii. **LoginUtenteServlet**: accetta richieste POST e permette il login di un Utente
        iii. **LogoutUtenteServlet**: accetta richieste GET e termina la sessione di un Utente
        iv. **RichiediPagineServlet**: accetta richieste GET e, in base ad un parametro ricevuto nella richiesta, permette lo smistamento della stessa verso i componenti più indicati.

3. Il package “**db**” è costituito dalle classi:
    a. **DBConnector**: gestisce la connessione al DBMS facendo uso del data source “**context.xml**”
    b. **DBQueries**: contiene tutte le query necessarie al funzionamento dell'applicazione
4. Il package “**models**” è costituito da **Java Beans**, mappando le relazioni nel DB:
    a. **Utente**: contiene le caratteristiche che Clienti e Distributori hanno in comune, ovvero **id**, **email**, **password**, **tipo utente** e **stato**
    b. **Cliente** extends **Utente**: aggiunge alle caratteristiche di base dell’Utente quelle specifiche per il Cliente
    c. **Distributore** extends **Utente**: aggiunge alle caratteristiche di base dell’Utente quelle specifiche per il Distributore
    d. **Prodotto**: racchiude le informazioni sui prodotti inseriti in ciascun Distributore
    e. **Ordine**: racchiude le informazioni sugli ordini effettuati da ciascun Cliente
5. Il package “**utils**” è costituito dalle classi:
    a. **ConversioniData**: contiene dei metodi che, ricevuta in input una stringa, ne formattano il contenuto in un particolare formato. Questa classe è utile per salvare senza problemi le date nel DB.
    b. **SHA256**: contiene un metodo che, ricevuta in input una stringa, genera l’hash della stessa. Questa classe serve ad evitare di salvare le password in chiaro nel DB.

DATA MODEL

Schema logico del DB



NOTE

Eventuali viste verranno, se necessario, aggiunte per migliorare la completezza dell'applicazione.