Applicazioni per dispositivi mobili course a.y. 2015/2016

Loveltaly

Technical documentation[1,2]

Team Members[3]		
Name	**Student Number**	**E-mail address**
Marco Petricca	*240391*	*marco.petricca@student.univaq.it*

[1] The max length of this document is 10 pages
[2] The structure of this document is fixed, it cannot be changed in any way
[3] The team leader is listed as first member in this table

Architecture

MVC architecture

Di seguito viene presentata l'architettura da un punto di vista ad alto livello per poter comprendere l'entità della realizzazione dell'applicazione e il funzionamento della sua architettura.

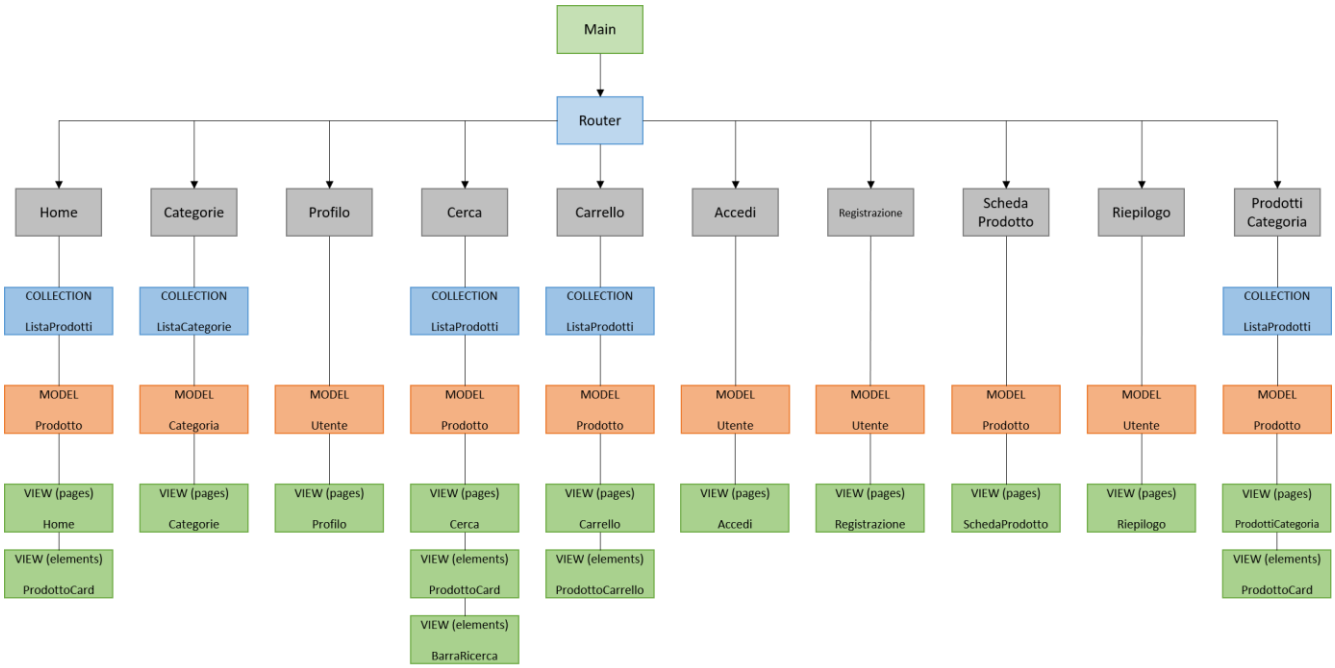


Figura 1: Architettura dell'applicazione ad alto livello

L'applicazione è stata sviluppata, come da immagine, rispettando la struttura dettata da Backbone. Scandagliandola in maniera analitica e sintetica top-down abbiamo i seguenti punti.

- Nel Main vengono inclusi i path di tutte le librerie usate e inizializzate alcune loro proprietà, come dettato dalle librerie stesse.

- Nel Router sono stati inclusi i path di tutte le pagine dell'applicazione, e ci sono i relativi metodi per ognuna di esse; il compito del router è di accedere alle pagine richieste dall'utente, con l'aggiunta di un controllo per la connessione dati; per ogni route abbiamo le rispettive pagine corrispondenti.

- Le View possono essere:
    - pagine (esse sono situate nella cartella "views/pages" come da struttura del boilerplate), e possono essere composte ulteriormente da eventuali elementi componenti;
    - elementi componenti: sono sempre View ovviamente, e sono situati nella cartella "views".
      [*Un esempio è l'elemento "ProdottoCardView.js" che rappresenta un singolo prodotto visualizzabile sottoforma di Card.*]

  Inoltre, ogni pagina View può avere al suo interno una Collection o un Model.

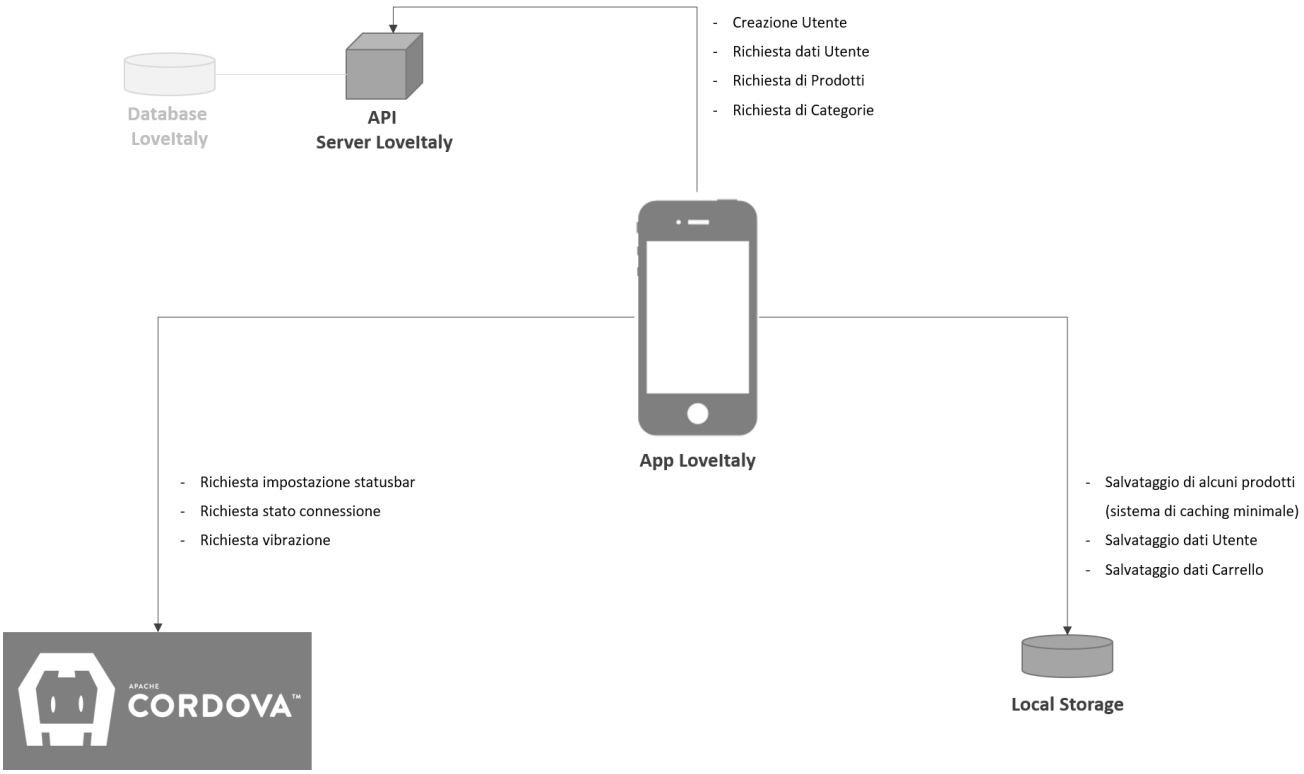
- Le Collection sono composte da molteplici Model omogenei:
    - ListaProdotti.js
    - ListaCategorie.js
    - Carrello.js
      [*Un esempio è "ListaProdotti.js" che dal punto di vista architetturale e comportamentale è un Adapter che funge da interfaccia di comunicazione con il server di LoveItaly; tramite questa Collection è possibile effettuare qualsiasi tipo di richiesta che coinvolga i Prodotti, che infatti vengono ritornati al suo interno.*
      Analogamente il discorso vale per "ListaCategorie.js".
      "Carrello.js" naturalmente è una Collection di Prodotti.]

- I Model:
    - Prodotto.js
    - Utente.js
    - Categoria.js

      [*Più attenzione va posta su "Utente.js" in quanto si interfaccia con le API di LoveItaly e adegua il modello snello del client, con le sole proprietà necessarie, con il modello molto più corposo lato server; inoltre c'è bisogno di una conversione XML necessaria solo nell'operazione di salvataggio dell'Utente (chiamato "customer" lato server), mentre nelle operazioni di semplice richiesta, imponiamo una risposta comodamente in JSON.*]

Data sources

Di seguito viene mostrato un diagramma che serve a identificare con chiarezza il sistema dal punto di vista dei dati.



L'applicazione si può interfacciare con le API del server di LoveItaly, per effettuare:

- Creazione Utente, che serve per poter effettuare la registrazione dell'Utente
- Richiesta dati Utente, che serve a controllare la correttezza dei dati inseriti dall'Utente nel Login
- Richiesta di Prodotti e di Categorie, i quali sono richiesti dall'applicazione durante la normale fruizione della stessa da parte dell'Utente

L’applicazione può salvare nel (e caricare dal) Local Storage i seguenti dati:

- Prodotti:

  in alcuni casi, possono esserci chiamate verso il server superflue, quindi è stato realizzato un sistema minimale di caching per evitare ulteriori scambi inutili di dati

- Dati Utente:

  serve a memorizzarne i suoi dati, visibili dal Profilo e per capire se sia loggato o meno

- Carrello:

  i prodotti contenuti in esso vengono mantenuti in modo da poter essere ritrovati ad un successivo avvio dell’applicazione

L’applicazione, inoltre, si interfaccia con le API fornite da Cordova per poter accedere ad alcune funzionalità proprie del dispositivo:

- Statusbar plugin:

  è stato sovrascritto il colore nero di default della statusbar di sistema, personalizzato con il verde caratteristico della palette

- Connessione plugin:

  si è impostato un Listener che rimane in ascolto di assenza totale di connessione o, nello specifico, anche di perdite di connessione dati

- Vibrazione plugin:

  si fa uso di una piccola vibrazione del dispositivo (se ne è dotato), come feedback tattile secondo buona regola del Material Design, laddove sia adatta (es. Viene usata all’aggiunta di un Prodotto nel Carrello)

Used libraries and frameworks

Ovviamente è stato utilizzato il set minimale richiesto dalle competenze del corso:

- Backbone
- Require
- Handlebars

Inoltre, sono state utilizzate le seguenti ulteriori librerie:

- Materialize: molto adatta allo scopo di realizzare un'interfaccia basata sullo stile del Material Design (come deciso in fase progettuale). Questa libreria è stata scelta dopo un confronto con altre simili in quanto meno macchinosa e di più facile utilizzo.
    - Al suo interno utilizza HammerJS per le gesture.

      [*È stato da me rilevato un bug della libreria e corretto, nonché migliorata una routine per il calcolo del drag del menu laterale.*]
- Md5 (fornita durante il corso): per il calcolo dell'md5, necessaria per la codifica della password degli utenti.
- jQuery: per accedere al DOM e manipolarlo, cioè è stata usata per modificare, aggiungere, rimuovere gli elementi grafici e le loro proprietà.

Tools

- Balsamiq Mockups 3 per realizzare il Lo-Fi Wireframe (e la Sitemap), in quanto dopo aver provato anche altri tool simili (Pencil, ad esempio) risultava essere il più efficace allo scopo e soprattutto di immediato utilizzo.
- Pencil, per realizzare la schermata Hi-Fi.
  Al contrario del Lo-Fi, e per una singola schermata, è senza dubbio una scelta molto efficace e rapida rispetto a un'eventuale scelta di programmi molto più pesanti come Adobe Photoshop o simili.
- Eclipse (ver. Mars2) con plugin JBoss Tools (tool consigliato durante il corso); in questo modo è possibile creare progetti Cordova e installare plugin direttamente dall'IDE, per cui non c'è la necessità di usare la riga di comando. Inoltre il deploy dell'applicazione sullo smartphone è semplicissima effettuarla, in quanto basta spingere un semplice pulsante "Avvia su dispositivo".
- Ripple Emulator come emulatore per far girare l'applicazione su pc, in combinazione con Google Chrome in modo da poter ispezionare l'html e/o css per le modifiche di stile, ecc. Senza dubbio risulta essere molto di aiuto per la parte implementativa, ma più che altro necessaria.
- Microsoft Office per la creazione dei documenti testuali di progetto, usato sfruttando la licenza studente universitaria.
- Screencast per l'acquisizione del video sul dispositivo Android.
- Avidemux per quanto riguarda:
  - postprocessing del video con applicazioni di filtri (double speed e fadeout finale)
  - encoding del video con codec H.264 con preset medio personalizzato e CRF=30.