

**Documentazione per il caso di studio di**

**Sviluppo di Mobile Software**

**(SMS)**

**Applicazione per la Promozione Turistica**

Sommario

[Descrizione del Gruppo 3](#_Toc363837)

[Nome del Gruppo 3](#_Toc363838)

[Componenti 3](#_Toc363839)

[Introduzione al Progetto 4](#_Toc363840)

[Nome del Progetto 4](#_Toc363841)

[Descrizione 4](#_Toc363842)

[Logo 4](#_Toc363843)

[Funzionalità 5](#_Toc363844)

[Consultazione Meteo 5](#_Toc363845)

[Scelte Implementative 5](#_Toc363846)

[Visualizzare lista categoria 5](#_Toc363847)

[Classi modello: Luogo ed Evento 5](#_Toc363848)

[Scelta della categoria 6](#_Toc363849)

[Scelte Implementative 7](#_Toc363850)

[Consultazione Degli Eventi nella Provincia di Bari 8](#_Toc363851)

[Informazioni Utili 8](#_Toc363852)

[Creazione e consultazione di luoghi preferiti 8](#_Toc363853)

[Consultazione dei Coupon Utilizzabili 8](#_Toc363854)

[Requisiti per l’Utilizzo 8](#_Toc363855)

[Scelte Implementative Generali 8](#_Toc363856)

# Descrizione del Gruppo

## **Nome del Gruppo**

4Designers

## Componenti

|  |  |  |  |
| --- | --- | --- | --- |
| **Cognome** | **Nome** | **Matricola** | **Email** |
| Castellano | Graziano | 559926 | grazianocas90@gmail.com |
| Di Pierro | Davide | 661292 | ddipierro745@gmail.com |
| Gigante | Domenico | 661582 | dom.gigante0@gmail.com |
| Lisco | Federica | 664323 | fedelis1997@gmail.com |



# Introduzione al Progetto

## Nome del Progetto

Barintondo

## Descrizione

Barintondo è un’applicazione nata con uno scopo preciso: favorire il turismo nella città di Bari.

A tal punto, l’applicazione dovrà offrire un insieme di funzionalità che accompagni e supporti il turista in tutte le fasi di esplorazione: dalla ricerca dei punti d’interesse fino alla prenotazione di locali gastronomici, passando per la consultazione del meteo e l’utilizzo di coupon in determinati luoghi di interesse.

## Logo



# Funzionalità

## Consultazione Meteo

Barintondo permette di consultare in maniera rapida le informazioni principali relative al meteo di Bari in tempo reale.

In particolare, le informazioni mostrate sono:

* Temperatura
* Descrizione sintetica del meteo
* Immagine rappresentativa del meteo

Inoltre, è possibile consultare informazioni più dettagliate cliccando sull’apposito pulsante, il quale aprirà una pagina web con numerose informazioni aggiuntive (previsioni giornaliere e dei giorni a venire).

### Scelte Implementative

Le informazioni relative al meteo sono mostrate all’interno di un LinearLayout, composto da una “ImageView” per l’immagine rappresentativa e più textView per mostrare le singole informazioni. Il pulsante per accedere alle informazioni aggiuntive è un “Chip” di tipo “Action”.

Per ottenere le informazioni in tempo reale, si è deciso di utilizzare le API gratuite del sito OpenWeatherMap, per i seguenti punti di forza:

* Informazioni fornite gratuitamente per un massimo di 60 richieste al secondo, al contrario di altre API disponibili pubblicamente ma con notevoli limiti (60 richieste/ora, informazioni troppo limitate, …)
* Supporto alle informazioni multilingua
* Mette a disposizione una classe Java di nome OperWeatherMapHelper, utilizzabile tramite opportune dipendenze nel file build.gradle, che automatizza le richieste al server ed effettua automaticamente il parsing dei risultati, lasciando al programmatore solo l’onere di selezionare le informazioni desiderate
  + Tutte le richieste sono effettuate automaticamente in maniera asincrona

Le informazioni aggiuntive sono consultabili in una WebView, a cui si accede cliccando sull’apposito “Chip” all’interno del suddetto LinearLayout.

## Visualizzare lista categoria

Barintondo permette di ottenere informazioni circa luoghi ed eventi presenti nel capoluogo pugliese e nei dintorni. Per gestire queste entità sono state create due classi.

### Classi modello: Luogo ed Evento

Luogo ed Evento sono le classi che contengono le istanze degli oggetti utilizzati dall’applicazione.

Evento è un’estensione di Luogo in quanto possiede tutti gli attributi di un luogo più una data di inizio e una di fine, e eventualmente un Luogo in cui avviene l’evento.

Entrambe le classi implementano l’interfaccia Parcelizable in quanto gli oggetti di queste classi vengono passati tra le activity negli intent.

### Scelta della categoria

Attraverso Barintondo è possibile consultare una lista aggiornata di tutti i punti di interesse presenti nella città di Bari suddivisi in cinque categorie, che comprendono a loro volta elementi di diverse sottocategorie:

* Attrazioni
* Chiese
* Monumenti
* Teatri
* Musei
* Discoteche
* Lidi Balneari
* Luoghi adatti a Famiglie
* Mangiare
  + Pizzerie
  + Ristoranti
  + Bar
* Dormire
  + B&B
  + Hotel
* Vicino Bari

(Possiede le stesse categorie di Attrazioni, con la differenza che i luoghi elencati sono situati in altre città della provincia)

* Eventi
  + A Bari
  + Fuori Bari

Barintondo è anche in grado di adattarsi alle preferenze dell’utente: qualora il suo utilizzatore dovesse scegliere più frequentemente una determinata categoria di luoghi, Barintondo offrirà la possibilità di spostare tale categoria all’inizio della lista, così da renderne più agevole l’individuazione.

Inoltre, è possibile ricercare un particolare luogo all’interno della lista, indicandone (in parte o completamente) il nome, la categoria, o la città.

Ogni elemento della lista mostrerà delle informazioni sintetiche per ciascun punto di interesse:

* Nome
* Valutazione, sulla base delle recensioni precedentemente aggiunte dai turisti
* Categoria
* Informazioni sull’apertura (ove possibile)
* Informazioni sulle date di inizio e fine qualora si tratti di un evento

### Scelte Implementative

L’activity che mostra l’elenco è LuoghiListActivity, implementata dall’ononima classe.

Alla creazione dell’activity verrà prelevato dall’intent che ha portato ad essa l’informazione sulla categoria richiesta. Sarà quindi interrogato il DB remoto, grazie alla classe **ControllerRemoteDB.java** che include tutti i metodi Volley che comunicano con gli script php del server. Con il passaggio di un listner, il metodoVolley restituirà una volta ottenuta la rispsota dal server la lista aggiornata dei luoghi, che verrà aggiornata sulla UI.

Per realizzare la UI relativa alla schermata contenente la lista dei luoghi:

* Toolbar, contenente una “Action” di tipo “SearchView”, sempre visibile e che permette all’utente di cercare un determinato elemento nella lista in base a nome, categoria o città. Il meccanismo di ricerca è implementato tramite i metodi “getFilter()” e “publishResults()” nella classe **LuogoAdapter.java** del package **utils**. Tale classe implementa l’interfaccia “Filterable”, che richiede una propria implementazione dei suddetti metodi.
* LinearLayout contenente un ChipGroup, con la logica di selezione mutuamente esclusiva (al più un solo chip attivo in ogni istante). Ogni Chip è di tipo “Action” e gestisce la selezione tramite un’opportuna implementazione dell’interfaccia onClickListener.

Il meccanismo di personalizzazione dell’ordine delle categorie è implementato **CONTINUA DAVIDE**

* La lista è implementata tramite l’utilizzo di una “Recycler View”, con le relative componenti:
  + **Luogo.java**: classe che descrive gli elementi del modello
  + **LuogoAdapter.java**: classe che rappresenta l’adapter utilizzato dalla “Recycler View”. Tale classe ottiene i dati dal server remoto tramite l’utilizzo dei metodi della classe **ControllerRemoteDB.java**, la quale contiene i metodi in cui è implementata la logica di comunicazione con il DB Remoto tramite framework “Volley”.
  + **MyViewHolder** (classe interna di **LuogoAdapter.java**): implementazione del componente ViewHolder, utilizzato dalla Recycler View. L’adapter si preoccuperà di comunicare all’activity, tramite un listner, dell’evento di selezione di un determinato elemento. L’activty, dopo aver effettuato un controllo se si tratta di un Luogo o un Evento, lancerà un intent all’activity che mostra il dettaglio dell’elemento selezionato, LuogoDetailActivity per i luoghi ed EventDetailActivity per gli eventi.
* Per gli elementi della “Recycler View” è stato definito un apposito layout, descritto nel file **luogo\_list\_content.xml**. Tale layout prevede
  + una “ImageView” per mostrare un’immagine rimpicciolita del luogo;
  + un particolare layout, gestito dalla classe **FrameVoteStars.java**, classe creata per permettere un riuso della sua implementazione in più parti dell’applicazione. Questa classe ottiene in automatico nel costruttore i riferimenti alle view del suo frame e con il metodo setStars() imposta il voto da visualizzare;
  + Un insieme di “TextView” per mostrare le informazioni del luogo;
  + Una “ImageView” che mostra una stella solo se il luogo è presente nella lista dei preferiti.

## Visualizzazione dettaglio luogo

Per realizzare la UI relativa alla schermata contenente il dettaglio di luogo sono stati utilizzati i seguenti componenti:

* “FAB” per l’aggiunta di un luogo alla propria lista di interessi; implementata tramite apposita implementazione dell’interfaccia onClickListener del “FAB”. Il salvataggio dell’interesse nel DB remoto è effettuato tramite opportuno metodo della classe **ControllerRemoteDB.java**
* Il layout precedentemente descritto per mostrare il voto complessivo del luogo
* “ImageView” per mostrare una foto del luogo selezionato
* Tre “Button”, cliccando i quali verrà mostrato un particolare fragment, ciascuno dei quali contiene un particolare sottoinsieme di informazioni:
  + Il primo (partendo da sinistra), selezionato di default, contenente informazioni generiche sul luogo:
* Orari di apertura e chiusura (se disponibili)
* Indirizzo
* Indicazioni stradali su come raggiungerlo (tramite apposito pulsante che lancia un intent all’appliczione maps)
  + Il secondomostra una lista di tutti i luoghi entro il raggio di 3km dal luogo selezionato. La lista non è altro che una “Recycler View” che utilizza gli stessi componenti sopra descritti. Tuttavia in questa sezione LuogoAdapter modificherà il layout degli elementi della lista per mostrare informazioni aggiuntive. Dato che in questa lista le varie categorie sono tutte presenti, un’icona sulla destra mostra la categoria di appartenenza del luogo (Attrazione, Mangiare o Dormire), mentre un’ulteriore TextView mostra l’informazione sulla distanza tra il luogo di cui si sta visualizzando il dettaglio e l’elemento della lista.
  + Il terzo mostra le recensioni relative al luogo selezionato. Anche in questo caso sarà utilizzata una “Recycler View”, supportata questa volta da **ReviewAdapter.java**, e dalla classe modello **Review.java.** Il fragment una volta collegato all’activity, richiederà al server, sempre tramite la classe **ControllerRemoteDB.java,** tutte le recensioni relative al luogo. Una volta ricevuta la lista tramite l’apposito listner, aggiorna la recyclerView. Il layout di riga utilizzato è descritto nel file **recensione\_list\_content.xml,** che mostra la data della recensione, il nickname dell’autore, il voto (impostato grazie alla classe **FrameVoteStars.java** già descritta) dato ed il testo.

In basso è presente un “FAB” che, quando cliccato, mostra la sezione per l’inserimento di una recensione, e due button: uno per confermare l’invio della recensione, l’altro per annullare e tornare alla lista delle recensioni. Quando l’utente compila una recensione e preme il button per l’invio, il fragment, tramite la classe **ControllerRemoteDB.java,** invia la recensione al server. Se il server risponde con risultato affermativo, il fragment chiude la sezione di inserimento recensione e aggiorna la lista delle recensioni con il meccanismo precedentemente descritto. In caso di risposta negativa notifica all’utente l’errore, e chiede di riprovare in seguito.

## Visualizzazione eventi

## Informazioni Utili

## Creazione e consultazione di luoghi preferiti

## Consultazione dei Coupon Utilizzabili

Consultazione lista e dettaglio, utilizzo tramite BT

## Requisiti per l’Utilizzo

Registrazione con dati sintetici, login, logout

# Scelte Implementative Generali

Classe per parsing immagini, volley per le richieste al server, listener per gestire gli eventi in seguito a richieste al server, nav drawer, componenti material design, invio feedback tramite intent ad applicazione di posta, modalità offline, un utente per volta