UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

PROGETTO E SVILUPPO DI UN'APPLICAZIONE ANDROID PER IL MONITORAGGIO DELL'EROGAZIONE DELLA TERAPIA FARMACOLOGICA

DESIGN AND DEVELOPMENT OF AN ANDROID APPLICATION FOR MONITORING THE SUPPLY OF PHARMACOLOGICAL THERAPY

Relatore:

Prof. Ennio Gambi

Tesi di:

Claudia Molinelli

Correlatore:

Prof. Adelmo De Santis

Anno accademico 2019/2020

Alla mia famiglia

SOMMARIO

ELENCO DELLE FIGURE

1 – INTRODUZIONE

Al giorno d'oggi, con una società che si basa sempre più sulla tecnologia, aumenta progressivamente l'utilizzo di smartphone, che risultano molto comodi in quanto permettono di svolgere più funzionalità tramite l'utilizzo di applicazioni in esso installate.
In informatica, con il termine applicazione si indica un programma, un software o una serie di programmi che sono in fase di esecuzione, con lo scopo di erogare un servizio utile ad un qualsiasi utente.
Combinando il risultato di un software alle risorse hardware e all'interazione di un utente, l'applicazione crea un output di qualsiasi genere, in base allo scopo per cui è stata creata.

1.1 – Obiettivo del lavoro di tesi

L'obiettivo del seguente lavoro di tesi consiste nella progettazione e nello sviluppo di un'applicazione destinata al monitoraggio dell'erogazione della terapia farmacologica di pazienti.
In altre parole, un paziente tramite l'utilizzo dell'applicazione, scannerizzando il codice a barre di un farmaco riceve delle indicazioni su di esso.
In particolare, vengono fornite informazioni sulla correttezza del farmaco e dell'orario di assunzione e sulle dosi prescritte nella terapia.
La piattaforma utilizzata per tale lavoro è Mit App Inventor, un ambiente di sviluppo per applicazioni Android creato da Google, ma ora proprietà del Massachusetts Institute of Technology.

1.2 – Tecnologia utilizzata

I codici a barre furono ideati nel 1948 da Norman Joseph Woodland e Bernard Silver, due studenti di ingegneria dell'università di Drexel, in seguito alla richiesta da parte del presidente di un'azienda del settore alimentare di automatizzare e rendere più veloci le operazioni di cassa.
Si possono distinguere due tipi di codice a barre: i codici lineari (unidimensionali) e i codici bidimensionali.
All'inizio si era pensato di utilizzare il codice Morse stampato ed esteso in senso verticale, così da formare delle barre di spessore diverso. Successivamente si iniziarono ad usare anche codici a barre ovali, ma questa idea fu poco utilizzata.

Un primo tentativo di lettura dei codici a barre venne fatto mediante l'utilizzo di un fotomoltiplicatore, originariamente utilizzato per la lettura ottica delle bande audio dei film, ma presto abbandonato a causa dell'eccessivo rumore, del calore e del peso dovuto ai suoi componenti.
Successivamente, grazie allo sviluppo della tecnologica laser, la costruzione dei lettori di codici a barre fu resa più economica e accessibile.

1.2.1 – Codici a barre in ambito farmaceutico

Attualmente, per la codifica dei prodotti farmaceutici, il Ministero della Sanità italiano utilizza il Farmacode, anche detto Code 32.
È una rielaborazione del Code 39, un codice a barre che permette la rappresentazione di 43 caratteri, comprendendo le 26 lettere maiuscole dell'alfabeto (dalla A alla Z), i caratteri numerici (da 0 a 9) e alcuni caratteri speciali, compreso lo spazio.
Il Farmacode fa parte della famiglia dei codici a barre bidimensionali ed è composto da due tipi di barre: strette e larghe. Lo spessore di tali barre può essere di 0,250 o 0.254 mm.

Figura 1-Esempio di Farmacode

È composto da 4 sezioni:

- Carattere di start;
- Codifica dei dati;
- Codice di controllo (check-digit);
- Carattere di stop.

Il carattere di start/stop, in genere, è l'asterisco (*) e serve per delimitare il codice stesso.
L'ultimo carattere rappresenta il codice di controllo.
Il resto è formato da 9 caratteri in base 32 con codifica binaria: questi caratteri corrispondono ad altrettanti caratteri in base 10, dei quali il primo è 0, poi seguono 7 caratteri che identificano il prodotto.

La codifica in base 32 utilizza le lettere dell'alfabeto anglosassone (escludendo le lettere A, E, I, O) e 10 numeri decimali (da 0 a 9), ottenendo la corrispondenza con la base 10 tramite la tabella che segue.

Tabella di conversione:

Carattere in base 32	Valore in base 10		Carattere in base 32	Valore in base 10
0	0		J	16
1	1		K	17
2	2		L	18
3	3		M	19
4	4		N	20
5	5		P	21
6	6		Q	22
7	7		R	23
8	8		S	24
9	9		T	25
B	10		U	26
C	11		V	27
D	12		W	28
F	13		X	29
G	14		Y	30
H	15		Z	31

Figura 2-Tabella di conversione

Il codice di controllo si calcola seguendo determinate operazioni:

- Siano c1, c2, c3, c4, c5, c6, c7, c8 le prime 8 cifre in base 10 a sinistra del codice;
- Si determinano i seguenti prodotti:
    1. x1=2*c2
    2. x2=2*c4
    3. x3=2*c6
    4. x4=2*c8
- Si calcola la somma S1 dei quozienti e dei resti ottenuti dividendo x1, x2, x3 e x4 per 10;
- Si calcola la somma S2 delle cifre c1, c3, c5 e c7 del codice;
- Si calcola la somma S=S1+S2;
- La cifra di controllo è il resto della divisione di S per 10.

1.3 – Struttura della tesi

La presentazione del lavoro effettuato si suddivide nelle seguenti parti:

- Il secondo capitolo introdurrà la piattaforma utilizzata per la realizzazione dell'applicazione, descrivendone le caratteristiche;
- Nel terzo capitolo verrà illustrato nel dettaglio il progetto, mostrando anche degli esempi di comportamento;
- Infine, il quarto capitolo concluderà il lavoro di tesi con delle considerazioni sulla tecnologia utilizzata e illustrerà i possibili sviluppi futuri dell'applicazione;

2 - DESCRIZIONE DELLA PIATTAFORMA

2.1 – Generalità

Nel seguente capitolo è illustrato l'ambiente di programmazione Mit App Inventor utilizzato per la realizzazione dell'applicazione.
Mit App Inventor consente di realizzare applicazioni per Android in modo molto più semplice e intuitivo rispetto alla programmazione classica in Java.
Molti ambienti di sviluppo sono in grado di realizzare applicazioni compatibili sia con il sistema operativo Android che con iOS. Diversamente, Mit App Inventor si propone come un software unicamente dedicato alla piattaforma Android, ma nonostante questo limite ha comunque ottenuto un grande successo ed è attualmente utilizzato da milioni di utenti.
Nonostante la sua semplicità, Mit App Inventor è un'opportunità anche per i programmatori più esperti, grazie alla potenza dell'interfaccia grafica che consente di risparmiare tempo e di velocizzare alcuni processi di sviluppo attraverso l'integrazione della libreria Java Open Blocks.

2.2 – Architettura di Mit App Inventor

Questo ambiente di sviluppo si compone di due schermate principali, che sono quelle in cui l'utente andrà a lavorare:

- Designer View: nella parte sinistra di questa schermata c'è la Components Palette, dove si trovano tutti i componenti, divisi per categoria, necessari per la realizzazione dell'applicazione e soprattutto dell'interfaccia grafica: in questa schermata si sviluppa il layout dell'applicazione.
  Essi permettono inoltre di gestire le periferiche dello Smartphone; infatti consentono l'interfacciamento con i social, la gestione della memoria e della connettività, ecc.
  Al centro si trova la sezione che rappresenta l'area di lavoro (Viewer) in cui si progetta il layout; infatti è rappresentata la sagoma dello schermo dello smartphone a cui è collegato.
  Dalla Components View è possibile gestire le proprietà dei componenti ed è divisa in due parti: la sezione Components, da cui si può rinominare o eliminare gli elementi e la sezione Properties, dalla quale è possibile accedere alle vere e proprie proprietà del componente (colore, font ecc.).

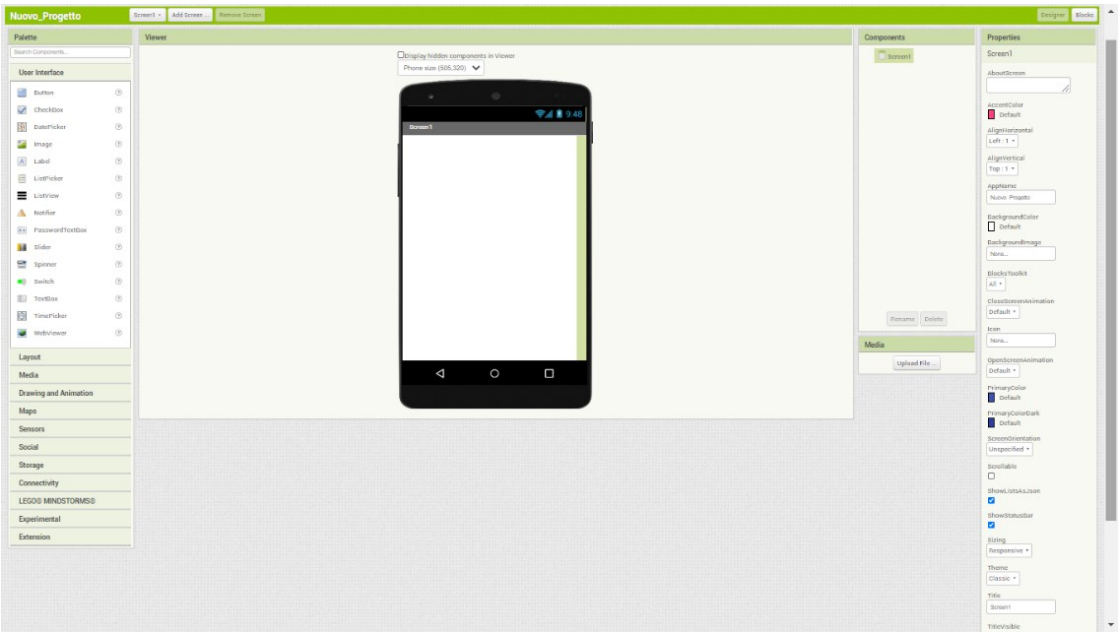


Figura 3-Esempio di Designer View

- Blocks View: dove si costruisce il codice vero e proprio.

  A sinistra dello schermo, nella Blocks Palette, ci sono i blocchi per l'implementazione dell'applicazione. Questa parte è divisa in due sezioni: la prima (Built-in) che contiene i blocchi di base, come quelli per il controllo dell'esecuzione, i blocchi logici e matematici e tutte le varie operazioni di base. La seconda sezione invece contiene i blocchi specifici delle varie componenti scelte in precedenza nella Designer View.

  Selezionando qualsiasi componente si apre un menù contenente tutti i rispettivi blocchi, ognuno con una funzione diversa.

  I blocchi possono essere incastrati tra di loro nella Blocks View come se fossero pezzi di un puzzle e possono essere raggruppati in funzioni, le quali possono essere richiamate più volte senza dover ripetere gli stessi blocchi.

  Le operazioni all'interno delle funzioni vengono eseguite consecutivamente, una alla volta, dall'alto verso il basso.

  Infine è possibile salvare e conservare i blocchi trascinandoli in alto a destra nel Backpack, che è in comune tra i vari progetti.

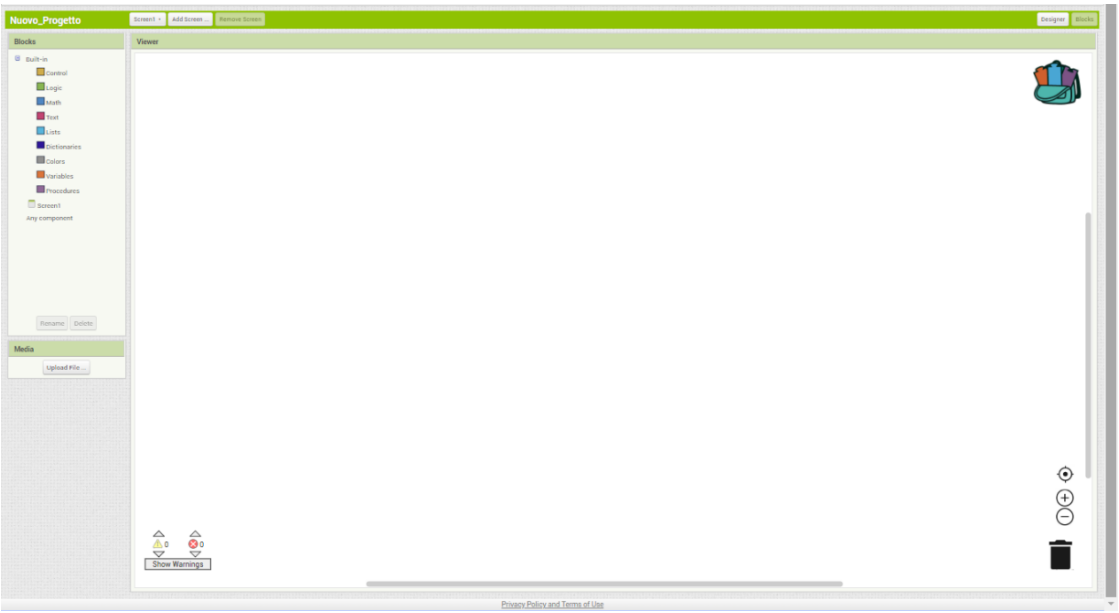


Figura 4-Esempio di Block View

2.3 – Debug e emulazione

Per poter testare l'applicazione ci sono tre modalità:

- test in tempo reale su un dispositivo Android attraverso la connessione WiFi;
- test in tempo reale tramite l'utilizzo di un emulatore direttamente dal computer;
- test in tempo reale su un dispositivo Android attraverso una connessione USB;

Il terzo metodo non è altro che un sottoinsieme del primo.

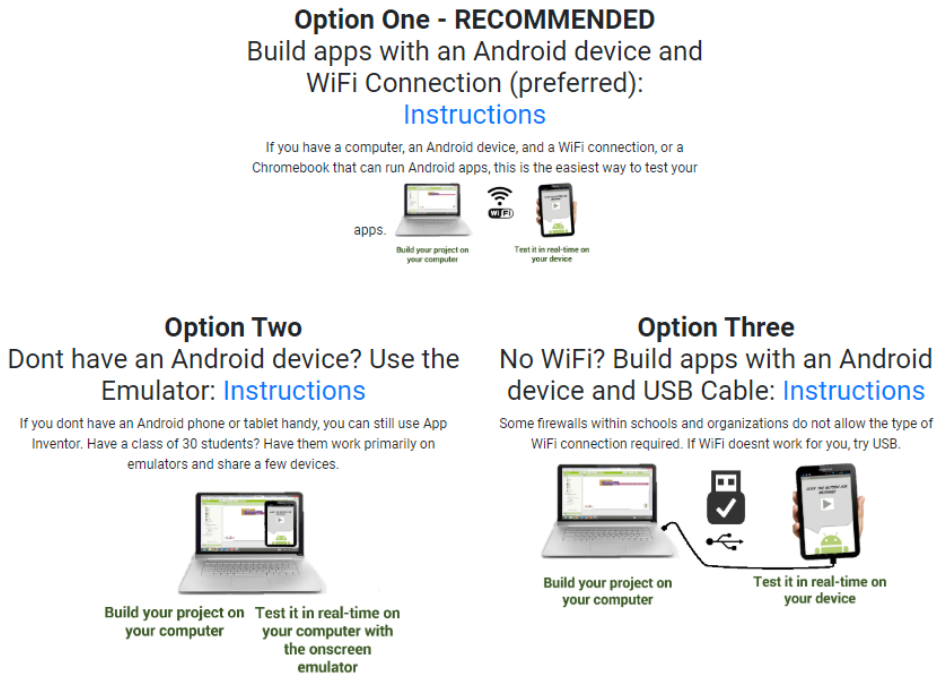


Figura 5-Metodi per testare l'app

2.3.1 – Test in real-time su device Android con WiFi

Per quanto riguarda la prima opzione, è necessario installare sul proprio dispositivo l'applicazione "MIT AI2 Companion", reperibile dal Play Store e connettere il computer e il dispositivo alla stessa rete WiFi.

Dalla Main Bar, la barra principale dove è possibile gestire il progetto, cliccando su "Connect" e successivamente su "AI Companion", apparirà una finestra di dialogo contenente un QR code da scannerizzare dal dispositivo tramite l'applicazione installata in precedenza.

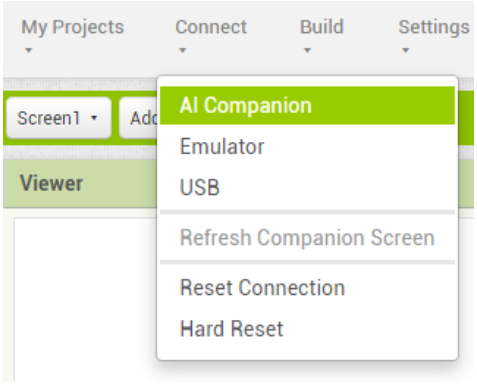


Figura 6-Finestra di dialogo

Una volta collegato, sul dispositivo sarà visualizzata l'app che si sta progettando, aggiornandosi ogni qual volta avvenga una modifica su di essa.

Figura 7-QR code

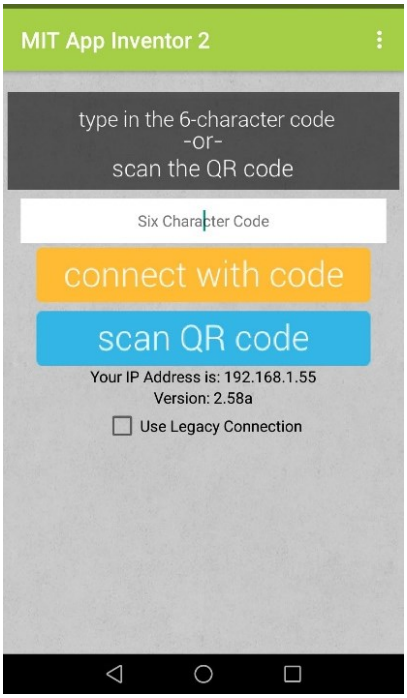


Figura 8-Visualizzazione sul dispositivo

2.3.2 – Test in real-time con emulatore

Attraverso un emulatore è possibile testare l'applicazione senza l'utilizzo di un dispositivo aggiuntivo, ma direttamente sul computer in cui si sta lavorando.
Per prima cosa è necessario installare il programma aiStarter, l'helper che consente al browser di comunicare con l'emulatore.
In seguito, dalla Main Bar, occorre cliccare "Connect" e successivamente "Emulator". Una volta connesso, l'emulatore verrà avviato e mostrerà l'app aperta in App Inventor.

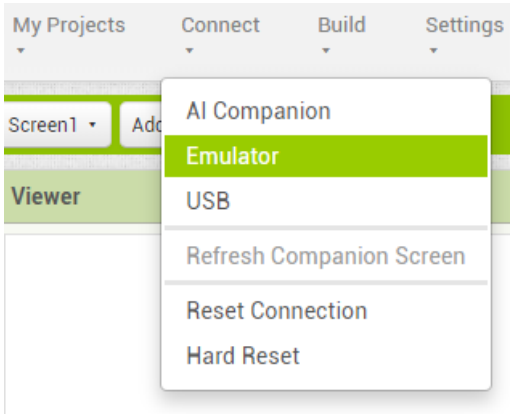


Figura 9-Finestra di dialogo

Figura 10-Emulatore

2.3.3 – Installazione app su device e pubblicazione su Google Play Store

Una volta terminata l'app e testata attraverso i metodi precedentemente citati, è possibile scaricarla sul proprio dispositivo.
Per farlo è necessario eseguire il build dell'app in formato .apk e installarlo sul proprio smartphone o tablet (Android). Per utilizzare questa modalità occorre impostare il dispositivo in modo da permettere l'installazione di file non provenienti da Google Play Store.
Mit App Inventor mette a disposizione la possibilità di pubblicare le app su Google Play, facendole diventare parte dell'archivio globale di app per sistemi operativi Android.

Per prima cosa bisogna attribuire all'app un VersionCode, un numero intero che indica la versione dell'app stessa e di un VersionName, un nome generico, anche se si è soliti scegliere un numero decimale.
Ogni volta che ci sarà un aggiornamento dell'app su Google Play, VersionCode e VersionName verranno modificati, in particolare il primo verrà incrementato.
Anche in questo caso si utilizza l'estensione .apk del file dell'applicazione.
Le procedure da seguire sono:

- Collegarsi al seguente link: https://play.google.com/apps/publish/signup;
- Accedere con il proprio account Google;
- Completare la registrazione alla piattaforma;
- Pagare la quota di iscrizione di 25 dollari, per essere abilitato al ruolo di sviluppatore Google.

Una volta effettuati questi passaggi si avrà la possibilità di caricare l'applicazione nel Play Store e in seguito occorrerà compilare una dettagliata scheda tecnica dell'app, certificando i contenuti trattati e definendo poi il prezzo, la lingua, la categoria di appartenenza e molto altro.
Ultimata la compilazione ed effettuato il caricamento si dovrà aspettare la verifica di idoneità da parte di Google, e nel caso di esito positivo dopo qualche giorno sarà possibile visionare l'applicazione nello Store.

2.4 – Variabili

Uno degli aspetti di maggiore importanza di un linguaggio di programmazione è la gestione delle strutture dati, ovvero le tipologie di dati supportate e le operazioni che possono essere eseguite su di esse.
In Mit App Inventor la gestione delle strutture dati è semplificata: infatti le variabili possono essere globali e locali, mentre i tipi di dati supportati sono booleani, numerici, stringhe, liste e colori.

- Per definire una variabile globale è sufficiente andare a sinistra dello schermo su "Variables" e prelevare il blocco "initialize global name to" rinominando la sezione "name" con il nome che si desidera abbia la variabile e inizializzandola.

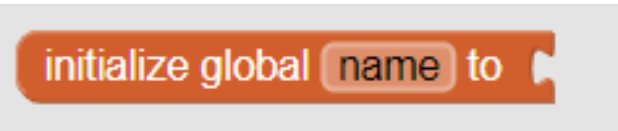


Figura 11-Variabile globale

- Le variabili locali invece, sono limitate ad una specifica procedura. Esse possono essere definite tramite due blocchi distinti; la differenza sostanziale sta nel modo in cui il blocco

vuole essere utilizzato. Uno dei blocchi può essere utilizzato nella sezione do di un blocco superiore (figura 12), mentre l'altro può essere collocato nella sezione return di un blocco collegato (figura 13).

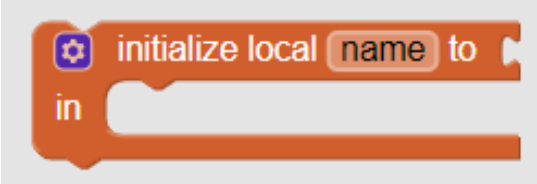


Figura 12- Variabile locale 1

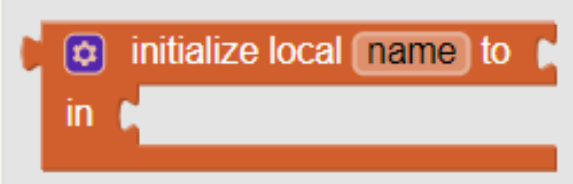


Figura 13-Variabile locale 2

2.5 – Descrizione dei componenti utilizzati

Tra i tanti componenti di cui Mit App Inventor è fornito, in questo paragrafo verranno illustrati quelli utili alla realizzazione dell'applicazione.

Per quanto riguarda la parte che si interfaccia con l'utente, i componenti sono i seguenti:

- Button: è un pulsante in grado di rilevare quando viene cliccato. Oltre alle proprietà di base (colore, font, testo, visibilità ecc), c'è la possibilità di renderlo attivo (Enabled) o meno.
- Label: l'etichetta è un componente utilizzato per mostrare il testo specificato nella proprietà Text.
- List View: è un componente che consente di visualizzare un elenco di elementi. Questo elenco può essere impostato usando la proprietà "ElementsFromString" o "Elements".
- TextBox: è una casella di testo dove l'utente può scrivere. Attraverso la proprietà Hint è possibile fornire all'utente un suggerimento sull'argomento da digitare. La TextBox può essere limitata al caso di soli input numerici impostandola su "NumbersOnly".
  Questa componente è in genere utilizzata insieme alla componente Button.
- PasswordTextBox: è un caso particolare del componente precedente in cui il testo che è stato digitato all'interno della casella è nascosto.

Per la gestione del layout dell'applicazione sono stati adottati due tipi di disposizione:

- Horizontal Arrangement: permette di disporre i componenti in orizzontale, da sinistra verso destra.
- Table Arrangement: permette di disporre gli elementi in forma tabulare scegliendo a priori il numero di righe e di colonne.

I sensori utilizzati sono:

- Barcode Scanner: è un elemento non visibile che permette di codificare il codice a barre o QR code restituendo come risultato una stringa. Questo componente permette di scegliere se usare o meno uno scanner esterno.
- Clock: è un componente non visibile che fornisce l'istante di tempo utilizzando l'orologio interno del dispositivo. Può far scattare un timer a intervalli regolari ed eseguire calcoli e conversioni di tempo: infatti sono disponibili dei metodi per convertire un instante di tempo in un testo.

Infine, per quando riguarda lo storage, ovvero la memorizzazione dei dati, si è utilizzato il TinyDB, un componente non visibile che memorizza i dati in maniera persistente sul dispositivo rendendoli disponibili ogni volta che viene eseguita l'applicazione. Il funzionamento del TinyDB si avvale di una coppia Tag/Value, dove Tag è il nome univoco con cui vengono identificati i dati, mentre Value è il valore vero e proprio che si vuole salvare.

Nel caso in cui venga letto un Tag inesistente o al quale non è associato nessun Value, viene restituita una stringa di lunghezza zero.

Inoltre può esserci un solo archivio per ogni applicazione, quindi anche se si utilizza più di un TinyDB, essi usano lo stesso spazio e gli stessi Tag/Value. Per ottenere archivi diversi occorre utilizzare nomi diversi per i vari Tag.

Infine, con questo componente non è possibile passare dati fra due applicazioni diverse anche se sono sullo stesso dispositivo.

3 - DESCRIZIONE DEL PROGETTO

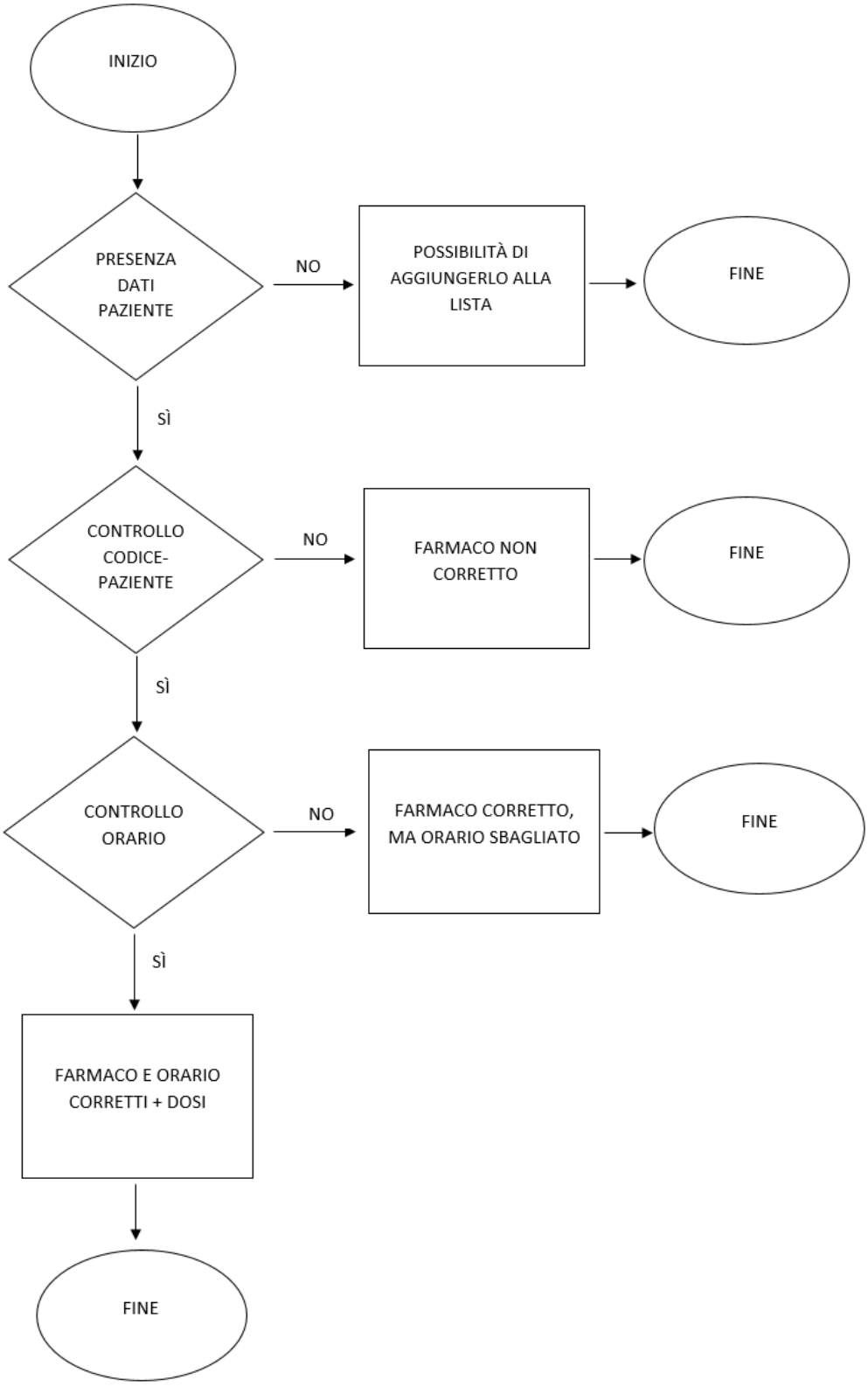
Il seguente capitolo mostra nel dettaglio il progetto sviluppato. In particolare verranno illustrati singolarmente i vari blocchi di codice e le loro funzionalità.

3.1 – Panoramica del progetto

L'applicazione progettata nasce dall'esigenza di rendere più agevole e veloce l'assunzione di farmaci prescritti nella terapia farmacologica di un paziente.
La funzionalità di base dell'applicazione è quella di leggere il codice a barre presente in un medicinale e di fornirne l'esito.
Per prima cosa viene chiesto all'utente di inserire i propri dati personali, in particolare il nome e il cognome e successivamente di scannerizzare il codice a barre del farmaco in questione.
Quindi, i dati che vengono acquisiti dall'applicazione sono i seguenti: nome, cognome, codice a barre e orario in cui viene acquisito il codice.
Il primo controllo che viene fatto dall'applicazione è quello di verificare se l'utente sia presente o meno nella lista dei pazienti, memorizzata in un archivio tramite il TinyDB e alla quale si può accedere mediante una password.
A questo punto, si possono avere due casi in base all'esito:

- Esito positivo: il paziente è presente nella lista. Vengono fornite le informazioni sul farmaco scannerizzato, andando a ricercare all'interno della lista se il codice a barre in questione è associato al paziente stesso e, in caso affermativo, viene poi verificato se l'orario di scansione è compreso nella fascia oraria di assunzione del farmaco.
  Infine, se le verifiche effettuate danno tutte un esito positivo, l'utente visualizzerà anche le dosi prescritte.
- Esito negativo: il paziente non è presente nella lista. In questo caso l'applicazione dà l'opportunità di aggiungere nuovi dati alla lista.

Rappresentazione grafica:



3.2 – Analisi del codice

In questo paragrafo vengono riportati tutti i blocchi utilizzati per programmare l'applicazione con le caratteristiche descritte in precedenza; di ognuno vengono spiegate le loro funzioni e l'ordine logico con cui vengono utilizzati.

Il codice è stato sviluppando suddividendo l'applicazione in 4 schermate (screen)**.**

La piattaforma Mit App Inventor infatti, dà la possibilità di creare diverse schermate e di lavorarci in modo indipendente, aggiungendo ogni volta i componenti utili a quella parte di lavoro.

Per quanto riguarda le variabili invece, sono tutte di tipo globale e definite per ogni schermata.

3.2.1 – Screen 1

La prima schermata è quella di interazione con l'utente, dove inserisce nelle apposite caselle di testo il proprio nome e cognome.

Successivamente, attraverso il pulsante "SCANNERIZZA", l'applicazione aprirà lo scanner del dispositivo in cui è installata e acquisirà il codice a barre. Dopodiché si aprirà la seconda schermata.

Dallo screen 1 è inoltre possibile accedere alla lista in cui sono registrati i dati dei pazienti, tramite una password. Dopo aver cliccato sul pulsante "Vedi lista" si aprirà la quarta schermata con le relative informazioni.

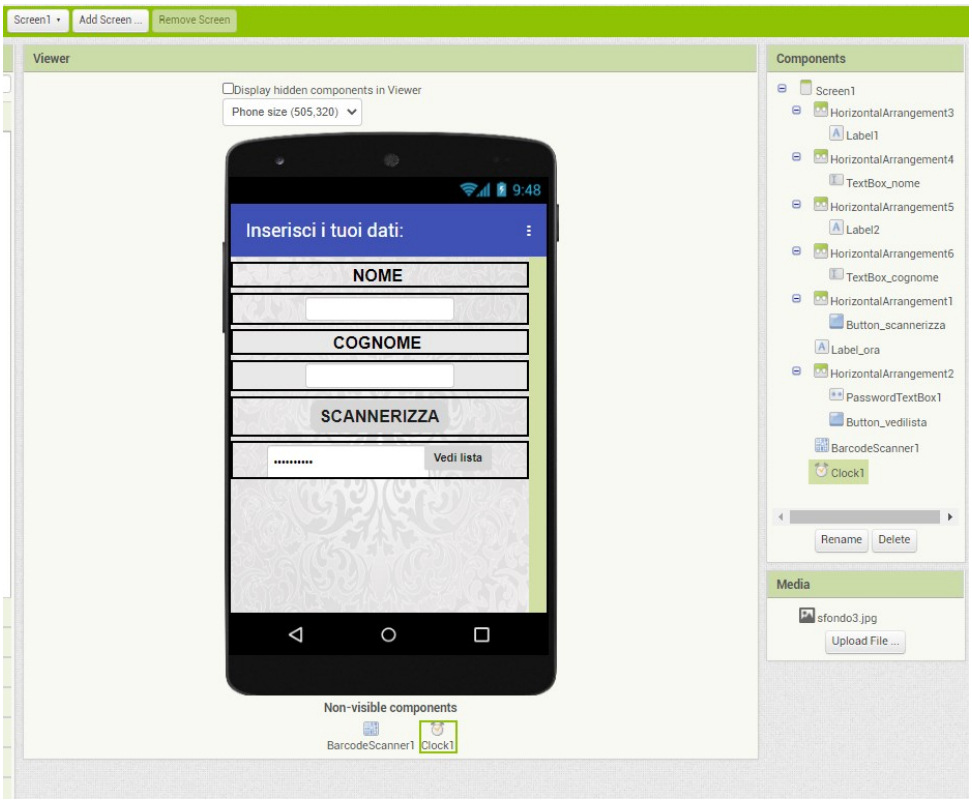


Figura 12-Screen 1

3.2.1.1 – Scannerizzazione

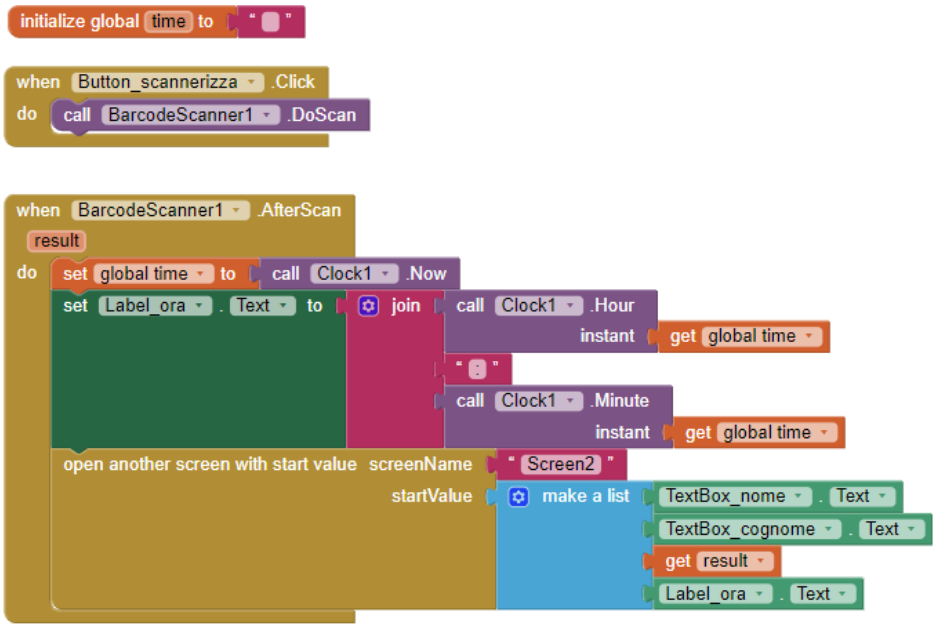


Figura 13-Blocco per scannerizzazione

La prima operazione svolta dall'applicazione quando viene avviata è quella di inizializzare la variabile globale "time" come una stringa vuota.

Il primo blocco illustra il comportamento del pulsante Button_scannerizza dopo essere stato premuto: infatti viene eseguita una chiamata al BarcodeScanner1 e si apre lo scanner del dispositivo.

Il secondo blocco invece, illustra che cosa accade dopo aver effettuato la scannerizzazione del codice a barre.

In ordine avvengono le seguenti operazioni:

- Viene attribuito alla variabile "time" l'istante di tempo del dispositivo.
- All'etichetta Label_ora, che è stata precedentemente impostata a non visibile, viene assegnato l'istante temporale indicando l'ora e i minuti.
- Si apre la schermata 2 passando i valori acquisiti da una schermata all'altra e organizzandoli in una lista. In particolare, i valori passati sono: il nome e il cognome inseriti dall'utente nelle apposite caselle di testo, il codice a barre acquisito e il valore dell'etichetta Label_ora.

3.2.1.2 – Lista

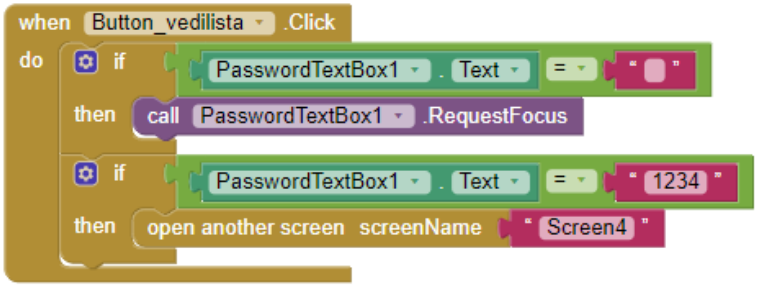


Figura 14-Blocco per visualizzazione lista

Come già anticipato in precedenza, da questa schermata è possibile accedere alla lista dei pazienti che si trova nell'ultima schermata dell'applicazione.

Per fare ciò occorre cliccare sul pulsante Button_vedilista.

In questo blocco di codice sono presenti due blocchi condizionali, entrambi eseguono un confronto tra il testo inserito nella PasswordTextBox1 e una stringa predefinita. In particolare:

- Il primo confronto è con una stringa vuota: se non è ancora stata inserita nessuna password, l'attenzione viene posta sulla casella di testo.
- Il secondo confronto invece, viene fatto con la vera password: se le stringhe coincidono allora si apre la schermata 4.

Figura 15-Screen 1 su smartphone

3.2.2 – Screen 2

In questa schermata viene fatto un riepilogo dei dati, visualizzando quindi i valori che vengono ricevuti dalla schermata precedente.

In questo modo l'utente può accorgersi immediatamente se i dati che ha inserito sono corretti o meno e, nel caso in cui non lo fossero, tramite il pulsante "INDIETRO" può tornare alla schermata precedente per modificarli.

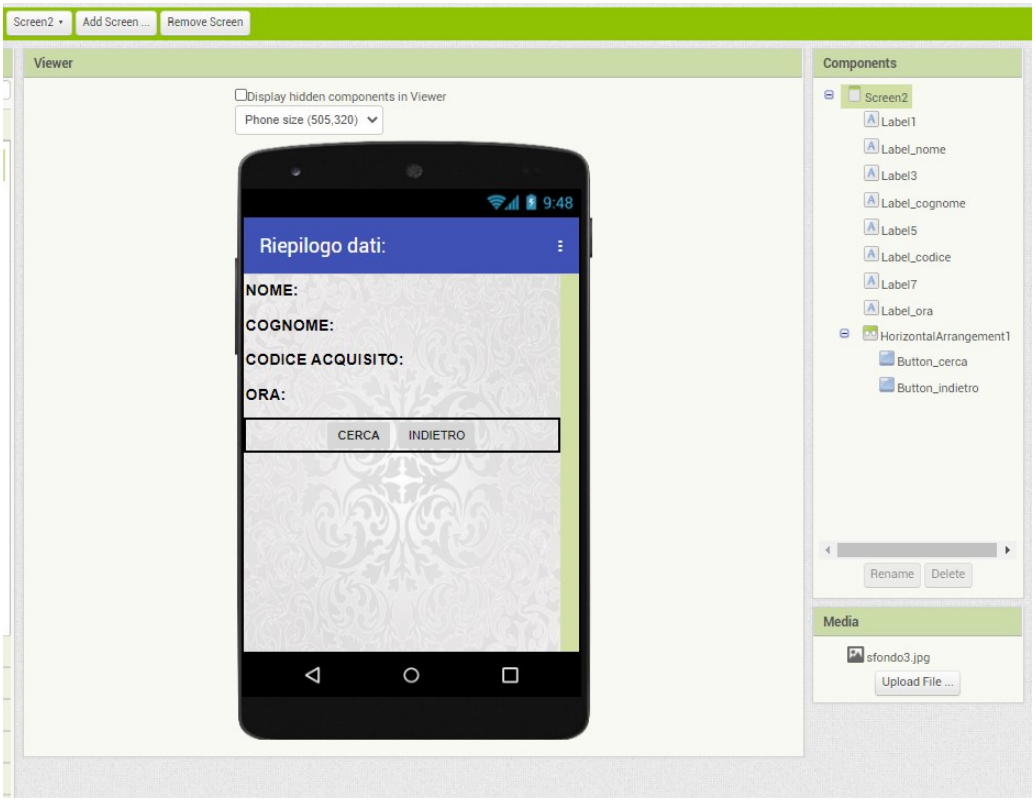


Figura 16-Screen 2

3.2.2.1 – Inizializzazione

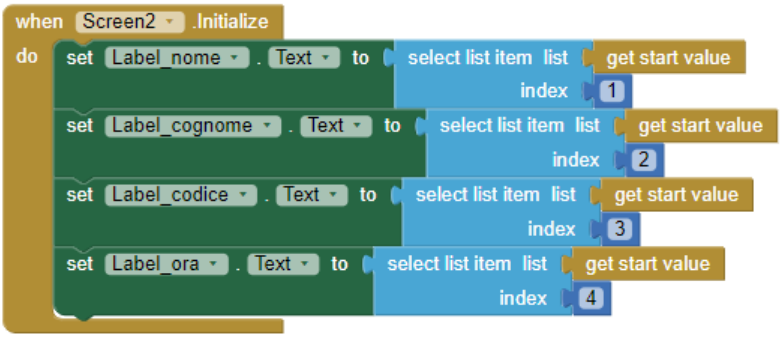


Figura 17-Blocco inizializzazione

Quando si avvia la schermata, in automatico vengono eseguite le seguenti operazioni:

- Sullo schermo, l'etichetta Label_nome viene impostata con il testo che corrisponde al primo elemento della lista (nome dell'utente) precedentemente creata per il passaggio dei valori da una schermata all'altra.
- All'etichetta Label_cognome viene assegnato il cognome dell'utente.
- Label_codice mostra il codice a barre del farmaco.
- Label_ora mostra l'istante di acquisizione del codice.

3.2.2.2 – Pulsanti di ricerca e indietro

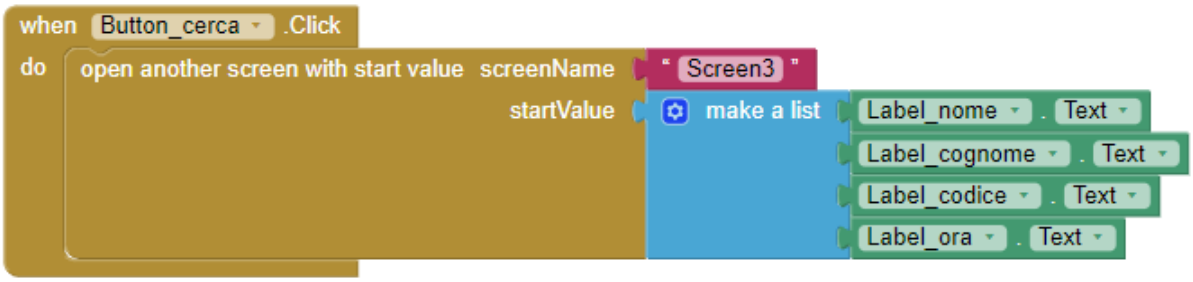


Figura 18-Pulsante di ricerca

Alla pressione del tasto Button_cerca si apre lo schermo successivo e, analogamente al caso precedente, vengono passati gli stessi valori riorganizzandoli in una lista.

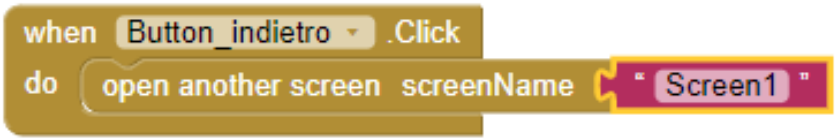


Figura 19-Pulsante indietro

Come si vedrà più avanti, all'utente viene sempre data la disponibilità di tornare alla pagina iniziale tramite un pulsante apposito, così da correggere eventuali errori dovuti all'inserimento dei dati.

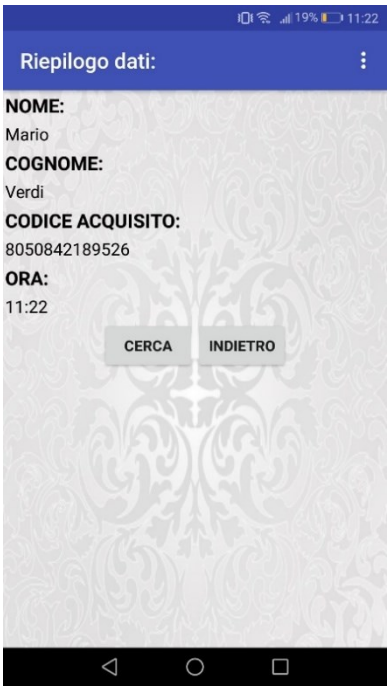


Figura 20-screen 2 su smartphone

3.2.3 – Screen 3

Questa schermata è il cuore dell'applicazione: qui è dove si effettua la ricerca vera e propria e vengono forniti i risultati. Per il passaggio dei valori, molte etichette sono state impostate come elementi non visibili, poiché sono utili solo al fine della progettazione delle righe di codice.

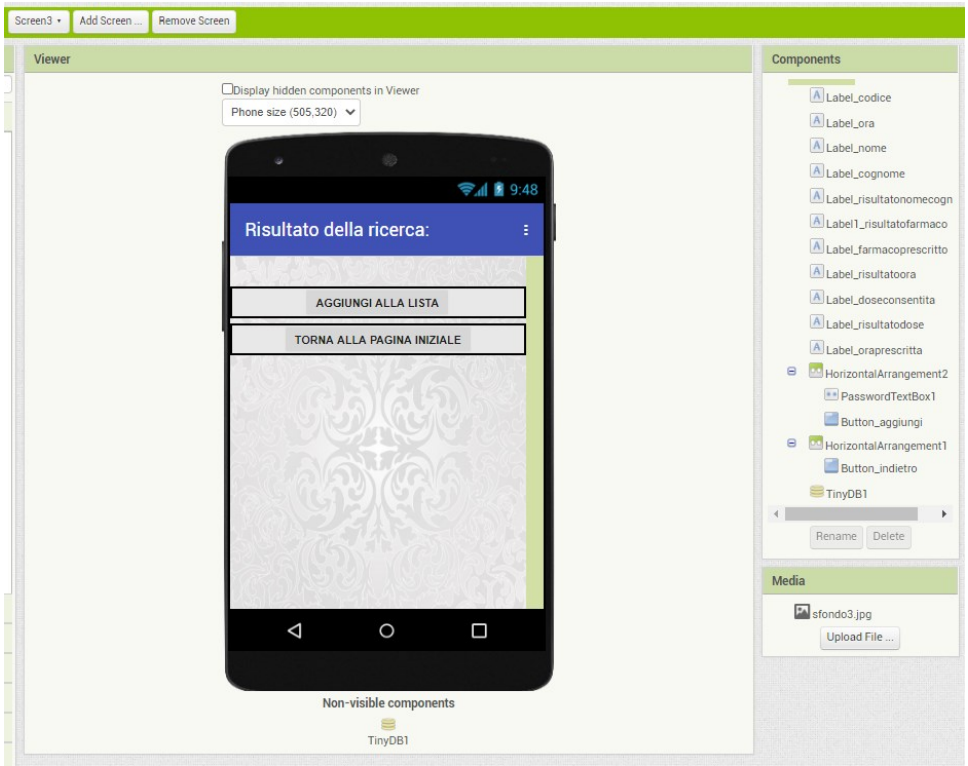


Figura 21-Screen 3

3.2.3.1 – Ricerca

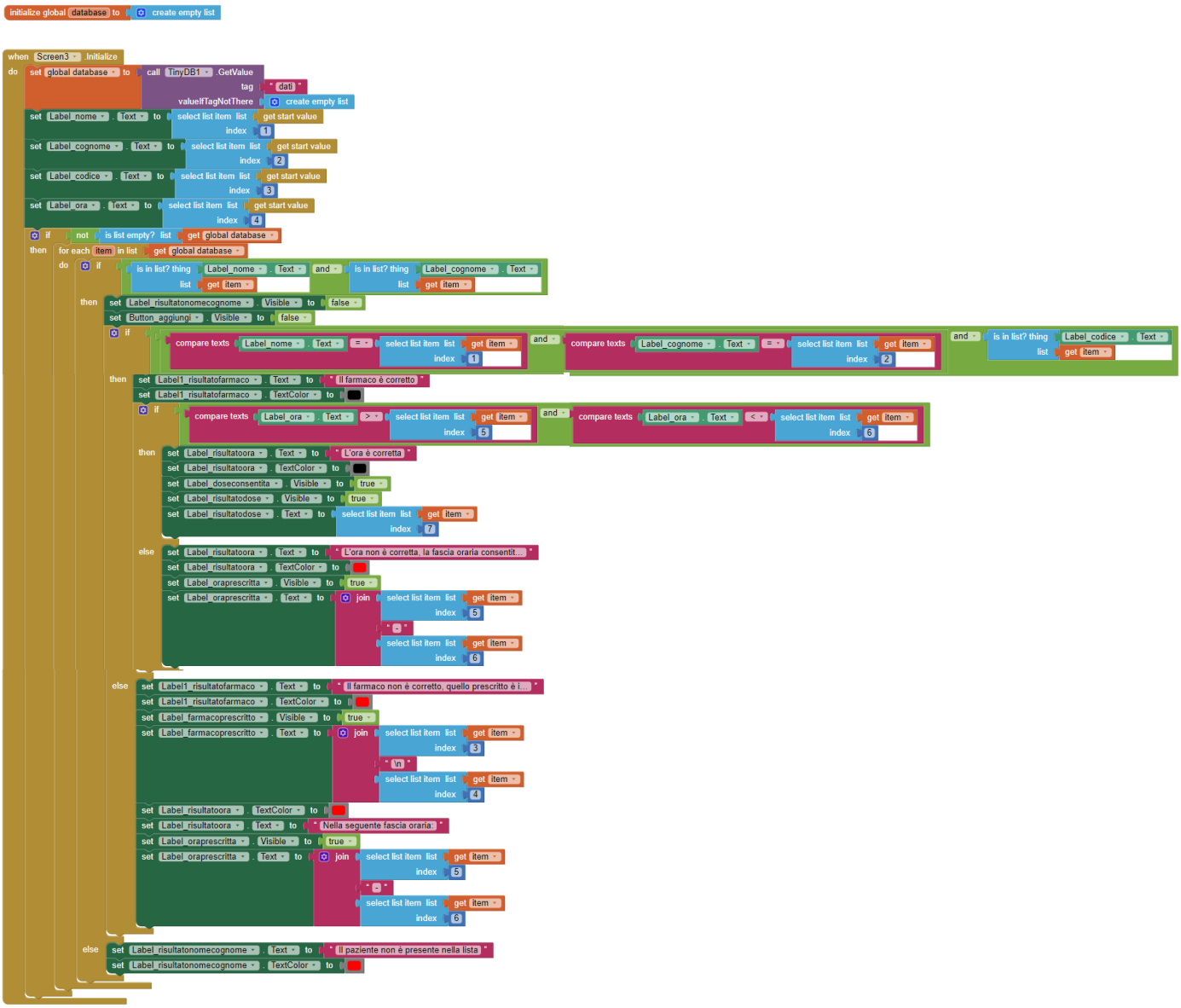


Figura 22-Blocco ricerca

Dovendo fare una ricerca all'interno di una lista condivisa tra più schermi, la prima cosa da fare è salvarla in un archivio al quale è possibile accedere da qualunque schermata.

Grazie al componente TinyDB descritto in precedenza, questo è possibile.

Per prima cosa viene definita la variabile globale "database" e inizializzata come una lista vuota. Quando lo screen 3 si apre, le operazioni che vengono eseguite solo le seguenti:

- Viene richiamato il TinyDB e i valori contenuti nell'archivio di memorizzazione sotto al tag "dati", vengono assegnati alla variabile "database". In questo modo, la lista di riferimento sarà proprio la lista "database" e ogni qual volta si vorrà accedere ad un elemento bisognerà indicarne l'indice.

- Le quattro operazioni successive sono le stesse descritte in precedenza per il passaggio dei valori tra schermate diverse. I dati sono gli stessi del caso precedente.
- Il primo controllo che viene eseguito dall'applicazione è quello di verificare che la lista sia popolata, ovvero che ci sia almeno un elemento per poter eseguire una ricerca.
- Se la lista non è vuota, viene eseguito un ciclo tramite cui l'applicazione va a esaminare ogni singolo elemento di essa.
- Il secondo controllo viene fatto verificando che l'utente sia presente nella lista, quindi tramite l'operatore logico "and" si controlla che sia il nome che il cognome siano registrati. In caso affermativo si procede agli ulteriori controlli, altrimenti sullo schermo apparirà un'etichetta con il seguente testo: "Il paziente non è presente nella lista". In tal caso viene data la possibilità di aggiungere nuovi dati alla lista tramite il pulsante "Button_aggiungi", il quale, dopo aver inserito la password, aprirà la schermata contenente la lista vera e propria.
- Si verifica poi la correttezza del farmaco andando a controllare che il codice a barre scansionato in precedenza sia presente tra i valori della lista e inoltre che sia associato a quel determinato paziente.
- Una volta verificata la correttezza del farmaco, si procede al controllo dell'orario andando a verificare che l'ora di acquisizione del codice a barre del farmaco sia compresa nella fascia oraria indicata nella lista.

  Sia in questo ultimo controllo che in quello precedente, viene utilizzato il blocco "compare texts" che mette a confronto due testi. In particolare, va a verificare se i due testi sono uguale o se uno è maggiore dell'altro. In programmazione, quando si ha a che fare con dei testi, questi vengono trasformati in numeri convertendoli attraverso l'uso di una tabella, detta Tabella ASCII: in questo modo ad ogni carattere (e non solo) corrisponderà un valore numerico.

Ogni volta che viene effettuato un controllo, ne viene fornito l'esito attraverso un'etichetta. In particolare, in caso di esito negativo, il testo si colorerà di rosso e verranno visualizzati dall'utente i dati corretti.

Infine, se tutti i controlli effettuati restituiscono un esito positivo, verranno fornite informazioni sulle dosi del farmaco prescritte nella terapia farmacologica del paziente.

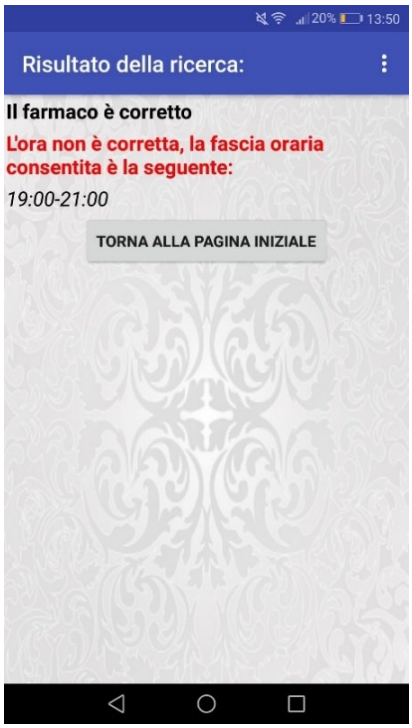


Figura 25-Esempio esito 1

Figura 26-Esempio esito 2

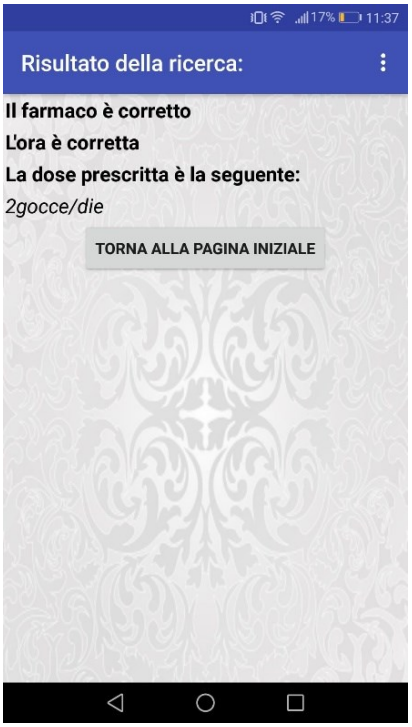


Figura 27-Esempio esito 3

3.2.4 – Screen 4

In questa parte dell'applicazione, alla quale è possibile accedere solo tramite l'inserimento di una password, sono memorizzati i dati dei pazienti in una ListView. In particolare, di un paziente vengono fornite le seguenti informazioni:

- Nome;
- Cognome;
- Nome del farmaco;
- Codice a barre del farmaco;
- Fascia oraria di assunzione (ora minima e ora massima);
- Dosi.

All'interno di questa lista è possibile fare due tipi di ricerche: una attraverso il codice a barre, mentre l'altra attraverso nome e cognome.

Inoltre si possono aggiungere nuovi pazienti in qualsiasi momento.

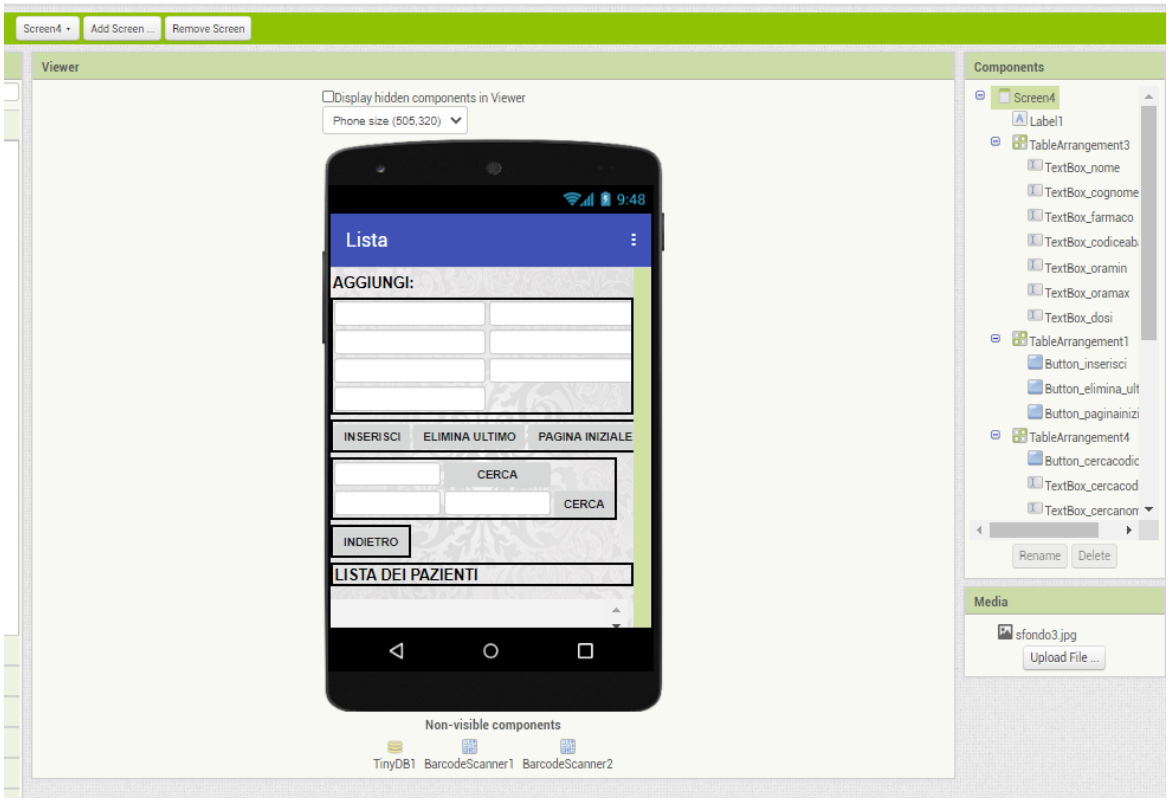


Figura 23-Screen 4

3.2.4.1 – Memorizzazione dei dati

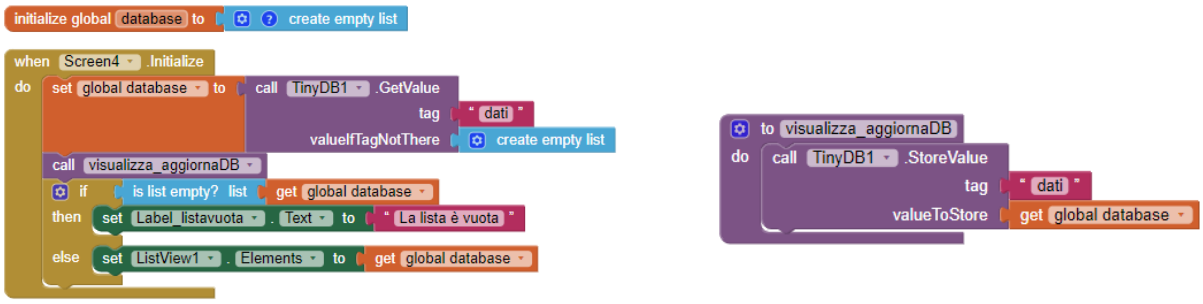


Figura 24-Inizializzazione e memorizzazione

Quando si apre la schermata, come già visto in precedenza, la variabile “database”, inizializzata a lista vuota, viene impostata con il valore memorizzato nel TinyDB con il tag “dati”. Nel caso in cui non fosse stato ancora memorizzato nulla, verrà creata una lista vuota.

Successivamente si fa una chiamata alla procedura “visualizza_aggiornaDB”: in Mit App Inventor una procedura non è altro che una sequenza di blocchi che vengono raggruppati insieme. È possibile utilizzare ripetutamente la sequenza di blocchi richiamando la procedura.

Attraverso “visualizza_aggiornaDB” si chiama nuovamente l’elemento TinyDB, ma questa volta con il metodo StoreValue, che consente di memorizzare il valore.
Ovviamente il tag deve essere sempre lo stesso e in questo caso il valore che si vuole memorizzare è quello contenuto nella lista “database”.
In seguito si controlla che la lista in questione non sia vuota e, nel caso in cui lo fosse, l’utente viene avvisato tramite l’etichetta “label_listavuota”, altrimenti tutti i valori della lista vengono memorizzati e riportati nella ListView.

3.2.4.2 – Inserimento e cancellazione

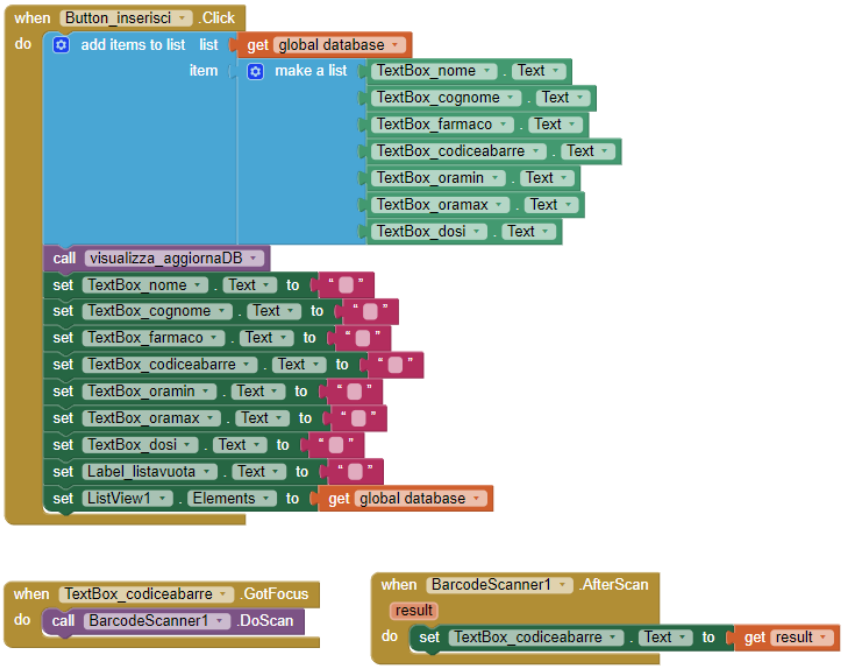


Figura 25-Pulsanti inserimento

Per quanto riguarda l’inserimento di nuovi pazienti alla lista, questo è possibile compilando gli appositi campi messi a disposizione e cliccando sul pulsante “INSERISCI”.
Dopo la pressione del pulsante si eseguono le seguenti operazioni:

- Alla lista “database” viene aggiunto il contenuto delle caselle di testo compilate, creando una lista in cui ad ogni indice corrisponde un campo (Nome, cognome, farmaco ecc.).
- Viene fatta una chiamata alla procedura “visualizza_aggiornaDB” per aggiornare l’archivio con i nuovi valori.
- Tutte le caselle di testo vengono svuotate dei testi inseriti, così da essere pronte in caso di inserimento di nuovi dati.
- Infine la ListView viene aggiornata, in modo da poter visualizzare anche i dati appena inseriti.

Per rendere più veloce l'inserimento del codice a barre, che spesso può essere lungo e poco agevole, alla pressione della casella di testo opportuna, si aprirà lo scanner del device e il risultato della scannerizzazione sarà visualizzato nella stessa casella di testo.

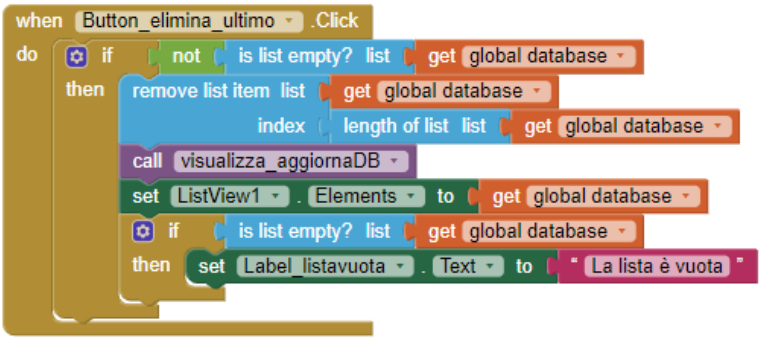


Figura 26-Pulsante eliminazione

Il blocco in questione consente all'utente di eliminare tutta l'ultima riga di dati attraverso un solo click.

Se vengono eliminate tutte le righe, sullo schermo si leggerà il seguente testo: "La lista è vuota".

3.2.4.3 – Ricerca

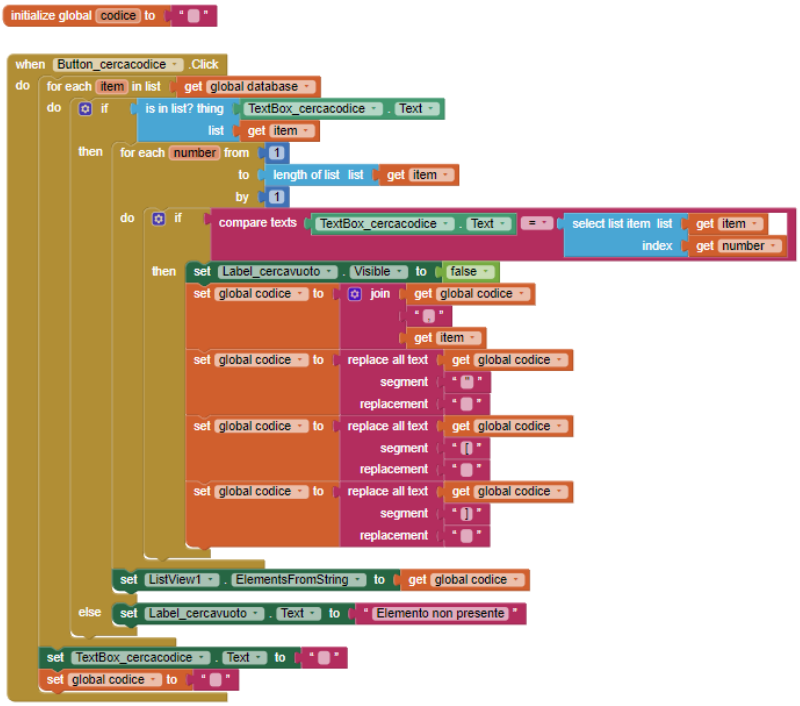


Figura 27-Blocco ricerca tramite barcode

Attraverso questo blocco è possibile ricercare i pazienti mediante il codice a barre del farmaco che devono assumere.

Viene analizzato ogni singolo elemento della lista e, nel caso in cui il farmaco fosse assegnato a più di un paziente, essi verranno visualizzati tutti.

Analogamente per il blocco seguente, nel quale però si fa una ricerca attraverso il nome e il cognome.

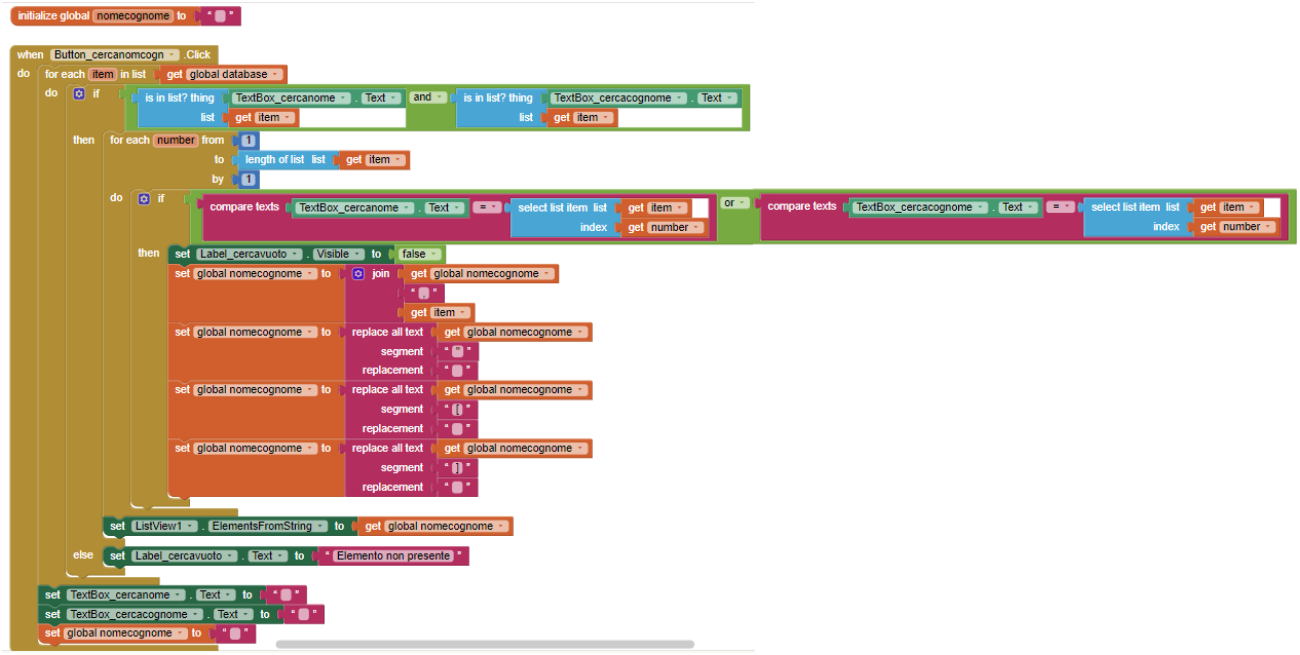


Figura 28-Ricerca tramite nome e cognome

Oltre a poter visualizzare tutti i dati della lista, cliccando su una riga della ListView apparirà solamente la riga in questione.

Infine, cliccando sul pulsante "INDIETRO", la ListView tornerà come era inizialmente.

Figura 29-Screen 4 su smartphone

4 – CONCLUSIONI E SVILUPPI FUTURI

In questo lavoro di tesi si è analizzata l'implementazione e il funzionamento di un'applicazione che utilizza come tecnologia di base i codici a barre: attraverso l'utilizzo di questa applicazione sarà possibile agevolare le persone nell'assunzione dei propri farmaci, facendo risparmiare tempo.
I codici a barre sono sempre più utilizzati al giorno d'oggi, perché racchiudono in essi molte informazioni nonostante siano molto piccoli. Inoltre la spesa per la loro stampa è spesso trascurabile, dato che il relativo disegno è solitamente inserito direttamente nel bozzetto dell'etichettatura o dell'imballaggio della merce.
Tra le esigenze di mercato riguardo i barcode vi è la richiesta che essi siano sempre più piccoli, in modo da occupare il minor spazio possibile e poter essere collocati ovunque.
Questo però, può essere fattibile fino ad un certo limite, infatti il fattore fondamentale è che possano essere leggibili dalle fotocamere degli smartphone.
Al giorno d'oggi attraverso questa tecnologia sono possibili molte azioni, ad esempio:

- Si può effettuare la spesa aggiungendo l'oggetto ad un carrello virtuale.
- All'interno di ristoranti, per permettere una maggiore interazione del cliente con i piatti cucinati.
- All'interno di musei per avere più informazioni riguardo agli oggetti esposti.
- Sui monumenti, così da fornire ai visitatori molte più informazioni di quante se ne possano scrivere su una classica didascalia.

Grazie alla versatilità dei codici a barre l'applicazione può essere estesa a qualsiasi altro contesto.
Alcuni sviluppi futuri potrebbero essere:

- Aggiungere una sezione dedicata alla descrizione dei farmaci. L'utente scannerizzando il codice a barre potrà visualizzare il foglietto illustrativo del farmaco in questione;
- Aggiungere la possibilità di inserire i propri sintomi ed ottenere una sorta di diagnosi e una terapia. Ovviamente questa applicazione non si sostituisce al lavoro del medico, ma lo supporta nell'esercizio della sua professione;

La tecnologia sta rivoluzionando sempre di più la vita quotidiana, diventando ormai alla portata di tutti.

BIBLIOGRAFIA

1. https://appinventor.mit.edu/
2. http://ai2.appinventor.mit.edu/reference/components/userinterface.html
3. https://www.codiceabarre.it/bccode32.html
4. http://nuovolabs.fauser.edu/~marfer/docs/2018/4AI/arduino/progetti/Corso%20App_Inventor(ElettronicaIN).pdf

RINGRAZIAMENTI

Grazie ai miei genitori e a mio fratello Michele, per avermi dato la possibilità di intraprendere questo percorso, supportandomi nelle mie decisioni e spronandomi ad andare avanti.

Grazie ad Andrea, per essere stato al mio fianco fin dall'inizio di questo percorso e per non aver mai dubitato di me, ascoltandomi ed aiutandomi nei momenti più difficili.

Grazie al professor Ennio Gambi, che mi ha supportato durante tutto il lavoro di tirocinio e tesi, dandomi la possibilità di mettermi in gioco.

Grazie ai miei amici, per esserci stati nel momento del bisogno. Un grazie particolare ai miei compagni di università, con i quali ho vissuto questa esperienza e con i quali mi sono confrontata.

Infine, un grazie a me stessa, che nonostante tutto sono riuscita a raggiungere questo traguardo che reputo importantissimo e che mi ha fatto maturare, facendomi diventare la persona che sono ora.