

**Documentazione per il caso di studio di**

**Sviluppo di Mobile Software**

**(SMS)**

**Applicazione per la promozione turistica**

Sommario

[Descrizione del gruppo 3](#_Toc373311)

[Nome del gruppo 3](#_Toc373312)

[Componenti 3](#_Toc373313)

[Introduzione al progetto 4](#_Toc373314)

[Nome del progetto 4](#_Toc373315)

[Descrizione 4](#_Toc373316)

[Logo 4](#_Toc373317)

[Funzionalità 5](#_Toc373318)

[Consultazione Meteo 5](#_Toc373319)

[Scelte implementative 5](#_Toc373320)

[Visualizzare lista categoria 5](#_Toc373321)

[Classi modello: Luogo ed Evento 5](#_Toc373322)

[Scelta della categoria 6](#_Toc373323)

[Scelte implementative 7](#_Toc373324)

[Visualizzazione dettaglio luogo 8](#_Toc373325)

[Visualizzazione eventi 9](#_Toc373326)

[Informazioni utili 10](#_Toc373327)

[Visualizzazione interessi 10](#_Toc373328)

[Consultazione dei coupon utilizzabili 10](#_Toc373329)

[Requisiti per l’utilizzo 11](#_Toc373330)

[Scelte implementative generali 11](#_Toc373331)

# Descrizione del Gruppo

## **Nome del Gruppo**

4Designers

## Componenti

|  |  |  |  |
| --- | --- | --- | --- |
| **Cognome** | **Nome** | **Matricola** | **E-mail** |
| 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 consumazione di coupon in determinati luoghi, passando per la consultazione del meteo.

## 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 sessanta richieste al secondo, al contrario di altre API disponibili pubblicamente ma con notevoli limiti (sessanta richieste/ora, informazioni troppo limitate, …);
* Supporto alle informazioni multilingua;
* Messa a disposizione di una classe Java di nome OpenWeatherMapHelper, 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 Parcelable in quanto gli oggetti di queste classi vengono passati tra le activity negl’ 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;
  + Hote;l
* Vicino Bari

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

* Eventi
  + 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, l’app 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

**Utenti dell’app**

Al primo avvio, l’app richiederà di inserire e-mail e password per accedere a Barintondo. Se non si possiede un account basterà andare nella specifica sezione per crearne uno. Gli unici dati sono richiesti sono: un nickname, un indirizzo e-mail (univoco) e una password di almeno otto caratteri.

Quando si effettua la registrazione, l’app esegue in automatico il primo accesso in modo da non dover reinserire le credenziali.

Ogni volta che si apre l’app, se esiste un account, non sarà necessario rieseguire la fase di login. Le credenziali saranno richieste solo in caso di logout esplicito da parte dell’utente all’interno dell’app.

In locale è stato memorizzato in un database SQLite l’unico account connesso di quel dispositivo. In caso di logout, l’unico record verrà cancellato per far spazio al nuovo account di quel dispositivo.

Tutti gli utenti sono salvati comunque sul server remoto.

L’app deve essere fruibile sia per gli italiani che per gli stranieri. Per questo motivo, Barintondo parla la lingua dell’utente. Con l’uso del multilingua, l’app è disponibile sia in italiano che in inglese. Quest’ultima è chiaramente la lingua di default.

L’app possiede una sezione Profilo in cui è possibile aggiornare il nickname e/o la password dell’utente. Per ovvi motivi, la password è nascosta ed è modificabile o visibile solo previo inserimento di quella attuale.

Se in fase di login l’utente non ricorda la sua password, potrà recuperarla attraverso il suo indirizzo e-mail. Verrà inviata una mail con la nuova password di accesso (generata casualmente). Dopo aver effettuato l’accesso, è consigliabile che l’utente la modifichi.

L’invio della mail per il recupero è seguito da una notifica che suggerisce che l’operazione è andata a buon termine.

**Visualizzazione luoghi**

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

Alla creazione dell’activity verrà prelevata dall’intent 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 listener, il metodo Volley restituirà una volta ottenuta la risposta dal server la lista aggiornata dei luoghi, che verrà aggiornata sulla UI.

La schermata relativa alla lista dei luoghi è composta da:

* 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 utilizzando le SharedPreferences. È un meccanismo che consente di memorizzare informazioni di tipo chiave-valore e sono utili per memorizzare le preferenze dell’utente.

Nel file “BarintondoPrefs” viene memorizzato, per ogni categoria, il numero di volte in cui l’utente l’ha selezionata. Quando una categoria supera tutte le altre, e non è al primo posto, appare un Dialog all’utente chiedendogli se vuole avvicinarla.

Per evitare che questo popup possa essere invasivo, è presente una Checkbox che consente di disabilitare questo Dialog facendo sì che l’ordine dei Chip resti invariato.

* La lista è implementata tramite l’utilizzo di una RecyclerView, con le relative componenti:
  + **Luogo.java**: classe che descrive gli elementi del modello
  + **LuogoAdapter.java**: classe che rappresenta l’adapter utilizzato dalla RecyclerView. 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 listener, dell’evento di selezione di un determinato elemento. L’activity, 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 RecyclerView è stato definito un apposito layout, descritto nel file **luogo\_list\_content.xml**. Tale layout prevede
  + Un’ 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;
  + Un’ 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:

* Un FAB per l’aggiunta di un luogo alla propria lista di interessi; implementata tramite apposita implementazione dell’interfaccia OnClickListener. 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
* Un’ ImageView per mostrare una foto del luogo selezionato
* Tre “Button” che, tramite click, mostrano un particolare fragment, ciascuno dei quali contenente un particolare sottoinsieme di informazioni:
  + A sinistra, selezionato di default, le 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’applicazione maps);
  + Al centro una lista di tutti i luoghi entro il raggio di tre chilometri dal luogo selezionato. La lista non è altro che una RecyclerView che utilizza gli stessi componenti sopra descritti. 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 RecyclerView, 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 listener, 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, se 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

**Slider eventi**

È stato inserito uno slider nella HomeActivity dell’applicazione che mostra in sequenza gli eventi del prossimo mese, indicando la città in cui si svolgono e i giorni che mancano all’inizio dell’evento.

Per realizzare lo slider è stata implementata la classe SliderAdapter.java, che estende la classe PageAdapter. L’activity richiede al server tramite ControllerRemoteDB.java la lista degli eventi, e una volta ricevuta la lista informa l’adapter di aggiornare la lista.

HomeActivity contiene nella sua UI un elemento ViewPager, nel quale viene inserito, tramite l’adapter, il layout contenuto in slider\_image\_content.xml, che a determinati intervalli di tempo, si aggiorna con le informazioni dell’evento successivo della lista. L’adapter predispone il ViewPager a generare un intent verso l’activity EventoDetailActivity nel momento in cui viene selezionando, passando come paramentro il codice dell’evento che in quel momento era visualizzato.

EventDetailActivity

Questa activity mostra il dettaglio di un evento, sfrutta gli stessi fragment di LuogoDetailActivity, con la differenza che non è presente la sezione delle recensioni, e in più mostra, se presente, l’informazione sul luogo in cui si svolge l’evento, con la possibilità di selezionarlo e aprire l’acivity di dettaglio di tale luogo.

## Informazioni Utili

Per aiutare un eventuale turista nelle situazioni di emergenza, gli si danno a disposizione dei numeri utili per l’evenienza quali:

* Carabinieri;
* Polizia;
* Emergenza infanzia;
* Vigili del fuoco;
* Emergenza sanitaria;

L’activity mostrerà una RecyclerView in cui ogni voce farà partire un intent di tipo DIAL che comporrà il numero sulla tastiera e ne consentirà la chiamata.

## Visualizzazione interessi

L’activity si comporta come LuogoListActivity, con la differenza che utilizza un metodo diverso della classe ControllerRemoteDB.java, per richiedere al server la lista dei luoghi di interesse dell’utente. Utilizza quindi una RecyclerView e LuogoAdapter per mostrare la lista. Utilizza, come LuogoListActivity, dei chips per applicare dei filtri, questa volta solo per categoria, in quanto tra i preferiti sono elencati tutti i tipi di luogo insieme.

## Consultazione dei Coupon Utilizzabili

Consultazione lista e dettaglio, utilizzo tramite BT

**Material Design**

Il Material Design è il design sviluppato da Google per migliorare l’esperienza d’uso dell’utente nell’interazione con le app.

Nei layout utilizzati sono stati utilizzati spesso componenti Material Design come i MaterialButton e FAB o strumenti che migliorano la visualizzazione quali elevazione, animazioni e ombre.

I MaterialButton dispongono già di bordi arrotondati, colori che si adattano al click ecc.

L’elevazione è lo strumento utilizzato per “avvicinare” gli elementi all’utente e far sì che i diversi widget vengano visti su piani differenti.

Le animazioni consentono di guidare l’utente verso l’interazione e di rendere più gradevole l’esperienza generale.

Nel nostro progetto, nella cartella res/anim sono stati inseriti i file XML che rappresentano le diverse animazioni; prevalentemente si tratta di transizioni.

## 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.