Mobile computing course a.y. 2015/2016

LOVE ITALY APP

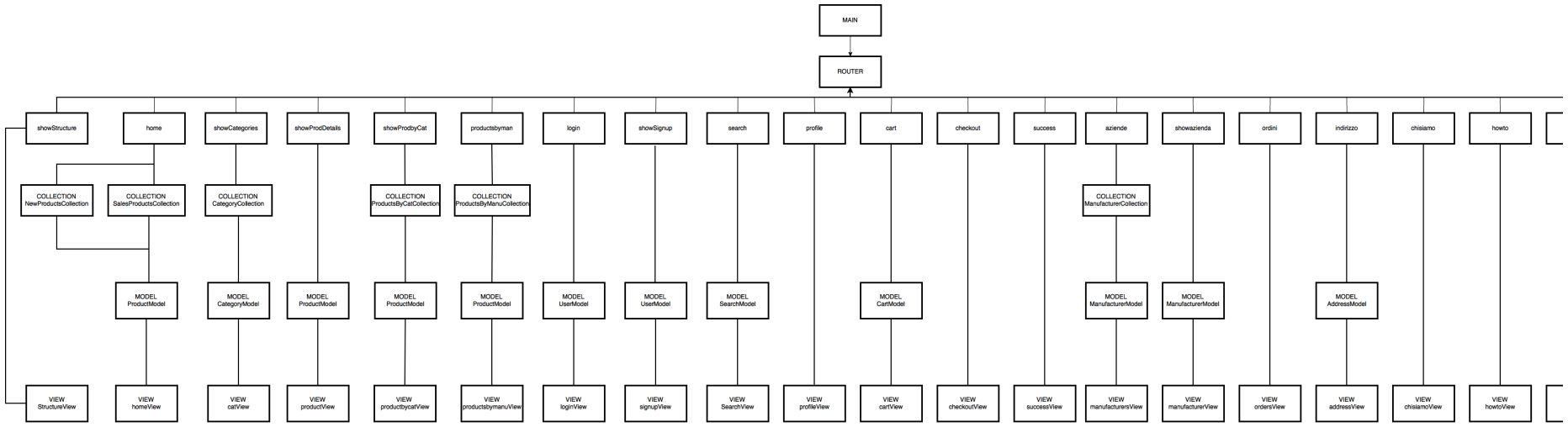
Technical documentation[1,2]

Team Members[3]		
Name	**Student Number**	**E-mail address**
Luca Del Gallo	*220760*	*delgalloluca@gmail.com*
Alessandro Franceschelli	*229718*	*franceschelli.ale93@gmail.com*

[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 (immagine grafico in allegato)



Descrizione:

Per la realizzazione dell'app Love Italy abbiamo usato Backbonejs, un framework che ci ha permesso di adottare il pattern arichitetturale MVC. La parte che potremmo identificare come logica di business, in Backbonejs è formata da model e collection, mentre la parte relative alla logica di presentazione è formata dalle varie view.

Nello specifico i model che abbiamo implementato sono :

- ProductModel : modello che rappresenta un singolo prodotto(dato id-product facendo la fetch si ricevono tutti i dati dal server)
- AddressModel : modello per indirizzo di un customer
- CartModel : modelo usato per il carrello , la sua principale funzione è quella di settare il prezzo totale del carrello
- CategoryModel : modello per una categoria (dato id-category facendo la fetch si ricevono tutti i dati dal server)
- ManufacturerModel : modello per un'azienda (dato id-manufacturer facendo la fetch si ricevono tutti i dati dal server)
- SearchModel: modello per la ricerca di un prodotto, contiene la "query" di ricerca
- UserModel : modello per un utente

Le collection implementate sono:

- NewProductsCollection e SalesProductsCollection: collection di prodotti per lo slider della home page
- CategoryCollection : collection per le categorie
- ManufacturerCollection : collection per le aziende
- ProductsByCatCollection : collection di prodotti in base alla categoria
- ProductsByManuCollection: collection di prodotti in base all'azienda

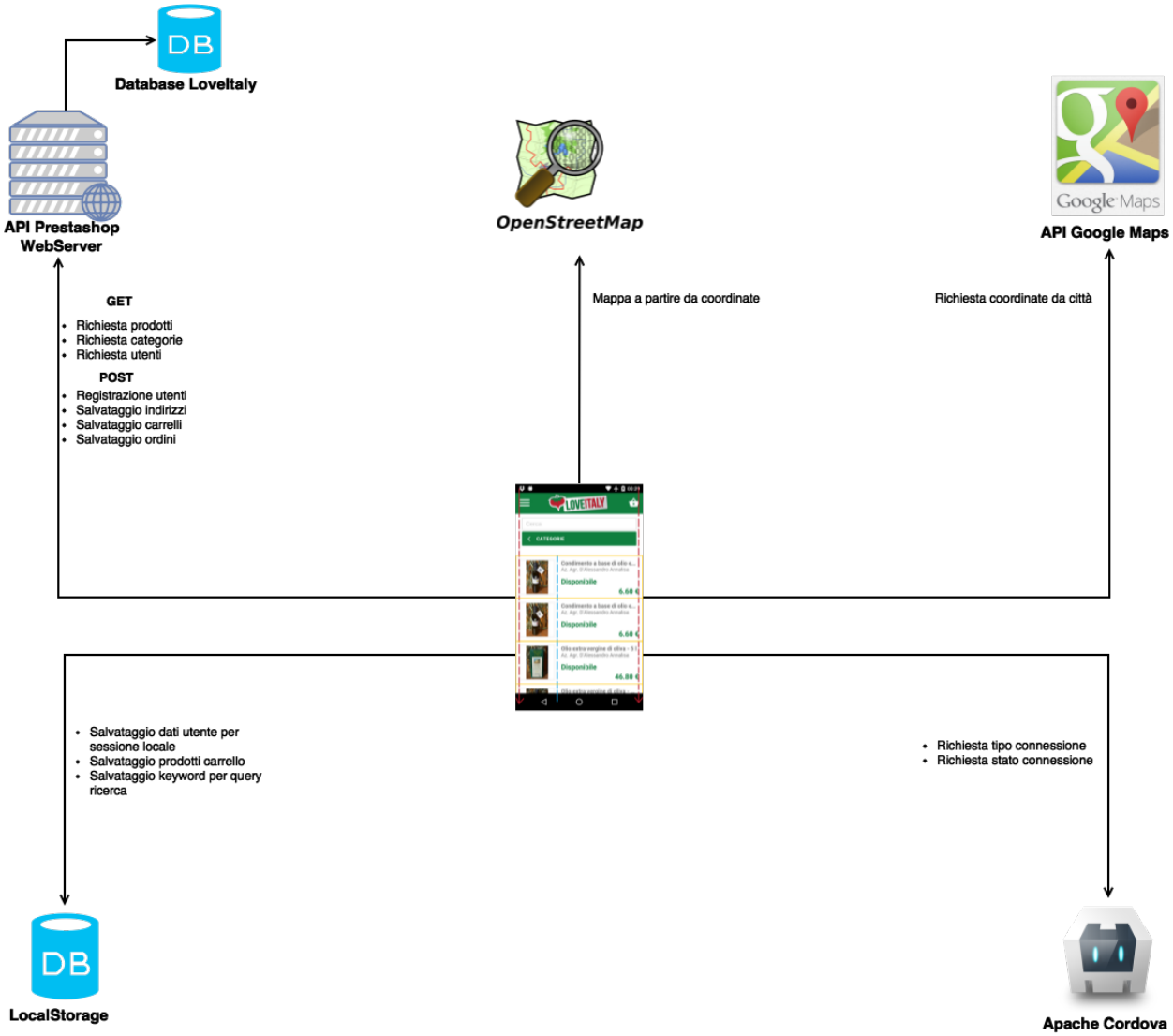
Model e collection rappresentano quindi la parte dati della nostra app cioè quella che si occupa di fornire alle view dati da mostrare all'utente (nei model e collection sono dichiarate le url che servono per ricevere i dati dal server).

La logica di presentazione, che si occupa di prendere i dati dei model e collection e renderizzarli, è composta dalle seguenti view Backbone :

- addressView : view che mostra indirizzo di un utente nella sezione "area personale"
- cartView : view che mostra il carrello
- catView : view che mostra la lista delle categorie
- checkoutView : view che mostra la fase di checkout
- chisiamoView : view che mostra la pagina "chi siamo"
- faqView : view che mostra le domande frequenti
- homeView : view view che mostra la homepage
- howtoView : view che mostra delle informazioni su come effettuare un acquisto
- loginView : view che mostra la pagina di login/signup
- ManufacturersView : view che mostra la lista delle aziende

- ManufacturerView : view che mostra la pagina di dettaglio di un'azienda
- OrdersView : view che mostra gli ordini effettuati da un cliente nella sezione "area personale"
- productbycatView : view che mostra la lista di prodotti di una categoria
- productsbymanuView : view che mostra la lista di prodotti di un'azienda
- productView : view che mostra la pagina di dettaglio di un prodotto
- profileView : view che mostra la pagina del profillo di un utente
- SearchView : view che mostra la lista dei prodotti in base alla ricerca effettuata
- signupView : view per la registrazione di un utente
- successView : view che mostra la riuscita dell'invio di un ordine
- StructureView : view che mostra in ogni altra view la navigation bar e il menu laterale; presente in tutte le view ad esclusione della view di login e signup

Data sources (immagine grafico allegata)



- **API Prestashop:** utilizzate per ottenere tutti i dati presenti nell'app
- **API Cordova :** utilizzate per verificare lo stato della connessione internet ed anche il tipo
- **API Google Maps:** utilizzate per trasformare un indirizzo in coordinate (geocoding)
- **API Open Street Map :** utilizzate per renderizzare la mappa attraverso le coordinate
- **LocalStorage :** utilizzato per salvare dati utente al momento del login per creare una sorta di "sessione" , per salvare il carrello e per il passaggio della stringa di ricerca digitata dall'utente

Used libraries and frameworks

- **Backbonejs :** framework Javascript utilizzato per adottare il pattern architetturale MVC, aiuta lo sviluppatore a strutturare meglio il codice.
- **Requirejs :** loader per libreria js , aiuta lo sviluppatore nel caricare correttamente le varie librerie utilizzate con le loro dipendenze.
- **Handlebars :** motore di template che rende possibile l'inserimento di "placeholder" all'interno dell'html, aiuta la divisione tra "grafico" e sviluppatore e consente di scrivere template che saranno poi riempiti con dati a run time.
- **Leaflet:** libreria open-source che consente di inserire mappe interattive in modo molto semplice
- **Materialize :** framework css responsive basato sul material design di Google che aiuta lo sviluppatore ad adottare un look and feel di un app mobile.
- **SlickSlider :** libreria js utilizzata per creare uno slider responsive
- **Md5:** libreria js utilizzata per decodidicare la password di un utente
- **Jquery:** libreria js utilizzata per la manipolazione di elementi del DOM e per parsare ad esempio la descrizione di un prodotto (eliminazione tag html).

Tools

- **Sublime Text 2 :** editor utilizzato per scrivere codice Javascript
- **Google Chrome :** browser utilizzato per debugging dell'applicazione
- **Advanced Rest Client :** client rest utilizzato per provare le chiamate al server di LoveItaly
- **VirtualBox :** tool utilizzato per virtualizzare la macchina virtuale di LoveItaly che ci è stata fornita
- **Komodo :** editor utilizzato per composizione codice html e css