**Applicazioni Web: Linguaggi e Architetture**

Relazione di Laboratorio

Bertoldi Noemi

Musetta Garion

1) ISTRUZIONI PER L’AVVIO:

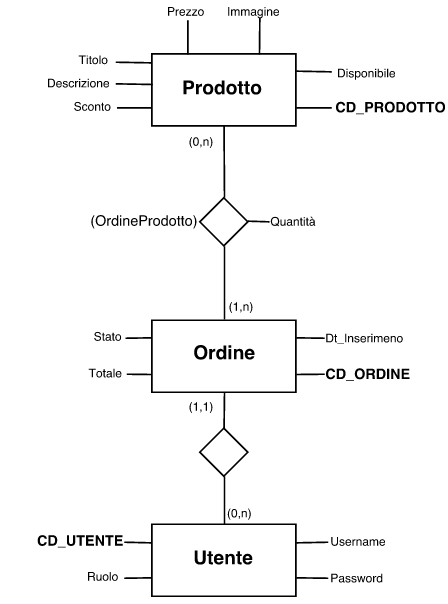
* Assicurarsiche il servizio SQL SERVER (SQLEXPRESS) sia attivo, prima di far partire l’applicazione
* Modificare eventualmente la stringa di connessione che si trova in ../AppWebProj/appsettings.json, sostituendo i campi relativi a “User ID” e “Password" con le proprie credenziali SQL Server Authentication.  
  Potrebbe essere necessario sostituire “User ID” e “Password” con “Trusted\_Connection=Yes” nel caso si usasse Windows Authentication.
* Eseguire lo script di creazione del database riportato nel file ddl.txt
* Aprire lo script di popolamento delle tabelle (dml.txt),sostituire il percorso riportato nella clausola DECLARE @ROOT VARCHAR(500) con il percorso su cui si è salvata la cartella del progetto ed eseguire lo script, ad esempio: ‘C:\Users\(user)\...\appwebcs-master’;
* Aprire la soluzione con Visual Studio (potrebbe essere necessario aprirlo come amministratore) eseguire la BUILD del progetto ed infine eseguire il programma.

2) SCELTE IMPLEMENTATIVE

* Si è scelto di non usare frameworkcome Kendo e AspNetAuthenticationper cercare di capire meglio il funzionamento di ASP.NET a più basso livello, perciò sono state implementate manualmente le pagine che mostrano i risultati delle query e il meccanismo di autenticazione, rispettivamente,usando delle semplici tabelle html econ il controllo del ruolo dell’utente salvato in sessione. Quest’ultimo punto permette di bloccare l’accesso alle pagine riservate da parte di utenti non autorizzati.
* Il Carrello dei prodotti è memorizzato interamente in sessione, rendendolo così temporaneo. Il Carrello, infatti, esiste sia per utenti non loggati, sia perutenti che abbiano “user” come ruolo. Questa scelta implica il fatto che una volta “chiusa” la sessione, non sarà possibile recuperare il contenuto del Carrello, pertanto l’acquisto dovrà essere terminato durante la sessione.
* Si è cercato, in generale, di usare molte tecniche diverse per cercare di imparare il più possibile non scegliendo necessariamente la tecnica migliore.

3) DESCRIZIONE MODELLO RELAZIONALE

Lo scherma E-R segue quanto illustrato prima: il carrello non è memorizzato sul database e pertanto sono presenti solo 3 entità: Utente, Ordine e Prodotto. La tabella OrdineProdotto (nel database) è il risultato della relazione n : m fra Ordine e Prodotto.

**

4) DESCRIZIONE MODELLO A OGGETTI

Per ogni dettaglio più specifico sull'implementazione si faccia riferimento direttamente sul codice, dove ogni classe e metodo è corredato da opportuni commenti.

Il progetto segue il pattern MVC:

* **Il Model** è il risultato del mappaggio automatico del database in oggetti. Per agevolare l'operazione si è seguito il tutorial: http://www.learnentityframeworkcore.com/walkthroughs/existing-database

Le classi **Utente, Ordine, OrdineProdotto e Prodotto** rappresentano i Model del contesto.

* **I Controllers** modificano e processano i model e li passano alle Views. E' presente un controller per ogni categoria di pagine (uno per ogni sub-folder delle view), nel quale ogni view corrisponde a un metodo di un controller. Quando una pagina è richiesta, il controller ascolta la richiesta, la processa eseguendo operazioni specifiche per ogni pagina, e poi indirizza alla View adatta, passando un Model.

In ordine alfabetico:

* **CarrelloController** gestisce le operazioni sul carrello: aggiunta, modifica e rimozione prodotti
* **CrudController** è una "superclasse" che raccoglie tutti i controller che eseguono operazioni CRUD sui model: questi controllers (OrdineController, ProdottoController e UtenteController) fanno uso dei metodi del CrudController per eseguire le suddette operazioni.
* **HomeController** gestisce la home page e fornisce la query sulla top 10 dei prodotti venduti questo mese.
* **OffertaController** pilota la view Offerte, che espone i prodotti in offerta.
* **OrdineController** permette la creazione (e la modifica, da parte degli admin) di nuovi ordini da parte degli utenti.
* **PrivateHomeController** è collegato all'area di amministrazione degli utenti admin.
* **ProdottoController** controlla tutte le pagine relative ai prodotti: ricerche, catalogo e dettagli prodotto. Gestisce anche le modifiche dei dati di un prodotto da parte di utenti admin.
* **UtenteController** permette ai nuovi utenti di registrarsi e agli admin di modificare i ruoli degli altri utenti.
* Le **Views** rappresentano la parte di front-end e l'esposizione dei dati preparati dai controllers. Permettono anche l'interazione con i controller stessi tramite links e forms, che chiamano alcune funzioni dei controller.