POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione

Corso di Studi in
INGEGNERIA INFORMATICA

Ekologi
Progettazione e sviluppo di una applicazione per smartphone per la gestione della raccolta differenziata

Relatrice: Prof.ssa Francalanci Chiara

Tesi di Laurea di:
Bonacina Alberto
matr. 755834

Anno Accademico 2013-2014

INDICE

ELENCO DELLE FIGURE

SOMMARIO

In questo lavoro di tesi verrà descritta la mia esperienza nello sviluppo dell'applicazione Ekologi e del suo backoffice di gestione realizzata presso Web3king S.n.c. . Nella tesi verranno descritti tutti i componenti del progetto utilizzando un linguaggio tecnico e facendo uso di diagrammi, schemi e immagini delle componenti realizzate.

PRIMO CAPITOLO presentazione dello stato dell'arte, confronto con altre piattaforme e applicazioni presenti nel mercato;

SECONDO CAPITOLO presentazione del progetto, descrizione della struttura e degli elementi principali, descrizione dei requisiti funzionali e non funzionali;

TERZO CAPITOLO realizzazione del backoffice: identificazione degli scenari e use case, gestione dei dati persistenti nel database, descrizione architettura del sistema;

QUARTO CAPITOLO realizzazione dell'applicazione mobile: identificazione degli scenari e use case, gestione dei dati persistenti nel database, descrizione architettura del sistema;

QUINTO CAPITOLO conclusioni e sviluppi futuri.

1

STATO DELL'ARTE

L'idea alla base di questa tesi nasce qualche anno fa, nell'estate del 2012, quando Web3king S.n.c. e la Cooperativa La Ringhiera videro nella progettazione e realizzazione di una app per la raccolta dei rifiuti un'idea molto interessante che portò qualche mese dopo al rilascio della prima versione dell'applicazione mobile, disponibile su Android e iOS, come aiuto per gli utenti nello smaltimento dei rifiuti. Di quella prima versione è rimasto poco all'interno dell'attuale versione che si è evoluta diventando customizzabile dai comuni a cui viene venduta. In questo ultimo anno sono arrivate altre applicazioni sugli store degli smartphone che ricalcano l'idea alla base di Ekologi. Si vogliono in questo primo capitolo descrivere le caratteristiche principali delle varie app, dei servizi da loro offerti e che valore aggiunto fornisce Ekologi.

Molte delle applicazioni che si possono trovare sugli store di Android e iOS forniscono solamente informazioni sulla categorie di rifiuti e sulla associazione rifiuto-categoria basandosi su informazioni generiche disponibili in rete. Ekologi viene interamente customizzata direttamente dal comune che può scegliere in maniera molto granulare le politiche di smaltimento e di associazione rifiuto-categoria. L'esperienza della Cooperativa La Ringhiera nel campo della raccolta differenziata ci ha portato a realizzare uno strumento che possa adattarsi perfettamente al singolo comune con la possibilità di modificare, in totale liberta, ogni aspetto della raccolta differenziata: dalla nomenclatura delle categorie alle modalità di smaltimento, dalla gestione di rifiuti speciali alla nomenclatura degli stessi.

Alcune applicazioni permettono all'utente di suggerire nuovi prodotti e di indicarne la categoria di smaltimento; Ekologi ha questa funzionalità ma con il valore aggiunto di notificare a tutti i comuni iscritti del nuovo prodotto suggerito e lasciare completa libertà di scelta nell'associazione rifiuto-categoria secondo le politiche del singolo comune, se un comune già tratta quel rifiuto con un diverso nome può eliminare il suggerimento.

Un ristretto numero di applicazioni fornisce la funzionalità di calendario per conoscere il giorno e la categoria di raccolta per un determinato giorno del mese; Ekologi si è spinta oltre con la possibilità per il comune, in maniera del tutto autonoma, di gestire eventuali zone di raccolta e associare le vie corrispondenti per migliorare l'efficacia del calendario specie in comuni di grandi dimesioni. All'utente dell'applicazione viene

data la possibilità di selezionare la via a cui è interessato e il calendario mostrerà solo le informazioni inerenti alla sua zona di raccolta.

Tutte le informazioni presenti in Ekologi sono facilmente modificabili da un pannello web, che nel corso della tesi verrà chiamato backoffice, in totale autonomia da parte del comune, diminuendo i costi e i tempi di gestione delle modifiche. Viene fornito ad ogni comune un login e una password e tutta l'assistenza necessaria nelle prime fasi di inserimento delle informazioni.

Diversamente dalle altre applicazioni, Ekologi riporta anche le principiali informazioni per il comune selezionato e i contatti utili al cittadino: numero di telefono, email, informazioni e contatti di tutta l'amministrazione comunale e numero per le emergenze. Cliccando sul numero di telefono Ekologi fa partire la chiamata per quel numero, cliccando sulla mail Ekologi lancia il programma di posta predefinito presente sullo smartphone.

Per rendere Ekologi uno strumento ancora più moderno e utile sia al comune che al cittadino è stata integrata all'interno dell'applicazione la funzionalità di sottoscrizione alle notifiche in push del comune, utilizzabile da quest'ultimo come strumento diretto e tempestivo di comunicazione con i propri cittadini.

Considerando l'esperienza utente, diverse applicazioni rappresentano la versione mobile di siti internet che hanno lo svantaggio di essere consultabili solo con una connessione ad internet e hanno qualche problema di visualizzazione e velocità, per risolvere questo problema Ekologi è una applicazione nativa sui due sistemi operativi, Android e iOS, che ha bisogno della connessione ad internet solo in fase di sincronizzazione delle informazioni, tutte le funzionalità sono successivamente fruibili anche senza connessione.

2

INTRODUZIONE

2.1 DESCRIZIONE DEL PROGETTO

Il lavoro di tesi riguarda la progettazione e lo sviluppo dell'applicazione mobile Ekologi, disponibile su Google Play Store e Apple Store e del relativo backoffice, raggiungibile dal sito `www.ekologi.it` che ho realizzato presso l'azienda Web3king S.n.c. con cui collaboro da 3 anni. Con questa sistema si vuole fornire ai comuni italiani uno strumento completo per il servizio della raccolta differenziata dei rifiuti e permettere ai cittadini di essere sempre informati attraverso l'applicazione mobile, andando così a sostituire il calendario cartaceo per la raccolta differenziata che viene distribuito in molti comuni.

L'utilizzo del backoffice, che può essere raggiunto da un apposito sito web tramite un login, permette all'impiegato comunale, in maniera facile e veloce, di gestire tutte le informazioni su:

- raccolta differenziata dei rifiuti;
- categorie di raccolta;
- calendario della raccolta;
- piattaforma ecologica;
- contatti e personale comunale;
- messaggi e notizie per i cittadini.

L'applicazione mobile, sviluppata per Android e iOS e scaricabile gratuitamente dagli appositi store, dà la possibilità al cittadino di avere sempre a disposizione informazioni aggiornate sulla raccolta differenziata, sul calendario della raccolta, sulle modalità di smaltimento dei rifiuti, sui contatti per il proprio comune di residenza. I valori aggiunti dell'applicazione mobile e del backoffice rispetto al solo calendario cartaceo sono:

- possibilità per il cittadino di avere sempre a disposizione, anche in mobilità, uno strumento aggiornato e personalizzato per il proprio comune di residenza;
- possibilità per il comune di una gestione precisa e puntale della raccolta differenziata con la possibilità di modifica e aggiornamento costante delle informazioni;

- possibilità per il comune di utilizzare le funzionalità di notifica, presenti su tutti gli smartphone in commercio, per informare i propri cittadini di: iniziative, news, variazioni periodiche del calendario a costi contenuti. Al comune viene data la possibilità di acquistare pacchetti con incluso un quantitativo prefissato di notifiche e news;
- funzionalità integrata all'interno dell'applicazione del lettore di notizie che vengono create e condivise dal comune;
- possibilità per il cittadino di sincronizzare l'applicazione per essere sempre aggiornati sulle modifiche effettuate dal proprio comune;
- risparmio sulla realizzazione dei calendari che vengono distribuiti ai cittadini.

2.2 SISTEMA REALIZZATO

Il sistema realizzato è quello schematizzato in Figura 1 in cui sono riportati i pricipali componenti e il posizionamento degli stessi sui device hardware

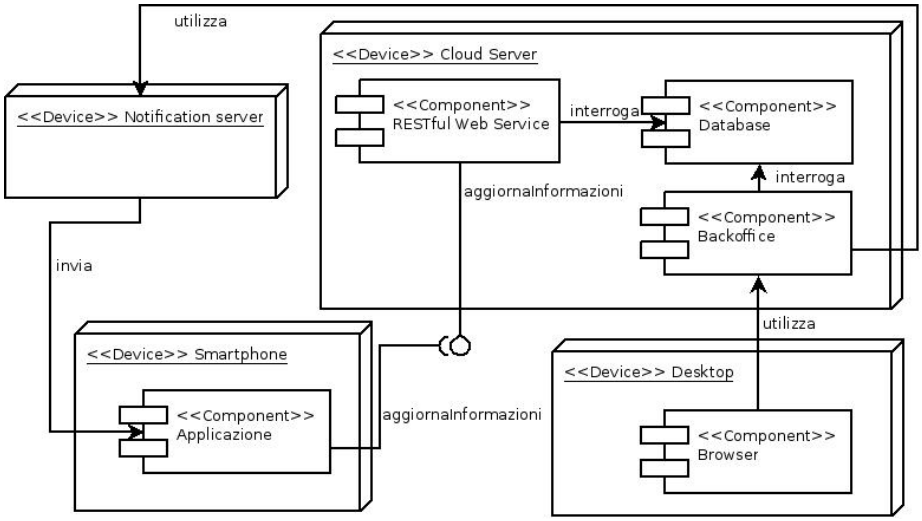


Figura 1: Sistema realizzato

Per quanto riguarda i dispositi hardware il sistema è composto da:

CLOUD SERVER : server in internet su cui è stato installato il web server Apache con il supporto a PHP. Su questo server girano sia il backoffice che il RESTFul Web Service. Oltre al web server è stato installato anche un database MySQL per gestire la persistenza dei dati del backoffice. Dati che sono utilizzati dal RESTFul

Web Service per la comunicazione con l'applicazione e per il suo aggiornamento;

NOTIFICATION SERVER : server che riceve le richieste di invio notifiche effettuate dal backoffice e le recapita ai dispositivi;

DESKTOP : macchina fisica utilizzata dal dipendente comunale che con un browser può collegarsi al backoffice e gestire direttamente tutte le informazioni sulla raccolta differenziata. Il collegamento al backoffice direttamente da un browser web permette più flessibilità all'accesso che è disponibile su tutti i sistemi operativi, non necessita di configurazione o installazione di software;

SMARTPHONE : dispositivo mobile, Android o iOS, in possesso del cittadino con installata l'applicazione. Il dispositivo per sincronizzare le informazioni dell'applicazione utilizza il RESTFul Web Service installato sul Cloud Server in cui le informazioni vengono trasmesse in formato JSON grazie a chiamate via HTTP, nella Figura 1 questa caratteristica è rappresentata dall'interfaccia `aggiornaInformazioni`.

2.3 IDENTIFICAZIONE DEGLI UTENTI

In questa sezione verrà data una prima descrizione degli utenti che interagiscono con il sistema:

AMMINISTRATORE BACKOFFICE : utente che accedendo al backoffice gestisce la creazione di un nuovo comune e la validazione degli acquisti di pacchetti di invio notifiche e news;

DIPENDENTE COMUNALE : utente che accedendo al backoffice gestisce tutte le informazioni su: raccolta differenziata dei rifiuti, categorie di raccolta, calendario della raccolta, informazioni sul comune e sull'amministrazione comunale (sindaco, segretario, giunta, consiglio comunale), creazione e invio di news e notifiche;

UTENTE APPLICAZIONE : utente che utilizza l'applicazione mobile installata sul proprio dispositivo e che accede a tutte le informazioni messe a disposizione dal comune.

2.4 REQUISITI FUNZIONALI

Per la descrizione dei requisiti sono stati presi in considerazione i componenti del sistema riportato in Figura 1 ed in partcolar modo:

Backoffice, RESTFul Web Service e Applicazione Mobile. I requisiti risultano essere:

- Backoffice
  - creazione di un comune;
  - modifica password per l'accesso al backoffice da parte di un dipendente comunale nel caso di smarrimento;
  - abilitazione pacchetto di notifiche e messaggi acquistato dal dipendente comunale.
  - creazione, eliminazione e modifica di un rifiuto e relativa associazione con la categoria di raccolta;
  - modifica delle informazioni di una categoria di raccolta ed eliminazione della categoria;
  - creazione, modifica ed eliminazione delle zone di raccolta;
  - creazione, modifica ed eliminazione delle vie del comune e loro associazione con eventuali zone di raccolta;
  - gestione delle date di raccolta dei rifiuti per ogni categoria e zona di raccolta;
  - gestione di tutte le informazioni riguardanti il comune, il sindaco, il segretario comunale, la giunta comunale e i gruppi di maggioranza e minoranza;
  - creazione ed invio dei messaggi e delle notifiche;
  - gestione degli acquisti di pacchetti di notifiche e messaggi.
- RESTFul Web Service
  - invio di tutte le informazioni su un comune per la sincronizzazione con l'applicazione;
  - invio dei comuni disponibili nell'applicazione;
  - iscrizione e disiscrizione al servizio di notifiche push.
- Applicazione Mobile
  - visualizzazione informazioni sul comune, il sindaco, il segretario comunale, la giunta comunale e i gruppi di maggioranza e minoranza;
  - ricerca di un prodotto e visualizzazione della categoria di smaltimento associata;
  - visualizzazione delle categorie di raccolta e modalità di smaltimento;

- visualizzazione calendario di raccolta associato alla propria zona;
- ricerca della via di residenza e associazione con la propria zona di raccolta;
- iscrizione al sistema di notifiche push;
- ricezione delle notifiche, se iscritto al sistema, e messaggi da parte del comune;
- modifica del comune di residenza;
- sincronizzazione dell'applicazione per il download di dati aggiornati.

2.5 REQUISITI NON FUNZIONALI

In questa sezione verranno riportati i requisiti che sono considerati come non funzionali in quanto non riguardano le funzionalità del sistema ma solo l'usabilità e le prestazioni dello stesso.

2.5.1 *Interfaccia utente*

Nella realizzazione dell'interfaccia utente si deve fare una distinzione tra interfaccia del backoffice e quella dell'applicazione mobile in quanto sono state utilizzate due tecnologie diverse per la loro realizzazione e l'utilizzo delle due viene fatto su due dispositivi molto diversi, rispettivamente desktop/notebook e smartphone.

Interfaccia utente backoffice

Come detto in precedenza tutto il backoffice è accessibile da un browser web per ridurre al minimo le configurazioni necessarie per potervi accedere, che si traducono al solo collegamento ad internet. Tutte le pagine del backoffice sono pagine web con standard HTML5 e CSS3, realizzate con il framework Bootstrap [9]. Per la strutturazione delle pagine si sono presi come riferimento i trend grafici per le pagine di amministrazione che si possono trovare sul web. Una schermata d'esempio è riportata in Figura 2 in cui si può vedere la suddivisione in una colonna laterale contentente il menu di navigazione e una centrale/destra con riportati i contenuti veri e propri.

In tutto il backoffice è stato utilizzato lo stesso schema di colori, si è mantenuta l'interfaccia il più pulita possibile, tutti i bottoni riportano un'icona per rendere il significato del bottone chiaro e immediato, tutti i campi di inserimento delle informazioni sono stati realizzati con

dei form e per i campi in cui il testo poteva avere della formattazione aggiuntiva (elenchi puntati, tabelle, etc.) è stato predisposto in corrispondenza del campo un editor WYSIWYG (What You See Is What You Get) che si integra nello stile grafico della pagina.

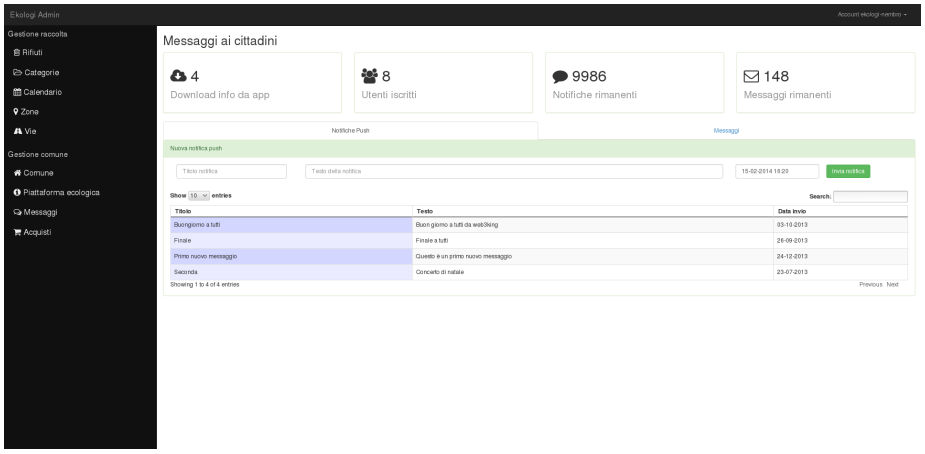


Figura 2: Schermata d'esempio del backoffice

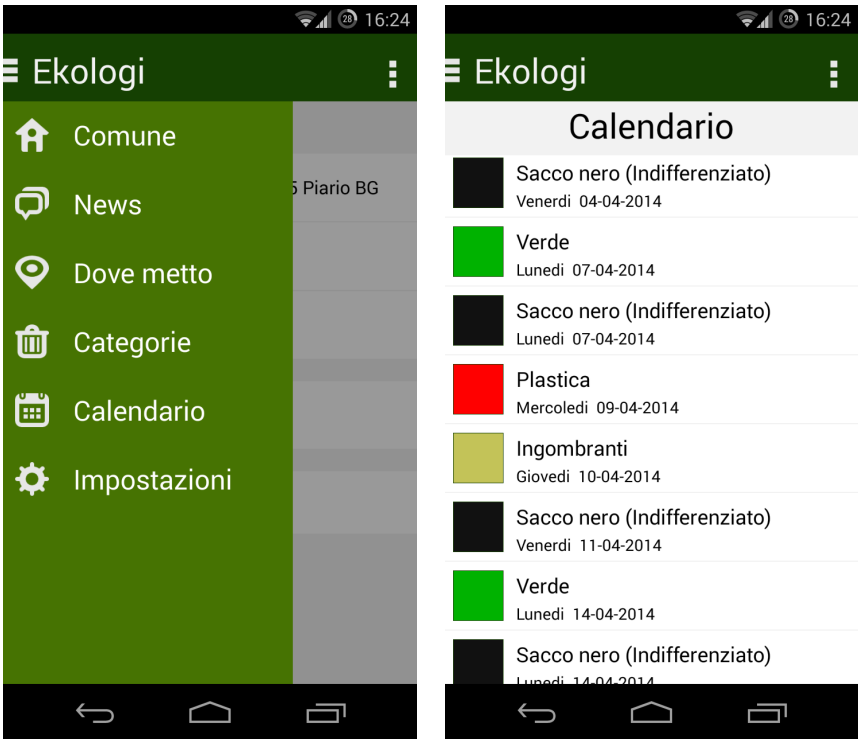
L'interfaccia utente del backoffice è responsive per adattarsi automaticamente al tipo di dispositivo con cui ci si collega, in ogni caso l'utilizzo del backoffice è consigliato su schermi grandi come quelli di notebook e desktop.

Interfaccia utente applicazione

L'applicazione è stata realizzata sia per Android che per iOS utilizzando la piattaforma di sviluppo Titanium [4] di Appcelerator Inc.[2] che permette la creazione di applicazioni native per i due sistemi operativi mobili partendo da un'unica base di codice JavaScript.

Utilizzando l'SDK messo a disposizione si ha accesso ad API per l'interfacciamento con l'hardware dei dispositivi e Servizi Cloud per l'utilizzo di sistemi di notifica. Per i due sistemi operativi le linee guida per la realizzazione di applicazioni sono abbastanza diverse [8][5], si è cercato di rendere la struttura dell'applicazione il più possibile simile creando una grafica che includesse elementi presenti in entrambi senza snaturare troppo le linee guida.

Il risultato è quello riportato in Figura 13 molto simile a quello visto per il backoffice con un menu laterale a sinistra in Figura 3a, che in questo caso è a scomparsa, e l'intero contenuto della pagina nella finestra principale in Figura 3b.



(a) Menu laterale

(b) Finestra principale applicazione mobile

Figura 3: Interfaccia utente dell'applicazione su Android

Considerando la grande diffusione di tablet Android e iPad anche l'applicazione mobile adatta la sua struttura se viene aperta su dispositivi con schermo grande. In questo caso la barra laterale per il menu è sempre presente con una interfaccia molto simile a quella vista per il backoffice come si può vedere in Figura 4.

2.5.2 *Cloud Server*

Il backoffice e il RESTFul Web Service sono installati su un server cloud GNU/Linux con distribuzione Debian 6.0.6. Entrambi i servizi sono scritti in PHP e utilizzano un database MySQL, le rispettive versioni sul server sono `5.3.3-7+squeeze17` e `5.1.63-0+squeeze1`.

2.5.3 *Notification Server*

Il server che si occupa delle funzionalità di notifica è gestito direttamente da Appcelerator Inc. [2] e vengono utilizzate le funzionalità

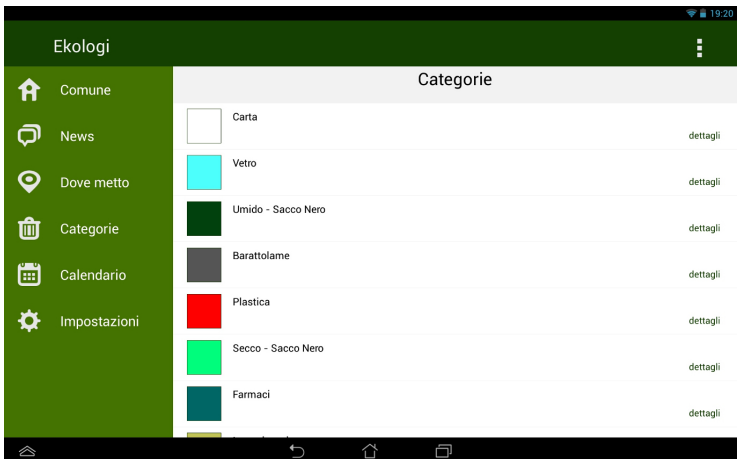


Figura 4: Interfaccia utente dell'applicazione su Tablet

esposte come iscrizione e disiscrizione alle notifiche utilizzando le API fornite dai Cloud Services [3], questo ha consentito durante le fasi di sviluppo di realizzare solo la comunicazione tra applicazione ed API senza preoccuparsi dei dettagli implementativi che sono proprietari di Appcelerator Inc.

3 BACKOFFICE

In questo capitolo verrà presentato lo sviluppo del backoffice, presentando prima l'identificazione degli scenari di utilizzo del backoffice (3.1), la definizione degli use case (3.2), la gestione dei dati persistenti per il salvataggio delle informazioni dei comuni nel database MySQL (3.3) e infine verrà descritta l'architettura del sistema (3.4). L'ultima sezione è stata dedicata alla descrizione del RESTFul Web Service (3.5) che rappresenta una parte fondamentale del sistema in quanto consente all'applicazione mobile di scaricare e sincronizzare tutte le informazioni messe a disposizione dal comune.

3.1 IDENTIFICAZIONE SCENARI

3.1.1 *La città di Milano si unisce al database di Ekologi*

A seguito della delibera del consiglio comunale il comune di Milano si unisce ai comuni che hanno sottoscritto un contratto con Web3king S.n.c. La signora Carla, dipendente del comune di Milano e incaricata della gestione delle informazioni del backoffice, telefona ad Alberto, che è l'amministratore del backoffice, chiedendo la creazione del comune e dell'utenza per l'accesso al backoffice. Alberto fa il login nella pagina di amministrazione, compila tutti i campi per la creazione del nuovo comune, crea il canale per le notifiche push sul Notification Server e comunica alla signora Carla l'utenza e la password per l'accesso al backoffice.

3.1.2 *Inserimento informazioni sulla raccolta differenziata*

La signora Carla utilizzando il nome utente e la password forniti da Alberto accede al backoffice e inserisce tutte le informazioni su: raccolta differenziata, categorie di raccolta, vie e zone di raccolta, calendario suddiviso per zona e categoria, contatti per l'amministrazione comunale.

3.1.3 *Nuova news per i mercatini dell'antiquariato*

La signora Carla vuole mandare a tutti i cittadini la comunicazione sui mercatini dell'antiquariato che si terranno Domenica 17 Agosto in Piazza

Leonardo; si collega al backoffice e crea una nuova news riportando gli orari della manifestazione, i circoli di appassionati presenti e gli orari degli spettacoli per l'intrattenimento dei bambini. Formatta la news con l'editor WYSIWYG e pubblica la notizia rendendola visibile su tutti i dispositivi che hanno scaricato l'applicazione e scelto Milano come comune.

3.1.4 *Percorso alternativo causa incidente in Via Ponzio*

La signora Carla deve avvertire tempestivamente tutti i cittadini della chiusura temporanea del traffico in via Ponzio causa incidente stradale; si collega al backoffice e crea una notifica push per avvisare della modifica alla circolazione stradale. Invia la notifica e nel giro di pochi minuti i cittadini iscritti al servizio di notifica verranno avvisati sul proprio dispositivo.

3.1.5 *Modifica della gestione del rifiuto CD/DVD*

A seguito del cambio di politica di gestione del rifiuto CD/DVD la signora Carla si collega al backoffice, cerca all'interno dei rifiuti creati il CD/DVD e con pochi click modifica la categoria del comune da Secco a RAEE. Nelle informazioni della categoria Secco elimina la nota per il trattamento dei CD/DVD e sposta la nota nelle informazioni per la categoria RAEE. Salva tutte le modifiche e al successivo aggiornamento i cittadini si troveranno le informazioni aggiornate sulla gestione del rifiuto CD/DVD.

3.2 USE CASE

3.2.1 *Creazione di un nuovo comune*

ATTORI Amministratore backoffice

INPUT L'amministratore deve aggiungere un nuovo comune

EVENTI
- L'amministratore accede al pannello di amministrazione inserendo username e password nella form di login;
- Inserisce tutte le informazioni richieste per la creazione del nuovo comune;
- Conferma la creazione del comune;
- Esegue il logout dal pannello di amministrazione.

OUTPUT L'amministratore ha creato un nuovo comune nel database di ekologi

3.2.2 *Creazione di un rifiuto*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve creare un nuovo rifiuto

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione dei rifiuti;
- Inserisce il nome del nuovo rifiuto;
- Sceglie la categoria a cui deve essere associato;
- Conferma la creazione del rifiuto;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha creato un nuovo rifiuto e lo ha associato ad una categoria di raccolta gestita dal comune

3.2.3 *Modifica descrizione di una categoria di raccolta*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve modificare la descrizione di una categoria di raccolta

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione delle categorie di raccolta;
- Sceglie la categoria da modificare;
- Modifica il testo della descrizione;
- Conferma la modifica effettuata;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha modificato la descrizione di una categoria di raccolta

3.2.4 *Compilazione calendario di raccolta*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve compilare il calendario di raccolta per la categoria `Secco` nella `Zona 1` per l'anno 2014

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione del calendario;
- Seleziona la categoria `Secco`;
- Seleziona la `Zona 1`;
- Seleziona l'anno 2014;
- Seleziona tutti i giorni di raccolta;
- Conferma le modifiche apportate al calendario;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha creato il calendario di raccolta per la categoria `Secco` nella `Zona 1` per l'anno 2014

3.2.5 *Acquisto di un pacchetto di notifiche e news*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve acquistare un pacchetto per l'invio di notifiche push e news ai cittadini

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina degli acquisti di pacchetti per notifiche e news;
- Seleziona il pacchetto desiderato;
- Conferma l'operazione;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha acquistato un pacchetto di notifiche e news

3.2.6 *Conferma acquisto di un pacchetto di notifiche e news*

ATTORI Amministratore backoffice

INPUT L'amministratore deve abilitare un pacchetto di notifiche e news precedentemente acquistato dal dipendente comunale

EVENTI
- L'amministratore accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione degli acquisti;
- Seleziona l'acquisto effettuato dal dipendente comunale ;
- Conferma l'acquisto;
- Esegue il logout dal pannello di amministrazione.

OUTPUT L'amministratore ha confermato l'acquisto di un pacchetto di notifiche e news

3.2.7 *Creazione di una notifica push*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve creare una notifica push

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione delle notifiche push;
- Inserisce il titolo e il testo della notifica;
- Seleziona la data e l'ora di invio;
- Invia la notifica al `Notification Server`;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha creato una notifica push e l'ha inviata al `Notification Server`

3.2.8 *Creazione di una news*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve creare una news

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;

- Si sposta nella pagina di gestione dei messaggi;
- Inserisce il titolo e il testo della news;
- Invia la news a tutti i cittadini;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha creato una news e l'ha inviata a tutti i cittadini

3.2.9 *Aggiunta di un assessore*

ATTORI Dipendente comunale

INPUT Il dipendente comunale deve aggiungere un nuovo assessore alla giunta

EVENTI
- Il dipendente comunale accede al pannello di amministrazione inserendo username e password nella form di login;
- Si sposta nella pagina di gestione degli assessori;
- Nella form di creazione di un nuovo assessore inserisce tutti i campi richiesti;
- Conferma la creazione dell'assessore;
- Esegue il logout dal pannello di amministrazione.

OUTPUT Il dipendente comunale ha creato un nuovo assessore

3.3 GESTIONE DATI PERSISTENTI

3.3.1 *Entità*

In riferimento alla Figura 5 si possono elencare le entità:

- Giorno Raccolta: rappresenta il singolo giorno di raccolta di una categoria di rifiuto in una determinata zona;
- Comune: rappresenta il singolo comune che si è iscritto ai servizi forniti da Ekologi;
- Persona: rappresenta il singolo dipendente comunale di cui si vogliono dare informazioni all'interno dell'applicazione mobile: sindaco, segretario comunale, assessore, consigliere comunale del gruppo di maggioranza e minoranza;

- Piattaforma ecologica: contiene tutte le informazioni sulla piattaforma ecologica del comune;
- Messaggio: rappresenta il messaggio che viene creato dal dipendente comunale e viene visualizzato all'interno dell'applicazione nella sezione news;
- Notifica: rappresenta la notifica push che viene creata dal dipendente comunale, diversamente dal messaggio la notifica presenta anche un campo ora in quanto il suo invio può essere customizzato e schedulato nel tempo;
- Pacchetto: rappresenta il pacchetto che può essere acquistato dal dipendente comunale e permette di invare un certo numero di messaggi e notifiche push;
- Rifiuto: rappresenta il singolo rifiuto che è presente nella base dati dell'applicazione;
- Categoria: rappresenta la categoria di raccolta associata ad un insieme di rifiuti, grazie ai campi descrizione e note si possono prevedere informazioni generali, modalità di smaltimento, eccezioni, pericoli nel trattamento, etc;
- Zona: rappresenta la zona di raccolta per un particolare insieme di vie. In molti comuni non è presente la suddivisione in zone di raccolta ma Ekologi permette la creazione di calendari di raccolta personalizzati per zona;
- Via: rappresenta la via del comune che può appartenere ad una particolare zona;
- Utente: rappresenta l'utente avente login e password che può collegarsi al backoffice e gestire tutte le informazioni.

3.3.2 *Associazioni*

In riferimento alla Figura 5 si possono elencare le associazioni che vengono presentate in forma attiva:

APPARTIENE TRA PIATTAFORMA ECOLOGICA E COMUNE una piattaforma ecologica appartiene ad un solo comune, la sua relazione inversa è *possiede* in cui un paese può avere più di una piattaforma ecologica;

LAVORA TRA PERSONA E COMUNE un singolo dipendente comunale (sindaco, segretario, assessori e consiglieri) lavora in un solo comune, la relazione inversa è *possiede* in cui un comune ha più di un dipendente comunale;

APPARTIENE TRA UTENTE E COMUNE un utente che accede al backoffice può gestire le informazioni solo di un comune;

INVIA TRA UTENTE E MESSAGGIO un utente invia zero o più messaggi a tutti gli utenti dell'applicazione;

INVIA TRA UTENTE E INVIA un utente invia zero o più notifiche a tutti gli utenti dell'applicazione che si sono registrati al servizio di notifiche;

ACQUISTA TRA UTENTE E PACCHETTO un utente acquista zero o più pacchetti per l'invio di messaggi e notifiche ai cittadini;

CONTIENE TRA ZONA E VIA una zona può contenere una o più vie, la relazione inversa *è contenuta* rappresenta il fatto che una via può appartenere solo ad una zona;

ASSOCIATO TRA GIORNO RACCOLTA, ZONA E CATEGORIA ogni giorno di raccolta deve essere associato ad una e una sola zona e categoria di raccolta;

CONTIENE TRA CATEGORIA E RIFIUTO una categoria può contenere più rifiuti al suo interno, la relazione inversa *è associato* rappresenta il fatto che un rifiuto può essere associato ad una e una sola categoria di raccolta.

3.3.3 *Modello Relazionale*

Con riferimento allo schema in Figura 5 sono state costruite le seguenti tabelle della base di dati:

- calendario(<u>id</u>, giorno, tipo, zona, nome_giorno, paesi_id)
- categorie(<u>id</u>, nomecat, color, descrizione, note, paesi_id)
- items(<u>id</u>, nomecat, color, categorie_id, paesi_id)
- messaggi(<u>id</u>, titolo, testo, data, paesi_id, inviato)
- notifiche_push(<u>id</u>, titolo, testo, data, ora, paesi_id, inviato, nr_destinatari)

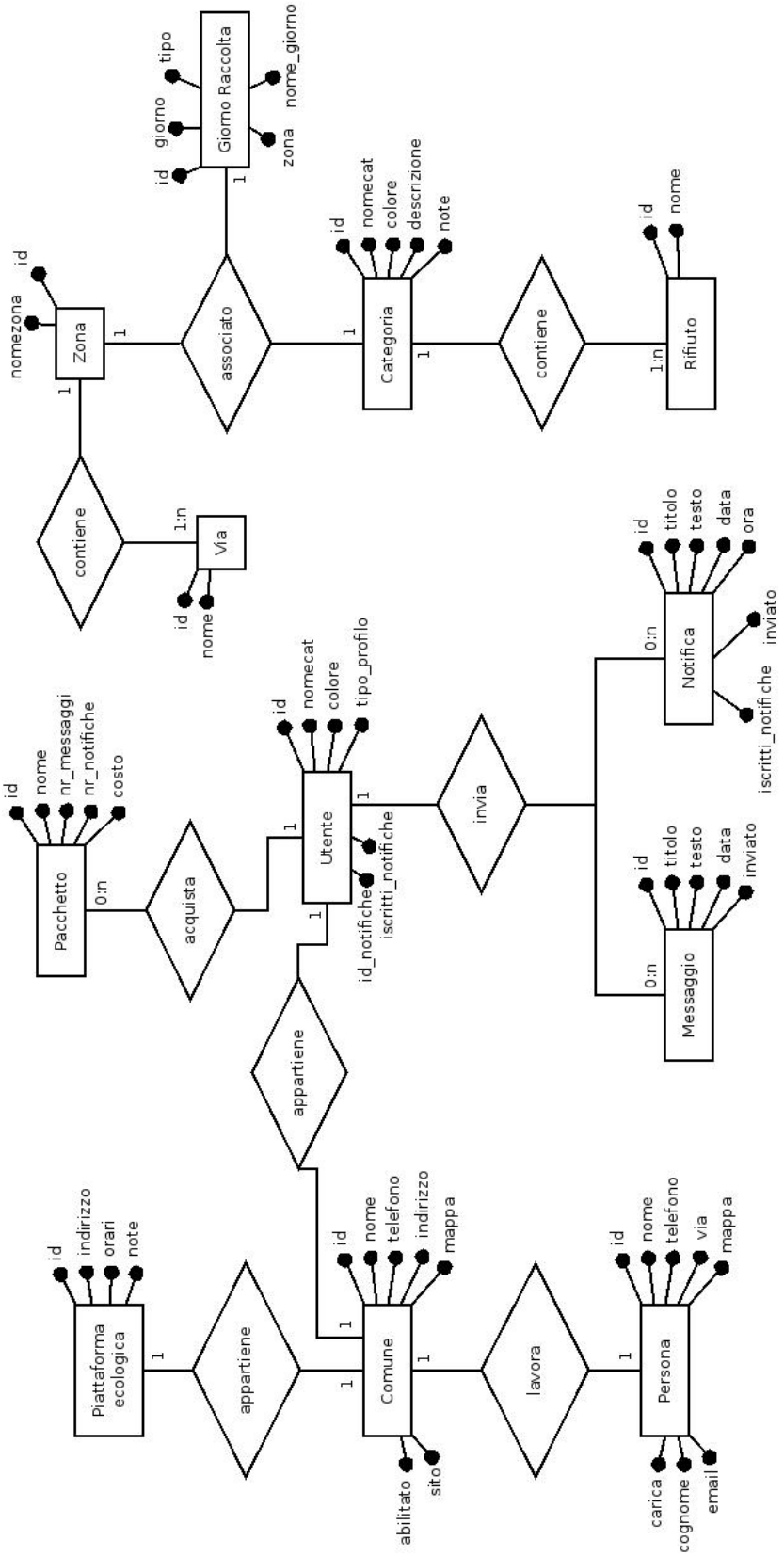


Figura 5: Schema entità associazioni

- pacchetti(id, nome, nr_messaggi, nr_push, costo)
- pacchetti_acquistati(id, pacchetti_id, paesi_id, data, abilitato)
- paesi(id, nome, nomedb, telefono, telefono, indirizzo, mappa, sito, abilitato)
- persone(id, nome, cognome, email, telefono, via, mappa, carica, tipopersone_id, paesi_id)
- piattaformaecologica(id, indirizzo, orari, note, paesi_id)
- tipopersone(id, nome)
- utenti(id, username, password, tipo, paesi_id, tipoprofilo_id, titanium_id, iscritti_notifiche)
- utenti_tipo(id, nome)
- vie(id, nome, zone_id, paesi_id)
- zone(id, nomezona, zone_id, paesi_id)

A cui si aggiungono le seguenti precisazioni:

- tutte le righe in tutte le tabelle hanno un campo id che all'interno della singola tabella è chiave;
- il campo `paesi_id` presente in quasi tutte le tabelle gestisce l'associazione della singola riga nella tabella con il proprio comune in modo che con un unico database si possano gestire tutti i comuni ed estrarre le informazioni richieste filtrando il campo `paesi_id`. L'`id` del singolo comune è quello che corrisponde al campo `id` nella tabella `paese`;
- le associazioni tra tabelle possono essere individuate attraverso il nome dei campi nelle tabelle stesse in cui il pattern `<text>_id` rappresenta una relazione con il campo `id` della tabella `text`;
- nella tabella `paesi` il campo `abilitato` viene utilizzato in fase di startup iniziale del comune per lasciar fare le modifiche al dipendente comunale ma non rendere visibile il comune all'interno dell'applicazione, quando sono state aggiunte tutte le informazioni il comune viene definitivamente abilitato e diventa visibile nell'applicazione mobile;
- il campo `titanium_id` viene utilizzato dal backoffice per gestire il corretto invio delle notifiche solo agli utenti di un determinato comune;

- il campo `iscritti_notifiche` riporta il dato, aggiornato ad ogni invio di notifica, dei cittadini che hanno sottoscritto il servizio di ricezione di notifiche da parte del comune.

3.4 ARCHITETTURA DEL SISTEMA

Il backoffice è stato realizzato con CodeIgniter [6] un framework PHP realizzato da Ellislab Inc. [7]. Il pattern architetturale di CodeIgniter è Model-View-Controller in cui si separa la logica dell'applicazione dalla sua rappresentazione:

MODEL rappresentano le classi con cui viene fatta l'interazione con la base di dati, i model non eseguono nessuna operazione sui dati ma si preoccupano solo di ricevere le richieste dai controller, eseguire le relative query e rispondere con i dati presenti nelle tabelle del database;

VIEW ricevono i dati dai controller e si occupano solo della loro visualizzazione nella pagina web. Le view sono state utilizzate anche per strutturare la pagina web in modo che una porzione di pagina ricorrente, come per esempio il menu di navigazione, fosse scritta solo una volta e poi inclusa in tutte le pagine;

CONTROLLER rappresentano le classi che eseguono le operazioni sui dati estratti dal database attraverso le chiamate ai model e li preparano per la loro visualizzazione all'interno delle view.

Lo sviluppo del backoffice è stato effettuato seguendo la struttura data dal pattern architetturale di CodeIgniter, questo ha consentito di avere una buona qualità del codice, buona manutenibilità e adattabilità. Soprattutto quest'ultima caratteristica ha reso il backoffice di Ekologi la base per la realizzazione di altri backoffice realizzati da Web3king S.n.c. Nella Figura 6 si è rappresentata l'architettura generale del sistema mostrando le classi. Le comunicazioni avvengono solo tra View-Controller e tra Controller-Model, i dati passati sono sempre degli array associativi chiave-valore.

3.4.1 *Model*

Le classi del model hanno il compito di gestire l'interazione con le tabelle delle basi di dati, i model ricalcano l'impostazione data nella descrizione delle entità e delle tabelle nel database in cui ogni model gestisce un'entità separata per rendere il codice più manutenibile.
Nella Figura 7 sono mostrati i model che sono stati implementati

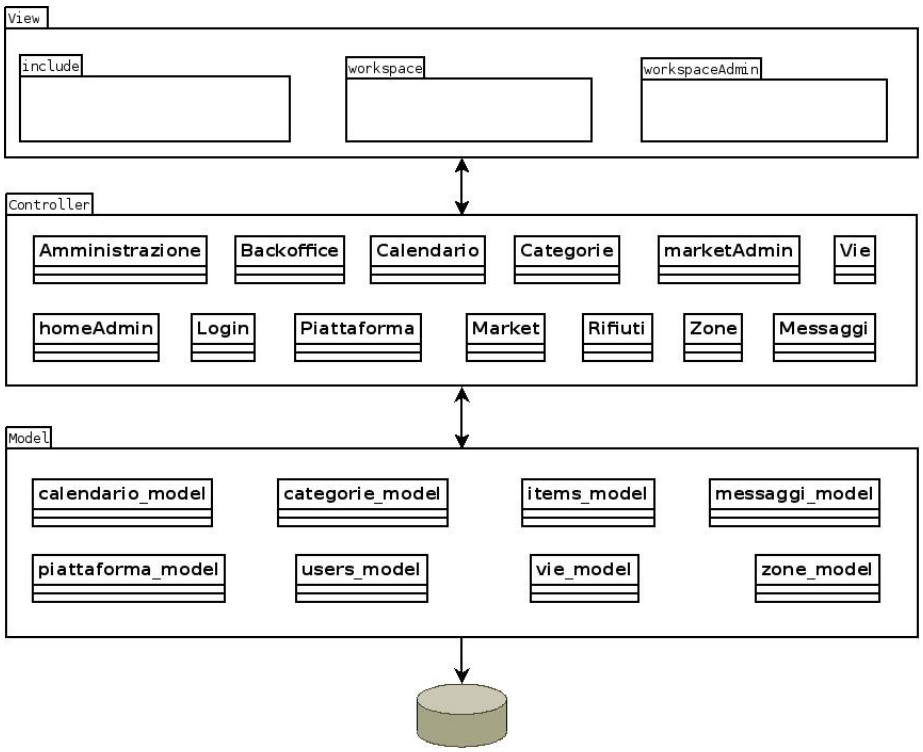


Figura 6: Architettura del sistema

CALENDARIO_MODEL fornisce le funzioni per operare sulla tabella `calendario`. E' possibile estrarre tutti i giorni di raccolta di un determinato rifiuto per anno e zona, aggiungere, modificare ed eliminare i giorni di raccolta;

CATEGORIE_MODEL fornisce le funzioni per operare sulla tabella delle `categorie`. E' possibile modificare o eliminare una categoria, sono presenti funzioni per estrarre il nome di una categoria dall'id per la compilazione corretta delle pagine del backoffice, la numerosità dei rifiuti in una categoria per motivi statistici e l'inizializzazione di categorie di default durante la creazione di un nuovo comune;

ITEMS_MODEL fornisce le funzioni per operare sulla tabella `items`. E' possibile gestire i rifiuti: modificarli, aggiungerli ed eliminarli. Con la funzione `getDifferenzeRifiuti` viene fornita anche una funzionalità per vedere quali rifiuti sono proposti dal backoffice ma non sono ancora stati aggiunti all'interno di quelli gestiti dal comune;

MESSAGGI_MODEL fornisce le funzioni per operare sulle tabelle che riguardano la gestione di messaggi e notifiche: `notifiche_push`,

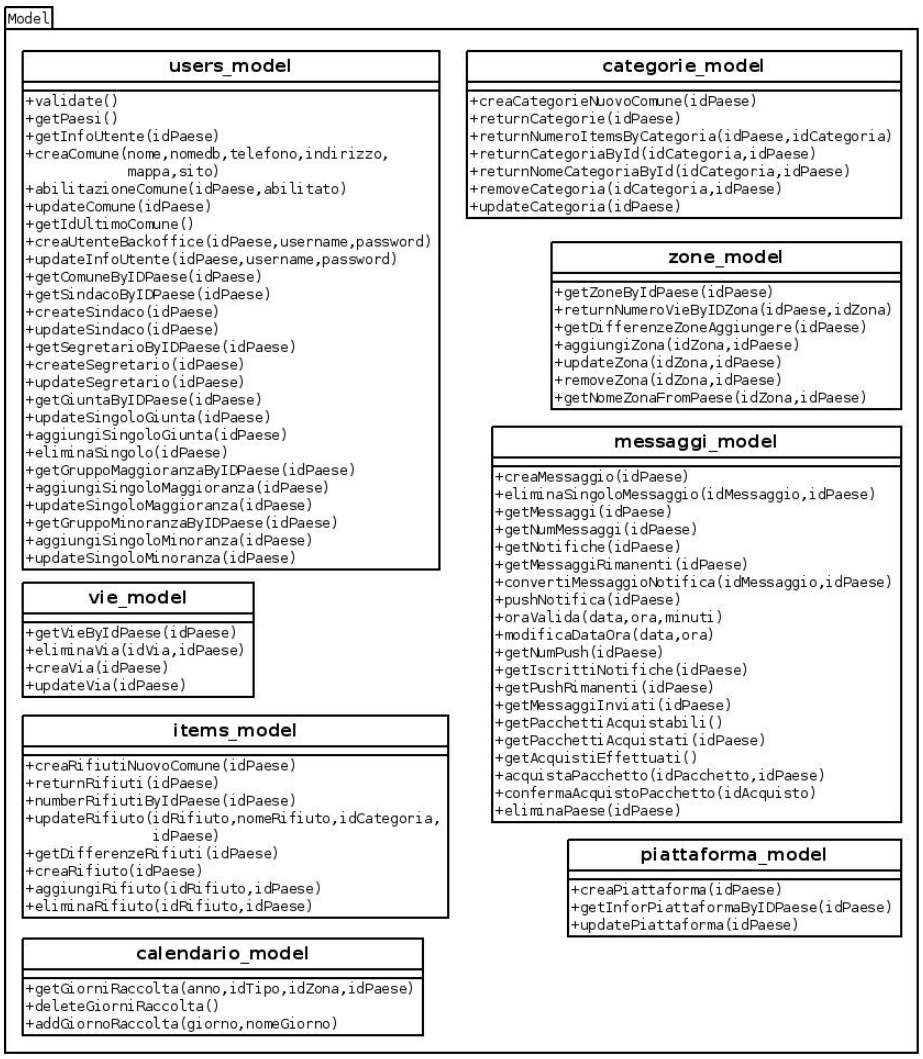


Figura 7: Schema dei model

`pacchetti_acquistati` e `messaggi`. E' possibile creare e modificare sia i messaggi che le notifiche ma è possibile eliminare solo i messaggi, trasformare un messaggio in notifica, gestisce l'acquisto e l'abilitazione di pacchetti di messaggi e notifiche. Fornisce delle funzioni a fini statistici come il numero di iscritti alle notifiche e funzioni per tenere sotto controllo il numero di messaggi e notifiche rimanenti in base ai pacchetti acquistati;

PIATTAFORMA_MODEL fornisce le funzioni per operare sulla tabella `piattaformaecologica`. Creare, modificare ed estrarre informazioni sulla piattaforma come indirizzo, orari e note;

USERS_MODEL fornisce le funzioni per operare sulla tabella `utenti`. In

questo model sono contenute le funzioni per creare, modificare ed eliminare tutti i dipendenti comunali e le informazioni sul comune;

VIE_MODEL fornisce le funzioni per operare sulla tabella `vie`. E' possibile creare, modificare, eliminare una via e modificare la sua zona di appartenenza;

ZONE_MODEL fornisce le funzioni per operare sulla tabella `zone`. E' possibile creare una zona, modificarne il nome e estrarre tutte le vie appartenenti a quella zona.

3.4.2 *View*

Nella Figura 8 sono mostrate le view che sono state implementate

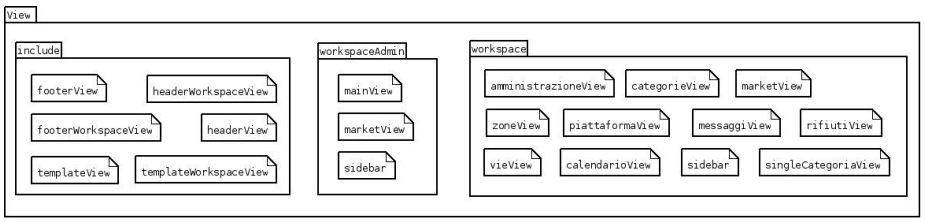


Figura 8: Schema delle view

include

Si vogliono presentare qui alcune viste che non sono direttamente legate alle funzionalità del backoffice ma sono state utilizzate per una migliore gestione della pagina, per l'inclusione degli script e dei fogli di stile.

TEMPLATEVIEW vista per l'impostazione della struttura della pagina nella homepage;

TEMPLATEWORKSPACEVIEW vista per l'impostazione della struttura della pagina nel backoffice;

HEADERVIEW vista per l'impostazione dell'header nella homepage, qui si può trovare l'inclusione dei foglio di stile e la form per il login;

HEADERWORKSPACEVIEW vista per l'impostazione dell'header nel backoffice, qui viene impostata la struttura principale della pagina in due colonne: sinistra per il menu di navigazione e centrale/destra per i contenuti principali;

FOOTERVIEW vista per l'impostazione del footer nella homepage;

FOOTERWORKSPACEVIEW vista per l'impostazione del footer nel backoffice, vengono richiamate le funzioni per gestire le parti interattive del sito con l'utilizzo di jQuery come i menu a tendina, i form per inserimento della data in maniera semplice, i modal per la modifica delle informazioni e gli script per la gestione delle tabelle;

workspaceAdmin

SIDEBAR vista per il menu di navigazione nella colonna di sinistra;

MAINVIEW vista principale dell'amministratore in cui si possono visualizzare i comuni gestiti ed è presente il form per la creazione di un nuovo comune;

MARKETVIEW vista per la gestione degli acquisti fatti dai comuni con la possibilità di abilitare i pacchetti acquistati.

workspace

SIDEBAR vista per il menu di navigazione nella colonna di sinistra;

AMMINISTRAZIONEVIEW vista per la gestione di tutte le informazioni sul comune e sui dipendenti comunali. Tutte le informazioni possono essere visualizzate e modificate attraverso dei form semplici e immediati, per migliorare la visualizzazione della pagina la stessa è stata organizzata con una navigazione a schede in cui ogni scheda rappresenta una specifica categoria: comune, sindaco, segretario, giunta, gruppo maggioranza e minoranza;

CALENDARIOVIEW vista per la modifica dei calendari di raccolta. Grazie ad un form può essere scelto il singolo calendario che si vuole modificare scegliendo la zona, l'anno e la categoria di raccolta;

CATEGORIEVIEW vista che consente di visualizzare le categorie gestite, eliminarle e scegliere la categoria che si vuole modificare;

MARKETVIEW vista che consente la visualizzazione dei pacchetti che possono essere acquistati e quelli già acquistati;

MESSAGGIVIEW vista, riportata in Figura 2, che consente la creazione di messaggi e notifiche push. Come per la vista dell'amministrazione, per migliorarne la visualizzazione è stata organizzata a schede, una per i messaggi e una per le notifiche. All'interno della vista sono riportati anche il numero di messaggi e notifiche rimanenti e il numero di utenti dell'applicazione mobile che sono iscritti al sistema di notifiche;

PIATTAFORMAVIEW vista che permette di modificare le informazioni sulla piattaforma ecologica, per i campi `orari` e `note` è disponibile un editor WYSIWYG per facilitare la formattazione e la creazione di elenchi puntati e tabelle;

RIFIUTIVIEW vista che permette la creazione, modifica ed eliminazione dei singoli rifiuti e la loro associazione con la categoria di raccolta;

SINGLECATEGORIEVIEW vista che permette la modifica di una sola categoria. E' presente un form con cui si possono modificare le informazioni e per i campi `descrizione` e `note` è disponibile un editor WYSIWYG per facilitare la formattazione e la creazione di elenchi puntati e tabelle;

VIEVIEW vista che permette la creazione, modifica ed eliminazione delle vie e della loro associazione con le zone di raccolta;

ZONEVIEW vista che permette la gestione delle zone di raccolta.

3.4.3 *Controller*

Nella Figura 9 sono mostrati i controller che sono stati implementati; tutti i controller, tranne `Login` che non è associato a nessuna vista, hanno la funzione `index()` che gestisce la vista di default associata al controller.

AMMINISTRAZIONE controller che gestisce tutte le modifiche delle informazioni sul personale del comune fatte dal dipendente comunale. Le funzioni che vengono messe a disposizione permettono la creazione, modifica ed eliminazione del personale del comune: sindaco, segretario, giunta, gruppo di maggioranza e minoranza utilizzando le funzioni di `users_model`. Le funzioni del controller predispongono queste informazioni per la visualizzazione nella vista `amministrazioneView`;

BACKOFFICE controller che gestisce la creazione della pagina iniziale di presentazione dell'applicazione;

CALENDARIO controller che possiede tutte le funzioni per la modifica del calendario di raccolta effettuate dal dipendente comunale

- `updateCalendario()` funzione che prende tutti i giorni che sono stati selezionati per una determinata zona e categoria di raccolta, formatta il risultato e chiama la funzione `addGiornoRaccolta` del model `calendario_model` per aggiungerli nel database;

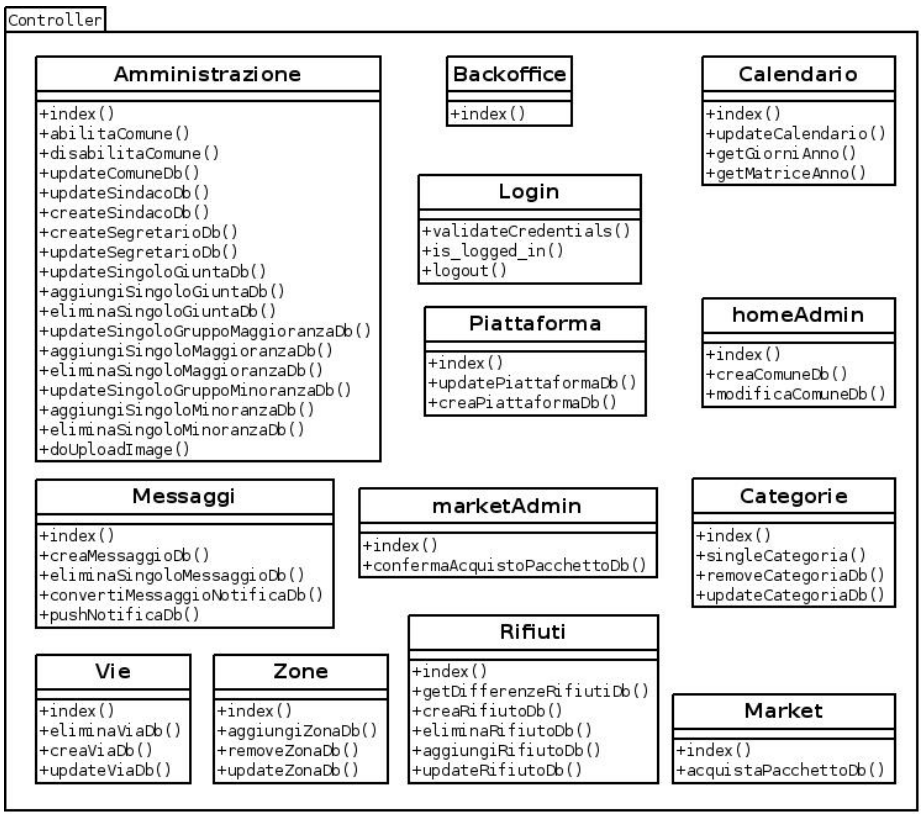


Figura 9: Schema dei controller

- `getGiorniAnno()` funzione che con l'utilizzo della funzione `getGiorniRaccolta` del model `calendario_model` predispone i giorni di raccolta per la visualizzazione nella pagina;
- `getMatriceAnno()` funzione per la creazione della matrice dei giorni di un determinato anno.

CATEGORIE controller che possiede tutte le funzioni per la modifica delle categorie di raccolta effettuate dal dipendente comunale

- `singleCategoria()` funzione che riceve dal model `categorie_model` le informazioni sulla singola categoria e passa le informazioni alla vista `singleCategoriaView` per la visualizzazione;
- `removeCategoria()` funzione che utilizzando il model `categorie_model` chiama la sua funzione `removeCategoria` per la rimozione di una categoria dal database del comune;
- `updateCategoria()` funzione che riceve le modifiche fatte alla categoria di raccolta e le passa alla funzione

`updateCategoria` del model `categorie_model`.

HOMEADMIN controller utilizzato dall'amministratore del backoffice per la gestione dei comuni

- `creaComuneDb()` funzione che prende i dati immessi nel form di creazione di un nuovo comune e inserisce tutte le informazioni nel database utilizzando le funzioni di `users_model`, `categorie_model` e `items_model`;
- `modificaComuneDb()` funzione che permette all'amministratore di modificare la password di accesso del dipendente comunale. Tutte le altre informazioni sul comune non sono modificabili dall'amministratore.

LOGIN controller che gestisce la parte di login al backoffice, validazione dell'accesso e logout

- `validateCredentials()` funzione che, utilizzando la funzione `validate` del model `users_model`, verifica l'esistenza di un utente, crea i parametri della sessione e reindirizza l'utente alla pagina corretta del backoffice;
- `is_logged_in()` funzione di sicurezza utilizzata dai controller per verificare l'esistenza di una sessione di login valida, in caso contrario l'utente viene automaticamente reindirizzato alla homepage;
- `logout()` funzione che svuota i parametri della sessione e reindirizza l'utente alla homepage.

MARKET controller che gestisce la visualizzazione e l'acquisto, attraverso la chiamata alla funzione `acquistaPacchettoDb()`, di nuovi pacchetti di messaggi e notifiche push da parte del dipendente comunale;

MARKETADMIN controller utilizzato dall'amministratore del backoffice per l'abilitazione degli acquisti effettuati dai comuni. Attraverso la funzione `confermaAcquistoPacchettoDb()` il pacchetto di messaggi e notifiche push acquistato viene abilitato e può essere utilizzato.

MESSAGGI controller per la gestione della parte legata alla messaggistica all'interno dell'applicazione

- `creaMessaggioDb()` funzione che prende le informazioni immesse nel form per la creazione di un nuovo messaggio, utilizzando la funzione `creaMessaggio` del model `messaggi_model` salva il messaggio nel database per la sincronizzazione con l'applicazione mobile;

- `eliminaSingoloMessaggioDb()` funzione che rimuove il singolo messaggio dal database utilizzando la funzione `eliminaSingoloMessaggio` del model `messaggi_model`;
- `convertiMessaggioNotificaDb()` funzione che trasforma un messaggio creato precedentemente in una notifica push;
- `pushNotificaDb()` funzione che prende le informazioni immesse nel form per la creazione di una nuova notifica push, chiama la funzione `pushNotifica` del model `messaggi_model` per l'aggiunta della notifica push al database.

PIATTAFORMA controller che gestisce le informazioni immesse dal comune sulla piattaforma ecologica

- `updatePiattaformaDb()` funzione che prende le informazioni immesse nel form per la modifica della piattaforma ecologica, chiama la funzione `updatePiattaforma` del model `piattaforma_model` per il salvataggio nel database;
- `creaPiattaformaDb()` funzione chiamata la prima volta che il dipendente comunale accede al backoffice per la creazione delle informazioni per la piattaforma ecologica, chiama la funzione `creaPiattaforma` del model `piattaforma_model` per il salvataggio delle informazioni inserite nel database.

RIFIUTI controller per la gestione dei rifiuti e delle loro associazioni con la categoria di raccolta

- `getDifferenzeRifiutiDb()` funzione che ritorna tutti i rifiuti che vengono messi a disposizione da Web3king S.n.c. e che non sono ancora stati categorizzati dal comune;
- `creaRifiutoDb()` funzione che, attraverso l'utilizzo delle funzioni `returnCategoriaById` e `creaRifiuto` rispettivamente di `categorie_model` e `items_model`, permette al dipendente comunale di creare un nuovo rifiuto e di associarlo alla categoria di raccolta;
- `eliminaRifiutoDb()` funzione che, utilizzando la funzione `eliminaRifiuto` del model `items_model`, elimina un rifiuto da quelli gestiti dal comune;
- `aggiungiRifiutoDb()` funzione che permette al dipendente comunale di aggiungere all'elenco di rifiuti gestiti dal comune un rifiuto messo a disposizione da Web3ing S.n.c., l'aggiunta viene fatta chiamando la funzione `aggiungiRifiuto` del model `items_model`;

- `updateRifituoDb()` funzione che consente al dipendente comunale, utilizzando le funzioni `returnCategoriaById` e `updateRifiuto` di `categorie_model` e `items_model`, di modificare il nome di un rifiuto e la sua categoria di smaltimento.

VIE controller che permette la gestione delle vie da parte del dipendente comunale

- `eliminaViaDb()` funzione che permette l'eliminazione di una via chiamando la funzione `eliminaVia` del model `vie_model`;
- `creaViaDb()` funzione che permette la creazione di una via e l'associazione con la relativa zona chiamando la funzione `creaVia` del model `vie_model`;
- `updateViaDb()` funzione che permette la modifica del nome di una via e l'associazione con la zona chiamando la funzione `updateVia` del model `vie_model`.

ZONE controller che permette la gestione delle zone da parte del dipendente comunale

- `aggiungiZonaDb()` funzione che permette l'aggiunta di una zona chiamando la funzione `aggiungiZona` del model `zone_model`;
- `removeZonaDb()` funzione che permette l'eliminazione di una zona utilizzando il model `zone_model` e la sua funzione `removeZona`;
- `updateZonaDb()` funzione che permette la modifica del nome della zona chiamando la funzione `updateZona` del model `zone_model`.

3.5 RESTFUL WEB SERVICE

Una sezione separata è stata dedicata alla descrizione del RESTFul Web Service in quanto è di fondamentale importanza per la sincronizzazione delle informazioni presenti nel backoffice e quelle dell'applicazione.

Il Representational state transfer (REST) è un tipo di architettura software per i sistemi di ipertesto distribuiti in cui i termini *representational state transfer* e *REST* sono stati introdotti nel 2000 nella tesi di dottorato di Roy Fielding [1]. In questa architettura è presente un client e un server che nel caso di Ekologi sono rispettivamente l'applicazione mobile e il Cloud Server su cui è installato il backoffice. La parte fondamentale in REST sono le risorse che sono salvate sul Cloud Server a cui l'applicazione mobile accede contattando un determinato URL attraverso una

comunicazione HTTP e il server invia una rappresentazione di queste risorse sotto forma di file testuale; come descritto nell'architettura REST la comunicazione è stateless in quanto il server non tiene traccia del contesto del client e ogni comunicazione è indipendente dalle altre.

3.5.1 *Descrizione architettura*

L'architettura del RESTFul Web Service di Ekologi è schematizzata in Figura 10 in cui è possibile trovare la schematizzazione dei due componenti dell'architettura software e le comunicazioni tra i due. Il client richiede al server le informazioni inserite all'interno del backoffice dal dipendente comunale attraverso degli URL via HTTP e il server risponde attraverso dei file JSON. Per quanto riguarda le risposte è stato utilizzato il formato JSON in quanto è particolarmente semplice il parsing di questo formato nel linguaggio Javascript con il quale è stata scritta l'applicazione mobile.

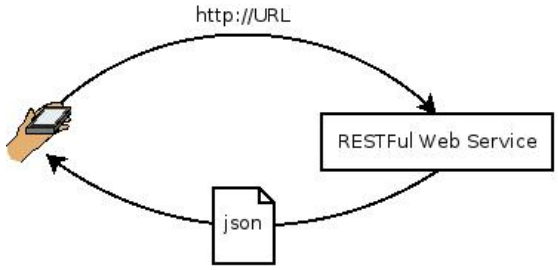


Figura 10: Architettura RESTFul Web Service

Richieste via HTTP

Le richieste vengono fatte dal client, l'applicazione mobile, contattando un particolare indirizzo. Tutte le richieste rispettano questa sintassi

```
http://<hostname>/<controller>/<risorsa>/[<opzioni>/<id>]
```

- `<hostname>` indirizzo internet che l'applicazione contatta;
- `<controller>` controller all'interno del backoffice che gestisce la richiesta;
- `<risorsa>` risorsa richiesta dall'applicazione;
- `<opzioni>/<id>` opzioni per specificare la risorsa richiesta. E' stato messo in parentesi quadre per mostrare che in alcune richieste questo campo non è presente.

Le chiamate RESTFul, di cui vengono riportate solo le parti relative a `<risorsa>` e `<opzioni>/<id>`, effettuate dall'applicazione sono le seguenti

`paesi/`
: chiamata HTTP che ritorna la lista di tutti i paesi che sono gestiti dall'applicazione

`info/id/<idPaese>/`
: chiamata HTTP che ritorna tutte le informazioni inserite nel backoffice per il comune con id `<idPaese>`

`news/id/<idPaese>/`
: chiamata HTTP che ritorna i messaggi inviati dal comune con id `<idPaese>`

`subscribe/token/<tokenDispositivo>/canale/<paese>/os/<osDispositivo>/`
: chiamata HTTP che iscrive al servizio di notifiche push il dispositivo con token `<tokenDispositivo>` al canale per il comune `<paese>` con sistema operativo del dispositivo `<osDispositivo>`

`unsubscribe/token/<tokenDispositivo>/canale/<paese>/`
: chiamata HTTP che rimuove la sottoscrizione alle notifiche push per il dispositivo con token `<tokenDispositivo>` al canale per il comune `<paese>`

Risposte in JSON

Il RESTFul Web Service risponde alle richieste effettuate via HTTP con dei file JSON formattati in questo modo

```
{
  "risorsaTabella-1" : [
    {"chiave-11" : "valore-11",
     "chiave-12" : "valore-12",
     ...,
     "chiave-1n" : "valore-1n"
    },
    {"chiave-21" : "valore-21",
     "chiave-22" : "valore-22",
     ...,
     "chiave-2n" : "valore-2n"
    },
    ...,
    {"chiave-n1" : "valore-n1",
     "chiave-n2" : "valore-n2",
      ...,
      "chiave-nn" : "valore-nn"
    }
  ],
  ...,
```

```
  "risorsaTabella-n" : [...],
  "risorsaRiga1" : {
    "chiave-1" : "valore-1",
    "chiave-2" : "valore-2",
    ...,
    "chiave-n" : "valore-n"
  },
  ...,
  "risorsaRiga-n" : {...}
}
```

`risorsaTabella` rappresenta un array di risultati. L'array è delimitato da parentesi quadre e ogni elemento è racchiuso all'interno delle parentesi graffe. Facendo un confronto con i database relazionali `risorsaTabella` rappresenta una tabella nel database e il singolo elemento dell'array di risultati è una riga della tabella;

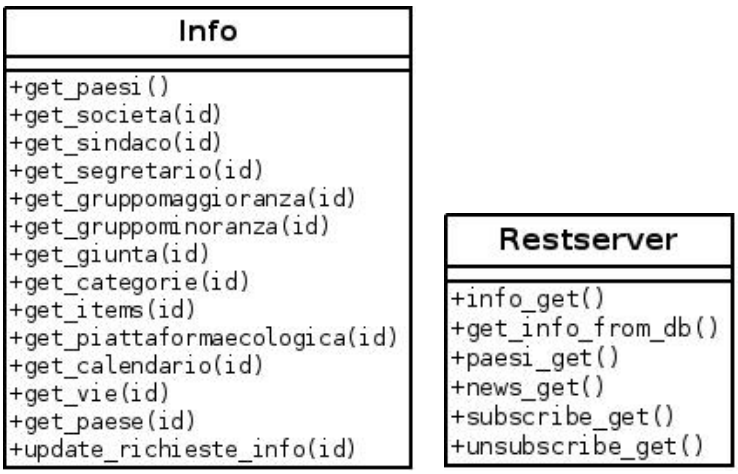
`risorsaRiga` rappresenta un array. L'array è delimitato dalle parentesi graffe e gli elementi sono separati da virgole.

3.5.2 *Implementazione*

L'implementazione del RESTFul Web Service è stata eseguita utilizzando il codice del repository di GitHub: codeigniter-restserver [10]. Il repository mette a disposizione una libreria, un file di configurazione e un controller che hanno consentito l'implementazione del RESTFul Web Service direttamente nell'albero delle directory del backoffice. Per rendere più manutenibile il codice è stato implementato un model che gestisce tutte e sole le interrogazioni al database effettuate dal controller del RESTFul Web Service . Il model e il controller, rispettivamente `Info` e `Restserver`, sono rappresentati in Figura 11

Model

Il model `Info` in Figura 11a fornisce tutte le funzionalità affinchè il controller `Restserver` abbia a disposizione le informazioni per la compilazione del file JSON da mandare in risposta alla chiamata HTTP fatta dall'applicazione mobile. Per le informazioni specifiche per un determinato comune le interrogazioni fatte dal model `Info` utilizzano l'id del paese come parametro in ingresso alle funzioni.
A fini statistici e su richiesta dei comuni, è stata implementata la funzione `update_richieste_info` che viene utilizzata dal backoffice per aggiornare il numero di download di informazioni fatta dai cittadini di un determinato comune.



(a) Model (b) Controller

Figura 11: Model e controller per RESTFul Web Service

Controller

Il controller in Figura 11b fornisce le funzionalità per tradurre le richieste HTTP fatte dall'applicazione mobile in file JSON e il nome delle funzioni visualizzate rispecchia le richieste HTTP che sono state descritte in 3.5.1. Diversamente dalle funzioni presenti nei controller riportati in 3.4.3 in queste funzioni è stato aggiunto il suffisso `_get` con il quale viene specificato che deve essere letto il contenuto dell'URL richiesto via HTTP, eseguita una query sul database utilizzando i parametri specificati nell'URL e mandata una risposta in formato JSON. Per la formattazione della risposta proveniente dal model `Info` in formato JSON viene utilizzata la funzione `json_encode` propria del linguaggio PHP.

4 APPLICAZIONE MOBILE

In questo capitolo verrà presentato lo sviluppo dell'applicazione `Ekologi`, presentando prima l'identificazione degli scenari di utilizzo dell'applicazione (4.1), la definizione degli use case (4.2), la gestione dei dati persistenti per il salvataggio delle informazioni dell'applicazione nel database SQLite (4.3) e infine verrà descritta l'architettura del sistema (4.4).

4.1 IDENTIFICAZIONE SCENARI

4.1.1 *Download dell'applicazione e primo avvio*

Claudia vuole scaricare l'applicazione Ekologi per il suo smartphone Android. Apre il Play Store, cerca Ekologi tra le applicazioni e conferma di volerla scaricare ed installare. Una volta che l'applicazione è stata scaricata ed installata Claudia la apre, sceglie il suo comune di residenza e si iscrive al servizio di notifiche push per ricevere direttamente sul suo smartphone le comunicazioni inviate.

4.1.2 *Ricerca rifiuti per corretto smaltimento*

Claudia non ricorda quale sia la categoria associata al rifiuto `Cartone del latte`. Apre l'applicazione, si sposta nella pagina per la ricerca di un prodotto, inserisce la parola `Cartone` nella barra di ricerca e l'applicazione gli mostra i risultati corrispondenti alla sua ricerca mostrandole che il `Cartone del latte` è associato, per il suo comune di residenza, alla categoria `Carta`. Claudia chiude l'applicazione e correttamente butta il rifiuto nel contenitore della `Carta`.

4.1.3 *Claudia vuole informazioni sulle ultime news dal comune*

Claudia ha visto sulla bacheca comunale che il giorno 30 giugno ci sarà uno spettacolo all'aperto in piazza e viene comunicato che maggiori informazioni si possono trovare nell'applicazione Ekologi. Apre quindi l'applicazione, aggiorna le informazioni, si sposta nella pagina con le notizie, apre la notizia e vede direttamente nell'applicazione quale sarà lo spettacolo, gli artisti coinvolti e gli orari.

4.1.4 *Visualizzazione corretta del calendario per la propria via di residenza*

Claudia vuole avere sul suo smartphone il calendario di raccolta per la sua via di residenza. Apre l'applicazione e si sposta nella pagina che le mostra tutte le vie del proprio comune, nella barra di ricerca inserisce `Meucci` e seleziona dall'elenco la sua via di residenza. Si sposta nella pagina del calendario che le mostra i prossimi giorni di raccolta per la via selezionata.

4.1.5 *Invio mail al segretario comunale*

Claudia vuole mandare una mail al segretario comunale per chiedere una informazione sulle prossime elezioni del sindaco. Apre l'applicazione, si sposta nella pagina con la lista dei dipendenti comunali e clicca sul bottone per l'invio di una mail direttamente all'indirizzo del segretario. L'applicazione apre il programma di posta configurato sullo smartphone, Claudia immette l'oggetto del messaggio e il testo e invia la mail.

4.2 USE CASE

4.2.1 *Scelta del comune di residenza al primo avvio*

ATTORI Utente applicazione

INPUT L'utente ha appena scaricato l'applicazione e vuole selezionare il proprio comune di residenza

EVENTI
- L'utente avvia l'applicazione;
- L'applicazione mostra l'elenco dei comuni disponibili in `Ekologi`;
- L'utente seleziona il proprio comune;
- L'applicazione scarica tutte le informazioni per il comune prescelto;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha selezionato il proprio comune e l'applicazione ha scaricato tutte le informazioni

4.2.2 *Ricerca del prodotto Lattina*

ATTORI Utente applicazione

INPUT L'utente vuole conoscere la categoria di smaltimento del prodotto `Lattina`

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina di ricerca di un prodotto;
- Cerca il prodotto nella barra di ricerca;
- L'applicazione mostra tutti i rifiuti che contengono la parola `Lattina`;
- L'utente visualizza la categoria di smaltimento associata al prodotto `Lattina`;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha visualizzato la categoria di smaltimento del prodotto `Lattina`

4.2.3 *Ricerca degli assessori comunali*

ATTORI Utente applicazione

INPUT L'utente vuole conosce la composizione del consiglio comunale

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina di visualizzazione delle informazioni del comune;
- Si sposta nella pagina di visualizzazione del personale;
- L'applicazione mostra l'elenco di tutti gli assessori e le modalità di contatto via mail e telefono;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha visualizzato l'elenco degli assessori e le loro informazioni di contatto

4.2.4 *Chiamata al numero di telefono del comune*

ATTORI Utente applicazione

INPUT L'utente vuole chiamare il numero del comune

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina di visualizzazione delle informazioni del comune;
- Clicca sul numero di telefono del comune;

- Conferma di voler chiamare il numero selezionato;
- L'applicazione fa partire la chiamata;
- L'utente chiude la chiamata;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha fatto una chiamata al numero del comune

4.2.5 *Visualizzazione dettagli per la categoria di raccolta*

ATTORI Utente applicazione

INPUT L'utente vuole visualizzare la descrizione di una categoria di raccolta

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina di visualizzazione dell'elenco delle `Categorie`;
- Seleziona la categoria di cui vuole visualizzare la descrizione;
- L'applicazione mostra la pagina con la descrizone della categoria selezionata;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha visualizzato la descrizione per la categoria di raccolta

4.2.6 *Selezione via di residenza*

ATTORI Utente applicazione

INPUT L'utente deve selezionare la sua via di residenza

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina delle `Impostazioni`;
- Clicca sul bottone per la scelta della via;
- Cerca la via nella barra di ricerca;
- Seleziona la nuova via;
- L'applicazione salva la via scelta e modifica il calendario in base alla selezione effettuata;
- L'utente chiude l'applicazione;

OUTPUT La via di residenza è stata impostata nell'applicazione

4.2.7 *Cambio comune di residenza*

ATTORI Utente applicazione

INPUT L'utente vuole cambiare il comune predefinito in `Ekologi`

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina delle `Impostazioni`;
- Clicca sul bottone per la scelta del comune;
- Cerca il nuovo comune nella barra di ricerca;
- Seleziona il nuovo comune;
- L'applicazione scarica i dati del nuovo comune e la fine del download viene notificata a video;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha cambiato il comune predefinito in `Ekologi`

4.2.8 *Abilitazione notifiche push*

ATTORI Utente applicazione

INPUT L'utente vuole iscriversi alle notifiche push del suo comune

EVENTI
- L'utente avvia l'applicazione;
- Si sposta nella pagina delle `Impostazioni`;
- Clicca sul bottone per l'iscrizione alle notifiche push;
- L'applicazione notifica all'utente l'iscrizione alle notifiche push;
- L'utente chiude l'applicazione;

OUTPUT L'utente si è iscritto alle notifiche push del suo comune

4.2.9 *Aggiornamento dati dell'applicazione*

ATTORI Utente applicazione

INPUT L'utente deve aggiornare i dati dell'applicazione

EVENTI
- L'utente avvia l'applicazione;
- Clicca sul bottone per l'aggiornamento dei dati;
- Attende la fine del download che viene notificata a video;
- L'utente chiude l'applicazione;

OUTPUT L'utente ha aggiornato i dati dell'applicazione

4.3 GESTIONE DATI PERSISTENTI

4.3.1 *Entità*

In riferimento alla Figura 12 si possono elencare le entità che sono molto simili a quelle descritte in 3.3.1:

- Rifiuto: rappresenta il singolo rifiuto che è possibile cercare all'interno dell'applicazione;
- Categoria: rappresenta la categoria di raccolta associata ad un insieme di rifiuti che viene gestita dal comune;
- Piattaforma ecologica: contiene tutte le informazioni sulla piattaforma ecologica del comune;
- Via: rappresenta la via del comune che può appartenere ad una particolare zona;
- Zona: rappresenta la zona di raccolta per un particolare insieme di vie;
- Giorno Raccolta: rappresenta il singolo giorno di raccolta di una categoria di rifiuto in una determinata zona;
- Messaggio: rappresenta il messaggio visualizzato all'interno dell'applicazione nella sezione news;
- Utente: rappresenta l'utente dell'applicazione di cui si vogliono salvare informazioni e preferenze di utilizzo;
- Info: rappresenta le informazioni da registrare per un particolare comune all'interno dell'applicazione;
- Sindaco: rappresenta le informazioni sul sindaco;
- Segretario: rappresenta le informazioni sul segretario comunale;
- Giunta: rappresenta le informazioni sugli assessori che compongono la giunta comunale;
- Gruppo Maggioranza: rappresenta le informazioni sui componenti del gruppo di maggioranza;
- Gruppo Minoranza: rappresenta le informazioni sui componenti del gruppo di minoranza.

4.3.2 *Associazioni*

In riferimento alla Figura 12 si possono elencare le associazioni che vengono presentate in forma attiva:

APPARTIENE TRA VIA E ZONA una via appartiene ad una e una sola zona;

APPARTIENE TRA RIFIUTO E CATEGORIA un rifiuto può appartenere ad una e una sola categoria di smaltimento gestita dal comune;

ASSOCIATO TRA GIORNO RACCOLTA, ZONA E CATEGORIA ogni giorno di raccolta deve essere associato ad una e una sola zona e categoria di raccolta.

4.3.3 *Modello relazionale*

Con riferimento allo schema in Figura 12 sono state costruite le seguenti tabelle della base di dati interna all'applicazione:

calendar(id, giorno, tipo, zona, nomegiorno)

general_info(comune, indirizzo, telefono, maps, site, nomedb, iddb, username)

giunta(id, carica, nome, email, telefono)

gruppomaggioranza(id, nome, email)

gruppominoranza(id, nome, email)

item_info(id, prodotto, riferimento)

item_type(id, nome, descrizione, color, note)

messaggi(id, titolo, testo, data, letto)

piattaformaecologica(id, indirizzo, orari, note)

road(id, via, zona)

segretariocomunale(id, nome, email, telefono, via, mappa)

sindaco(id, nome, email, telefono, via, mappa)

user(rowidstreet, name, surname, email, days, push)

zone(color, number)

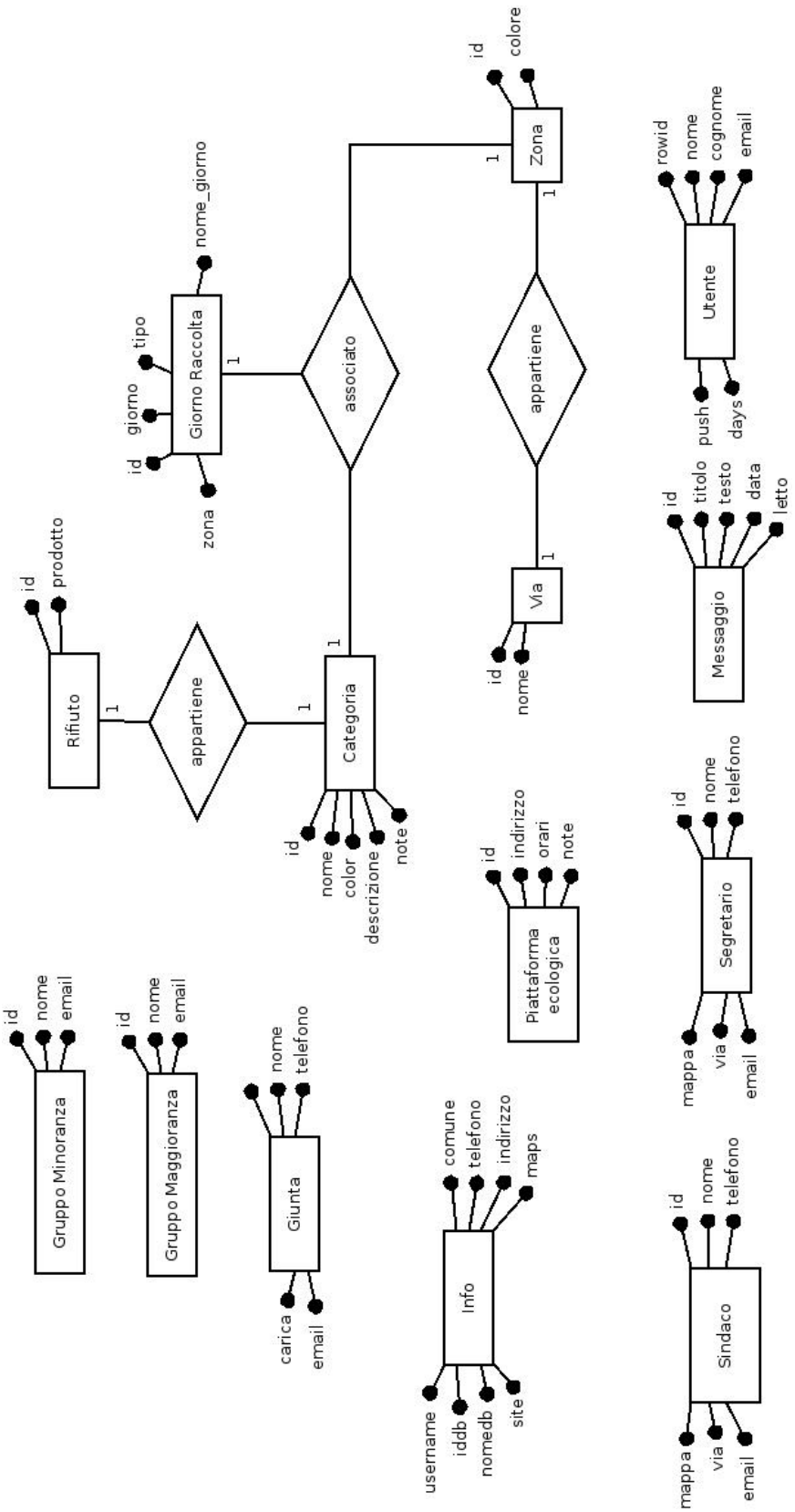


Figura 12: Schema entità associazioni applicazione

A cui si aggiungono le seguenti precisazioni:

- l'associazione tra la via e la zona avviene grazie al campo `number` di `zone`;
- l'associazione tra `item_info` e `item_type` viene fatta grazie al campo `riferimento` della prima tabella che corrisponde al campo `id` della seconda;
- il campo `id` presente in quasi tutte le tabelle è chiave in quella specifica tabella;
- all'interno del database dell'applicazione non è presente il campo `paesi_id` nelle tabelle come accadeva nel database del backoffice in quanto di volta in volta l'applicazione tiene memorizzate le informazioni di un solo comune;
- in riferimento al modello relazionale 3.3.3 del backoffice in cui ogni dipendente del comune viene salvato nella stessa tabella, nell'applicazione ogni tipologia di dipendente è stato salvato in una tabella dedicata: `sindaco`, `segretariocomunale`, `giunta`, `gruppomaggioranza`, `gruppominoranza`.

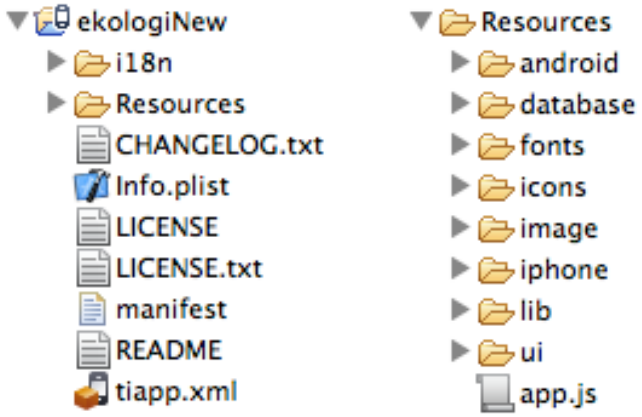
4.4 ARCHITETTURA DEL SISTEMA

In questa sezione verrà presentata la struttura di base della app richiesta da Appcelerator (4.4.1) e le modifiche apportate durante lo sviluppo per l'implementazione delle funzionalità richieste (4.4.2).

4.4.1 *Struttura base della app*

Nella creazione di una applicazione con Titanium di Appcelerator Inc. viene creato un predefinito albero delle directory, con le modifiche apportate allo stesso la struttura base della app è quella riportata in Figura 13a in cui le parti principali sono costituite dal file `tiapp.xml` e dalla cartella `Resources`. Il file `tiapp.xml` è il file di configurazione di tutta l'applicazione con cui vengono specificati: autore, società, licenze, certificati per gli store, moduli utilizzati, versione dell'applicazione, piattaforme supportate dall'applicazione.
Nella cartella `Resources` sono presenti tutte le cartelle con il codice sorgente dell'applicazione e i file necessari come riportato in Figura 13b di cui si da qui una breve descrizione:

ANDROID cartella contenente i file specifici per la piattaforma Android: icone e splash screen alle risoluzioni richieste;



(a) Albero delle directory principale (b) Alberto delle directory nella cartella Resources

Figura 13: Alberi delle cartelle nell'applicazione mobile

DATABASE cartella che contiene il file del database interno all'applicazione nel formato sqlite;

FONTS cartella che contiene i file per font personalizzati;

ICONS cartella che contiene i file immagine per le icone presenti nell'applicazione;

IMAGE cartella contenente i file immagini i loghi presenti nell'applicazione;

IPHONE cartella contenente i file specifici per la piattaforma iOS: icone e splash screen alle risoluzioni richieste;

LIB cartella contenente i file sorgenti per le librerie e le variabili globali dell'applicazione;

UI cartella contenente i file per le diverse pagine dell'applicazione;

APP.JS file di avvio dell'applicazione.

4.4.2 *Sviluppo applicazione*

Il codice dell'applicazione mobile è suddiviso nelle cartelle `lib`, `ui` e nel file `app.js`. E' stata effettuata questa suddivisione per ragioni di pulizia del codice in modo da avere nelle diverse cartelle file con funzioni simili. Il file `app.js` è il file di default con cui viene avviata l'applicazione, in questo file sono stati inclusi: l'installazione e inizializzazione del database, caricamento delle librerie descritte nel prossimo paragrafo,

codice di controllo per intercettare il primo avvio dell'applicazione dai successivi avvii. Quest'ultima funzionalità permette le chiamate a codice custom che deve essere eseguito solo al primo avvio come la schermata di presentazione e introduzione della app.

Cartella lib

In questa cartella sono contenute le librerie e le variabili necessarie per la strutturazione dell'interfaccia dell'applicazione sulle diverse piattaforme e dimensioni dello schermo. Viene riportata ora una descrizione del contenuto dei file senza scendere nei dettagli delle singole funzioni:

ISTABLET.JS file contenente le funzioni per identificare il tipo di schermo su cui viene avviata l'applicazione. Per il riconoscimento di un iPad Titanium mette già a disposizione delle API mentre per Android questo non è possibile e il controllo sul tipo di schermo fa uso delle dimensioni in pixel di altezza e lunghezza del display, risoluzione e densità dei punti sullo schermo;

MENU.JS file contenente le funzioni specifiche per la creazione del menu laterale e del menu di aggiornamento della app;

SUBSCRIPTION.JS file contenente le funzioni specifiche per la gestionde della iscrizione e deiscrizione al servizio di notifiche push utilizzando il RESTFul Web Service descritto in 3.5;

VAR.JS file contenente le variabili per la definizione dell'interfaccia sui diversi dispositivi. Le variabili presenti descrivono l'altezza e larghezza dei diversi elementi, la grandezza dei font utilizzando le funzionalità del file `isTablet.js`.

Cartella ui

In questa cartella sono presenti i file che gestiscono la visualizzazione di tutte le pagine presenti nell'applicazione in cui può trovare tutto il codice necessario per la creazione della pagina, la logica e l'interazione con il database. Uno schema che rappresenta la struttura della app è riportato in Figura 14 in cui `main.js` rappresta l'unica finestra dell'applicazione a cui vengono aggiunte tutte le viste che contengono le varie pagine dell'applicazione. Viene riportata ora una descrizione del contenuto dei file senza scendere nei dettagli delle singole funzioni:

MAIN.JS file contenente il codice per la gestione della struttura della app. In questo file si trova il codice eseguito durante il primo avvio e quello per i successivi. Durante il primo avvio viene eseguita la scelta del comune di default, il download delle informazioni

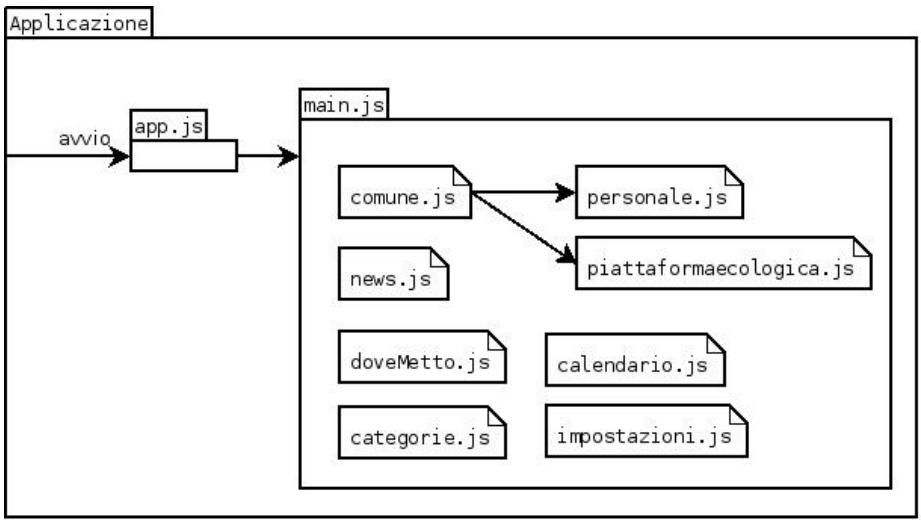


Figura 14: Struttura finestra e viste dalla app

e il loro salvataggio nelle tabelle del database dell'applicazione descritte in 4.3.3, per i successivi avvii dell'applicazione questo file contiene il codice per la predisposizione di tutte le viste, i menu e i listener per le azioni;

COMUNE.JS file contenente il codice per la gestione della vista per la pagina del comune, questa vista mostra le informazioni di contatto del comune e fornisce la possibilità di accedere alla pagina del personale e a quella delle informazioni sulla piattaforma ecologica;

NEWS.JS file contenente il codice per la gestione della vista che mostra le news inviate dal comune ai propri cittadini, per ogni click sulla notizia la vista apre una pagina dedicata che mostra il contenuto della notizia e mostra il tutto in una pagina web con la formattazione definita in fase di creazione della notizia nel backoffice dal dipendente comunale;

DOVEMETTO.JS file contenente il codice per la gestione della vista per la visualizzazione dell'elenco di tutti i rifiuti gestiti dal comune con la possibilità di vedere immediatamente la categoria associata. Nella vista è presente anche una barra di ricerca con cui, in modo semplice e veloce, si può ricercare un particolare rifiuto, per migliorare l'esperienza utente la ricerca viene fatta in tempo reale ad ogni nuova lettera inserita;

CATEGORIE.JS file contenente il codice per la gestione della vista con l'elenco delle categorie di raccolta gestite dal comune. Cliccando sulla riga corrispondente alla categoria scelta la vista apre una

pagina dedicata che mostra la descrizione e le note sulla categoria inserite nel backoffice dal dipendente comunale;

CALENDARIO.JS file contenente il codice per la gestione della vista per la visualizzazione del calendario di raccolta. Il calendario riporta i successivi 30 giorni di raccolta mostrando il nome della categoria, il giorno della settimana e la data in formato esteso;

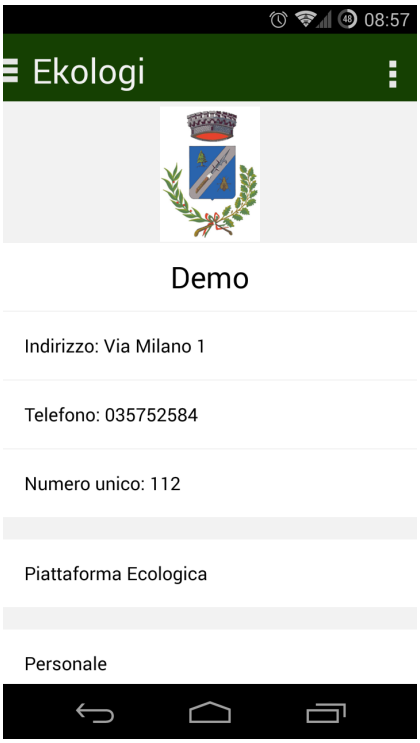
IMPOSTAZIONI.JS file contenente il codice per la gestione della vista con le impostazioni dell'applicazione. Questa vista mette a disposizione la possibilità di modificare la scelta del comune, la via di residenza e la iscrizione/deiscrizione al servizio di notifiche push. Come per la ricerca dei rifiuti anche quella del comune e della via vengono facilitate da una barra di ricerca che aggiorna i risultati in tempo reale ad ogni nuova lettera inserita;

INTERNAL/PERSONALE.JS file contenente il codice per la gestione della vista che riporta le informazioni su: sindaco, segretario comunale, giunta comunale, gruppo di maggioranza e minoranza. All'interno della vista viene data la possibilità all'utente di contattare direttamente via telefono o email i propri rappresentanti;

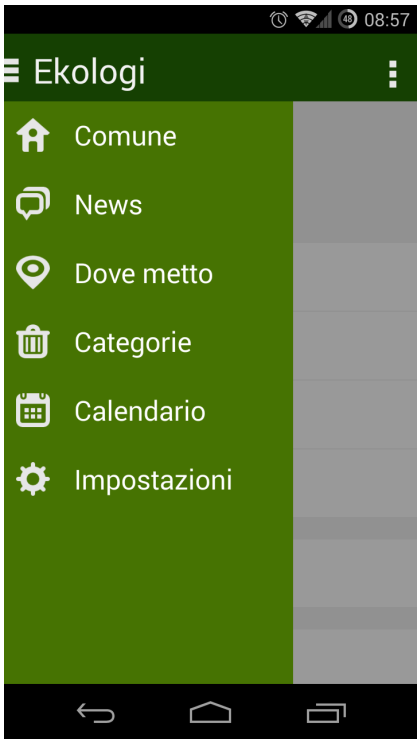
INTERNAL/PIATTAFORMAECOLOGICA.JS file contenente il codice per la gestione della vista per la visualizzazione delle informazioni sulla piattaforma ecologica del comune.

Alcuni esempi delle pagine descritte possono essere visualizzati in Figura 15

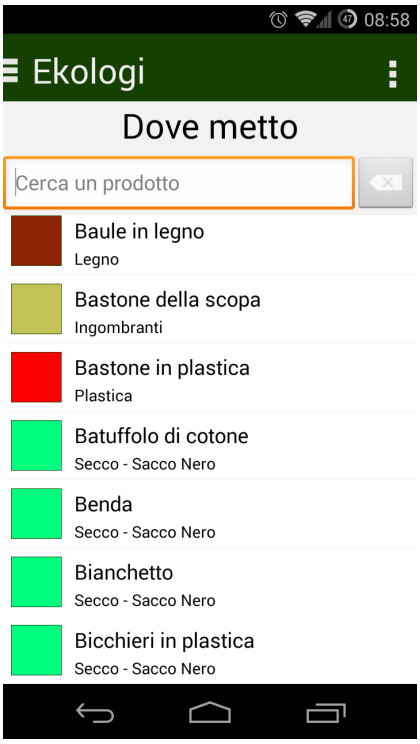
- Figura 15a: vista dell'applicazione, file `comune.js`, in cui si possono visualizzare le informazioni di contatto del comune;
- Figura 15b: menu laterale della app che consente la navigazione tra le diverse viste;
- Figura 15c: vista dell'applicazione, file `dovemetto.js`, in cui è possibile la ricerca di un rifiuto con la barra di ricerca;
- Figura 15d: vista dell'applicazione, file `calendario.js`, per la visualizzazione dei prossimi giorni di raccolta.



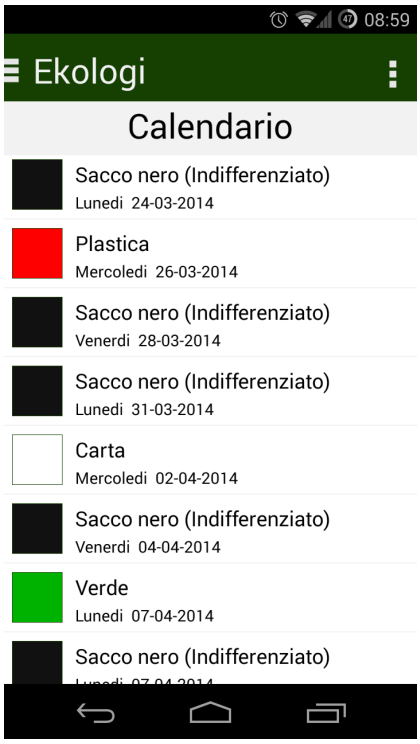
(a) Finestra principale applicazione mobile



(b) Menu laterale



(c) Ricerca prodotti



(d) Calendario raccolta

Figura 15: Interfaccia utente dell'applicazione

5 CONCLUSIONI E SVILUPPI FUTURI

Il progetto per la realizzazione di Ekologi e del suo backoffice ha raggiunto una discreta stabilità ed è ora in fase di vendita alle amministrazioni comunali. Come tutti i software non può mai dirsi concluso il suo sviluppo e insieme alle amministrazioni comunali si stanno valutando delle aggiunte per migliorare il servizio offerto ai cittadini. Uno dei servizi richiesti è la possibilità di utilizzare l'applicazione come strumento di comunicazione bidirezionale tra l'amministrazione e il cittadino per permettere a quest'ultimo di notificare al comune direttamente dall'applicazione eventuali comunicazioni, disservizi, richieste, feedback. Si stanno studiando le funzionalità di Appcelerator di interazione con la fotocamera degli smartphone, la possibilità di geotaggare le foto scattate per migliorare l'informazione legata alla foto nel caso di comunicazione di disservizi, l'integrazione con il calendario degli smartphone per migliorare l'esperienza utente e la possibilità di sincronizzazione automatica in background in caso di collegamento ad internet.

Ekologi è il punto di partenza per lo sviluppo di altri prodotti realizzati da Web3king S.n.c., la struttura grafica dell'applicazione è stata utilizzata per la realizzazione di altre app e le funzionalità presenti nel backoffice: login, invio schedulato di notifiche push, gestione multiutente, vengono utilizzate per implementare altri backoffice di gestione di servizi che spaziano dalla ristorazione, alle assicurazioni, alle commesse per aziende di arredamento.

Particolarmente formativa è stata l'implementazione del RESTFul Web Service che ha spostato la complessità di gestione degli aggiornamenti della app e delle notifiche push dall'applicazione al Cloud Server (2.5.2) permettendo una maggiore sicurezza delle comunicazioni, semplicità di sviluppo della app ed estendibilità futura.

Web3king S.n.c. è da sempre impegnata nella diffusione del software libero e a questo proposito si sta valutando la possibilità di rilasciare parti del codice realizzato con licenza libera. La parte che potrebbe essere rilasciata nel breve/medio termine è la struttura grafica dell'applicazione che può essere utilizzata da altri sviluppatori per la realizzazione di app ottimizzate sia per smartphone che per tablet utilizzando la stessa base di codice senza la necessità di realizzare due app separate.

BIBLIOGRAFIA

[1] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[2] Appcelerator Inc. Appcelerator. http://www.appcelerator.com/.

[3] Appcelerator Inc. Titanium cloud services. http://www.appcelerator.com/cloud/.

[4] Appcelerator Inc. Titanium mobile development environment. http://www.appcelerator.com/titanium/.

[5] Apple Inc. ios 7 design resources - ios human interface guidelines. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/.

[6] EllisLab Inc. Codeigniter. http://ellislab.com/codeigniter.

[7] EllisLab Inc. Ellislab. http://ellislab.com/.

[8] Google Inc. Android design guide. http://developer.android.com/design/index.html.

[9] Twitter Inc. Twitter bootstrap. http://getbootstrap.com/.

[10] Chris Kacerguis. Codeigniter rest server. https://github.com/philsturgeon/codeigniter-restserver.