Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

QR-VEY
Una piattaforma software per la creazione di sondaggi basata sulla tecnologia del QR-Code

Relatore:
Chiar.mo Prof.
Luciano Bononi
Co-relatore:
Dott.
Luca Bedogni

Presentata da:
Alessandro Zini

Sessione II
Anno Accademico 2015/2016

In ogni momento in cui ho scritto questa tesi
ho avuto qualcuno al mio fianco a supportarmi.
Con un semplice gelato, o un giro in moto,
un sorriso o un abbraccio di conforto..
Non ce l'avrei mai fatta senza di voi.

Introduzione

Nella società moderna, la vita di un individuo è fortemente condizionata dall'opinione e dalle idee di altri soggetti. La definizione stessa di società delinea l'esistenza di un organismo complesso che progredisce e che si evolve continuamente: l'opinione e la formulazione di un giudizio da parte di tutti gli individui particolari che ne fanno parte sono la chiave del progresso della società stessa.

Nel corso degli anni le metodologie per la raccolta delle idee e delle opinioni si sono evolute a pari passo con il progresso tecnologico: dalla semplice comunicazione orale si è passati alla diffusione cartacea, fino a che l'introduzione e l'evoluzione di internet non hanno portato alla digitalizzazione e informatizzazione del processo. Tale progresso ha reso possibile l'abbattimento di ogni barriera fisica, permettendo di fatto a qualsiasi individuo di esprimere la propria opinione in merito a qualsiasi argomento: se in precedenza la diffusione di un sondaggio era limitata dall'effettiva possibilità di distribuzione del sondaggio stesso, lo sviluppo della rete globale ha esteso tale possibilità (virtualmente) a tutto il mondo.

Il cambiamento può apparire meno significativo di quanto lo è stato in realtà. In precedenza, una società dislocata in varie parti del mondo aveva uno spettro di possibilità di diffusione enormemente più ampio di una piccola impresa locale o di un singolo individuo. L'abbattimento dei limiti fisici ha di fatto posto sullo stesso piano, dal punto di vista del potere sondaggistico, individui, aziende e multinazionali, nonché organizzazioni statali.

Nonostante tutti questi siano miglioramenti non indifferenti, è però im-

portante notare come la valorizzazione della diffusione su scala mondiale di un sondaggio abbia inevitabilmente portato a trascurarne le proprietà e i vantaggi intrinsechi legati alla sua diffusione prettamente locale. Facciamo un esempio: un sondaggio che mira a valutare la soddisfazione degli utenti riguardo alla recente introduzione di una nuova linea di laptop ottiene un guadagno enorme dall'informatizzazione, e dunque delocalizzazione, del processo di raccolta dati; dall'altro lato, un sondaggio che ha come scopo la valutazione dell'impatto sui cittadini di un recente rinnovamento degli impianti di illuminazione stradale ha un guadagno pressoché nullo.

Viene dunque da porsi una domanda: qual è la strada giusta da scegliere? L'avanzamento tecnologico porta da sempre con sé un naturale e progressivo abbandono delle metodologie classiche in favore delle nuove: l'ottimizzazione apportata dall'adozione di sistemi informatizzati per la raccolta, l'analisi, e la distribuzione di sondaggi non è trascurabile. È però necessario che tale adozione apporti miglioramenti senza causare la perdita di aspetti e funzionalità intrinseche importanti e vantaggiose.

Sono numerosi i casi in cui l'interazione sinergica tra classico e moderno, accompagnata da un'analisi relativa al caso d'uso particolare, conduce ai risultati migliori: se infatti lo studio del caso d'uso di un sondaggio per una linea di laptop suggerisce l'eliminazione cartacea in favore della diffusione prettamente informatizzata, la combinazione della metodologia cartacea, e quindi localizzata, accompagnata da un sistema automatico e informatizzato per l'elaborazione dei dati si rivela vincente nel caso d'uso relativo all'illuminazione stradale.

È dunque necessario trovare un ponte tra le due metodologie che permetta di sfruttare le potenzialità di elaborazione automatica dei sistemi informatizzati. Tale ponte è stato identificato nella tecnologia del ***QR-Code***: è possibile utilizzare un generico supporto che presenti proprietà di localizzazione, come un foglio cartaceo, ma anche un proiettore di una conferenza, come base per la visualizzazione di tale codice, ed associare alla scansione dello stesso funzionalità automatizzate di raccolta di dati e opinioni.

Il caso di studio che è stato affrontato nello sviluppo della piattaforma è quello relativo al tema di Green Engineering, avanzato dal ***World Harbour Project*** [1] del Sydney Institute of Marine Science. L'obiettivo del World Harbour Project (WHP) è quello di sviluppare una rete di comunicazione globale per permettere la collaborazione tra scienziati di tutto il mondo nel preservare e monitorare l'ecosistema marittimo indigeno in prossimità di grandi aree portuali urbane. Il gruppo ha come partner una quantità sempre crescente di porti mondiali, tra cui Sydney, New York, Rio De Janeiro, Singapore, Ravenna, e tanti altri.

Il team del WHP che si occupa del Green Engineering ha come scopo quello di condurre esperimenti per migliorare la qualità delle comunità di molluschi bivalvi autoctoni, nonché ridurre il numero di specie faunistiche non indigene in porti e strutture marittime artificiali. Oltre a misurare l'efficienza attraverso analisi scientifiche specializzate, WHP ha necessità di utilizzare un sistema semplice e localizzato che permetta la raccolta e l'analisi automatizzata di opinioni pubbliche riguardanti il loro progetto. La piattaforma software è stata costruita per soddisfare pienamente tale caso d'uso, senza perdere la flessibilità necessaria per essere utilizzata per casi d'uso differenti.

Figura 1: Località partner del progetto di Green Engineering del WHP (in corso di avvio anche a Ravenna).

Indice

Elenco delle figure

Elenco dei frammenti di codice

Capitolo 1

Lavori correlati

Esistono una quantità innumerevole di piattaforme online che permettono la creazione e la condivisione di sondaggi *ad-hoc* da parte dell'utente. La metodologia del sondaggio è infatti uno strumento fondamentale all'interno di una azienda: la raccolta di opinioni sull'attività svolta e sulla qualità del servizio offerto permette l'avvicinamento ad utenti e dipendenti, e crea un rapporto di interazione e crescita positiva con essi. Per questo motivo, la maggior parte delle piattaforme più popolari che andrò a menzionare è rivolta ad un pubblico di piccole e medie imprese, ma sono anche diverse le soluzioni per multinazionali e società globali.

Tali piattaforme si distinguono a seconda di caratteristiche particolari, quali:

- il grado di personalizzazione dei sondaggi
- l'interfaccia grafica e la semplicità di utilizzo
- le funzionalità di analisi e post-processing dei dati
- la presentazione in forma semplice ed intuitiva dei risultati
- il costo di utilizzo della piattaforma

Nonostante sia stato posto come ultimo parametro, il costo relativo alla fruizione dei servizi di una determinata piattaforma non è indifferente, ed è generalmente il primo fattore da considerare. Quasi tutte le piattaforme prevedono un piano senza costi, mentre si può arrivare da una media di 20$/mese anche a 200$/mese per le versioni a pagamento: è necessario analizzare in dettaglio i propri requisiti.

La presente è una descrizione riassuntiva delle principali piattaforme e dei servizi offerti; lo scopo di tale illustrazione è rendere possibile un veloce confronto con il servizio oggetto di tesi e collocarne correttamente il target di utenza.

La piattaforma più popolare e utilizzata è **SurveyMonkey** [2]. Oltre ad essere stato uno dei primi servizi professionali del genere, la sua popolarità è dovuta ai numerosi benefici della semplice registrazione gratuita, primo fra tutti la possibilità di creare un numero illimitato di sondaggi. Nonostante siano presenti limiti di 10 domande e 100 risposte per sondaggio, vengono offerte numerose funzionalità: per citarne alcune, sono presenti domande con risposta multipla, matrici di scelta, menu a tendina, caselle di testo e altro ancora; sono inoltre presenti modelli pronti all'uso. Vengono fornite facilitazioni per la distribuzione del sondaggio, che può avvenire via mail, incorporato nel sito personale o mostrato in un account social. Le domande possono essere modificate e incrementate in qualsiasi momento, anche dopo aver pubblicato il sondaggio, ed è possibile monitorare le risposte sia per singola domanda che per singolo utente. Le limitazioni sui sondaggi, la possibilità di esportare i dati dei sondaggi, e tante altre funzionalità sono sbloccabili attraverso la sottoscrizione di un abbonamento mensile.

Un'altra piattaforma largamente usata e considerata una solida alternativa alla precedente è **Surveygizmo** [3]. Il servizio offerto si differenzia per la possibilità di inserire logica nei sondaggi: è possibile nascondere domande, saltare intere pagine o terminare il sondaggio in base alle risposte selezionate dall'intervistato. Un'altra funzionalità aggiuntiva è la possibilità di personalizzare ogni aspetto delle domande da porre, ed sono infine disponibili

API pubbliche per una gestione completamente automatizzata dei sondaggi. Tali caratteristiche hanno fatto si che l'utente medio di Surveygizmo si sia spostato più verso il mondo delle medie-grandi imprese.

Typeform [4] è una piattaforma che utilizza come punto di forza il design e la semplicità di utilizzo, sia per gli utenti che creano sondaggi, sia per coloro che andranno a rispondere. L'interfaccia front-end è incredibilmente curata, e riflette il motto della compagnia

> "Asking questions should be easy, human, and beautiful."

Typeform offre nel piano gratuito un numero illimitato di domande e risposte, la possibilità di esportazione dei dati, e una totale personalizzazione delle domande.

Come ultima alternativa viene proposta la soluzione sviluppata da parte di Google, **Moduli** [5] (in inglese Google Forms). Oltre ad essere 100% gratuita, offrire sondaggi e risposte illimitati e ampie possibilità di personalizzazione, il vantaggio di tale piattaforma è l'integrazione nativa dei risultati dei sondaggi con i servizi di Google, come Google Documenti (Documents) o Fogli (Spreadsheets).

1.1 Utilizzo dei QR-Code

Le più popolari e principali piattaforme hanno già introdotto all'interno del loro ambiente la tecnologia del QR-Code, seppure in forma ridotta: esse offrono all'utente la possibilità di associare l'indirizzo della pagina web del sondaggio ad un QR-Code univoco; la scansione di tale QR-Code è semplicemente una facilitazione nella digitazione dell'indirizzo della pagina. Una volta caricato il sondaggio, l'utente prosegue la compilazione del sondaggio nel modo tradizionale, ovvero selezionando la/le risposte.

L'esempio più strettamente correlato al lavoro svolto è stato identificato nella piattaforma **SurveySwipe** [6].

Essa fornisce la possibilità di creare un semplice sondaggio con due risposte, una positiva e una negativa, a cui sono associati due QR-Code differenti. La scansione di un codice reindirizza l'utente ad una pagina web in cui viene registrata la preferenza corrispondente al QR-Code scansionato.

Figura 1.1: Esempio di sondaggio a risposta positiva o negativa

Capitolo 2

Stato dell'arte

2.1 Descrizione generale

QR-VEY è una piattaforma software per la creazione di sondaggi basata sulla tecnologia del QR-Code. Essa permette all'utente, previa registrazione di un account, di creare un sondaggio con un numero arbitrario di risposte.

L'idea alla base della piattaforma, ispirata dal gruppo di lavoro *World Harbour Project*, è quella di rendere possibile la creazione di un sondaggio, su supporto cartaceo, in cui ad ogni risposta è associato un QR-Code. Un utente interessato alla causa del sondaggio, per votare, non dovrà fare altro che trovarsi nei pressi di una delle sue copie cartacee e usare il proprio smartphone per scansionare il QR-Code relativo alla risposta desiderata: la scansione di tale codice porterà alla registrazione del voto, nonché alla richiesta completamente facoltativa di inserimento di dati personali (i.e. registrazione), da parte del votante, per finalità puramente statistiche. La scelta del votante di non inserire i propri dati non comporta la perdita del voto.

Oltre al già discusso vantaggio di una elaborazione automatizzata dei dati sulle risposte, la procedura proposta risulta una semplificazione notevole allo sforzo necessario ad un utente per votare. Complessi procedimenti, richieste obbligatorie di registrazione, pagine web non ottimizzate per l'utilizzo da dispositivo mobile, sono tutti fattori che potrebbero causare l'abbandono

prematuro della procedura di voto da parte del votante.

Un utente che ha creato un sondaggio ha la possibilità di visualizzare una pagina ad esso dedicata, la quale mostra i QR-Code associati ad ogni risposta e il numero di voti attualmente ricevuti. Per ogni risposta è stata inserita una modalità di voto alternativa alla scansione di un codice, attivabile attraverso la pressione, per ogni risposta, di un tasto dedicato, posizionato nei pressi del QR-Code. Il requisito di utilizzo di un supporto cartaceo per la distribuzione del sondaggio è soddisfatto attraverso la possibilità di download di un file in formato PDF appositamente strutturato per fornire una visualizzazione chiara e precisa delle informazioni chiave del sondaggio e per permettere una rapida scansione dei codici.

L'analisi e la visualizzazione dei dati raccolti è stata resa disponibile in due diverse modalità:

- attraverso la visualizzazione di grafici integrati nel sito, strutturati per tipologia di analisi (età dei votanti, orario di voto, ...).
- attraverso l'esportazione in formato CSV (*Comma-Separated Values*) dei dati raccolti; in questo modo è possibile una rapida manipolazione dei dati utilizzando un elaboratore di fogli elettronici esterno.

È stata sviluppata una versione mobile della piattaforma sia per dispositivi iOS che Android. L'utilizzo dell'applicazione fornisce un doppio vantaggio:

- fornisce una soluzione integrata e *full stack* alla piattaforma, integrando un lettore di codici QR ed eliminando qualsiasi tipo di necessità di applicazioni esterne.
- rende possibile la creazione "al volo" di sondaggi: ad esempio, se si ha necessità di valutare la qualità di una conferenza appena tenuta, è possibile creare velocemente un sondaggio, utilizzando l'applicazione dedicata, e rendere disponibile ai partecipanti un supporto dal quale sia possibile scansionare i QR-Code, come un tablet o un video-proiettore.

Figura 2.1: Esempio di funzionamento

2.2 Funzionamento di un QR-Code

La tecnologia del QR-Code è una tecnologia universale e largamente utilizzata in numerosi ambiti. Un **QR-Code**, abbreviazione di *Quick Response Code*, è un codice a barre bidimensionale composto da piccoli moduli neri disposti all'interno di uno schema di forma quadrata. Un modulo è l'equivalente di un bit.

Vennero per la prima volta introdotti nel 1994 dalla compagnia giapponese Denso Wave con lo scopo di ampliare le capacità di codifica limitate dei tradizionali codici a barre unidimensionali, i quali non erano in grado di codificare i caratteri degli alfabeti locali Kanji e Kana[7].

Al giorno d'oggi, un QR-Code può essere utilizzato per codificare stringhe composte da caratteri numerici, alfanumerici e binari, oltre agli originali caratteri giapponesi.

Figura 2.2: Struttura di un QR-Code

La figura 2.2 evidenzia la struttura di un QR-Code:

- i tre grandi quadrati colorati di rosso sono utilizzati per determinare la posizione e l'orientamento con cui si sta scansionando il codice; aiutano lo scanner a identificare i bordi del codice.

- il quadrato piccolo evidenziato in rosso è il punto di riferimento utilizzato dallo scanner per verificare l'allineamento del codice; nei codici più grandi se ne trova più di uno.

- le strisce rosse che collegano i quadrati grandi sono un pattern prefissato e identico in ogni qr-code, detto *timing pattern*, che guida lo scanner nell'identificazione di righe e colonne.

- le sezioni verdi determinano il formato del codice, indicando se la codifica è relativa ad un URL, semplice testo, ideogrammi cinesi, numeri o una combinazione di questi.

- le sezioni evidenziate in blu rappresentano il numero della versione di codifica utilizzata, la quale indica la capacità di codifica del codice, espressa in moduli. La versione 1 corrisponde ad uno schema 21x21, e si può arrivare fino alla versione 40, corrispondente ad uno schema 177×177. Se la dimensione della codifica è minore della capacità della versione 6, tale sezione non è necessaria in quanto lo scanner è in grado di definirla autonomamente.

- la porzione rimanente del codice viene raggruppata a gruppi di 8 moduli (8 bit, ovvero 1 byte). Ciò significa che cambiare anche un singolo modulo rende l'intero byte illeggibile.

Un codice QR, dunque, può memorizzare fino a un massimo di 4.296 caratteri alfanumerici, o 7.089 caratteri numerici. In caso il codice sia danneggiato ed alcuni moduli risultino illeggibili viene applicato l'algoritmo Reed-Solomon, un algoritmo di correzione degli errori che permette la corretta decodifica del codice anche in caso di moduli mancanti, modificati, o coperti da un'immagine; tale algoritmo viene frequentemente sfruttato per sovrapporre ai moduli del codice una piccola immagine o un logo che ne indichi la compagnia di riferimento.

2.3 Struttura

La piattaforma si suddivide in 3 macro-categorie:

- Back-end
- Font-end
- Applicazioni per dispositivi mobili
  - iOS
  - Android

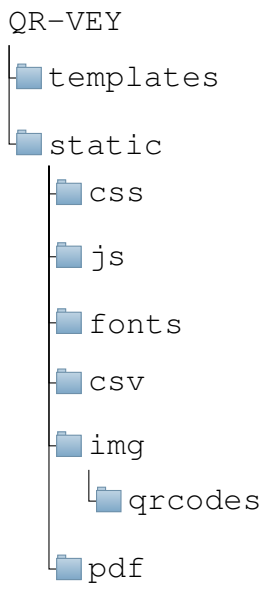
2.3.1 Back-end

Il framework utilizzato per la logica del server è stato **Flask** [8], un micro-framework scritto in Python basato sul motore di template Jinja2 e sul set di strumenti per interfacce web server Werkzeug. L'aggettivo *micro-framework* non è da intendersi come strettamente "povero di funzionalità", ma deriva dal fatto che Flask non richiede nessun particolare strumento o libreria per il suo funzionamento [9]: non presenta alcun *data abstraction layer* per la comunicazione con un database, non contiene alcun meccanismo di validazione di form, di upload/download di dati, etc. La filosofia degli sviluppatori è quella di mantenere il nucleo del framework semplice, ma fortemente estensibile: tutte le funzionalità aggiuntive sono gestibili esternamente, attraverso librerie di terze parti o manualmente. La scelta di tale framework è stata effettuata proprio in base a queste caratteristiche: è un framework estremamente leggero, ed è possibile strutturarlo per ottenere esattamente ciò che serve, senza ottenere funzionalità superflue e non utilizzate.

L'installazione di Flask e di tutte le librerie di terze parti avviene attraverso lo strumento da riga di comando **pip**, acronimo ricorsivo per *Pip Installs Packages*, il sistema di gestione di pacchetti caratteristico dell'ambiente di Python.

Per convenzione, Flask richiede una struttura dei file ben delineata: modelli e file statici sono memorizzati in sottodirectory all'interno della cartella principale del progetto, affiancati da un modulo Python che rappresenta il cuore logico del server. Nonostante tale configurazione possa essere personalizzata, è stata ritenuta perfettamente funzionale ai fini del progetto.

La struttura di directories risultante è stata dunque la seguente:



Il contenuto delle cartelle sarà discusso nelle sezioni più opportunamente correlate.

Sono state valutate diverse opzioni per la scelta dell'*host* per la piattaforma. Inizialmente sia la logica server che il database sono stati implementati su un server personale, ma la soluzione si è rivelata inadeguata. La valutazione delle effettive necessità di risorse della piattaforma e la natura accademica del progetto hanno portato alla migrazione dello stesso verso un server universitario interno al Dipartimento di Informatica.

Viste le rigide limitazioni imposte dalla politica di gestione degli account del Dipartimento, è stato deciso di installare Flask in un ambiente di lavoro virtuale costruito ad-hoc. A tale scopo è stato utilizzato **virtualenv** [10], uno strumento disegnato per isolare gli ambienti di diversi progetti Python.

Ciò ha permesso di risolvere i seguenti problemi:

- conflitti relativi alle molteplici e inerentemente differenti versioni di Python installate sulle macchine di laboratorio.
- impossibilità di installazione di librerie di terze parti con i permessi standard di un account studente.
- conflitti relativi a diverse versioni di librerie di terze parti già installate.

Infine, vista la natura fortemente strutturata dei dati da raccogliere, la scelta del *Database Management System* (DBMS) è stata ristretta alla famiglia dei database relazionali, e in particolare a **MySQL**: il criterio di scelta è stata la semplicità di integrazione nell'ambiente di Flask e l'implementazione fortemente rivolta a piccole e medie piattaforme web.

2.3.2 Front-end

L'interfaccia per l'utente finale è stata strutturata per fornire la massima chiarezza e semplicità d'uso.

Per fornire familiarità all'utente, è stato utilizzato come framework grafico **Materialize**: esso è un motore che si ispira al design recentemente introdotto dal sistema operativo per dispositivi mobili Android, ormai da un paio di anni il sistema più diffuso al mondo [11].

Le possibilità di utilizzo della piattaforma sono state suddivise nelle seguenti categorie:

- Homepage
- Registrazione
- Accesso
- Pannello utente
- Creazione di un nuovo sondaggio

- Visualizzazione dei dettagli di un sondaggio
- Visualizzazione del pdf relativo ad un sondaggio (in una pagina dedicata)
- Risultato di voto
- Errore

Seguendo il modello descritto nella Sottosezione 3.1.2 del capitolo relativo all'Implementazione, per ognuno di questi casi d'uso è stato creato un file template in cui è stato inserito il contenuto statico di presentazione, riassumibile nella seguente struttura tri-partita:

- header, contenente gli strumenti di navigazione (semi-dinamico)
- content, per la visualizzazione del contenuto (dinamico)
- footer, per le informazioni relative alla piattaforma (statico)

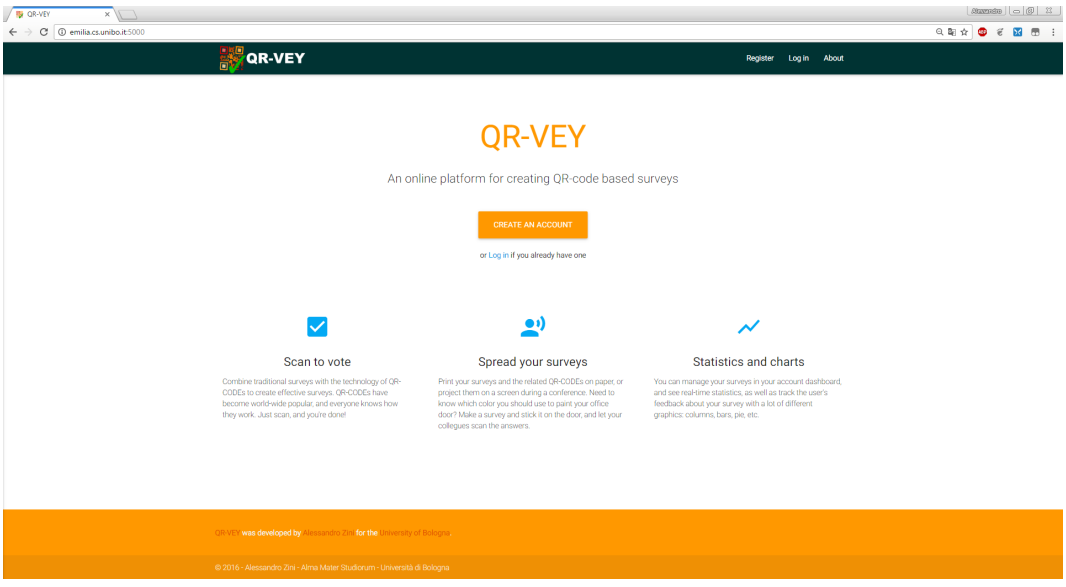


Figura 2.3: Homepage - la struttura tripartita è ben visibile

La natura fortemente *mobile-oriented* del progetto ha inoltre richiesto lo sviluppo di un'interfaccia responsiva a qualunque dimensione di schermo. A tale scopo sono stati utilizzati gli strumenti messi a disposizione dal framework Materialize: la pagina è suddivisa in 12 parti, dette colonne. È possibile specificare il numero di colonne che un contenitore (e.g. l'elemento `div`) occupa all'interno dello spazio della pagina utilizzando l'attributo HTML `class`; inoltre, è necessario posizionare l'elemento in un contenitore padre, il quale ha il ruolo di specificare quali elementi e quale porzione della pagina occuperà lo spazio desiderato.

Ad esempio, un contenitore che occupa tutto lo spazio della pagina:

Frammento 2.1: Esempio di griglia

```
<div class="row">
  <div class="col s12"><p>Elemento di 12 colonne</p></div>
</div>
```

Analizzando il frammento proposto, si nota che il numero di colonne che è stato specificato come larghezza del contenitore interno ha come prefisso la lettera "s": contrariamente a quello che si potrebbe pensare, tale lettera non significa *size*, ma *small*, ed è la chiave per la progettazione di un'interfaccia responsiva.

In Materialize sono stati definiti 3 prefissi di classi:

- `s` - si applica ai dispositivi mobili la cui larghezza è minore o uguale a 600px (~16cm); generalmente tali dispositivi sono smartphone.
- `m` - si applica a tutti i dispositivi mobili la cui larghezza è compresa tra 601px e 992px (~26cm); generalmente, tablet.
- `l` - si applica ai dispositivi la cui larghezza è maggiore di 993 pixel; generalmente, dispositivi desktop.

La progettazione dell'interfaccia responsiva si basa sull'utilizzo di tali prefissi: essi rendono possibile la specifica della quantità di spazio occupato

da un contenitore (e, di conseguenza, dagli elementi al suo interno) a seconda della dimensione dello schermo. Nell'esempio precedente è stato utilizzato il prefisso della classe *small* poiché la propagazione di tali classi avviene in modo crescente in base alle dimensioni.

Ad esempio, è possibile specificare una serie di elementi che in caso di schermo dalle dimensioni ridotte occupano l'intero spazio della pagina, e che vengono progressivamente affiancati man mano che la dimensione cresce:

Frammento 2.2: Layout responsivo

```
<div class="row">
  <div class="col s12"><p>Elemento di 12 colonne</p></div>
  <div class="col s12 m4 l2"><p>Elemento di 12 colonne su
     smarthpone, 4 su tablet e 2 su desktop</p></div>
  <div class="col s12 m4 l8"><p>Elemento di 12 colonne su
     smarthpone, 4 su tablet e 8 su desktop</p></div>
  <div class="col s12 m4 l2"><p>Elemento di 12 colonne su
     smarthpone, 4 su tablet e 2 su desktop</p></div>
</div>
```

La strutturazione dell'elemento *header* ha richiesto uno studio particolare. Come risultato, né stato modellato il comportamento secondo due casi d'uso:

- utilizzo da desktop: l'elemento propone una struttura bipartita, dove sul lato sinistro è stato inserito il logo della piattaforma, mentre il lato destro ospita un piccolo menù. Le opzioni di tale menù variano a seconda della pagina in cui ci si trova: in questo modo è possibile fornire all'utente una varietà di opzioni mirata e basata sul contesto.

- utilizzo da dispositivo mobile: l'elemento propone una struttura bipartita dove al centro è posizionato il logo della piattaforma, mentre il menù originariamente posizionato a destra è stato spostato sul lato sinistro e sostituito con l'icona classica dei menù a comparsa laterale. Il contenuto di tale menù è dinamico, in modo analogo al precedente.

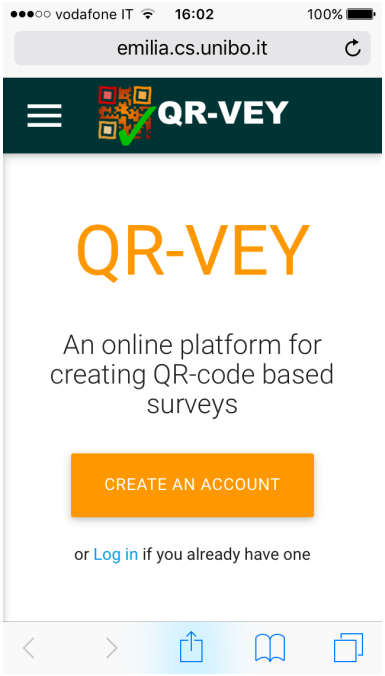


Figura 2.4: Homepage responsiva

2.3.3 Applicazioni per dispositivi mobili

La progettazione dell'applicazione per dispositivi mobili è stata effettuata inizialmente in ambiente iOS, ma sin dal principio si è cercato di sviluppare una strategia comune alle due piattaforme, vista la volontà di porting anche su sistema Android. A tale scopo, si è cercato di stabilire una corrispondenza tra gli elementi di interfaccia dei due sistemi operativi, analizzandone aspetti e proprietà comuni.

Il risultato è stato un layout strutturalmente differente in quanto a componenti, ma la cui familiarità si mantiene attraverso i diversi dispositivi. Per ogni dispositivo si è anche tenuto conto dello stile di presentazione a cui un utente è naturalmente abituato durante l'utilizzo del sistema.

Viene ora descritta questa struttura, tralasciando i dettagli implementativi e il funzionamento delle tecnologie utilizzate. Tali argomenti saranno discussi nel capitolo successivo.

Sono state raggruppate 5 possibilità di utilizzo dell'applicazione da parte di un utente registrato:

- Visualizzazione dei sondaggi
  - Visualizzazione dei QR-Code e delle statistiche delle risposte
- Visualizzazione delle statistiche generali dei sondaggi
- Possibilità di scansione di un QR-Code
- Possibilità di inserimento di un nuovo sondaggio
- Visualizzazione di opzioni aggiuntive (e.g. Logout, ...)

La visualizzazione dei sondaggi attivi corrisponde alla pagina principale nella quale l'utente verrebbe reindirizzato utilizzando la versione browser della piattaforma; entrambe le applicazioni permettono di visualizzare tale schermata solamente previa autenticazione o registrazione.

iOS

L'applicazione relativa al sistema iOS vede alla base del suo scheletro un *UITabBarController*: tale componente è l'elemento centrale attorno al quale ruota l'interfaccia utente; esso racchiude al suo interno 5 *UIView*, utilizzate con la semantica di "schede", corrispondenti ognuna ad una delle 5 possibilità di utilizzo descritte in precedenza. Nei casi in cui si è resa necessaria una navigazione nidificata all'interno di una scheda sono stati utilizzati componenti *UINavigationController* per la gestione della navigazione stessa.

Il completamento della procedura di autenticazione porta l'utente nella *UIView* principale, *HomeView*. La lista dei sondaggi attivi è presentata all'utente attraverso una *UITableView*: tale componente è stato scelto soprattutto per la visualizzazione pulita e strutturata che fornisce, caratteristiche che soddisfano i requisiti di semplicità prefissati. Ogni sondaggio ha una pagina dedicata in cui ne vengono presentati i dettagli quali Titolo, Descrizione, Data di sottomissione, e tutte le risposte con i QR-Code associati.

Il componente *UITableView* è stato utilizzato anche per la presentazione delle statistiche dei sondaggi, visualizzabili nella scheda *StatsView*.

Attraverso la scheda *ScanView* è invece possibile eseguire la scansione di un sondaggio: l'effettiva scansione viene effettuata in una UIView creata dinamicamente per ospitare l'utilizzo della fotocamera, e l'esito della procedura di voto viene dunque presentato nella schermata originale.

La sottomissione di un nuovo sondaggio avviene attraverso la *NewSurveyView*: l'inserimento delle risposte è stato effettuato anch'esso per mezzo di una *UITableView*, in cui l'utente può inserire tante righe quante le risposte desiderate.

Infine, la presentazione delle opzioni aggiuntive è stata realizzata nella scheda *MoreView*, in cui è presente anche l'opzione di disconnessione. In tale scheda sono stati inseriti dei *placeholder* relativi a funzionalità di cui è stata pensata, ma non ancora effettuata, l'implementazione, poiché l'utilità effettiva è ancora in corso di valutazione.

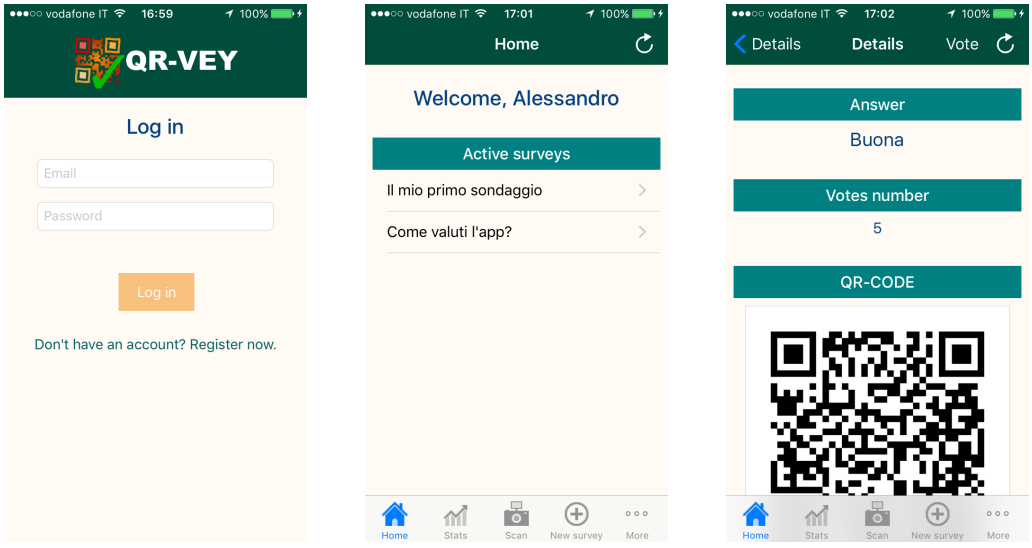


Figura 2.5: Applicazione iOS

Android

A differenza della controparte iOS, la navigazione tra le varie funzionalità nell'applicazione Android è stata implementata utilizzando un *Navigation Drawer*: la strutturazione dell'interfaccia presentata da tale componente è stata ritenuta quella più popolare in ambiente Android.

In un primo momento, la strutturazione interna era stata eseguita sullo stesso stile dell'applicazione iOS: era stata creata un'*Activity* (assimilabile come l'equivalente Android di una *UIView*) per ogni possibilità di utilizzo descritta. Un'attenta analisi e una maggiore comprensione del sistema utilizzato dal *Navigation Drawer* ha portato all'eliminazione delle diverse *Activity* in favore dell'impiego di *Fragments*. [12] Sono state mantenute invece le Activity relative alle procedure di accesso, registrazione, e per la visualizzazione dei dettagli di un sondaggio, la cui gestione ha richiesto un'Activity separata.

L'applicazione è quindi costruita attorno ad una *MainActivity*, la quale agisce come ponte tra le attività di accesso e quelle di effettivo utilizzo. Essa si occupa della gestione della navigazione fra i vari Fragment e della preparazione iniziale dell'applicazione nel momento in cui essa viene caricata.

La visualizzazione dei sondaggi avviene attraverso l'*HomeFragment*: esso permette la visualizzazione e la selezione dei sondaggi attivi attraverso l'utilizzo del componente *ListView*. Come nella versione iOS, ogni sondaggio ha una pagina dedicata in cui ne vengono presentati i dettagli quali Titolo, Descrizione, Data di sottomissione, e tutte le risposte con i QR-Code associati.

I componenti *StatsFragment* e *NewSurveyFragment* svolgono un ruolo equivalente alle controparti iOS.

La scansione di un QR-Code avviene attraverso lo *ScanFragment*: la strutturazione è stata eseguita in modo che il componente che permette la visualizzazione della fotocamera sia effettivamente un Fragment, e non un'Activity separata. Ciò ha reso possibile sviluppare una modalità di utilizzo della fotocamera non a schermo intero, facilitando l'implementazione del sistema di scansione e voto.

Contrariamente a quanto avviene nell'applicazione iOS, non è stato inserito un Fragment relativo alle opzioni aggiuntive: per tale scopo è stato utilizzato il sistema di *ActionBar* integrato nell'Activity principale. Attraverso questo metodo è stato possibile:

- integrare le opzioni aggiuntive, come la possibilità di disconnessione, in tutti i Fragment gestiti dalla *MainActivity*.
- sviluppare un'unica *ActionBar* con funzionalità comuni a tutti i Fragment.
- utilizzare un componente tipico dell'esperienza Android

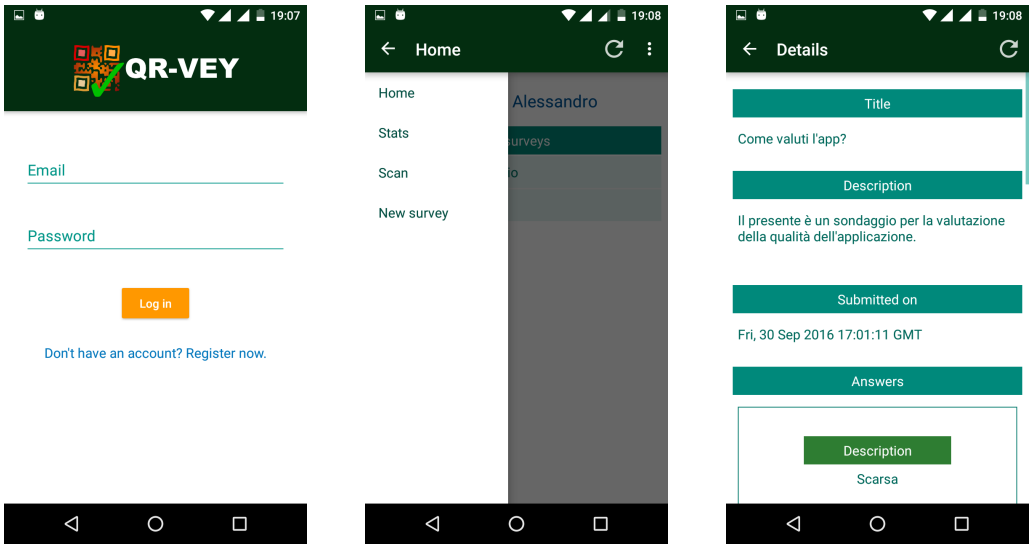


Figura 2.6: Applicazione Android

2.4 Requisiti

La piattaforma e il suo utilizzo sono inerentemente legati ad un servizio basato su internet, e dunque la presenza della connessione alla rete mobile o Wi-Fi è un requisito fondamentale.

Nonostante sia stato deciso di inserire una metodologia di voto alternativa a quella basata sulla scansione dei QR-Code, come accennato nella Sezione 2.1, tale metodologia è da considerarsi "di emergenza". Pertanto, un utilizzo normale della piattaforma richiederà l'accesso ad una fotocamera per la scansione dei QR-Code.

L'utilizzo della piattaforma attraverso browser richiede l'abilitazione all'utilizzo di

- JavaScript, utilizzato per la logica client-side e per alcune funzionalità grafiche
- Cookies, utilizzati per il meccanismo di voto e per quello di accesso

La minima versione di iOS richiesta per l'applicazione è la numero 8; tale scelta permette l'utilizzo della piattaforma sul 97% di tutti i dispositivi iPhone al mondo [13].

La minima versione richiesta per l'applicazione Android è invece la numero 4.0.3, denominata Ice Cream Sandwich, permettendo di raggiungere il 98,4% dei dispositivi Android [14].

Android		
Versione	**Identificativo**	**Distribuzione**
2.2	Froyo	0.1%
2.3.3 - 2.3.7	Gingerbread	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	1.4%
4.1.x - 4.3	Jelly Bean	15.6%
4.4	KitKat	27.7%
5.0 - 5.1	Lollipop	35%
6.0	Marshmallow	18.7%

Distribuzione del sistema Android

iOS	
Versione	**Distribuzione**
iOS 8	9%
iOS 9 - 10	88%
precedenti iOS 8	3%

Distribuzione del sistema iOS

Capitolo 3

Implementazione

Lo scopo di questo capitolo è fornire una descrizione pratica della strumentazione utilizzata e delle scelte implementative prese in corso di sviluppo.

La descrizione è suddivisa nelle macro-categorie introdotte nel capitolo precedente.

3.1 Back-end

Lo sviluppo dell'intera parte back-end è stata effettuata utilizzando come editor di testo Sublime Text 3, un semplice editor capace di adattarsi straordinariamente bene a diversi linguaggi e a diverse metodologie di sviluppo, grazie soprattutto all'ampio supporto per plugins.

3.1.1 DBMS

La prima parte dell'implementazione è stata rivolta alla progettazione del database. La configurazione finale vede l'utilizzo di 4 tabelle: SURVEY, ANSWER, REGISTERED_USER,VOTER.

In dettaglio (sono state sottolineate le chiavi primarie):

SURVEY				
id	name	email	submit_time	description

ANSWER				
id	survey_id	votes_number	description	qrcode

REGISTERED_USER					
email	hashed_password	first_name	last_name	age	sex

VOTER						
id	survey_id	answer_id	email	time	device	browser

Si procede ora con una breve descrizione delle caratteristiche di tale progettazione:

- i campi *id* delle tabelle SURVEY e VOTER sono stati definiti come auto-incrementali, in modo da permetterne la gestione automatica da parte del database.

- il campo *id* della tabella ANSWER è relativo al sondaggio in considerazione; per questo motivo, una risposta è identificata univocamente dalla coppia (*id*, *survey_id*).

- il campo *qrcode* della tabella ANSWER non contiene l'immagine effettiva, ma contiene il percorso del file di immagine, salvato sul server, rappresentante il QR-Code. Poiché una risposta è identificata univocamente dall'id del sondaggio e dell'id relativo della risposta, il salvataggio dei file immagine relativi ai QR-Code ha seguito la stessa logica di denominazione. Per questo motivo, l'inserimento del campo *qrcode* nella tabella è una ridondanza, ma è stato ritenuto utile inserirla per semplificare parte del codice lato server. Analogamente, anche il campo *votes_number* è stato introdotto in ridondanza per lo stesso obiettivo di semplificazione.

- come descritto nella sezione 2.1, i dati personali di un votante sono facoltativi. Il metodo di implementazione della votazione è stato gestito nel modo seguente: inizialmente, un votante viene registrato nella tabella VOTER con il campo *email* nullo, e viene creato un particolare cookie nel browser che indica la presenza di una registrazione di voto in attesa. A questo punto:
  - un votante che preferisca votare anonimamente non dovrà compiere nessuna azione; potrà chiudere la pagina o visitarne un'altra, provocando così la scadenza (temporale) del cookie.
  - un votante che decida di condividere i suoi dati personali:
    - se già in possesso di un account e già acceduto, il voto registrato nella tabella VOTER viene arricchito dell'email corrispondente all'account utente.
    - se già in possesso di un account ma non acceduto, o se non ancora in possesso di un account, l'utente viene reindirizzato alla pagina di accesso / registrazione; una volta effettuata tale operazione, il cookie verrà cancellato e il campo *email* sarà aggiornato con l'indirizzo corretto.
- nella tabella VOTER, i dati relativi ai campi *device* e *browser* sono automaticamente raccolti lato-server eseguendo un parsing della richiesta HTTP ed estraendo dall'header "User-Agent" le informazioni relative.
- il campo *hashed_password* della tabella REGISTERED_USER, come si evince dal nome, contiene un hash della password, con salt di lunghezza 8; la funzione di hash utilizzata è PBKDF2-HMAC-SHA1 [15].

Le funzionalità di creazione di un nuovo utente, aggiunta di un sondaggio e aggiunta di un votante sono state gestite interamente per mezzo di *Stored Procedure*. Una Stored Procedure è una serie di istruzioni SQL salvate direttamente sul DBMS Server; il vantaggio risiede nel fatto che un client che si connette ad un server di un database non dovrà eseguire ogni volta tutte

le operazioni contenute della procedura, ma potrà semplicemente invocare la chiamata alla procedura stessa. Inoltre, le Stored Procedure ottimizzano la comunicazione, minimizzando la quantità di dati da scambiare; come contro-effetto si ottiene un maggior carico di lavoro sul DBMS Server, ma visto il limitato regime di richieste tale conseguenza è stata ritenuta pienamente accettabile.

Per non abusare della disponibilità di spazio offerta dai server universitari, sono stati imposti dei limiti alle dimensioni delle informazioni, in particolare quelle relative a titolo e descrizione. Una futura migrazione verso un server esterno potrebbe valutare l'ampliamento di tali limiti.

L'integrazione delle funzionalità di comunicazione verso DBMS MySQL è stata inserita in Flask attraverso la libreria *flask-mysql*.

3.1.2 Flask

Il linguaggio utilizzato per l'implementazione della logica server-side è **Python 2.7**: è necessario precisare che, nonostante tale scelta sarebbe stata presa ugualmente, alcune librerie utilizzate non hanno esteso il supporto alla versione 3 di Python.

Per comprendere il funzionamento del framework Flask e della progettazione effettuata, viene proposto l'esempio classico dell'informatica: *Hello World*.

Frammento 3.1: Hello World in Flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/", methods=["GET"])
def hello_world():
   return "Hello, World!"
```

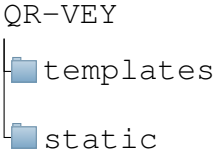
Il risultato di tale codice è la visualizzazione di una pagina web contenente la stringa "Hello, World!". Come prima operazione viene importata la classe Flask, istanziandone un oggetto. Il parametro specificato è la variabile speciale `__name__`: tale variabile è utilizzata dall'ambiente Python, e più in specifico da Flask, per identificare se il modulo corrente è stato eseguito come applicazione principale, o se è stato importato da un modulo esterno; nel primo caso, essa avrà valore ¨`__main__`¨, nel secondo conterrà il nome del modulo esterno da cui è stato importato il modulo corrente. Tale meccanismo è utilizzato da Flask per impostare correttamente i parametri di ricerca di templates, file statici, etc.

L'istruzione seguente fa uso di un'annotazione particolare, un *Decorator*, denotata dal simbolo `"@"`. In Python, un Decorator è un'aggiunta sintattica attraverso la quale è possibile alterare ed arricchire metodi e funzioni con informazioni aggiuntive. Nel caso di Flask, tale meccanismo è utilizzato per specificare a quale URL fa riferimento la definizione di funzione immediatamente successiva: la visita da parte di un browser dell'URL specificato innescherà la chiamata di funzione. Assieme all'URL, è possibile specificare il metodo HTTP di visita della pagina; nell'esempio è stato utilizzato il metodo `GET`.

Contenuti statici e contenuti dinamici: il motore Jinja2

Il valore di ritorno della funzione `hello_world` contiene il codice HTML della pagina che il browser andrà a caricare. Il frammento proposto è un esempio minimale di funzionamento; il caricamento di una pagina completa avviene attraverso diverse fasi.

Come descritto nella Sottosezione 2.3.1, la cartella principale del progetto contiene due sotto-directories:



Come si evince dal nome, la cartella *templates* contiene gli scheletri delle pagine web di cui è composta la piattaforma. Un template viene presentato all'utente attraverso la funzione `render_template("template_name")`, che ricerca nella sotto-directory *templates* il template con il nome specificato.

Il contenuto di un template base è speculare a quello di una normale pagina HTML: utilizzando esclusivamente template base, la semantica della funzione `render_template` è semplicemente quella di restituire il codice HTML contenuto nel file.

Flask permette di accompagnare questo comportamento statico ad un sotto-sistema di template interni ad ogni documento, permettendo di utilizzare di fatto un modello in cui il contenuto statico è popolato non solo da un eventuale contenuto dinamico lato client, ma anche da contenuto generato direttamente lato-server. I sotto-template sono indicati, nel codice HTML, attraverso l'utilizzo di doppie parentesi graffe, e il loro contenuto viene passato come parametro alla funzione RENDER_TEMPLATE.

Il precedente esempio di *Hello World* diventa dunque:

Frammento 3.2: Flask template - Python

```
from flask import Flask
app = Flask(__name__)

@app.route("/", methods=["GET"])
def hello_world():
   return render_template("hello_world.html",
      message="Hello, World!")
```

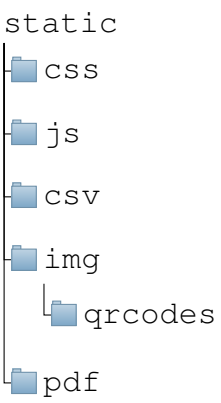
Mentre il file `hello_world.html` può essere strutturato come segue:

Frammento 3.3: Flask template - HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{message}}</title>
  </head>
  <body>
   <p>{{message}}</p>
  </body>
</html>
```

La cartella static

Il contenuto della cartella static è ripartito come segue:



L'unico scopo di tale cartella è il caricamento degli elementi statici che fanno parte della piattaforma:

- file di presentazione (css)
- logica client-side (js, i.e. JavaScript)
- file di supporto (csv, qrcodes, pdf, ...)

3.1.3 Deployment

Le possibilità di distribuzione messe a disposizione da Flask sono numerose. Per semplicità di utilizzo, e tenendo conto dei limiti evidenziati nella Sottosezione 2.3.1, è stato scelto di utilizzare un server con il supporto per applicazioni Python WSGI [16]. WSGI è una specifica di interfaccia standard attraverso la quale un'applicazione ed un server comunicano; essendo Flask scritto su tale specifica, è stato sufficiente scegliere un server *WSGI-compliant* in grado di servire richieste sulle macchine di laboratorio: la scelta è ricaduta sul server **Gunicorn**, un server WSGI HTTP per sistemi UNIX.

Note

Attualmente, la piattaforma è legata allo stato di uno specifico computer del laboratorio Ercolani, Emilia. È stato impostato Gunicorn per fare in modo che tutte le richieste sulla porta 5000 di tale macchina siano reindirizzate al server sviluppato, tuttavia è evidente come questa sia semplicemente una soluzione temporanea e di natura progettuale.

3.2 Front-end

Lo sviluppo Front-end, così come la parte Back-end, ha visto l'utilizzo come strumento principale dell'editor Sublime Text 3.

Per l'implementazione dell'interfaccia web è stato utilizzato **HTML** come linguaggio di markup, mentre la logica client-side è stata sviluppata utilizzando **Javascript**, al quale è stata affiancata la libreria **jQuery**. jQuery è una libreria Javascript, open-source ed estremamente leggera, nata con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML. La peculiarità di jQuery è l'utilizzo del simbolo `$`: esso è un alias del costruttore `jQuery()`, e fornendogli come parametro di input un selettore del DOM esso restituisce in output l'elemento jQuery corrispondente. Attraverso questo semplice meccanismo di selezione è possibile eseguire, su tutti gli elementi del DOM, le chiamate alle funzioni introdotte dalla libreria, che risultano generalmente molto semplificate rispetto alle controparti Javascript. L'utilizzo di jQuery ha permesso anche la notevole semplificazione dei meccanismi di comunicazione con il server, che verranno descritti più avanti.

3.2.1 jQuery Validation

La prima parte dello sviluppo è stata incentrata sull'implementazione delle metodologie di accesso e registrazione da parte dell'utente. La raccolta dei dati dell'utente è stata effettuata attraverso l'elemento HTML `form`; ad esso è stato applicato il plug-in **jQuery Validation**, utilizzato per l'implementazione di un meccanismo efficiente ed allo stesso tempo elegante di validazione dell'input. Il funzionamento di tale plug-in si basa sulla definizione di regole che vengono applicate ai campi di input del form in considerazione.

Ad esempio, è possibile definire un semplice form del tipo:

Frammento 3.4: jQuery Validation - HTML

```
<form>
  <input type="text" name="short_text">
</form>
```

A questo punto, si può impostare il plug-in per richiedere che l'input non sia vuoto, e che contenga un numero minimo di caratteri (e.g. 10 caratteri):

Frammento 3.5: jQuery Validation - JavaScript

```
$("form").validate({
  rules: {
    short_text: {
      required: true,
      maxlength: 10,
    }
  },
  messages: {
    short_text: {
      equalTo: "Insert between 1 and 10 characters."
    }
  },
  errorElement : "div",
  errorPlacement: function(error, element) {
    var placement = $(element).data("error");
    if (placement) {
      $(placement).append(error)
    } else {
      error.insertAfter(element);
    }
  }
});
```

La parte finale del codice è utilizzata per specificare come e dove visualizzare un eventuale messaggio di errore.

La reale utilità di tale plug-in risiede nella possibilità di impostare parametri di validazione personalizzati: ad esempio, la validazione in fase di registrazione di un indirizzo email prevede una richiesta asincrona al server, il quale controlla che l'indirizzo non sia già stato utilizzato.

3.2.2 AJAX

La seconda parte dell'implementazione dell'interfaccia ha visto lo sviluppo della comunicazione client-server. Il tipo di comunicazione che è stato sviluppato utilizza il modello **AJAX**, acronimo di *Asynchronous JavaScript and XML*. Una comunicazione di questo tipo si basa su uno scambio di dati asincrono tra due estremi di una comunicazione: i dati che vengono richiesti da un client ad un server vengono ricevuti in background, senza che tale trasferimento interferisca con il normale comportamento della pagina già caricata. Dunque, l'utilizzo di AJAX in ambito web consente il caricamento dinamico di contenuto in una pagina senza la necessità di ricaricamento dell stessa, permettendo di fatto la realizzazione di applicazioni web interattive. Nonostante il nome, una richiesta di questo tipo non è effettuabile esclusivamente utilizzando JavaScript e XML.

La libreria jQuery implementa un'interfaccia semplice per eseguire richieste AJAX. Ogni comunicazione tra client e server, fatta esclusione per quelle relative al caricamento di pagine, è stata realizzata attraverso richieste di questo tipo.

3.3 Applicazioni per dispositivi mobili

Le scelte implementative delle due applicazioni sono risultate simili nella parte di presentazione e visualizzazione dei dati, ma strutturalmente diverse per quanto riguarda la gestione delle comunicazioni con il server e il salvataggio dei dati stessi.

3.3.1 iOS

L'applicazione per il sistema iOS è stata sviluppata utilizzando lo strumento ufficiale **Xcode**, versione 7.3.1: esso è un ambiente di sviluppo integrato (i.e. IDE) che permette lo sviluppo di software per sistemi MacOS e iOS. Xcode supporta i linguaggi C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, Rex e Swift, ed è completamente sviluppato e mantenuto da Apple. Nonostante l'ambiente di sviluppo fornisca la possibilità di utilizzo di un emulatore, la maggior parte dei test è stata effettuata su un dispositivo fisico, per la precisione un iPhone 5S. Il linguaggio di sviluppo scelto è stato **Swift**.

Novità di Swift

Swift è un linguaggio di programmazione object-oriented introdotto da Apple nel 2014. Esso integra al suo interno alcune tra le migliori caratteristiche e funzionalità di diversi linguaggi di programmazione come C#, Python, Ruby, e molti altri. L'obiettivo principale dell'introduzione di Swift è stato quello di eliminare la pesantezza e i contenuti "*legacy*" del linguaggio C, di cui Objective-C era un super-set.

Tra le caratteristiche peculiari di Swift possiamo trovare:

- introduzione della *type inference*: non è necessario specificare esplicitamente il tipo di variabili e costanti se questo può essere determinato dal contesto.

- snella gestione della memoria, con un garbage collector automatico; in questo modo è stata eliminata la gestione esplicita di allocazione e de-allocazione di puntatori e memoria.
- non è più necessario avere un *header file* ed un *implementation file* separato per ogni classe: la definizione avviene in un unico file con estensione .swift.
- introduzione di optionals, collezioni, array strutturati come liste, dizionari, closures e generics
- (a partire dalla versione 2.2) open source e disponibile in ambiente UNIX

Tutte queste caratteristiche hanno reso determinante lo scarto di Objective-C in favore di Swift.

Salvataggio e caricamento dei dati utente

L'implementazione del meccanismo di accesso e memorizzazione dei dati utente è stato eseguito attraverso l'utilizzo della classe **NSUserDefaults**, la quale fornisce un meccanismo base per il salvataggio di dati persistenti all'interno di un'applicazione. Generalmente, tale classe viene utilizzata per il salvataggio di piccole quantità di dati, come ad esempio le impostazioni di un utente, mentre il salvataggio di dati più pesanti o complessi viene eseguito attraverso l'utilizzo delle API *Core Data*, il framework avanzato di iOS per la gestione dei dati. La scelta implementativa tra il sistema Android e il presente differisce in questo punto: nell'implementazione iOS si è deciso di utilizzare i salvataggio dei dati persistenti esclusivamente per i dati personali relativi all'utente; la lista dei sondaggi, le risposte, i QR-Code, ed altri dati non strettamente legati alle informazioni dell'utente sono stati memorizzati utilizzando un meccanismo di memoria cache. Il vantaggio di tale scelta, oltre alla semplificazione del processo di salvataggio, risiede nella possibilità da parte del sistema operativo di cancellare tali dati in caso di una

situazione di esaurimento di memoria; lo svantaggio risiede nella possibile presenza di piccoli periodi di latenza dovuti allo scaricamento in tempo reale dei dati. Nonostante la quantità decisamente limitata di dati che vengono memorizzati, l'assenza della disponibilità di un supporto di memorizzazione di massa esterno nei dispositivi iOS ha portato alla sperimentazione, per necessità più teorica che pratica, dell'utilizzo di tale memoria cache..

La soluzione proposta è stata testata su dispositivo reale utilizzando diverse tecnologie di trasferimento dati: la tecnologia 3G e 4G non hanno evidenziato ritardi apprezzabili, mentre la latenza introdotta dall'utilizzo della tecnologia EDGE 2G è risultata dell'ordine di 5 secondi, un risultato ritenuto accettabile.

Facendo riferimento alla struttura descritta nella Sottosezione 2.3.3, il processo di caricamento dei dati è stato suddiviso e delegato alle varie *UIView* che compongono l'applicazione. In dettaglio:

- la lista dei sondaggi attivi dell'utente viene caricata alla visualizzazione della *HomeView*.
- le statistiche generali dell'utente vengono caricate alla visualizzazione della *StatsView*.
- le informazioni dettagliate di un sondaggio vengono caricate nel momento effettivo in cui viene selezionato un sondaggio dalla *UITableView* della pagina principale; il caricamento delle risposte avviene in modo analogo.

Strumenti esterni

Nell'implementazione dell'applicazione sono state utilizzate 3 librerie esterne alle librerie standard:

- **IQKeyBoardManager**: già dalle fasi iniziali dello sviluppo si è notato come iOS non fornisca alcun meccanismo di gestione automatica e intelligente della tastiera, al contrario di quanto succede per Android;

si è reso quindi necessario lo sviluppo manuale di diversi di meccanismi di comportamento, come la scomparsa della tastiera in seguito ad un tocco esterno a qualsiasi sorgente di input, oppure la visualizzazione di una piccola toolbar che permetta la navigazione fra tali sorgenti. L'integrazione di questa libreria semplifica notevolmente la gestione e la personalizzazione di questi comportamenti.

- **QRCodeReader.swift**, utilizzata per il riconoscimento dei codici QR scannerizzati attraverso la fotocamera.

- **SwiftMessages**: diversamente dalla situazione di gestione della tastiera, iOS fornisce un metodo di default per la presentazione di brevi messaggi all'utente. Tuttavia, le possibilità di personalizzazione di tali messaggi sono state ritenute insufficienti; l'utilizzo di questa libreria permette la presentazione all'utente di messaggi largamente personalizzabili.

Tutte le librerie sono state installate attraverso l'utilizzo di **CocoaPods**, un sistema a livello applicativo per la gestione delle dipendenze, il quale fornisce una procedura ed un formato standard per l'installazione e l'utilizzo di librerie esterne.

iOS & App Transport Security

Sui dispositivi Apple è presente una funzionalità chiamata App Transport Security (ATS). Tale funzionalità, abilitata di default, aiuta a migliorare la privacy e l'integrità dei dati, assicurandosi che la comunicazione tra l'applicazione e il mondo esterno avvenga esclusivamente attraverso canali che utilizzano protocolli di sicurezza e meccanismi di cifratura standard industriali. In particolare, a partire da iOS 9 viene richiesto che l'applicazione comunichi utilizzando il protocollo HTTPS.

Si è reso necessario disabilitare la funzionalità, in quanto non è stato possibile utilizzare un protocollo di comunicazione cifrato per la connessione alla macchina di laboratorio ospitante.

3.3.2 Android

L'applicazione per il sistema Android è stata sviluppata utilizzando l'ambiente di sviluppo Android Studio 2.2, l'IDE ufficiale per la piattaforma Android sviluppato da Google. È basato su IntelliJ IDEA ed è liberamente disponibile sotto licenza Apache 2.0. Android Studio mette a disposizione un emulatore per ogni tipo di dispositivo Android, offrendo la possibilità di testing del proprio prodotto su un'ampia gamma di risoluzioni e versioni di sistema operativo differenti. Tale possibilità di emulazione è stata ampiamente sfruttata, affiancata all'utilizzo di alcuni dispositivi fisici. Come risultato, la fase di testing dell'applicazione ha utilizzato:

- emulatore, configurato con il fattore di forma di un LG Nexus 5, con sistema operativo Android 5.0 Lollipop.
- Motorola Moto G, Android 6.0 Marshmallow
- HTC Incredible S, Android 4.0.3 Ice Cream Sandwich

Il linguaggio utilizzato per lo sviluppo è Java.

Salvataggio e caricamento dei dati utente

Il meccanismo di memorizzazione dei dati dell'utente è stato implementato attraverso l'utilizzo della classe **SharedPreferences**, una classe che permette il salvataggio all'interno dell'applicazione di coppie chiave-valore di dati; il funzionamento è analogo alla controparte iOS, NSUserDefaults. Tale salvataggio avviene nelle due Activity relative all'autenticazione:

- *LoginActivity*, la quale in caso di accesso riuscito salva lo username dell'utente e attende i dati restanti dal server; una volta ricevuti, vengono salvati anch'essi.
- *SignupActivity*, la quale in caso di registrazione riuscita, avendo già a disposizione le informazioni dell'utente, si limita a salvarle; infine, il controllo viene passato alla *MainActivity*

La possibilità di sviluppo di due applicazioni diverse della stessa piattaforma ha reso possibile una diversa strategia di implementazione del meccanismo di caricamento dei dati. Come descritto nella Sottosezione 3.3.1, l'approccio utilizzato nella versione iOS è stato quello di un caricamento progressivo, effettuato esclusivamente al momento di una effettiva necessità di visualizzazione dei dati.

Per la versione Android è stato utilizzato un approccio differente: attraverso una API appositamente progettata, vengono richiesti *una tantum* al server tutti i dati relativi all'utente, quali sondaggi, risposte, codici QR, etc. Una richiesta di questo tipo avrà inevitabilmente ripercussioni sul tempo di attesa dell'utente a causa del caricamento dei dati, ma eliminerà qualsiasi tipo di latenza secondaria nella visualizzazione di statistiche e dettagli di sondaggi. La richiesta dei dati al server avviene al caricamento dell'Activity principale, la *MainActiviy*; in aggiunta, è stato implementato un semplice pulsante, inserito nella *Action Bar* principale, che permetta un nuovo caricamento di tutti i dati a discrezione dell'utente.

È stata inoltre differenziata la metodologia di salvataggio dei dati ricevuti: il meccanismo di memoria cache è stato abbandonato in favore dell'utilizzo di un metodo persistente anche per i dati non strettamente relativi alle informazioni dell'utente. A tale scopo, si è deciso di catalogare i dati in due categorie:

- dati strutturati: titolo di un sondaggio, data di sottomissione, etc.
- dati non strutturati: file immagine relativi ai QR-Code

Per le due categorie di dati sono stati implementati metodi di memorizzazione differente. Il salvataggio dei dati strutturati è stato effettuato attraverso l'uso del database standard messo a disposizione da Android per le applicazioni, **SQLite**. L'analogia di tale database con MySQL, utilizzato dalla piattaforma principale, ha concettualmente permesso di considerare il database interno come un sottoinsieme strutturato del principale, consentendo la definizione ed l'utilizzo di tabelle speculari tra i due sistemi. Per fornire

chiarezza all'implementazione sono state definite le classi *Survey* e *Answer*, rispettivamente relative a sondaggi e risposte: l'introduzione di tali classi ha semplificato estremamente la gestione dei dati sia in fase di inserimento nel database, sia in fase di lettura e presentazione.

Il salvataggio dei dati non strutturati è stato invece effettuato utilizzando le API messe a disposizione dal sistema Android per la lettura e la scrittura del filesystem. È importante notare come la decisione di salvare tali file sia stata dettata dal fatto che la dimensione di un'immagine rappresentante un QR-Code è inferiore al kilobyte: un'ipotetica situazione in cui un utente abbia all'attivo un numero di sondaggi tale da richiedere l'impiego di 1000 codici QR differenti richiederebbe una quantità di spazio di molto inferiore al megabyte.

Strumenti esterni

Lo sviluppo dell'applicazione è stato effettuato utilizzando le seguenti librerie aggiuntive:

- **Volley**, la libreria standard in ambiente Android per l'implementazione di richieste HTTP.
- **ZXing**, una libreria per la gestione delle funzioni di scansione di QR-Code dalla fotocamera del telefono.
- **ButterKnife**, una libreria che permette la creazione di *istanze* riutilizzabili delle viste che compongono una Activity o un Fragment. L'utilizzo di tale libreria è una semplificazione del metodo canonico utilizzato da Android per la selezione e il "dialogo" tra codice Java e componenti dell'interfaccia. Normalmente, per fare riferimento ad una vista del layout si utilizza il metodo `findViewById(id_della_view)`, che restituisce in output l'oggetto *View* richiesto (se trovato); è necessario ripetere questo procedimento in ogni parte del codice in cui si ha necessità di interagire con la vista in questione. ButterKnife utilizza un sistema di annotazioni per semplificare tale meccanismo, permettendo

di definire una sola istanza di una particolare vista, e di riutilizzare tale istanza in qualunque parte del codice relativa all'Activity o Fragment contenitore.

Ad esempio, è possibile definire un'istanza di una vista di tipo `EditText` in questo modo:

Frammento 3.6: Butterknife - definizione legame

```
@Bind(R.id.emailEditText) EditText _emailText;
```

dove, `R.id.emailEditText` è l'id relativo alla vista, e `_emailText` sarà l'istanza a cui si farà riferimento. A questo punto, se ad esempio si vuole effettuare un'operazione di lettura del testo all'interno della vista si potrà utilizzare semplicemente:

Frammento 3.7: Butterknife - lettura testo

```
_emailText.getText().toString();
```

Capitolo 4

Casi d'uso

Lo scopo della piattaforma è quello di fornire un servizio di creazione di sondaggi che utilizzino una metodologia di risposta basata sulla tecnologia del QR-Code.

Il caso d'uso principale per cui la piattaforma è stata ideata è stato ispirato dal gruppo di lavoro *World Harbour Project*: come descritto nell'introduzione della presentazione, il gruppo è attivo in alcune delle più grandi aree portuali mondiali con il duplice scopo di salvaguardare le specie autoctone di molluschi bivalvi e di limitare la diffusione e l'espansione di fauna non indigena.

La necessità del gruppo è quella di utilizzare una metodologia automatizzata di raccolta di dati e opinioni riguardo al lavoro svolto in una particolare area. A tale scopo, è possibile creare e stampare su un supporto cartaceo un sondaggio, che dovrà essere distribuito nelle aree interessate dall'indagine. Ad esempio, nell'ipotesi che sia stato effettuato un intervento lungo la riva di un fiume costeggiato da una strada, è possibile posizionare il sondaggio su supporti artificiali come pali di illuminazione, panchine, muretti.

L'utilizzo dei QR-Code come metodo di votazione richiederà ad un cittadino interessato il possesso di un smartphone con una fotocamera. Il procedimento di votazione è semplice e immediato: sarà sufficiente utilizzare la fotocamera per inquadrare il QR-Code relativo alla risposta desiderata.

Allo stesso tempo, la necessità della scansione fisica di un codice fornisce le proprietà di localizzazione richieste dal sondaggio, limitando la possibilità di votazione esclusivamente a individui posizionati nei pressi dell'area interessata dall'intervento.

Al termine del sondaggio, un operatore dovrà semplicemente rimuovere i supporti cartacei dalle aree interessate, e potrà analizzare i dati automaticamente raccolti.

Un secondo caso d'uso può essere descritto nel modo seguente. Si prenda come scenario quello di un relatore che ha appena tenuto una conferenza e ha necessità di valutarne la qualità. Utilizzando la piattaforma, tale relatore ha la possibilità di creare un semplice sondaggio e di proiettarlo all'assemblea mediante l'utilizzo di un video-proiettore.

Le dimensioni del supporto di proiezione sono generalmente sufficienti perché un partecipante che decida di esprimere la propria opinione possa utilizzare il proprio smartphone o tablet per la scansione della risposta desiderata anche da una lunga distanza.

Terminata la conferenza, il relatore potrà analizzare i dati raccolti ed utilizzarli per valutare, ed eventualmente migliorare, la qualità della sua esposizione.

Seguendo il modello dei casi d'uso descritti, è possibile estendere l'utilizzo della piattaforma ad un numero decisamente elevato di casi.

Capitolo 5

Conclusioni

Lo stato attuale di sviluppo della piattaforma soddisfa tutti i requisiti prefissati: la tecnologia del QR-Code ha rappresentato il ponte perfetto per permettere un'interazione sinergica tra moderne e tradizionali metodologie di raccolta e analisi di opinioni. Alle proprietà classiche di un supporto cartaceo è stato affiancato il vantaggio dell'elaborazione automatizzata dei dati raccolti, ed il tutto è stato presentato all'utente attraverso un'interfaccia semplice ed intuitiva.

Sono inoltre state fornite soluzioni mobili integrate e *full-stack* che permettono l'utilizzo della piattaforma anche esclusivamente in mobilità.

5.1 Lavori futuri

Nonostante sia possibile utilizzare il servizio esaustivamente, lo sviluppo della piattaforma è stato eseguito in un contesto di prototipazione progettuale. Di seguito sono elencate alcune possibilità di estensione e miglioramento:

- Possibilità di inserimento di sondaggi con quesiti multipli: tale funzionalità è al momento supportata parzialmente, in quanto è richiesta la creazione di un nuovo sondaggio per ogni quesito da inserire.
- Inserimento dei grafici nelle versioni per dispositivo mobile: sono state riscontrate non poche difficoltà nell'inserimento di un sistema di grafici

interattivi per la versione iOS. Viste le difficoltà, si è deciso di disabilitare temporaneamente tale funzionalità, la quale richiede uno studio più approfondito.

- Possibilità di visualizzazione e stampa di un sondaggio da applicazione mobile, utilizzando un servizio di stampa wireless.
- Analisi e valutazione delle diverse implementazioni di caricamento dei dati utilizzate su applicazione mobile, con conseguente eliminazione dell'opzione peggiore. (Sezioni 3.3.1 e 3.3.2)

Bibliografia

[1] http://www.worldharbourproject.org/

[2] https://www.surveymonkey.com/

[3] https://www.surveygizmo.com/

[4] http://www.typeform.com/

[5] https://www.google.com/forms

[6] http://www.surveyswipe.com/

[7] *Untold story of QR Code deveolpment* - http://www.qrcode.com/en/history/

[8] http://flask.pocoo.org/

[9] Flask documentation - *What does micro mean?* http://flask.pocoo.org/docs/0.10/foreword/#what-does-micro-mean

[10] The hitchhiker's guide to Python - *Virtual Environments* http://docs.python-guide.org/en/latest/dev/virtualenvs/

[11] http://www.mobileworld.it/2016/07/13/dati-kantar-maggio-2016-87335/

[12] https://developer.android.com/guide/components/fragments.html

[13] https://developer.apple.com/support/app-store/

[14] https://developer.android.com/about/dashboards/index.html#Platform

[15] Kaliski, B. - RFC2898. https://tools.ietf.org/html/rfc2898#section-5.2

[16] PEP 333 - *Python Web Server Gateway Interface* https://www.python.org/dev/peps/pep-0333/