SMART VENDING MACHINE

TEAM DI SVILUPPO

Vincenzo Fardella (0738045)

GENERAL DESCRIPTION

L'applicazione consente il funzionamento di un sistema di distributori automatici tramite interazioni online. La web-app è una Single Page App che si presenta tramite una parte pubblica, caratterizzata da un form per il login degli utenti e da un form per l'iscrizione dei clienti, e da una parte privata, accessibile solo previo accesso e distinta in base al tipo di utente che effettua l'accesso. Distinguiamo infatti tra:

- clienti, ossia coloro che, opportunamente registrati, possono instaurare una connessione con un distributore e procedere all'acquisto di beni;
- macchine, ossia i dispositivi attraverso cui i clienti effettuano l'ordine di un dato prodotto.

La parte di applicazione destinata ai clienti consente l'iscrizione e l'accesso alla propria area riservata, la possibilità di modificare i propri sistemi di pagamento (aggiungere/rimuovere carte di pagamento) e instaurare/interrompere una connessione con una macchina.

La parte di applicazione destinata alle macchine permette, successivamente all'accesso, di consultare lo stato del dispositivo e di eseguire le funzionalità necessarie alla presentazione e alla vendita dei prodotti.

SYSTEM ARCHITECTURE

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. Web Filters, per filtrare le richieste fatte al server e verificare che siano coerenti con le condizioni richieste dal sistema;
3. HTML, CSS (definito dall'utente e framework Bootstrap), JavaScript e jQuery, per la gestione dell'interfaccia utente;
4. Apache Tomcat, quale middleware per il deployment della web application.
5. Database relazionale MySQL, per fornire persistenza all'applicazione.

6. IntelliJ IDEA, come IDE per la scrittura del codice e la gestione delle risorse specificate precedentemente

FUNCTIONAL REQUIREMENTS

Generali:

1. dovrà presentare in primis un'interfaccia per il login degli utenti (sia cliente che macchina);
2. in alternativa, dovrà permettere una procedura per l'iscrizione (solo clienti);
3. effettuato il login, dovrà generare una vista personalizzata:
    a. per il cliente, essa sarà composta da una navbar tramite cui richiedere contenuti al server o procedere al logout, e da un container, che ospiterà effettivamente i contenuti;
    b. per la macchina, essa sarà composta da informazioni generali sulla macchina (numero, indirizzo e disponibilità), più la lista dei prodotti disponibili
4. la validazione degli input avviene tramite attributi HTML (lato client) e all'interno delle servlet, in modo da fornire parametri opportunamente elaborati a dei prepared statement necessari per le operazioni sul DB

Client side:

1. Vista loginForm: contiene un modulo per l'accesso utente (corredato di pulsante per il reset dello stesso) e un link per richiedere il modulo di iscrizione;
2. Modale loginFail: contiene un messaggio di errore qualora l'accesso non sia stato possibile;
3. Vista subscribeForm: contiene un modulo per l'iscrizione dei clienti (corredato di pulsante per il reset dello stesso) e un link per richiedere il modulo di login. I dati saranno controllati, dove necessario, tramite attributi HTML type, pattern e required;
4. Modale modaleIscrizione: è un box che viene mostrato al cliente che vuole iscriversi, mostra opportuni messaggi a seconda che l'iscrizione sia andata a buon fine o meno;
5. Vista dashboard: contiene un messaggio di benvenuto al cliente, un insieme di pulsanti per effettuare le operazioni di riguardo e, se il cliente ha registrato una o più carte, una tabella con le carte ad esso associate;
6. Vista aggiungiCarta: contiene un modulo per associare una carta al cliente che lo invia (il saldo viene generato casualmente dal server), con i dati che vengono validati lato client tramite attributi HTML type, pattern e required;
7. Modale modaleAggiungiCarta: è un box modale in cui viene mostrato un messaggio al cliente che prova ad aggiungere una carta per comunicare l'eventuale successo o fallimento dell'inserimento;
8. Vista rimuoviCarta: contiene un input di selezione per rimuovere eventuali carte associate all'utente, con la conferma data dall'inserimento della password dell'utente;

9. Modale modaleRimuoviCarta: è un box modale mostrato al cliente che prova a rimuovere una carta ad esso associata, mostrando opportuni messaggi a seconda del successo o del fallimento della rimozione;
10. Vista connettiMacchina: contiene un input di selezione con le macchine libere al momento, da cui l'utente può sceglierne una per instaurare la connessione;
11. Vista homeMacchina: visualizzerà numero, indirizzo e stato della macchina, più la lista dei prodotti disponibili: il form per effettuare l'ordine è mostrato solo se la macchina risulta connessa a un cliente;
12. Footer: semplice componente posto in fondo alla pagina, vengono riportati i credits dell'applicazione;
13. Componente head della pagina iniziale: contiene tutti i tag necessari a importare stili CSS (Bootstrap e personalizzati) e script (jQuery).
14. Aggiornamento automatico di alcuni input tag nei form tramite JS (es. nell'iscrizione, la data massima per essere maggiorenni sarà impostata tramite JS).
15. Validazione input client side (tramite attributi HTML).
16. Tecnologie:
    a. HTML
    b. CSS
    c. JavaScript
    d. jQuery
    e. AJAX
    f. Framework Bootstrap per stili e script per l'interfaccia

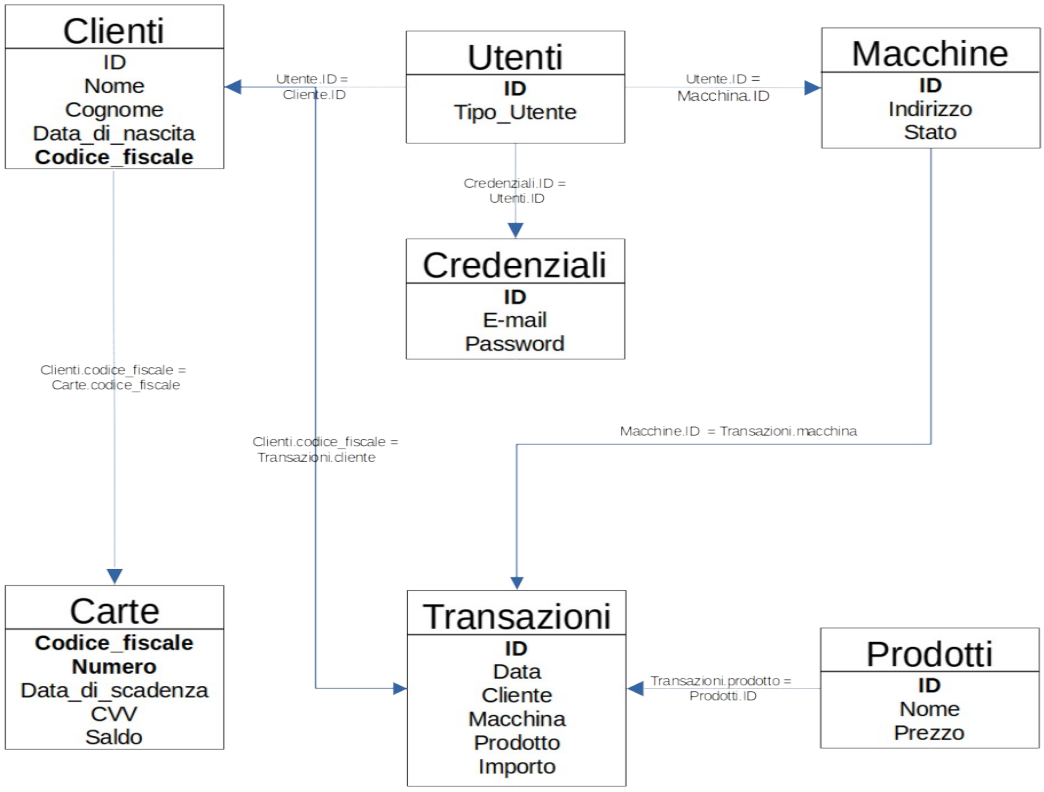
Server side:

1. Il model è costituito da Java Beans, mappando le relazioni nel DB
    a. Utente, racchiude le caratteristiche comuni a clienti e macchine (id e tipo dell'utente);
    b. Credenziali, racchiude le credenziali associate a un id;
    c. Cliente extends Utente, aggiunge delle caratteristiche specifiche per il cliente;
    d. Macchina extends Utente, aggiunge delle caratteristiche specifiche per la macchina;
    e. Carta, racchiude le informazioni di una carta associata al codice fiscale di un cliente;
    f. Prodotto, racchiude le informazioni base sui prodotti in vendita in ogni macchina;
2. Classe InterazioniDB, che gestisce la connessione con il database e contiene le query necessarie al funzionamento dell'applicazione;
3. Classe RequestContentServlet, che riceve un parametro nella richiesta e smista la richiesta verso i componenti opportuni;
4. Servlet Accesso: accetta richieste POST e permette l'accesso di un utente;
5. Servlet Uscita: accetta richieste GET e termina la sessione di un utente;
6. Servlet Iscrizione: accetta richieste POST e permette l'iscrizione ad un nuovo cliente;
7. Servlet AggiungiCarta: accetta richieste POST e permette l'associazione di una carta al cliente che la richiede;

8. Servlet RimuoviCarta: accetta richieste POST e permette la dissociazione di una carta dal cliente che la richiede;
9. Servlet ConnessioneMacchina: accetta richieste POST e permette di instaurare una connessione tra il cliente che la richiede e una macchina;
10. Le classi del package 'filters' permettono di filtrare le richieste secondo opportune condizioni necessarie alla corretta fruizione dell'applicazione;
11. Servlet Ordine: accetta richieste POST e permette di controllare che i parametri in input soddisfino le condizioni per effettuare un ordine;
12. Tecnologie:
    a. Java Servlet
    b. JSP
    c. Java Web Filter

DATA MODEL

Schema logico per il database.



N. B. : Le parole in **grassetto** sono le chiavi primarie di ogni tabella.