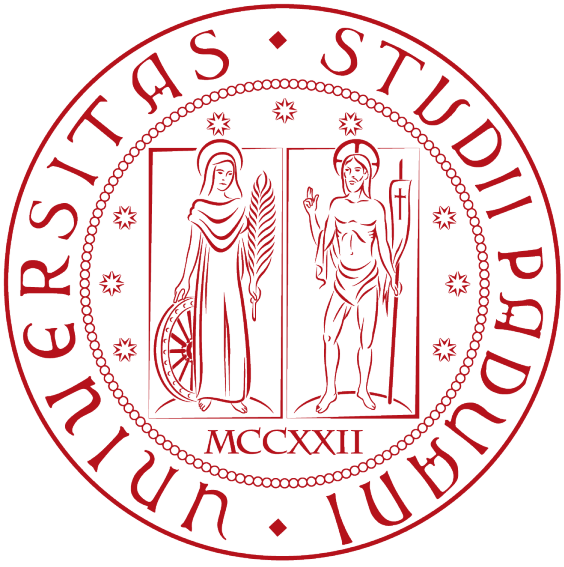
Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA

Corso di Mobile Programming and Multimedia



Relazione del progetto SmartOrder

Studenti:
Alberto Bezzon 1211016
Tommaso Carraro

Anno Accademico 2018-2019

Indice

Elenco delle figure

1 Introduzione

Il progetto consiste nello sviluppo di un'applicazione mobile, chiamata SmartOrder, progettata per permettere agli utenti registrati di effettuare ordini presso il proprio fornitore. L'applicazione è stata sviluppata esclusivamente per smartphone, al momento soltanto per l'ambiente Android, utilizzando il framework cross-platform PhoneGap. Per l'ambiente iOS è stato realizzato il progetto per XCode ma non essendo il gruppo in possesso di un Mac, non è stato possibile realizzare la versione iOS di SmartOrder. L'applicazione non prevede una registrazione esplicita, infatti le aziende che intendono acquistare il servizio devono comunicarlo al team di SmartOrder. Al momento della richiesta devono essere fornite le anagrafiche degli utenti che il fornitore intende registrare e in questo modo verranno creati gli account per tutti gli utenti, compresi di credenziali. Inoltre devono essere forniti i dati relativi a tutti gli articoli che il fornitore desidera rendere disponibili per gli utenti tramite il servizio. Una volta ricevute le credenziali, via mail, gli utenti possono iniziare ad usufruire del servizio. Il tutto sarà perfettamente configurato per fare in modo che ogni utente veda solamente gli articoli venduti dal proprio fornitore. Tramite l'applicazione sarà possibile:

- ricercare prodotti e aggiungerli al carrello;
- scansionare il codice a barre di un prodotto che si ha già acquistato per aggiungerlo al carrello più velocemente;
- modificare o eliminare i prodotti in carrello;
- inviare un ordine presso il proprio fornitore;
- visualizzare le informazioni relative a tutti gli ordini effettuati presso il proprio fornitore.

L'idea è nata per cercare di soddisfare un bisogno reale che proviene dallo zio di uno dei membri del gruppo. È stato richiesto se fosse possibile velocizzare il processo di rifornimento di libri di una libreria una volta terminati. L'idea è che è possibile velocizzare questo processo scansionando il codice a barre di un libro quando si è a conoscenza del fatto che sta per terminare. La possibiltà di aggiungere gli articoli dall'inventario è stata implementata per permettere all'applicazione di essere robusta rispetto all'aggiunta di nuovi clienti, in quanto essi non avendo ancora acquistato prodotti non hanno la possibilità di scansionare il codice a barre per ordinare i prodotti.

2 La scelta del framework cross-platform

Il framework cross-platform scelto per lo sviluppo dell'applicazione è PhoneGap. La scelta è ricaduta su questo framework in quanto un membro del gruppo aveva già familiarità con le tecnologie offerte dallo stesso. Inoltre il framework non avrebbe richiesto l'apprendimento di nuove tecnologie per l'implementazione dell'applicazione, in quanto è possibile utilizzare le tecnologie web. Altre considerazioni sulla scelta del framwork sono state inserite nelle conclusioni di questa relazione.

3 Accorgimenti sulle tecnologie web

Siccome è stato scelto il *framework PhoneGap*, il gruppo ha utilizzato i linguaggi *HTML5*, *CSS3* e *JavaScript*. In particolare, *HTML5* è stato scelto perché include un insieme di funzionalità

che permettono di valorizzare le interfacce *mobile*. Alcune di queste evidenziano come *HTML5* sia già per sua natura orientato al *mobile*. In particolare *HTML5* fornisce *API* per:

- **eventi *touch***: mentre i meccanismi di input nei PC consistono per lo più nella tastiera e nel mouse, nei dispositivi mobili quasi tutto passa per il *touch screen*, e avere funzionalità comode per gestire questo strumento consente un'interazione più ricca e senza limitazioni per l'utente. Le gestualità da attuare su un *display*, nel mondo *mobile*, costituiscono un vero e proprio linguaggio fondamentale nella *user experience*;
- **controllo batteria**: considerata l'importanza rivestita dalle risorse energetiche, l'esistenza stessa di questa libreria nel linguaggio dimostra come il suo impiego sia particolarmente mirato al panorama *mobile*.

Ciò che ha favorito la scelta di *CSS3* sono state le *media queries*. Esse permettono di definire regole stilistiche in base alla tipologia del mezzo di visualizzazione, delle sue dimensioni e della sua attuale disposizione (*portrait* o *landscape*). Ciò influisce non solo sull'aspetto esteriore degli elementi ma anche sul loro posizionamento e quindi sulla struttura stessa dell'interfaccia.

Per quanto riguarda il linguaggio *JavaScript* si è utilizzato *JavaScript* puro, senza l'utilizzo di *framework* o *JQuery*. Una particolarità del linguaggio, detta *AJAX*, ha reso possibile eseguire chiamate all'*API* del servizio *web* di *SmartOrder*. *AJAX*, acronimo di *Asynchronous JavaScript and XML*, è una tecnica di sviluppo *software* per la realizzazione di applicazioni *web* interattive (*Rich Internet Application*). Lo sviluppo di applicazioni *HTML* con *AJAX* si basa su uno scambio di dati in *background* fra *web browser* e *server*, che consente l'aggiornamento dinamico di una pagina *web* senza esplicito ricaricamento da parte dell'utente.

4 Informazioni utili all'utilizzo dell'applicazione

Per poter utilizzare l'applicazione è necessario essere connessi ad Internet, in quanto l'applicazione scarica i dati da un server. Si fa presente che la connessione viene bloccata dalla rete Eduroam e Studenti.math.unipd.it dell'Università, quindi si consiglia di utilizzare altre reti per usufruire dell'applicazione. Per ricevere le e-mail al momento dell'invio dell'ordine è necessario che la propria mail sia certificata Amazon. Questo è un limite di Amazon AWS che può essere aggirato inviando una semplice richiesta ad AWS. Poiché non sarà possibile quindi ricevere la mail, viene di seguito fornito uno screen della stessa. La mail presenta due versioni, a seconda che debba essere inviata all'azienda o all'utente che ha effettuato l'ordine.

Infine si fa notare che il loader dell'applicazione indica che SmartOrder è in fase di scaricamento dati dal server e sta momentaneamente attendendo una rispsota per la costruzione della pagina. Il loader è stato implementato proprio per fare in modo che l'utente non interagisca con l'applicazione prima della visualizzazione della pagina. Quindi, se si premesse qualche bottone prima dell'avvenuto caricamento, potrebbero accadere problemi di rendering.

5 Progettazione frontend

Il frontend è stato realizzato mediante utilizzo di tecnologie web. HTML e CSS hanno predisposto l'interfaccia grafica, mentre JavaScript ha permesso di implementare la logica applicativa. Le pagine realizzate sono le seguenti:

- *index.html*;
- *login.html*;

- *homepage.html*;
- *order.html*;
- *article.html*;
- *articles.html*;
- *newpage.html*;
- *inventory.html*.

Ad ognuna di queste pagine è associato un foglio di stile e un file Javascript; inoltre sono presenti un file CSS e un file Javascript generali. Nelle prossime sezioni vengono descritte le funzionalità di SmartOrder.

5.1 Login e tutorial

L'applicazione, una volta installata, è utilizzabile. All'avvio della stessa viene mostrata la pagina di login per accedere al servizio. In questa schermata devono essere inserite le credenziali fornite dal proprio fornitore in fase di registrazione. La pagina presenta una checkbox "Resta collegato" per evitare che ad ogni apertura dell'applicazione venga richiesto il login. In seguito al login, soltanto al primo avvio, viene mostrato un breve tutorial sul funzionamento dell'applicazione. In fondo al tutorial è presente un bottone che permette di iniziare ad utilizzare l'app. Nel caso in cui un utente si trovasse in difficoltà durante l'utilizzo dell'applicazione, è possibile trovare il tutorial nel menu della stessa. Completato il tutorial si viene reindirizzati al *Carrello*.

5.2 Carrello

La pagina *Carrello* è la pagina principale dell'applicazione e visualizza per ogni articolo in carrello la quantità decisa dall'utente, il prezzo e il prezzo parziale (prezzo unitario x quantità). Ogni articolo in carrello presenta due bottoni:

- **MODIFICA**: permette di modificare l'articolo. Alla sua pressione si viene reindirizzati alla pagina per la modifica dell'articolo, dove è possibile modificarne la quantità. Una volta effettuata la modifica è necessario confermarla per apportarla definitivamente, premendo sul bottone "Conferma", oppure annullarla, tramite il bottone "Annulla";
- **ELIMINA**: permette di rimuovere l'articolo dal carrello. Se tale bottone viene premuto, verrà chiesta conferma prima di effettuare l'operazione. (La spiegazione del perché viene rimandata al capitolo sul mobile design).

In questa pagina è presente inoltre un footer contenente:

- il prezzo totale di tutti gli articoli inseriti nel carrello;
- un pannello di bottoni utilizzabili per la gestione del carrello:
  - bottone *Scan*: la spiegazione di questo pulsante è presente nella sezione successiva;
  - bottone *Elimina*: permette di svuotare il carrello. In seguito alla pressione viene chiesta conferma dell'azione;
  - bottone *Invia*: permette di inviare un ordine composto dagli articoli nel carrello. In seguito all'ordine si riceverà una mail di conferma contenente le informazioni relative all'ordine effettuato.

5.2.1 Aggiunta di un articolo al carrello

È possibile aggiungere un articolo al carrello in due modi:

1. tramite scansione del codice a barre in seguito alla pressione del bottone *Scan* nella pagina *Carrello*, accedibile dal menu dell'applicazione;
2. dalla pagina *Inventario*, accedibile tramite il menu dell'applicazione, selezionando l'articolo che si desidera aggiungere e inserendone la quantità nella pagina di aggiunta dell'articolo. Una volta immessa la quantità desiderata, è necessario premere su "Conferma" affinché l'articolo venga inserito nel carrello, oppure su "Annulla" per annullare l'inserimento del nuovo articolo. Nella pagina *Inventario* è inoltre possibile ricercare un'articolo, per nome o per codice, digitando la chiave di ricerca nella textbox in alto.

La pagina per l'aggiunta di un articolo, che presenta la stessa interfaccia grafica della pagina per la modifica di un articolo, contiene le seguenti informazioni:

- nome dell'articolo: visualizzato in alto alla pagina;
- descrizione dell'articolo: visualizzata subito dopo il nome;
- pannello per l'inserimento della quantità: le modalità di interazione con questo pannello sono spiegate nella sezione dedicata al mobile design;
- prezzo dell'articolo: visualizzato alla fine della pagina. Si tratta di un campo dinamico in quanto varia a seconda della quantità inserita.

Attenzione: nel caso in cui si cercasse di aggiungere al carrello un articolo già presente in esso, il sistema mostrerà un messaggio nel quale chiederà se si vuole modificare la quantità dello stesso.

5.3 Visualizzazione ordini

Dal menu dell'applicazione è possibile accedere alla lista di tutti gli ordini effettuati premendo sulla voce *Ordini*. Alla sua pressione si verrà reindirizzati alla pagina *Ordini*, dove per ogni ordine, vengono mostrate le seguenti informazioni:

- codice: è il codice che identifica univocamente l'ordine;
- data: è la data in cui è stato effettuato l'ordine;
- totale: è il totale in euro dell'ordine.

. E' possibile consultare la lista degli articoli acquistati in un determinato ordine, tramite il bottone "DETTAGLI", situato sulla destra di ogni ordine. Alla sua pressione si verrà reindirizzati alla pagina contenente le informazioni degli articoli ordinati:

- nome;
- quantità;
- prezzo;
- prezzo parziale.

In questa pagina è possibile inoltre modificare la visualizzazione degli ordini, ordinandoli per data, dal più recente al meno recente, oppure per prezzo crescente. Per selezionare la tipologia di ordinamento è presente una select in fondo alla pagina.

6 Mobile design

In questa sezione vengono descritte le scelte implementative effettuate ai fini di una corretta progettazione dell'interfaccia dell'applicazione. Questa sezione è suddivisa in varie sottosezioni e ognuna di esse descrive un'elemento dell'interfaccia e le motivazioni alla base della sua progettazione.

La maggior parte delle pagine dell'applicazione sono progettate nel seguente modo:

- la parte superiore presenta il pulsante menu sulla sinistra e il titolo della pagina al centro;
- la parte centrale presenta il contenuto informativo;
- la parte inferiore presenta un footer che varia a seconda della pagina.

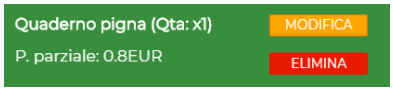
6.1 Bottoni

(a) Bottoni presenti nel footer della pagina carrello

(b) Bottoni presenti nel footer delle apgine aggiungi e modifica articolo

Figura 1: Bottoni nel footer

Nelle pagine *Carrello*, *Modifica* e *Aggiungi* sono presenti dei bottoni nella parte bassa dello schermo (Figura 1). L'applicazione richiede una frequente modifica dei dati quindi i controlli sono stati posizionati in una zona semplice da raggiungere e, seguendo la regola "Content always on top", si è preferito inserire questi bottoni in basso, infrangendo così la convenzione che su Android i controlli devono essere posizionati nella parte alta dello schermo.



(a) Bottoni nella pagina Carrello

(b) Bottoni nella pagina Ordini

(c) Bottoni nella pagina Inventario

Figura 2: Bottoni nella parte centrale delle pagine

I bottoni presenti in Figura 2 sono situati nella parte centrale dello schermo. Essendo disposti nella comfort zone sono facilmente raggiungibili e più piccoli rispetto ai bottoni del footer, in quanto si è seguita l'immagine in Figura 3. La dimensione è stata opportunamente scelta per fare in modo che l'utente non prema accidentalmente su di essi, in quanto possono cambiare lo stato del carrello oppure dare luogo all'apertura di nuove pagine. Nonostante la dimensione ridotta, sono comunque sufficientemente distanti da evitare un tap accidentali sui bottoni vicini. Infine, in caso si premesse sul bottone ELIMINA affianco ad un articolo nella pagina Carrello viene chiesta prima la conferma. Si tratta di un modo per rendere l'operazione reversibile.

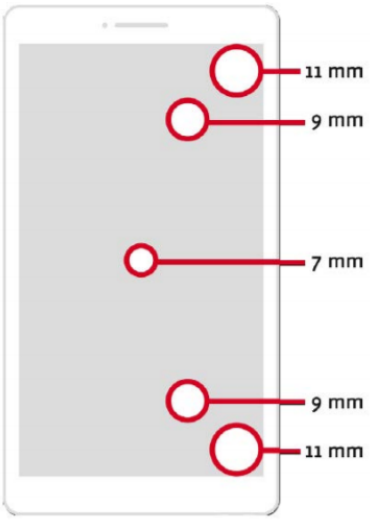


Figura 3: Dimensione vs. posizione dei bottoni

6.2 Menu

Figura 4: Menu dell'applicazione

Il menu dell'applicazione è rappresentato in Figura 4. Per accederci è presente il classico bottone "hamburger" in alto a sinistra di ogni pagina. Il menu è suddiviso in due parti:

- informazioni relative all'utente che ha effettuato il login nell'applicazione. In particolare vengono visualizzati il nome dell'utente e il codice azienda del fornitore dello stesso. Queste informazioni non sono modificabili e pertanto sono state disposte nella parte alta del menu;
- insieme di link che permettono di navigare tra le pagine dell'applicazione. In particolare sono disponibili i link al carrello, alla lista degli ordini e all'inventario. Inoltre è possibile

effettuare il logout dall'applicazione o visionare nuovamente il tutorial. Questi bottoni sono stati posizionati nella parte centrale del menu (comfort zone), in modo da essere facilmente accessibili quando il menu risulta aperto.

6.3 Progressive disclosure

In generale, è una buona prassi non affollare le interfacce. Al fine di perseguire questo obiettivo si è utilizzato il principio della progressive disclosure:

- nella pagina *Ordini* è possibile visualizzare le informazioni essenziali per poter identificare un ordine. Per accedere ai dettagli risulta necessario un ulteriore tap;
- nella pagina *Inventario* è possbile visualizzare le informazioni essenziali per poter identificare un articolo. Per accedere ai dettagli dello stesso, o per aggiungerlo al carrello, risulta necessario un ulteriore tap.

6.4 Uso della tastiera

L'utilizzo della tastiera è necessario è stato limitato a tre casi:

- inseriremento delle credenziali in fase di login;
- ricerca di un articolo;
- inserimento o modifica della quantità nelle pagine *Aggiungi* e *Modifica*, rispettivamente. In questo caso la tastiera permette un veloce inserimento di grandi quantità.

Per l'inserimento della quantità è stata predisposta una seconda modalità. Sono stati messi a disposizione due pulsanti (+/-), i quali permettono di aumentare o diminuire la quantità, rispettivamente. Tramite questa modalità è possibile inserire velocemente piccole quantità.

6.5 Aiuti per l'utente

Al primo accesso di un utente all'applicazione, viene visualizzato il tutorial della stessa. Esso fornisce istruzioni rilevanti per l'utilizzo dell'applicazione e risulta breve, in quanto l'applicazione è stata progettata per essere massimamente intuitiva. Alla fine del tutorial è presente un bottone che permette di iniziare ad utilizzare l'applicazione. Il tutorial sarà comunque disponibile dal menu, nel caso in cui l'utente non ricordasse come utilizzare l'app.

6.6 Elementi nativi

Gli elementi nativi dell'applicazione sono la fotocamera, per la scansione del codice a barre, e i dialog, per la visualizzazione di messaggi informativi o di conferma. Per rendere questi elementi nativi si sono utilizzati i seguenti plugin di PhoneGap:

- dialogs: fornisce un'interfaccia per convertire gli alert e i confirm delle pagine web in dialog box nativi. In questo modo è stato possibile customizzare i titoli e le etichette dei pulsanti nei dialog;
- barcodescanner: fornisce un'interfaccia per accedere nativamente alla fotocamera del dispositivo ed effettuare la scansione di un codice a barre.

6.7 Considerazioni sulla versione iOS

Nella versione iOS dell'applicazione è stato effettuato un solo cambiamento. Siccome sugli iPhone non esiste un tasto fisico o vistuale che permette di tornare indietro, si è dovuto aggiungere il classico pulsante indietro accanto al menu hamburger. È stata presa ispirazione per questa soluzione dall'applicazione di Amazon per iOS. Nel seguente screenshot è possibile vedere com'è stata implementata la soluzione.

6.8 Considerazioni finali

Si ritiene che l'applicazione soddisfi i primi tre livelli della piramide di Maslow rimappata sui bisogni degli utenti, ovvero che sia funzionale, affidabile e usabile.

7 Backend

Per l'implementazione del backend si è deciso di appoggiarsi ad Amazon AWS, in quanto tramite l'account universitario è possibile usufruire dei servizi offerti gratuitamente.

7.1 Architettura

Il backend della piattaforma presenta la seguente architettura:

- un server cloud Amazon AWS EC2 con installato Windows Server 2017;
- un'instanza Amazon AWS RDS contenente un database Microsoft SQL Server.

All'interno dell'istanza AWS EC2 sono stati installati:

- un server web Apache Tomcat: esso permette di eseguire un web service scritto in linguaggio Java, il quale si occupa di captare le richieste HTTP che provengono dall'applicazione, di elaborarle e di inviare delle risposte costruite sulla base di query effettuate su un database;
- il linguaggio Java: necessario per il funzionamento di Apache Tomcat e per l'esecuzione del web service.

7.2 Web service

Per lo sviluppo del web service si sono utilizzati gli oggetti servlet Java. Essi permettono di captare richieste HTTP e di rispondere ad esse tramite stringhe in formato JSON. Il servizio web è costituito dai seguenti package:

- dbConnection: package contenente una classe utilizzabile per connettersi ed interagire con un database SQL Server;
- servlet: package contenente le classi servlet del servizio web;
- utility: package contenente le classi utilità del servizio web

7.2.1 Servlet

In questa sezione vengono descritte le classi servlet e per ognuna di esse vengono indicate le seguenti caratteristiche:

- **End-point**: end-point che identifica il servlet. L'end-point è utilizzabile per comunicare con lo specifico servlet all'interno del web service;
- **Parametri**: lista dei parametri che devono essere inseriti nella richiesta HTTP per comunicare correttamente con il servlet;
- **Funzionalità**: una breve descrizione delle operazioni eseguite dal servlet e della tipologia di risposta che restituisce.

1. **AggiuntaModificaArticolo.java**
   - **End-point**: */AggiuntaModificaArticolo*
   - **Parametri**:
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
     - query: è la query di inserimento o modifica di un articolo nel carrello dell'utente loggato.
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito ed effettua la query fornita su di esso. Se la query va a buon fine restituisce una stringa JSON contenente un messaggio positivo, altrimenti un messaggio negativo.
2. **Autenticazione.java**
   - **End-point**: */Autenticazione*
   - **Parametri**:
     - username: è la username che l'utente ha inserito in fase di login;
     - password: è la password che l'utente ha inserito in fase di login.
   - **Funzionalità**: il servlet si connette al database ideato per la gestione dell'autenticazione e cerca all'interno di esso le credenziali fornite. Se le credenziali vengono identificate viene resituita una stringa JSON contenente codice 0, altrimenti ci possono essere le seguenti possibilità:
     - codice 1: la password inserita dall'utente non è quella corretta;
     - codice 2: la username inserita dall'utente è inesistente.
3. **EliminazioneArticoli.java**
   - **End-point**: */EliminazioneArticoli*
   - **Parametri**:
     - username: è la username dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
     - lista codici: è la lista dei codici a barre dei prodotti che devono essere rimossi dal carrello dell'utente loggato.

- **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito ed elimina dalla tabella *contenutoCarrelli* gli articoli corrispondenti ai codici e alla username forniti. Infine restituisce una stringa JSON contenente codice 1 se l'eliminazione è andata a buon fine, altrimenti codice 0.

4. **InvioOrdine.java**
   - **End-point**: */InvioOrdine*
   - **Parametri**:
     - username: è la username dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
     - totale: è il totale dell'ordine che l'utente ha deciso di inviare.
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e crea in database un nuovo ordine contenente gli articoli presenti nel carrello dell'utente identificato dalla username fornita. Infine invia una mail di notifica dell'ordine sia all'utente identificato dalla username fornito che all'azienda identificata dal codice azienda fornito.
5. **PrelevaArticoliAzienda.java**
   - **End-point**: */PrelevaArticoliAzienda*
   - **Parametri**:
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e restituisce una stringa JSON contenente le informazioni sugli articoli venduti dall'azienda.
6. **PrelevaDatiOrdine.java**
   - **End-point**: */PrelevaDatiOrdine*
   - **Parametri**:
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
     - codice ordine: è il codice dell'ordine di cui si vogliono conoscere i dettagli.
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e restituisce una stringa JSON contenente le informazioni relative all'ordine corrispondente al codice ordine fornito.
7. **PrelevaOrdini.java**
   - **End-point**: */PrelevaOrdini*
   - **Parametri**:
     - username: è la username dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;

- **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e restituisce una stringa JSON contenente le informazioni relative agli ordini effettuati dall'utente corrispondente alla username fornita.

8. **PrelievoInfoArticoli.java**
   - **End-point**: */PrelievoInfoArticoli*
   - **Parametri**:
     - username: è la username dell'utente loggato;
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e restituisce una stringa JSON contenente le informazioni relative agli articoli nel carrello dell'utente identificato dalla username fornita.
9. **PrelievoInfoArticolo.java**
   - **End-point**: */PrelievoInfoArticolo*
   - **Parametri**:
     - codice articolo: è il codice dell'articolo di cui si vogliono conoscere i dettagli;
     - codice azienda: è il codice che identifica univocamente il database del fornitore corrispondente all'utente loggato;
   - **Funzionalità**: il servlet si connette al database identificato dal codice azienda fornito e restituisce una stringa JSON contenente le informazioni relative all'articolo corrispondente al codice articolo fornito.

7.2.2 Utility

Il package *utility* contiene le seguenti classi utilità:

- **GetDb**: classe contenente un metodo statico che restituisce la stringa di connessione dal database di autenticazione;
- **SendMail**: classe che fornisce un'interfaccia per l'invio di e-mail tramite un server SMTP Amazon AWS.

7.3 Database

Per la gestione degli ordini, degli utenti e degli articoli venduti dai vari fornitori sono stati utilizzati dei database Microsoft SQL Server. In questa sezione vengono descritti i seguenti database:

- database di autenticazione;
- database aziendale.

7.3.1 Database per l'autenticazione

Il database per l'autenticazione contiene le credenziali e le anagrafiche degli utenti registrati e le informazioni dei fornitori registrati all'applicazione. Questo database contiene le seguenti tabelle:

- aziende: contiene le informazioni delle aziende registrate a SmartOrder. La tabella contiene i seguenti attributi:
  - codAzienda (chiave primaria): è un codice che identifica univocamente l'azienda. Esso viene fornito all'azienda in fase di registrazione all'applicazione;
  - nome: è il nome dell'azienda;
  - via: è la via dove è situata la sede principale dell'azienda;
  - civico: è il civivo dove è situtata la sede principale dell'azienda;
  - comune: è il comune dove è situata la sede principale dell'azienda;
  - provincia: è la provincia dove è situata la sede principale dell'azienda;
  - CAP: è il codice postale dove è situata la sede principale dell'azienda;
  - mail: è la mail che l'azienda ha fornito in fase di registrazione;
  - telefono: è il numero di telefono che l'azienda ha fornito in fase di registrazione.
- users: contiene le informazioni relative agli utenti registrati a SmartOrder. La tabella contiene i seguenti attributi:
  - username (chiave primaria): è il nome utente che è stato fornito all'utente in fase di registrazione;
  - password: è la password che è stata fornita all'utente in fase di registrazione;
  - mail: è la mail che l'utente ha fornito in fase di registrazione;
  - codAzienda: è il codice dell'azienda presso cui l'utente è cliente;
  - cellulare: è il numero di cellulare che l'utente ha fornito in fase di registrazione;
  - nome: è il nome dell'utente;
  - cognome: è il cognome dell'utente;
  - via: è la via di residenza dell'utente;
  - civico: è il civico di residenza dell'utente;
  - comune: è il comune di residenza dell'utente;
  - provincia: è la provincia di residenza dell'utente;
  - CAP: è il codice postale di residenza dell'utente.

7.3.2 Database aziendale

Il database aziendale contiene le informazioni relative agli articoli venduti e agli ordini effettuati presso uno specifico fornitore. Per cui nel server è presente un database di questa tipologia per ogni fornitore registrato all'applicazione. Questo database contiene le seguenti tabelle:

- articoli: contiene le informazioni relative agli articoli venduti dall'azienda. La tabella contiene i seguenti attributi:

  - nome: è il nome dell'articolo;
  - barCode (chiave primaria): è il codice a barre dell'articolo;
  - prezzo: è il prezzo dell'articolo;
  - descrizione: è la descrizione dell'articolo.

- contenutoCarrelli: contiene le informazioni relative al contenuto dei carrelli degli utenti dell'azienda. La tabella contiene i seguenti attributi:
  - idRiga (chiave primaria): è un identifico autoincrementante che funge unicamente da chiave primaria;
  - barCode: è il codice a barre dell'articolo in carrello;
  - quantità: è la quantità scelta dall'utente per l'articolo in carrello;
  - username: è lo username dell'utente che ha inserito in carrello l'articolo.

- contenutoOrdini: contiene le informazioni relative al contenuto degli ordini effettuati presso l'azienda. La tabella contiene i seguenti campi:
  - idRiga (chiave primaria): è un identifico autoincrementante che funge unicamente da chiave primaria;
  - codiceOrdine: è il codice dell'ordine in cui é presente l'articolo;
  - barCode: è il codice a barre dell'articolo all'interno dell'ordine;
  - quantità: è la quantità ordinata per l'articolo.

- ordini: contiene le informazioni relative agli ordini effettuati presso l'azienda. La tabella contiene i seguenti campi:
  - codiceOrdine (chiave primaria): è il codice che identifica univocamente l'ordine presso l'azienda;
  - data: è la data in cui l'ordine è stato effettuato;
  - username: è la username dell'utente che ha effettuato l'ordine;
  - totale: è il totale in euro dell'ordine.

8 Conclusioni

Vi è la consapevolezza che l'interfaccia grafica poteva essere resa più accattivante per lo scopo dell'applicazione. L'implementazione della stessa poteva risultare più semplice tramite l'utilizzo di un altro framework, come ad esempio React Native. Il framework scelto ha comunque permesso di realizzare una buona interfaccia grafica in tempi ristretti. Vi è la consapevolezza che un framework ibrido non dovrebbe essere la prima scelta nel caso si utilizzino i sensori del dispositivo (fotocamera in questo caso). Per giustificare la scelta di PhoneGap si ritiene che il tempo necessario per l'apprendimento delle tecnologie richieste dagli altri framework sia di gran lunga superiore e non avrebbe permesso al gruppo di consegnare il progetto alla consegna pianificata. Infatti erano stati presi in considerazione i framework *Xamarin* e *React Native*, i quali richiedevono l'apprendimento di *C#* e *React*, rispettivamente, linguaggi totalmente sconosciuti dai membri del gruppo. Nonostante questo problema il gruppo ritiene di aver realizzato una buona applicazione, che potrebbe risultare utile ad aziende ed utenti nel contesto per cui è stata progettata e implementata.