

ITIS M.DELPOZZO

Dipartimento di Informatica

ANNO SCOLASTICO 2013/2014

Classe 5^O

Progetto/Tesina :

Let's Move

Ad opera di:

MIGLIORE Giacomo,



INDICE

1. Obiettivo
2. Materiale neccessario (and translation)
3. Ambiente di sviluppo
4. Android
5. Java
6. Php
7. XML
8. Json
9. MySQL
10. SQL
11. Tecnologia NFC
12. Descrizione tecnica del progetto
13. Manuale Utente

1. Obiettivo

L'obiettivo del progetto è di sfruttare le nuove tecnologie per facilitare o risparmiare su sprechi che ormai potrebbero essere estremamente limitati. Nel nostro caso trattiamo del risparmio cartaceo dei biglietti del pullman (o qualunque altro mezzo pubblico su cui venga montato il necessario per poter utilizzare il nostro progetto): questo avviene grazie al pagamento delle tratte tramite apparecchiature mobili, quali smartphone o tablet. Oltre al pagamento degli utenti, abbiamo creato due altre applicazioni che permetteranno una al gestore dei trasporti di poter modificare le linee, gli orari e le fermate, mentre la terza applicazione sarà quella utilizzata dai controllori per verificare che tutti abbiano pagato la tratta.

2. Materiale Necessario

Il nostro progetto richiede l'utilizzo di una tecnologia NFC (che verrà spiegata oltre) ed un collegamento ad Internet wireless: queste due richieste limitano la quantità di possibili utilizzatori ma la nostra idea era di non fermarci ad una piccola realtà come è quella di Cuneo, dove sicuramente non ci sono le finanze necessarie e la tecnologia non è ancora delle più sviluppate; se si pensa già solo ad una città come Milano questo metodo, in maniere leggermente differenti, sta già venendo utilizzata.

Su ogni mezzo di trasporto deve essere installato un tablet portatile avente le caratteristiche citate sopra così l'utente, una volta salito, passerà il proprio dispositivo attaccato per pagare la corsa: questo tablet sarà lo stesso utilizzato dal controllore per verificare il pagamento di ogni persona.

Il linguaggio di programmazione utilizzato per creare queste applicazioni è l'ormai diffuso "Android", linguaggio che permette una personalizzazione del proprio lavoro molto elevata.

Required Material

Our project needs a device with an NFC sensor (that will be explained further) and a wireless Internet connection. These two requirements may limit the number of possible users in a small town like Cuneo; nevertheless our idea was to expand the project beyond the reality of Cuneo, where financial support and technology are not always available; rather we are thinking about expanding to a city like Milan, where this way of paying tickets is already used in a slightly different manner.

On every bus there must be a tablet with the features stated earlier. When the user gets on the bus, he has to pass his mobile phone near to the tablet to pay for the ride; this tablet will be the same used by the ticket inspector to verify that everyone has paid.

The operative system used to create this applications is "Android", a widespread operating system that allows a great deal of individual customization.

3. Ambiente di sviluppo

L'ambiente di sviluppo utilizzato è Eclipse per Android disponibile al link



<http://developer.android.com/sdk/index.html> . Eclipse è un ambiente di sviluppo multi-linguaggio e multiplatforma. Questo IDE può essere integrato con numerosi plug-in, che oltre allo sviluppo in java offrono mezzi per sviluppare, per esempio, in C++, PHP, Python ed altri. Tuttavia per sviluppare PHP abbiamo deciso di utilizzare un editor di testo minimale come per esempio notepad++ o wordpad, perché era già presente sui nostri computer, ed installare il plugin per Eclipse non vi avrebbe offerto ulteriori vantaggi.

4. Android

Il progetto Letsmove si basa sul sistema operativo Android, utilizzando quindi dispositivi mobili quali smartphones e tablets. Cosa è Android? Android è un sistema operativo sviluppato da Google, rilasciato per la prima volta nel 2008 con l'HTC Dream. Esso poggia le sue basi su una struttura open-source (escluse alcune versioni intermedie) sfruttando il kernel Linux. Essendo un sistema open-source si può scaricare, studiare, modificare ed eventualmente proporre il proprio codice per contribuire al progetto direttamente dalla piattaforma github (<https://github.com/android>).



5. Java

Le applicazioni per Android sono principalmente sviluppate in Java (nonostante sia possibile sviluppare anche in Python e C++). Java è un linguaggio di programmazione nato nel 1992 ed annunciato ufficialmente nel 1995, orientato alla programmazione ad oggetti. E' stato creato con l'intento di renderlo il più possibile indipendente dalla piattaforma di esecuzione. Questo concetto è espresso dal motto "write once, run everywhere(WORA)". Il codice viene compilato in un formato chiamato bytecode, per essere successivamente eseguito da una Java Virtual Machine. Android supporta la Dalvik virtual machine (DVM), una macchina virtuale ottimizzata per sfruttare la poca memoria offerta dai dispositivi mobili.



6. Php

Il php, per esteso PHP: Hypertext Preprocessor, è un linguaggio di programmazione



interpretato, utilizzato lato web server. Nel nostro progetto è utilizzato per connettere le applicazione lato mobile con il database salvato sul server. Il php esegue le sue interrogazioni al database, ed alla fine, o allo scatenarsi di un'eccezione, si restituisce un responso json. Questo responso contiene un numero che indica il successo o l'insuccesso più un messaggio ed eventuali ulteriori dati.

7.XML



Il linguaggio XML è utilizzato in Android con tre scopi: definire le informazioni di layout, ovvero grafica; abilitare il supporto multilingue (da noi non sfruttato perché il progetto è creato per essere utilizzato inizialmente in una singola cittadina); la definizione del file `AndroidManifest.xml`. Quest'ultimo file descrive l'applicazione al dispositivo, che lo utilizzerà al momento dell'installazione. Indica ogni componente che utilizza (ovvero ogni Activity, ogni Service ecc...), e i permessi che richiede.

L'xml, od in forma estesa eXtensible Mark up Language, è simile all' HTML che, a differenza di esso, supporta dei tag personalizzati. L'Xml definisce anche il concetto di namespace, ovvero un URI dove i tag sono definiti e resi condivisibili in modo che altri programmatori possano utilizzarli.

8.Json



JSON, acronimo di JavaScript Object Notation, è un formato adatto per lo scambio dei dati in applicazioni client-server.

Un esempio di json, è il seguente:

```
{
  "successo": "0",
  "messaggio": "PHP eseguito!",
  "dati": [
    {"val": "0", "azione": "Avvia"},
    {"val": "1", "azione": "Sospendo"},
    {"val": "2", "azione": "Spegni"}
  ]
}
```

I linguaggi php e Android hanno funzioni apposite per formattare e leggere i dati json.

9. MySQL

Abbiamo deciso di salvare i dati all'interno di un database relazionale MySQL, ospitato dalla piattaforma Altervista ed interrogato tramite il linguaggio SQL.



10. SQL

In informatica SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:



- creare e modificare schemi di database (DDL - Data Definition Language);
- inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);
- interrogare i dati memorizzati (DQL - Data Query Language);
- creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

Nonostante il nome, non si tratta dunque solo di un semplice linguaggio di interrogazione, ma alcuni suoi sottoinsiemi si occupano della creazione, della gestione e dell'amministrazione del database.

11. Tecnologia NFC

Near Field Communication (NFC) è una tecnologia che fornisce connettività wireless(RF) bidirezionale a corto raggio (fino a un massimo di 10 cm). La tecnologia NFC si è evoluta dai più semplici dispositivi RFID, che permettevano solamente la possibilità di una comunicazione unidirezionale: quando due dispositivi NFC vengono messi a contatto si crea una rete peer-to-peer, dove entrambi possono inviare e ricevere informazioni. Questa tecnologia è stata implementata ultimamente su tablet e smartphone di ultima generazione, ma la troviamo anche su chip integrati : la velocità di trasmissione massima è di 424 kbit/s e la sua frequenza è di 14,56 MHz.



12. Descrizione tecnica del progetto

Per sviluppare questa tesina sono stati utilizzati tre mezzi: il sistema operativo Android, il linguaggio di programmazione web lato server php, e la tecnologia NFC (Near Field Communication).

Il nostro progetto è strutturato in tre applicazioni tutte collegate ad un database centrale ospitato dai server di Altervista. Siccome Android non supporta la connessione diretta al database è stato necessario scegliere un linguaggio per collegarli. Noi abbiamo scelto il php perché avevamo già esperienza con esso e perché unico supportato dai server Altervista. Inoltre per collegare due dispositivi Android, più precisamente l'applicazione dell'utente e quella del controllore, abbiamo scelto di utilizzare la tecnologia NFC, che permette di collegare due dispositivi posti a breve distanza l'uno dall'altro in una connessione peer to peer.

Analisi del codice Android

Di seguito spieghiamo le classi che abbiamo utilizzato per completare le nostre applicazioni.

Cominciamo con l'analisi dell'applicazione utente:

All'apertura dell'applicazione parte un'Activity di controllo che legge da un file delle preferenze dell'applicazione (Shared Preference File) se l'utente ha effettuato l'accesso, e decide se far visualizzare all'utente l'Activity di login oppure la schermata Home.

La classe Shared Preference File permette, di creare e salvare all'interno di un file privato dell'applicazione, dati di tipo primitivo, quali Stringhe, Interi, Float ecc... in modo permanente. In questo modo i dati rimangono salvati sul dispositivo anche quando questo è spento. Il vantaggio rispetto ai file di testo normali è la facilità d'uso e soprattutto la riservatezza perché l'utente non ne ha l'accesso diretto.

Se l'utente non ha ancora effettuato il login, viene aperta la schermata di accesso. Da qua i dati vengono mandati alla pagina "login.php" situata sul server di Altervista.

Per aprire un'Activity in Android, c'è bisogno di utilizzare la classe Intent che permette di aprire nuove finestre dall'applicazione. Basta semplicemente dichiarare un oggetto di tipo Intent e passargli come parametro la classe da aprire.

Per mandare la richiesta alla pagina web i dati devono essere salvati in formato Json. Per fare ciò, è stato utilizzata la classe Json Parser, che si occupa di incapsulare i dati nel formato necessario e di spedire tutto alla pagina desiderata. Questa classe si occupa inoltre del ricevimento dei dati ovvero di fare il lavoro opposto a quello spiegato precedentemente.

Visto che la richiesta di login alla pagina php è un'operazione che necessita di un tempo relativamente lungo, abbiamo deciso di svolgere questa operazione in un thread separato da quello che si occupa della gestione grafica. Questo significa che l'operazione viene eseguita in parallelo all'applicazione.

La classe che gestisce il nuovo thread si chiama AsyncTask. Nel nostro progetto la classe che estende AsyncTask viene

utilizzata come inner class dove risulta necessaria una connessione al qualche pagina php.

Tutte le classi che estendono AsyncTask devono ereditare il metodo doInBackground che specifica quale operazione deve essere eseguita in parallelo all'applicazione stessa. Altri metodi da noi utilizzati all'interno di queste classi sono onPreExecute e onPostExecute, eseguiti rispettivamente prima e dopo doInBackground, che ci permettono di inizializzare tutti i parametri del progress Dialog e di chiuderlo dopo il completamento dell'operazione richiesta. Il metodo onPostExecute permette inoltre di aggiornare elementi sull'interfaccia grafica, mentre negli altri metodi quest'operazione risulta vietata.

Il progress dialog non è altro che una semplice finestra di dialogo con l'utente che spiega l'operazione in corso di svolgimento.

Se l'utente invece ha già effettuato l'accesso all'applicazione, viene mandato direttamente alla pagina Home dell'applicazione.

Tutte e tre le schermate della Home sono gestite da un'Activity soltanto. Mentre le varie schermate sono dei Fragment. Quest'ultimo non è altro che una porzione di un'Activity. Grazie ai Fragments infatti possiamo assegnare più schermate (layout) ad un'unica Activity.

Per passare da un Fragment all'altro viene utilizzato il TabActionBar ovvero dei pulsanti all'interno dell'action bar dell'applicazione (parte nera col titolo dell'applicazione).

La gestione del TabActionBar avviene all'interno della classe che estende l'Activity che si occupa di tutti i vari Fragments (classe Main nel caso dell'applicazione utente).

Un'altra parte significativa dell'applicazione utente è il fragment che gestisce gli orari delle varie linee dei pulman.

In esso vi sono una ExtendibleListView ed una EditText. La ExtendibleListView contiene nelle View principali il nome delle linee ed un bottone per visualizzare gli orari di partenza e ritorno. Le ChildView contengono tutte le fermate della linea. Per creare questa ListView è stato implementato un Adapter personalizzato, ovvero che estende BaseAdapter. Questa scelta è dovuta al fatto di aver dei layout personalizzati per le varie righe.

Nel progetto vi sono numerose altre ListView. La maggior parte di esse hanno un adapter che è personalizzato, estendendo BaseAdapter o ArrayAdapter. Altre invece

hanno un ArrayAdapter non personalizzato. Nell'applicazione GestioneLinea alcune delle righe della listView hanno un comportamento dinamico. Ciò significa che cambiano il layout a seconda del comportamento dell'utente: se l'utente clicca su una riga alcune ad alcune View verrà assegnato il valore GONE alla visibilità ed ad altre VISIBLE.

Come vengono popolate le ListView? I dati relativi alla ListView vengono prelevati da un database locale. Un metodo carica i dati del database dentro ad una Collection, solitamente un ArrayList. Con essa si carica poi l'ArrayAdapter. E' stata evitata la classe CursorAdapter, che permette di utilizzare i risultati di una query per popolare la ListView perché richiede una TextView per ogni colonna ritornata dalla query , condizione che sovente non era soddisfatta.

Le query svolte per popolare la ListView vengono eseguite su un database locale. Questo database contiene, in tutte e tre le applicazioni, una copia delle tabelle Linea, Tratta e Fermata presenti sul database online. Si è deciso di salvarle per motivi legati al tempo di richiesta sul database online. L'aggiornamento viene eseguito manualmente su tutte le applicazioni eccetto GestionLinea, su cui viene eseguito all'apertura.

Il database locale è dichiarato in una classe che estende `SQLiteOpenHelper`. Questa classe permette di creare le tabelle, aggiornarle e fornisce con dei metodi statici il nome delle varie colonne e tabelle. In `GestioneLinea` vi è una classe utente che contiene i metodi che svolgono le query. Nelle altre applicazioni questa classe non è stata implementata perché non si svolge mai due volte la stessa query in classe diverse.

Nell'applicazione `GestioneLinea` quando si modifica il database, se la modifica di quelli online si conclude con successo, viene modificato anche quello in locale: alla fine dell'operazione restano sincronizzati.

Per popolare la `ListView` contenuta nell'Activity `inserimentoOrari` dell'applicazione `GestioneLinea` viene invece utilizzato il `TimePickerDialog`. Questa finestra di dialogo permette di inserire un orario. Per semplificare l'inserimento di un elenco di orari in ordine crescente la finestra di dialogo ha come valore di default l'ultimo inserito, e non permette di inserire orari minori. Se d'altra parte l'ultimo inserito viene eliminato, il valore di default viene aggiornato.

Alla `ListView` che viene utilizzata per modificare la linea, sono associati due listener. Uno, `onLongClickListener`,

permette di eliminare la linea previa conferma; l'altro, il meglio conosciuto `onClickListener`, permette di modificare i parametri della linea desiderati.

Un'altra parte significativa del codice è la gestione della tecnologia NFC in android.

Per stabilire un collegamento p2p tra due applicazioni, una delle due (quella che invia il messaggio) deve implementare la classe `CreateNdefMessageCallback`. Grazie a questa classe è possibile scrivere codice all'interno del metodo `CreateNdefMessage`. Questo metodo si occupa di mandare il messaggio all'applicazione ricevente.

L'applicazione che riceve il messaggio deve contenere un intent filter ovvero un ascoltatore che apre l'applicazione controllore appena riceve un messaggio NFC. Una volta ricevuti tutti i parametri necessari, l'applicazione controllore può convalidare il biglietto dell'utente.

`CreateNdefMessageCallback` perché il collegamento avviene tra due dispositivi Android. Se questa connessione non fosse di questo tipo ma, tra un dispositivo ed un tag Nfc, non ci sarebbe bisogno di usare la classe `CallBack`.

Per stabilire il collegamento p2p, tra le due applicazioni, non serve altro che avvicinare i due dispositivi e confermare la volontà di inviare il messaggio.

Le altre due applicazioni, quali controllore e gestione linea, non contengono nuove classi che non siano già state spiegate parlando dell'applicazione utente.

Per maggiori informazioni sul codice scritto in android e su come esso è strutturato consultare i commenti scritti apposta all'interno di tutte le classi da noi utilizzate.

CODICE PHP

Tutto il nostro codice php è salvato online sui server di altermvista, insieme al database.

Ogni pagina web scritta da noi ha lo scopo di eseguire le query per caricare o prelevare dati dal server. Ogni pagina inoltre contiene una parte scritta in linguaggio html che consente un debug veloce senza la necessità di passare i parametri dalle applicazioni Android.

Inoltre tutte le pagine php restituiscono come risultato dei dati in formato Json che verranno successivamente elaborati dalla classe JsonParser.

Il codice php è quello che si occupa di fare tutti i controlli sui dati restituiti in modo da non dover fare più questo lavoro all'interno dell'applicazione android che, si deve già occupare della gestione grafica. In questo modo l'esecuzione della richiesta risulta anche più veloce.

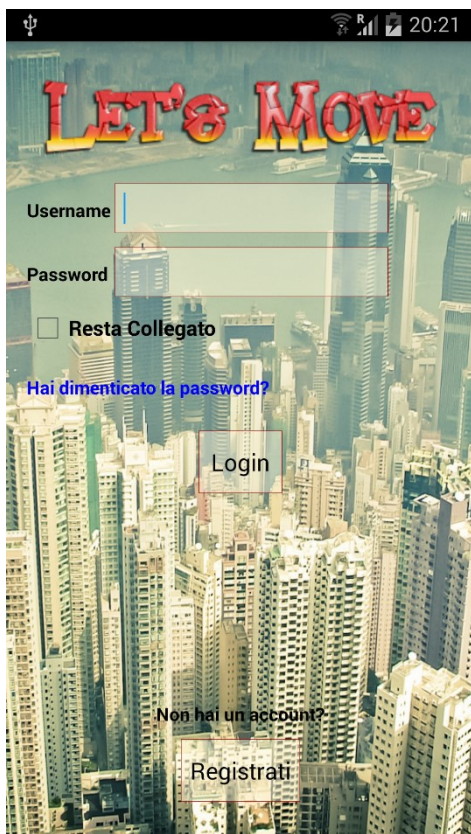
13. Manuale Utente



“Let’s Move”:

Il nome dell’applicazione era stato scelto quando ancora le idee erano poche e sparse, ma ora che il tutto è stato finito possiamo dire di aver scelto bene: con quel titolo si cerca di istigare le persone ad uscire, a girare; ed adesso grazie alla facilità di utilizzo della nostra applicazione tutti saranno più propensi a farlo.

Questa applicazione è divisa in 8 diverse schermate che ora andremo a mostrare e spiegarne le diverse funzionalità.



Questa sulla sinistra è la schermata che appare alla prima apertura dell’applicazione. Viene data la possibilità di effettuare un log-in se si è già in possesso di un account, o in caso contrario cliccando sul pulsante “Registrati” di andare a registrarsi.

La scritta in blu “Hai dimenticato la password?” permette, dopo aver inserito il proprio *Username* ed aver cliccato sulla scritta stessa, di ricevere sull’e-mail utilizzata alla registrazione una password nuova.

Se viene selezionato il *checkbox* “Resta collegato” una volta effettuato il log-in, anche se verrà chiusa l’applicazione una volta riaperta ci si troverà già loggati dentro.

Questa schermata la si ottiene quando, dalla schermata precedente viene premuto il pulsante “Registrati”.

Qui vengono richiesti, nei vari campi, tutti i dati che servono per garantire un po’ di sicurezza ed evitare inconvenienti sul pullman: il controllore possiede tutti i dati di ogni utente, quindi se il passeggero viene sorpreso senza il pagamento effettuato, per il riconoscimento basterebbe il codice del documento e si può risalire a tutto.

Registrazione

Codice Documento

Cognome

Nome

Nato il [aaaa-mm-gg]

Indirizzo

Località

Provincia

CAP

Username

Email

Password

Conferma Password

Registrazione

Ecco la schermata che troviamo quando si entra con il proprio account.



All'interno dell'applicazione le 3 schermate vengono gestite tramite l'utilizzo di *Tab Action Bar* e *Fragments*.

Questa schermata mostra, oltre che al nome dell'utente, quante corse ha ancora quest'ultimo o il tipo di abbonamento in possesso.

Sotto "Ultima Convalida" viene visualizzata l'ora dell'ultima convalida del biglietto. L'ora passata è quella del dispositivo del controllore per evitare che

qualcuno, modificando le ore sul proprio smartphone, porti errori nel database.

Invece, sotto a "Tempo Rimanente" viene visualizzata una *ProgressBar* che mostrerà graficamente quanto tempo manca al termine della validità del biglietto.

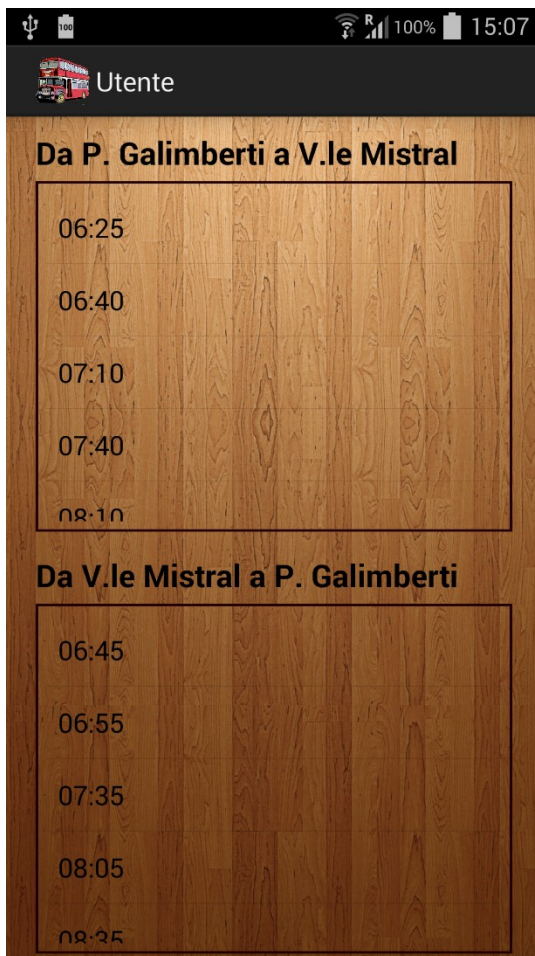
Infine, in fondo alla schermata, troviamo il numero di ore o giorni rimanenti alla scadenza del biglietto.

Questa seconda pagina contiene i nomi delle varie linee ed al loro interno il nome delle diverse fermate.

Al cliccare sui pulsanti “Orario” verranno aperti gli orari di partenza e di ritorno della prima e dell’ultima fermata: con città assai trafficate è usuale inserire solo l’orario di partenza e di ritorno, e non quella di ogni fermata.



Se nell’*EditText* con a fianco la lente si inserisce il nome di una fermata, o almeno una parte di essa, verranno nascoste tutte le linee che non contengono ciò che è stato ricercato: questo facilita tanto se si sa esattamente dove andare o in caso ci siano tante linee diverse.



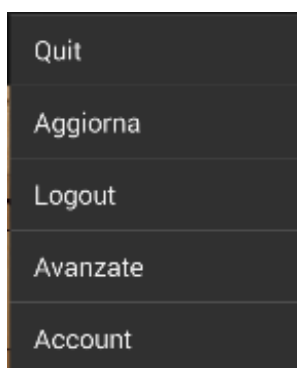
Questa è la schermata che comprare cliccando sul pulsante “Orario”. Nella parte superiore dello schermo viene visualizzata una lista di orari di partenza dalla prima fermata, mentre nella parte inferiore della schermata ci sono gli orari di ritorno. In questo modo la gestione degli orari risulta semplice e intuitiva per qualsiasi utente.



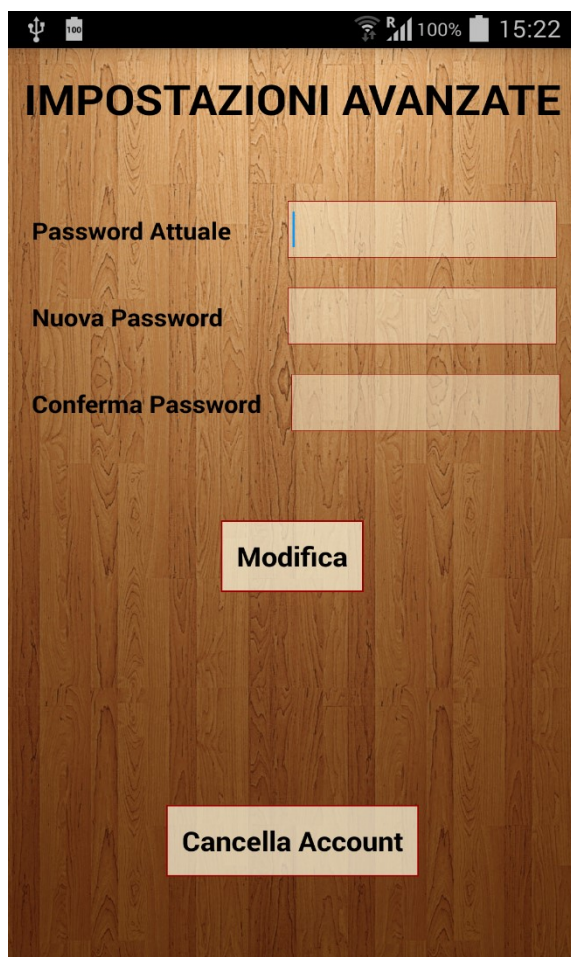
Questa è la terza schermata dove si ha la possibilità di acquistare il biglietto on-line tramite il dispositivo: la scelta, separata da quel segmento tratteggiato, si divide in quantità di corse o nella scelta di un abbonamento. Quest’ultimo parte al momento della prima convalida.

N.B. :

Per scelte di tempo e di difficoltà abbiamo sacrificato la parte della simulazione di pagamento, che vuole dire che al cliccare del bottone, dopo aver dato conferma dell'acquisto, non viene richiesta nessuna forma di pagamento, ma il biglietto viene acquistato.

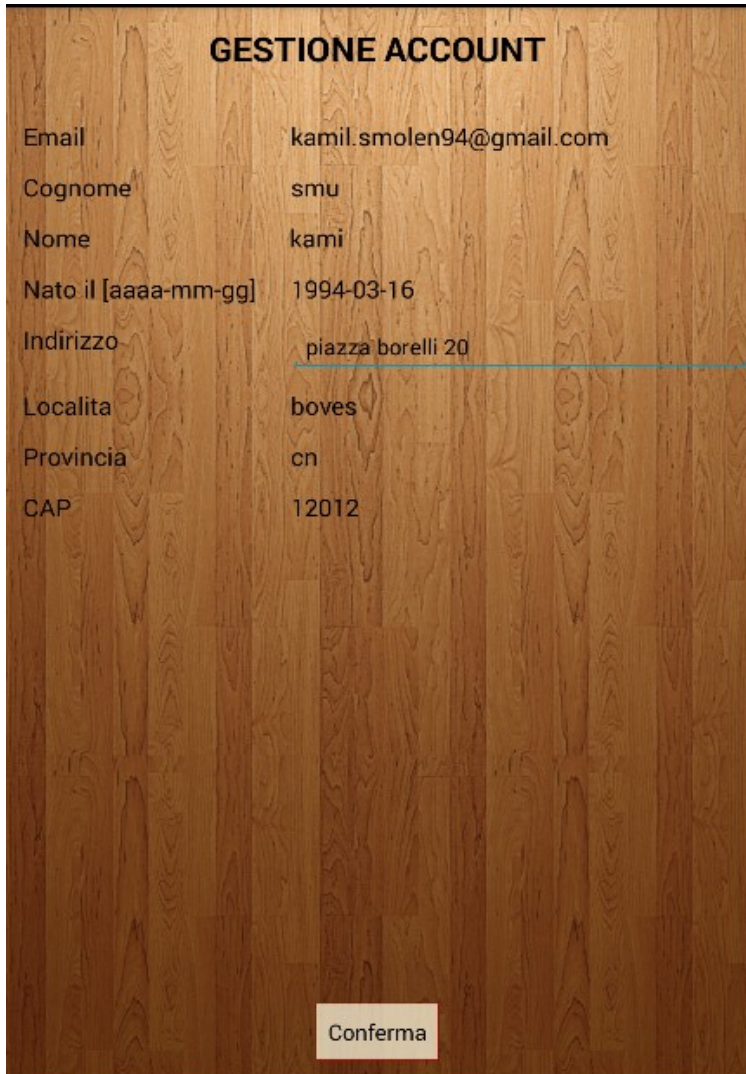


Cliccando sul tasto “opzioni” del proprio cellulare verrà data: la possibilità di uscire dal programma, di aggiornare il database interno, di effettuare il log-out, oppure di accedere a due sottomenu dell'applicazione.



Cliccando su “Avanzate” verrà aperta questa schermata, la quale ha lo scopo di permettere all'utente di accedere alle impostazioni avanzate del proprio account. Nella parte superiore della schermata, viene data all'utente la possibilità di cambiare la propria password, mentre nella parte inferiore,

clickando sul pulsante “Cancella Account” e confermando la propria scelta una seconda volta, l’utente ha la possibilità di cancellare il proprio profilo.



The screenshot shows a 'GESTIONE ACCOUNT' screen with a wooden background. It displays the following user information:

Email	kamil.smolen94@gmail.com
Cognome	smu
Nome	kami
Nato il [aaaa-mm-gg]	1994-03-16
Indirizzo	piazza borelli 20
Localita	boves
Provincia	cn
CAP	12012

At the bottom center, there is a button labeled 'Conferma'.

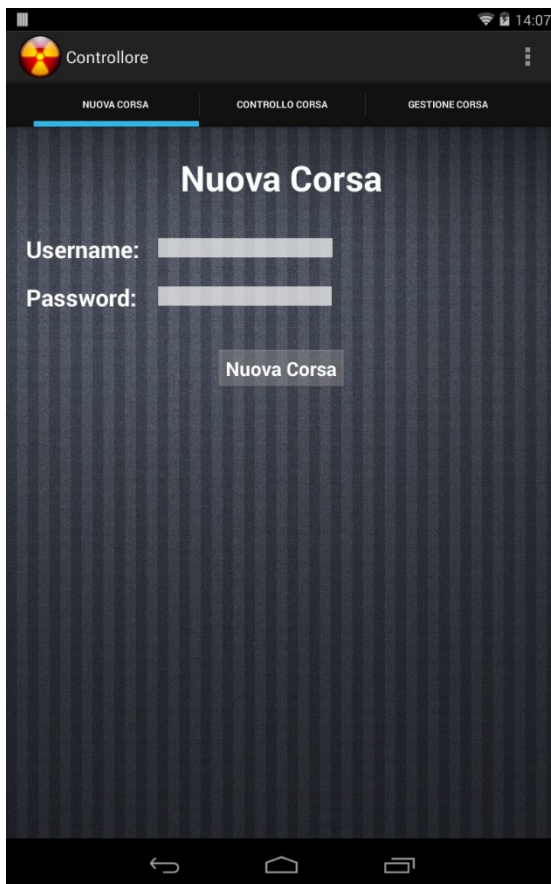
Quest’ultima schermata appare clickando su “Account”. Qui l’utente può controllare i dati inseriti al momento della registrazione: in caso ci sia la volontà di cambiare qualche dato, basta cliccare sulla riga contenente il dato stesso. La scritta verrà inserita all’interno di una “EditText”, permettendo all’utente di modificare il proprio

dato. Infine basta cliccare sul pulsante “Conferma” per salvare tutte le modifiche apportate ai dati personali.



“Controllore”:

Controllore è l'applicazione installata sui tablet posizionati sui mezzi di trasporto. La facilità di utilizzo di questa applicazione è notevole: la scelta è stata fatta pensando a del personale che potrebbe non essere qualificato nell'ambito tecnologico.

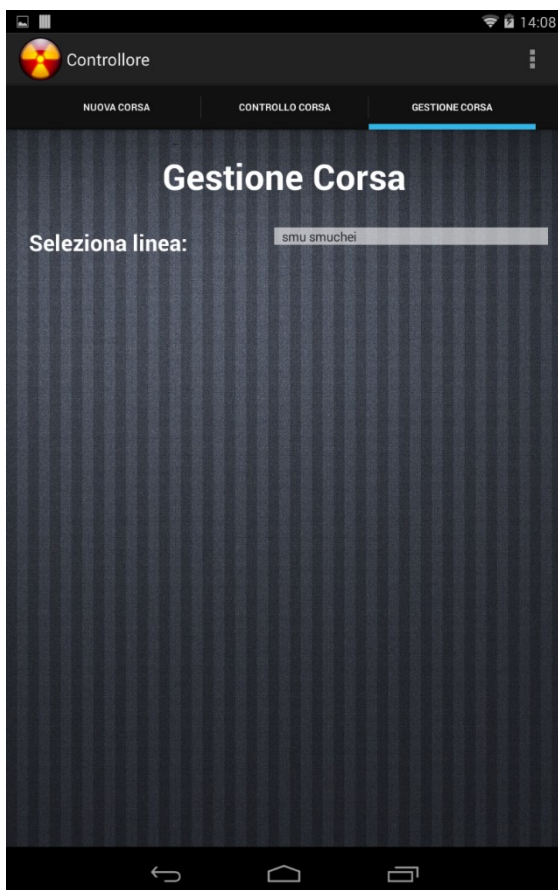
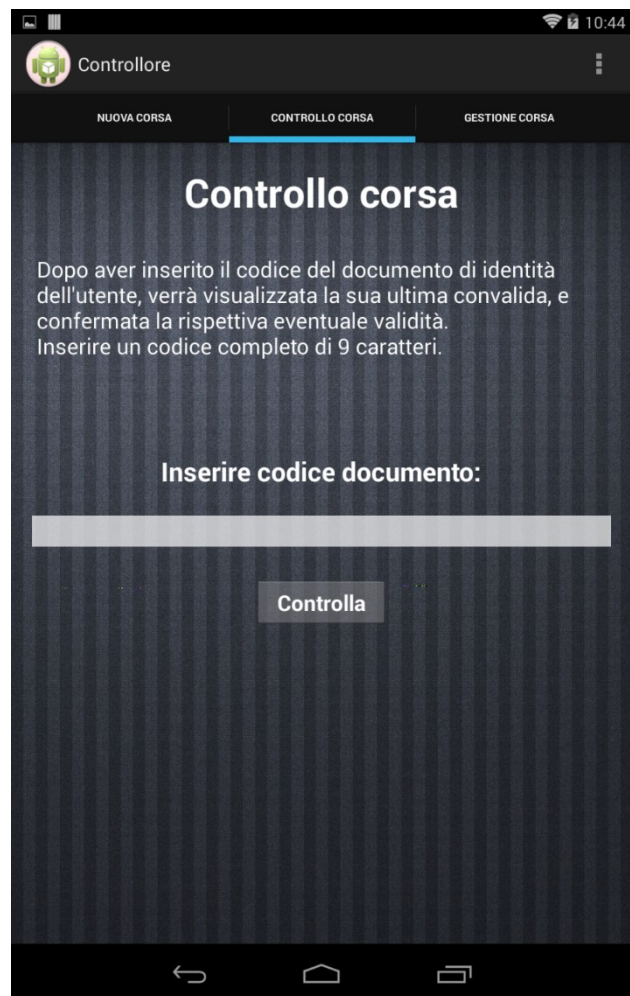


Se l'utente non può utilizzare il proprio dispositivo per effettuare la convalida del biglietto, può servirsi di questa schermata.

Questo permette al controllore di evitare scene come: *“Ho il cellulare scarico”* oppure *“Ho dimenticato il cellulare a casa”*.

La funzione della seconda schermata è simile alla precedente. Il controllore può verificare la validità del biglietto chiedendo soltanto il codice del documento all'utente.

Premendo il pulsante "Controlla", viene visualizzata la data e l'ora dell'ultima convalida insieme al tipo di abbonamento in possesso dal passeggero.



Questa ultima pagina deve essere modificata direttamente dall'autista alla partenza della

linea: una volta selezionata, al passare del cellulare gli utenti verranno registrati su quella linea.

Al premere del pulsante “menu” del dispositivo verranno visualizzate due alternative: “Esci” ed “Aggiornamento”, che non hanno bisogno di essere spiegate.

N.B. :

Quest'applicazione riceve tramite NFC l'username e la password dell'utente e fa la convalida del biglietto.

Infine, visualizza un Toast con l'esito della convalida.



“Gestione Linee”

Questa, terza ed ultima applicazione, permette la modifica del database da parte di qualche autorità, quale titolare dell’agenzia o tecnico di riferimento.

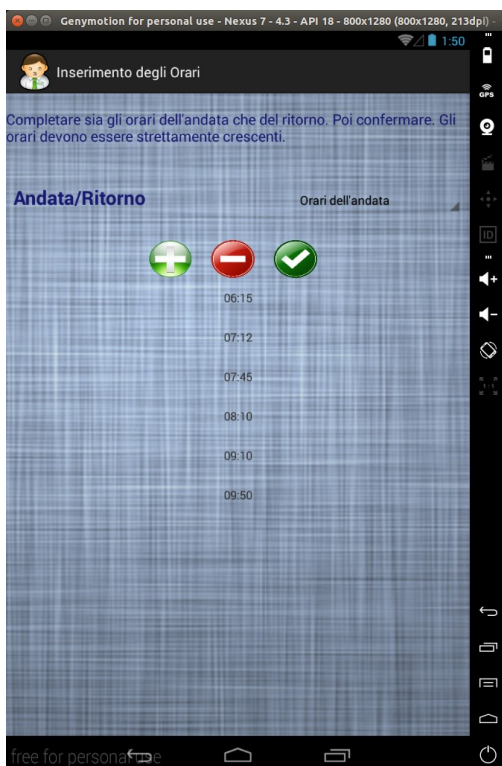
In “aggiungi linea” c’è la possibilità di inserire il nome della nuova linea ed il relativo codice: poi una ad una, si può andare ad inserire il nome delle fermate.

Una volta completata, cliccando il tasto rappresentante un “check” la linea viene aggiunta al database.

Fermate	
Nome della fermata	P. Torino - Cod: 2
Nome della fermata	V. Roma - Cod: 21
Nome della fermata	C. Nizza - Cod: 32
Nome della fermata	San Rocco - Cod: 24

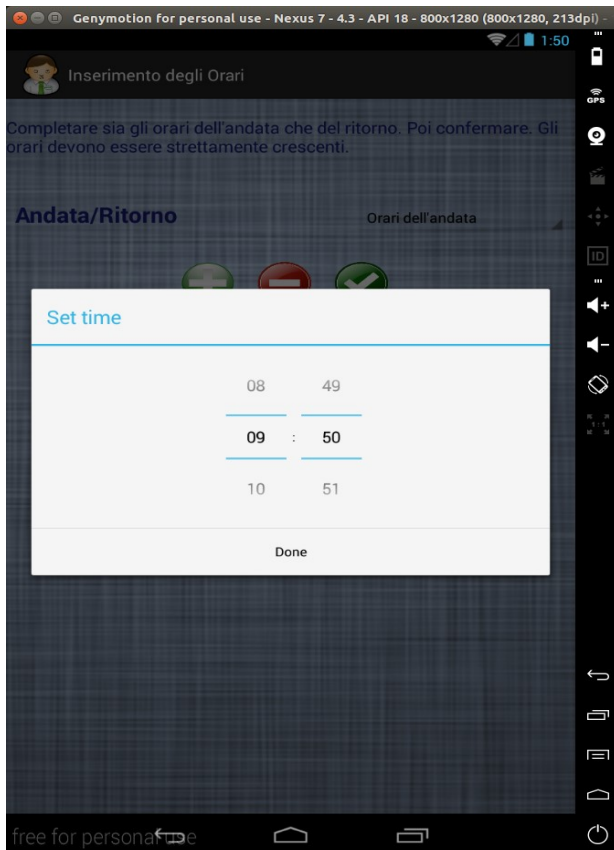


Per aggiungere le fermate alle linee nella pagina “Aggiungi Linea”, bisogna far sì che la fermata esista all’interno del database. Ciò è reso possibile in questa schermata: il testo inserito diventerà il nome della fermata.



Questa è la schermata che, all’aggiunta di una nuova linea, permette di inserirne gli orari.

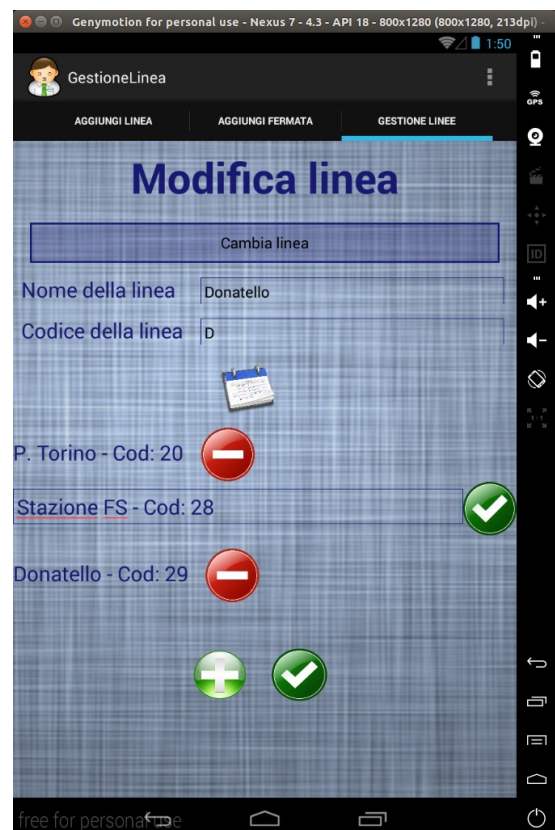
Il “più” aggiunge un nuovo orario, mentre il “meno” cancella l’ultimo inserito. Infine il “check” conferma le modifiche effettuate.

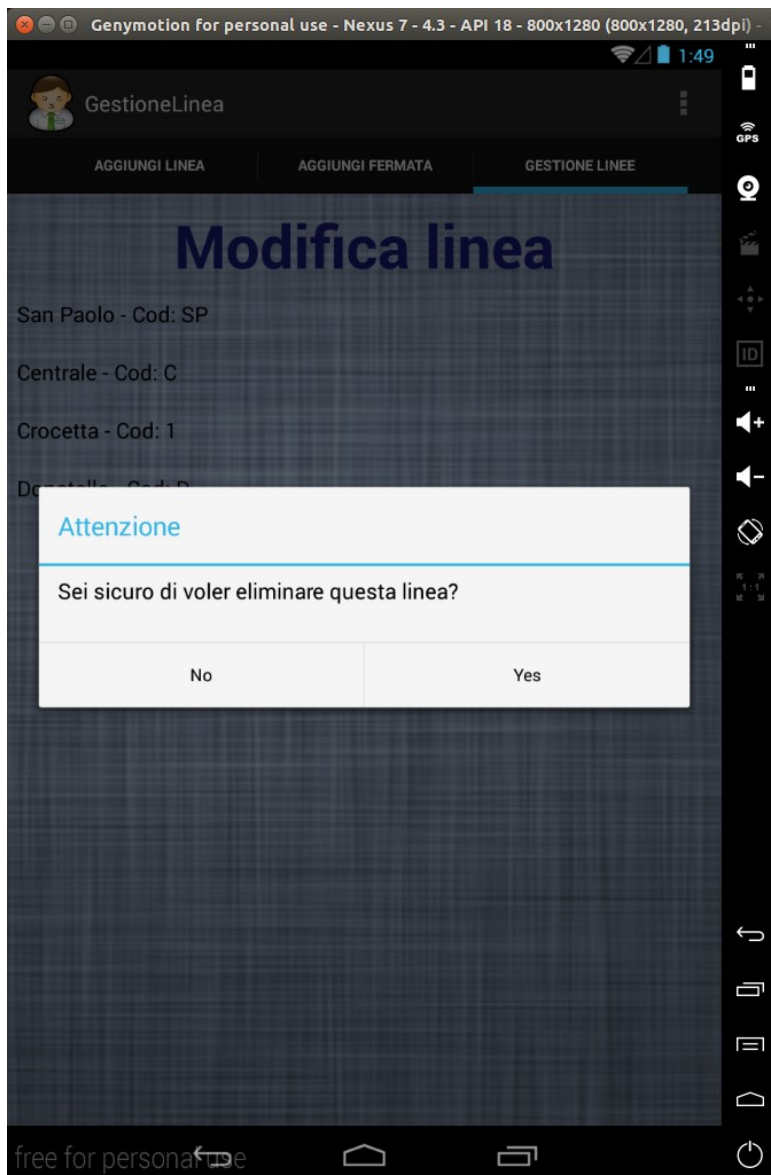


Una volta cliccato sul “più” verrà aperta questa schermata che permette di inserire, intuitivamente, l’orario della fermata.

La schermata “Modifica Linea” permette, a chi si occupa di gestione, di modificare le linee già esistenti, scegliendole tra quelle salvate nel database. Per eliminare una fermata bisogna cliccare sul pulsante rosso, mentre per modificarne una, non serve altro che cliccare sulla fermata stessa.

Alla fine di tutte le modifiche bisogna soltanto confermare il tutto cliccando sul pulsante “check”.





Quest'ultima schermata rappresenta la finestra di conferma per eliminare una linea dal database. Viene chiesto all'utente se è davvero sicuro di farlo. Una volta ottenuta la conferma la linea viene eliminata. L'accesso a questa schermata si ottiene dal m