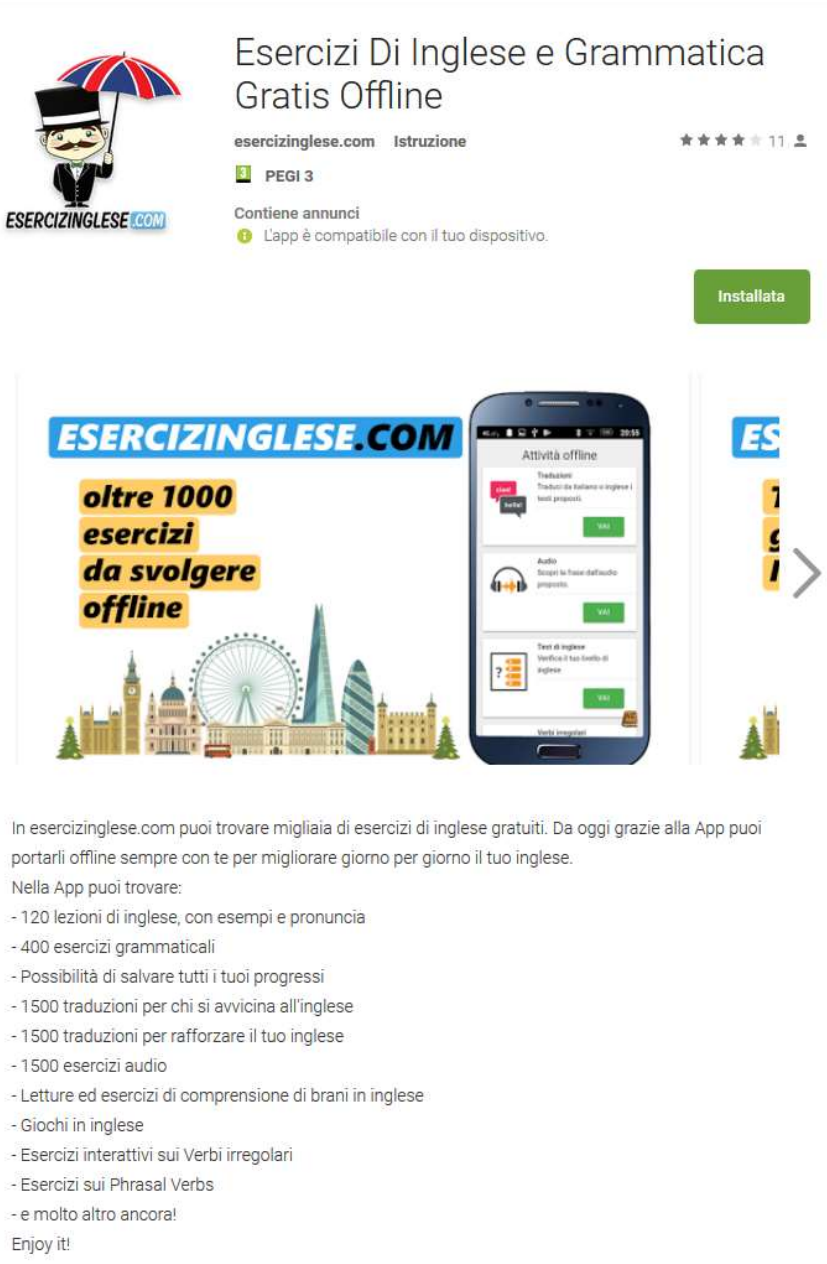
Sviluppo di un'applicazione Android per l'apprendimento della lingua inglese



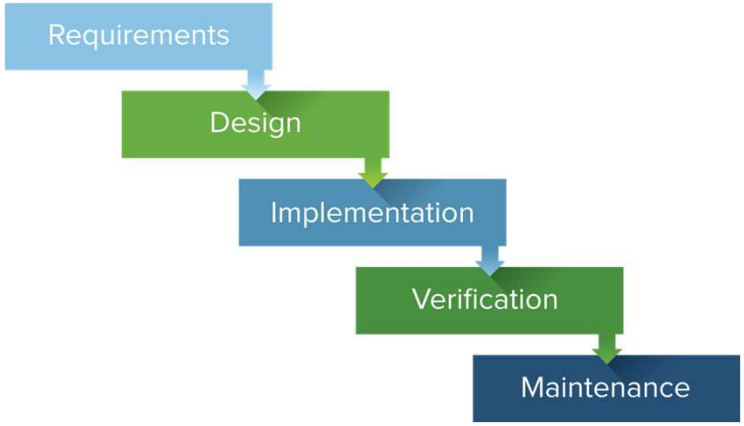
Il progetto

L'applicazione che ho sviluppato propone in modalità offline tutte le diverse attività del sito internet esercizinglese.com:

Come è strutturato il progetto

Per lo sviluppo del software si deciso di utilizzare il «modello a cascata» che è composto da 5 fasi:

1. analisi dei requisiti
2. progetto
3. sviluppo
4. collaudo
5. manutenzione



Si è scelto modello a cascata perché ha una struttura lineare che mostra tutti i passaggi che si incontrano nello sviluppo del software.

Fase 1

La prima fase è **l'analisi dei requisiti** il cui scopo principale è quello di definire tutte le funzionalità desiderate nel software.

Requisiti funzionali	Requisiti non funzionali
Test di inglese Salvataggio degli esercizi Esercizi di Traduzioni Audio …	Tempi di risposta reattivi Sicurezza Grafica accattivante Facilità di utilizzo dell'App …

- I *requisiti funzionali* tutto ciò che la App deve implementare [56]
- I *requisiti non funzionali* descrivono le proprietà che il software dovrà possedere

Output > *specifica dei requisiti*

Fase 2

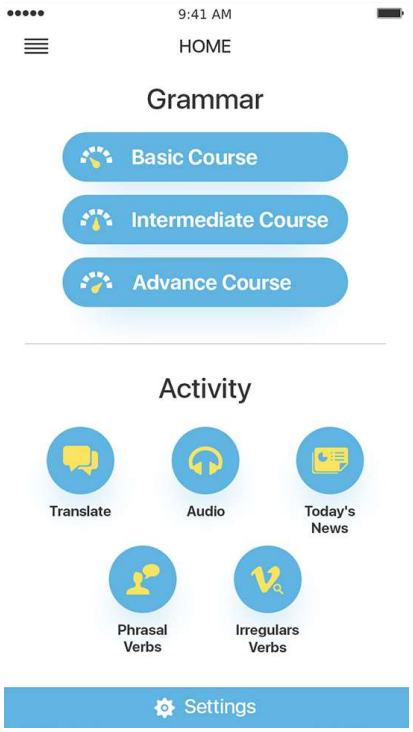
La fase di progettazione, partendo dalla **specifica dei requisiti**, definisce come tali requisiti saranno soddisfatti.

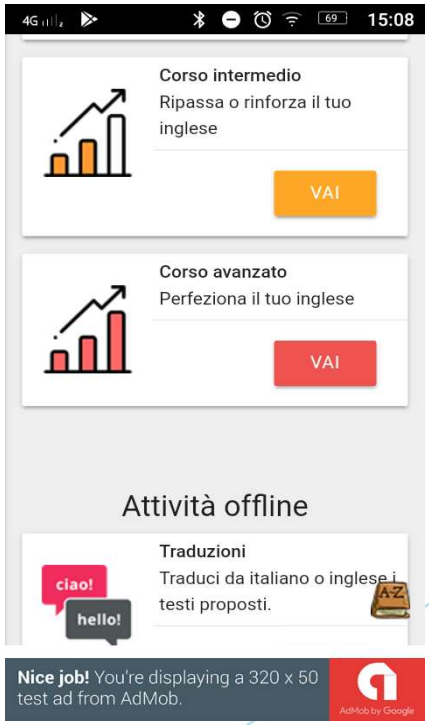
In questa fase sono stati progettati:

- Interfaccia utente
- Struttura dei database
- Simulazione di un Web Server (Java)
- Servizio di API (C#) per far comunicare il sito con la App

2.1 - Interfaccia utente

Queste 3 figure mostrano le fasi dello sviluppo dell'interfaccia utente.
Img 1: prima bozza disegnata a mano partendo dalla *specifica dei requisiti*
Img 2: prototipo suggerito dal grafico
Img 3: risultato finale che appare nella App



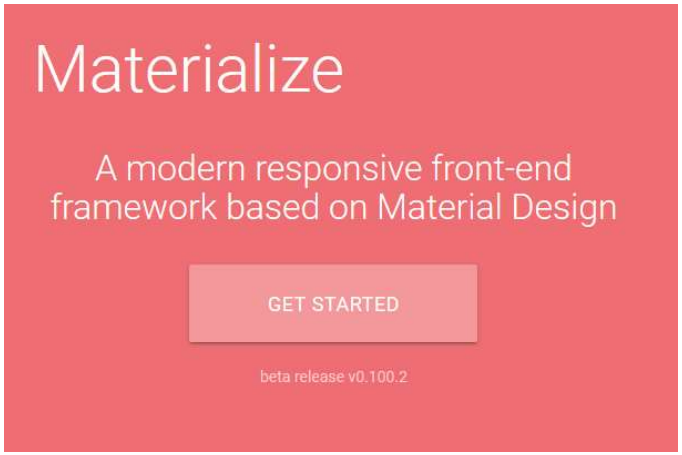


2.1 - Come realizzare il layout

- Per visualizzare tutti i contenuti si è deciso di utilizzare solamente una **webView**, un componente che permette di rappresentare HTML direttamente nel layout dell'Activity.

+ Velocizza il lavoro di sviluppo

+ Riutilizzo del codice HTML e Js del sito



Per creare i vari componenti html (pulsanti, textbox,menù …) si è scelto di utilizzare il framework front-end **Materializecss** che si ispira al design suggerito da Google per le App di Android

2.2 - Database

Per poter offrire all'utente un'esperienza completamente offline, l'App avrà bisogno che tutti i dati necessari siano salvati sul dispositivo quindi si ha la necessità di un database Sql.

- *FASE1:*

Tutti i dati venivano scaricati direttamente da internet alla prima apertura per avere:

1. App sempre aggiornata
2. minor spazio occupato sul device.

Ma con il crescere delle funzionalità implementate la fase di installazione è risultata troppo lenta.

- *FASE 2:*

I database sono stati creati localmente ed inseriti direttamente nel file apk quindi, al primo avvio vengono copiati nella cartella «databases».

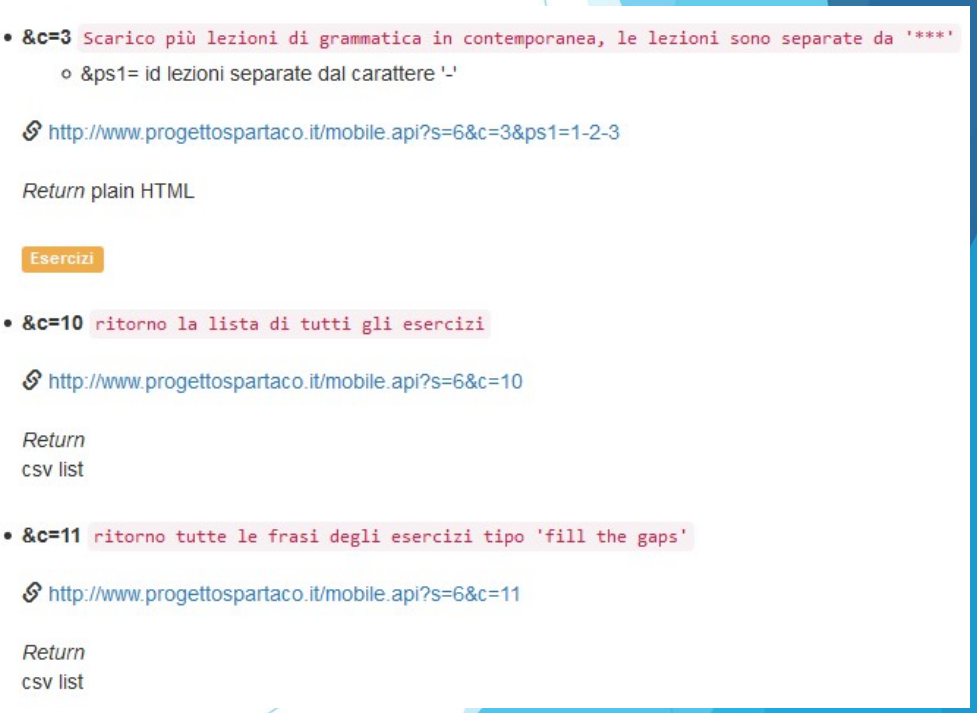
Successivamente si controlla se ci sono eventuali aggiornamenti attraverso un **numero di versione**.

Grazie a questo cambiamento il tempo di installazione è passato da 2 minuti a 20 secondi

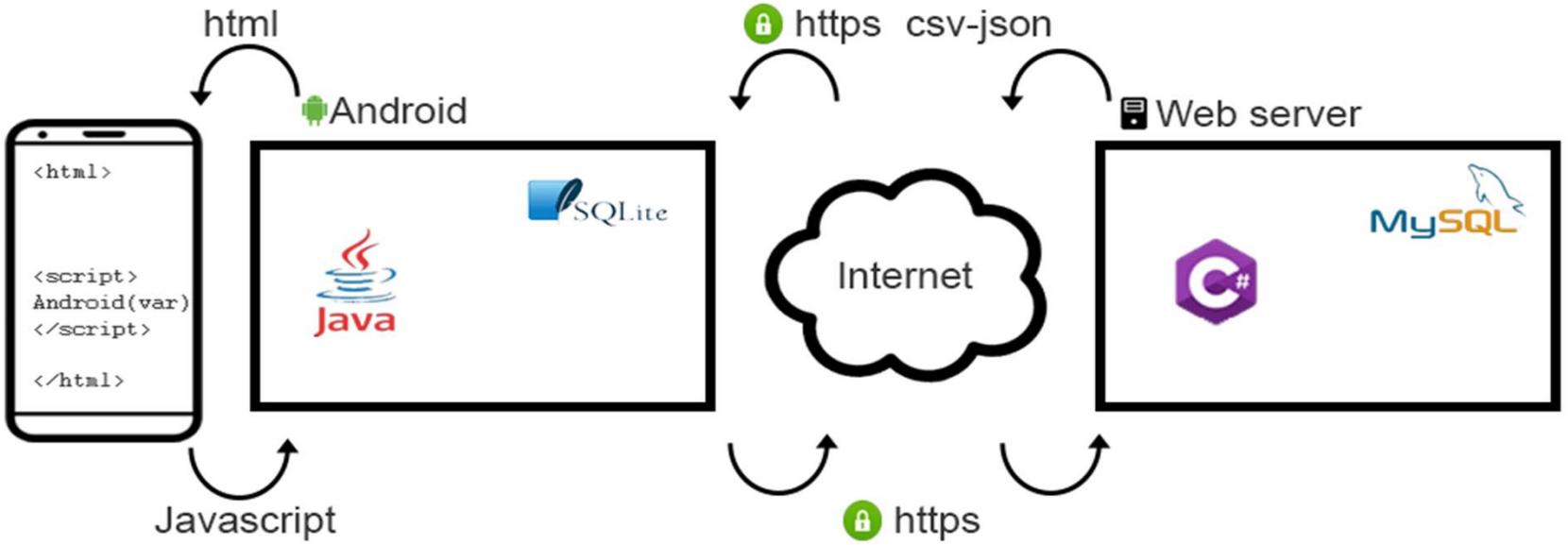
2.4 Web Server C#

Per far dialogare l'App con il sito si è dovuto progettare anche un servizio di API per:

1. Accedere direttamente ai dati immagazzinati nel database Mysql
2. Sincronizzare i risultati degli esercizi tra il sito e la App
3. Mostrare della pubblicità



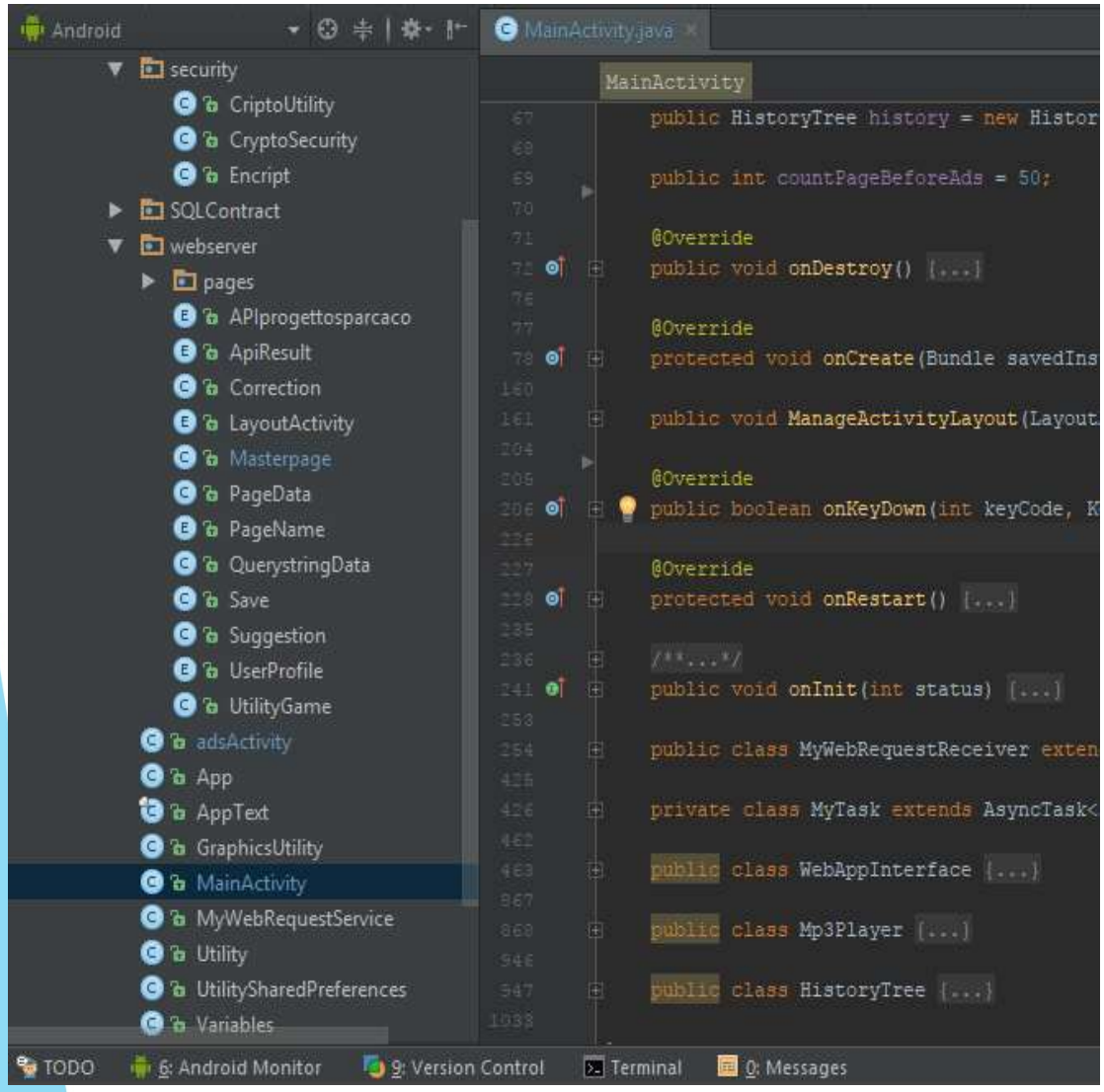
- Nell'immagine si può vedere come è stato progettato il sistema e le tecnologie utilizzate.



La pagina HTML che l'utente visualizza contiene il Javascript che richiama il codice Java.
La parte Java

1. recupera le informazioni dalla base di dati SQLlite, le elabora e restituisce direttamente alla webview l'html prodotto localmente
2. si interfaccia con le API del webserver C# che, recuperando le informazioni sul database Mysql, le restituisce in formato csv o Json.

4 Sviluppo



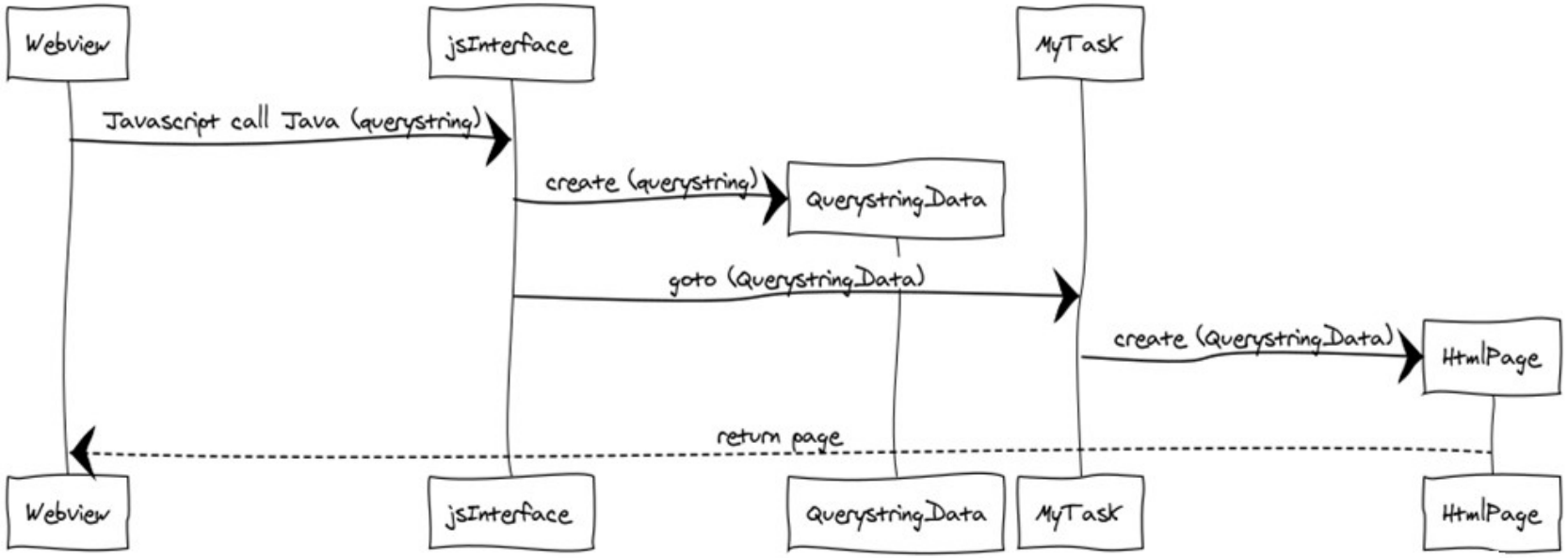
La fase successiva alla progettazione è la fase di sviluppo in cui viene scritto il codice.

Per il lavoro di sviluppo ho utilizzato

- **Andorid studio** per i file .java
- **Visual studio** per i file .cs del webserver, html css e js
- **Git** per il controllo di versione
- **DB Browser for SQLite** per creare e gestire i database

4.1 Creazione della pagina html (1)

In questa immagine possiamo vedere il **Sequence Diagram** della creazione della pagina ad alto livello



Il codice Javascript richiama il codice Java passandogli una stringa in cui si specifica cosa l'utente vuole effettuare.

Questa stringa è usata per creare l'oggetto *QueryStringData* che semplicemente struttura le informazioni.

L'oggetto *QueryStringData* viene passato alla classe MyTask che crea il codice HTML in un thread separato.

Infine il codice prodotto viene ritornato alla webview che lo visualizza.

4.1 Creazione della pagina html (2)

Nel codice riportato si vede parte il lavoro effettuato nella classe *MyTask*.

Sfruttando la gerarchia delle classi tipica dei linguaggi ad oggetti, viene creata una nuova classe *PageData* opportunatamente inizializzata in base al parametro PageName.

```
1 public Masterpage(QuerystringData myQuerystringData) {
2 
3         boolean needShowUserScore = false;
4 
5         switch (myQuerystringData.myPageName) {
6 
7             case HOMEPAGE:
8                 myPageData = new HomePage( );
9                 break;
10            case LISTGRAMMAR:
11                myPageData = new ListGrammarLessons(myQuerystringData.lessonID);
12                break;
13            case LESSON:
14                myPageData = new GrammarLesson(myQuerystringData.lessonID);
15                needShowUserScore= true;
16                break;
```

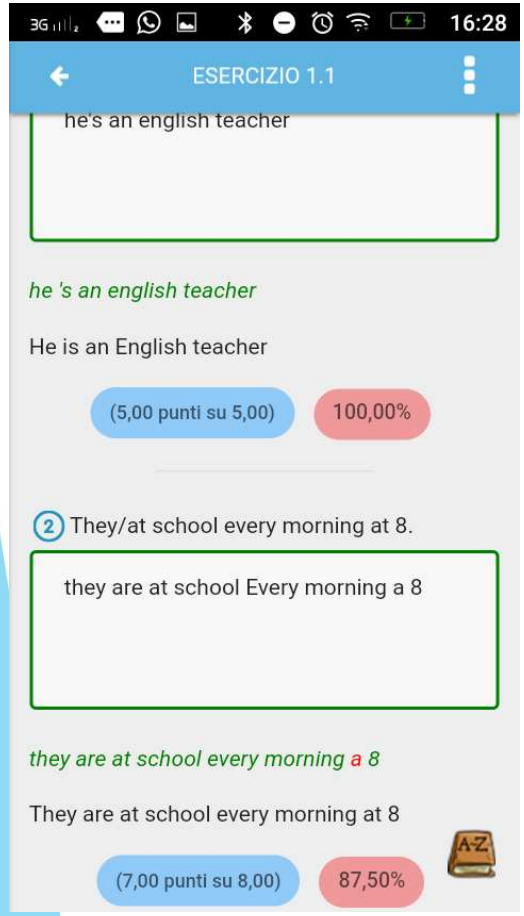
4.1 Creazione della pagina html (3)

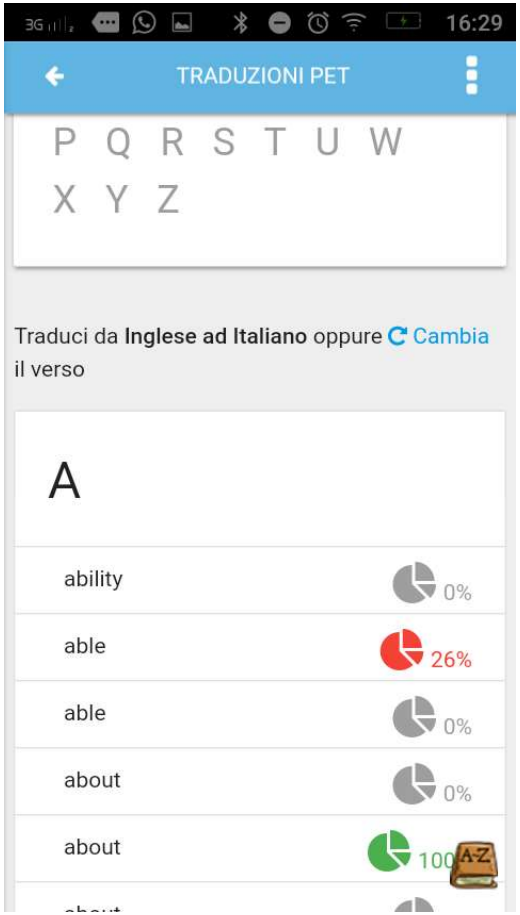
- In quest'altro pezzo di codice vediamo la creazione dell'esercizio audio. In base al valore di *isPostBack* il body della pagina mostra l'esercizio da svolgere oppure la correzione.
- *SetExerciseAudio* è la classe che interroga il database per recuperare tutte le informazioni necessarie
- I metodi *GetCSS* e *GetJS*, come il loro nome suggerisce, si occupano di caricare il CSS e il Javascript della pagina

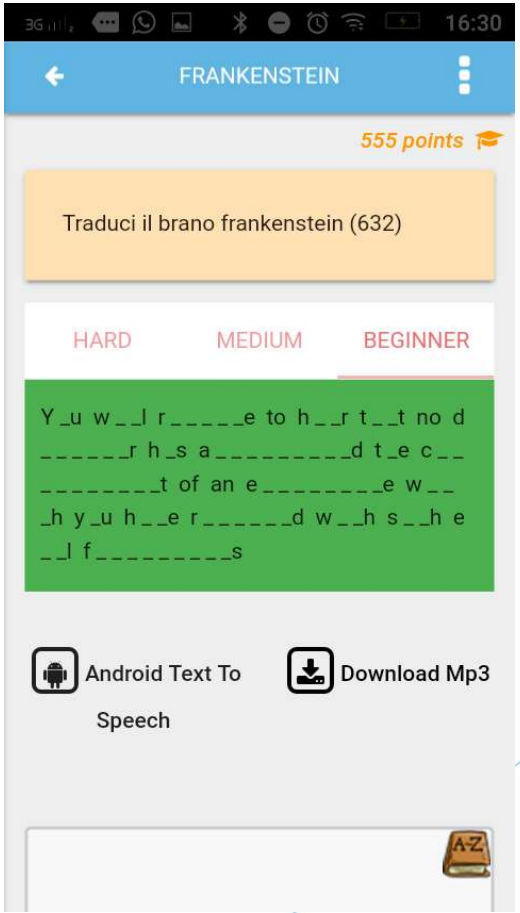
```
1    public ExerciseAudio(int lessonID, String userSolutions, boolean isPostBack) {
2
3        this.IDExercise=lessonID;
4
5        SetExerciseAudio mySetExerciseAudio = new SetExerciseAudio();
6        ExerciseAudioData myExerciseAudioData= mySetExerciseAudio.GetAudioExercise(App.get(),lessonID);
7
8        this.navbarPageName = myExerciseAudioData.book;
9
10       if(isPostBack==false){
11           this.body = ShowExercise(myExerciseAudioData.sentence);
12       }else{
13           this.body = ShowCorrection(userSolutions,myExerciseAudioData.sentence);
14       }
15
16       this.cssStyle = GetCSS();
17       this.javascript = GetJS();
18   }
```

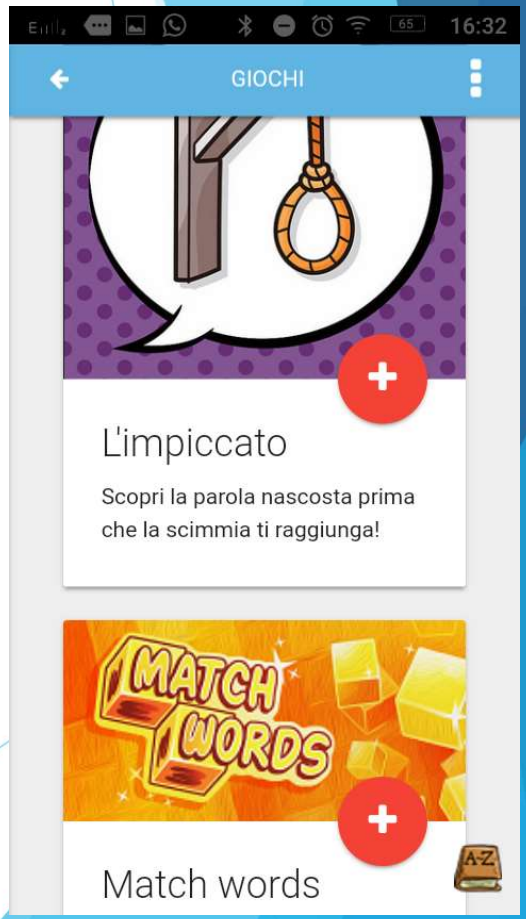
4.3 Creazione di tutte le pagine

Sono state create così tutte le 32 pagine della App



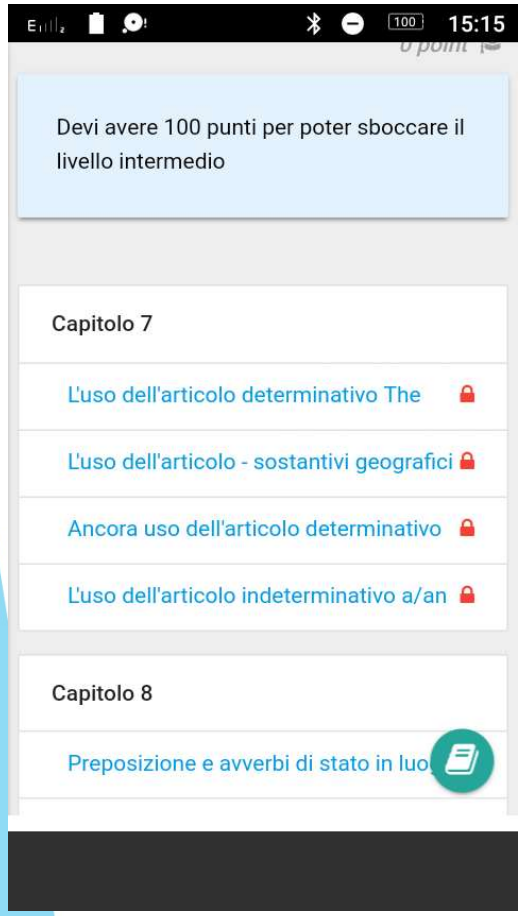


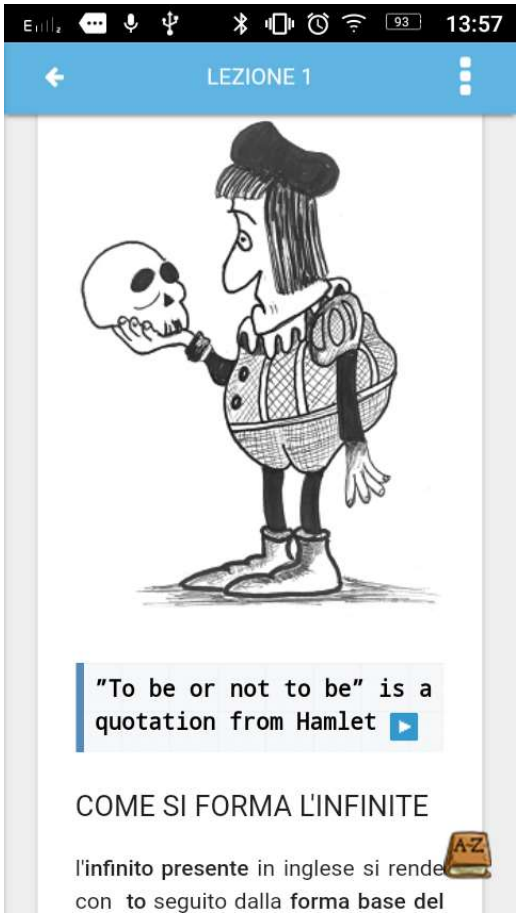


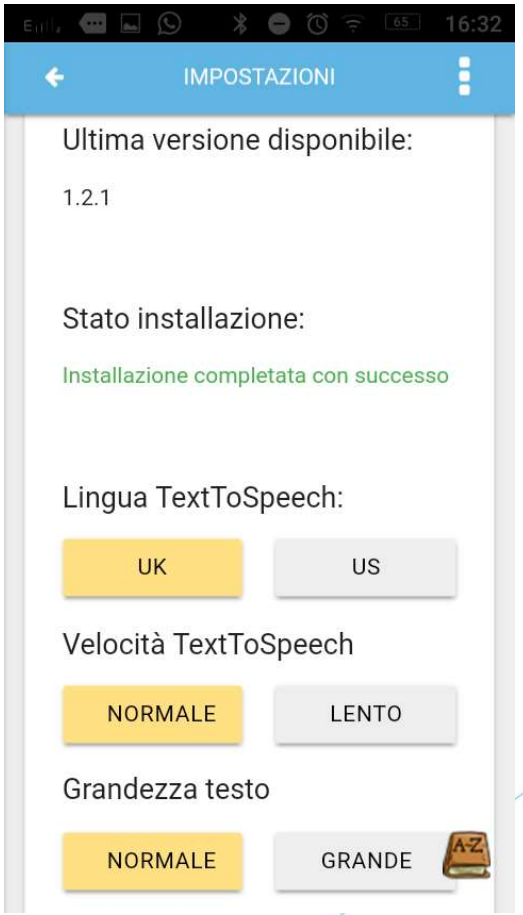


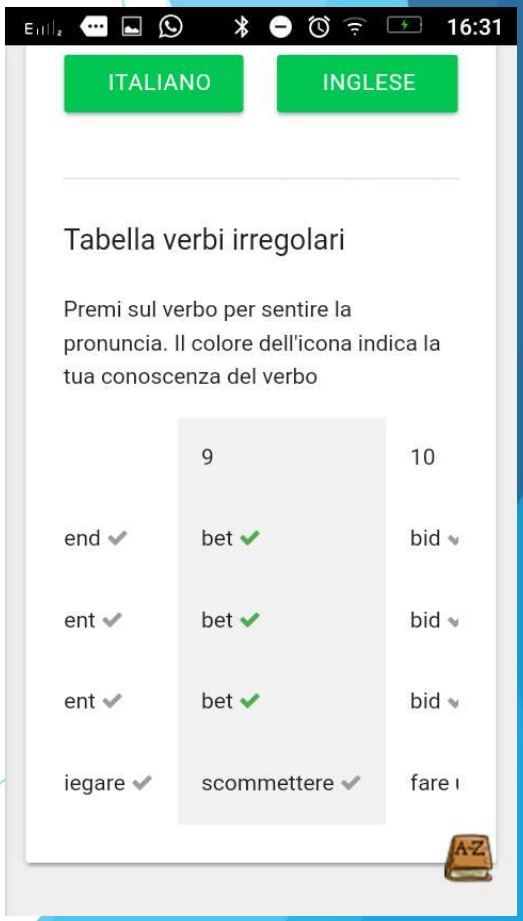
4.3 Creazione di tutte le pagine (2)

Altri screenshot

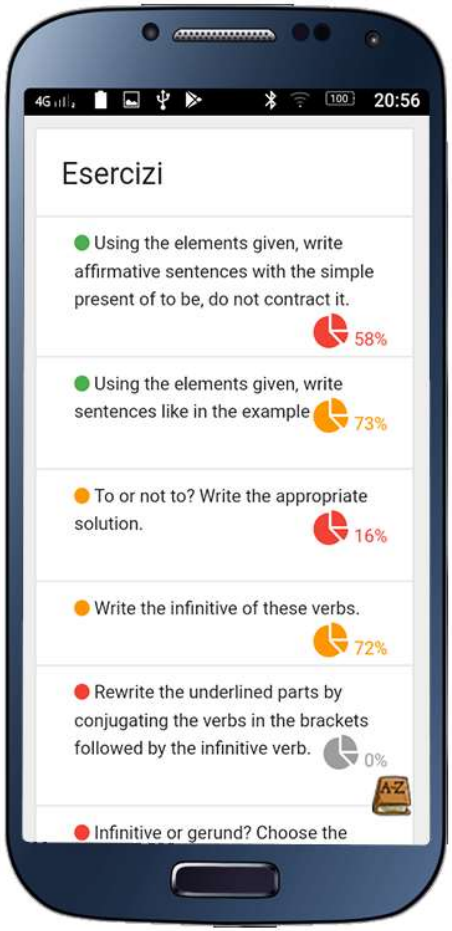








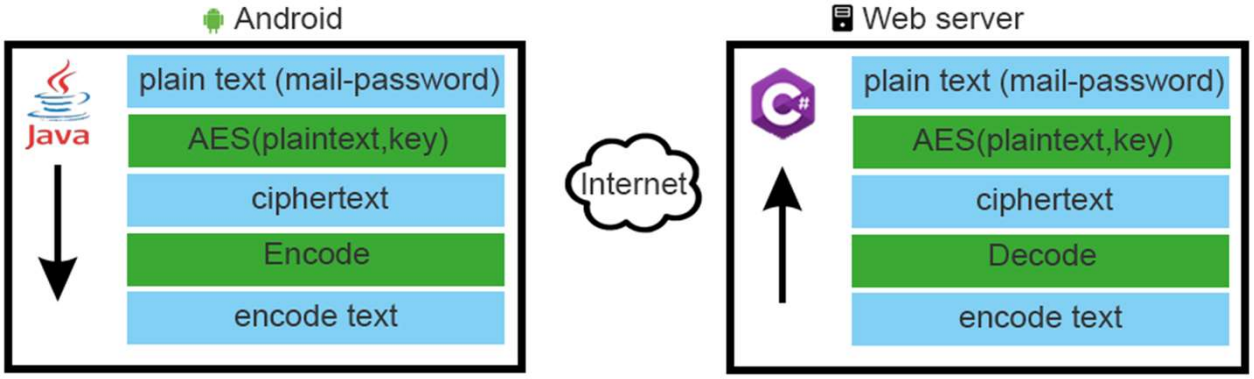
4.3 Salvataggio esercizi



- Per aumentare il coinvolgimento dell'utente si è sviluppato un sistema di *gamification* dell'esperienza: quando l'utente effettua un esercizio, in base al risultato ottenuto, guadagna dei punti.
- I punti che gli serviranno per sbloccare altri esercizi e attività
- È stato implementato di sistema di acquisto dei punti gestito con **Paypal**
- Per cambiare il colore e la percentuale delle icone dei vari esercizi come ad esempio nell'immagine della slide, viene utilizzato un piccolo script in Javascript, che viene eseguito dopo che la pagina è stata caricata.

4.5 Sicurezza

- Le richieste di autentificazione potrebbero esporre le credenziali dell'utente, nonostante il sito disponga di una connessione https. Nei log degli accessi si potrebbe trovare una stringa:
  https://api.website.com?mail=mail@yhaoo.com&password=mypassword
- Si deciso di criptare e-mail e password



Come si può vedere dalla figura il webserver fa esattamente il lavoro opposto effettuato dalla per ritrovare mail e password

Fortunatamente entrambi i linguaggi hanno delle librerie dedicate alla sicurezza. Per criptare si è usato l'algoritmo di cifratura *Advanced Encryption Standard*, mentre per la codifica e la decodifica del testo criptato (compatibile con un URL) si è usato l'algoritmo *Base64*.

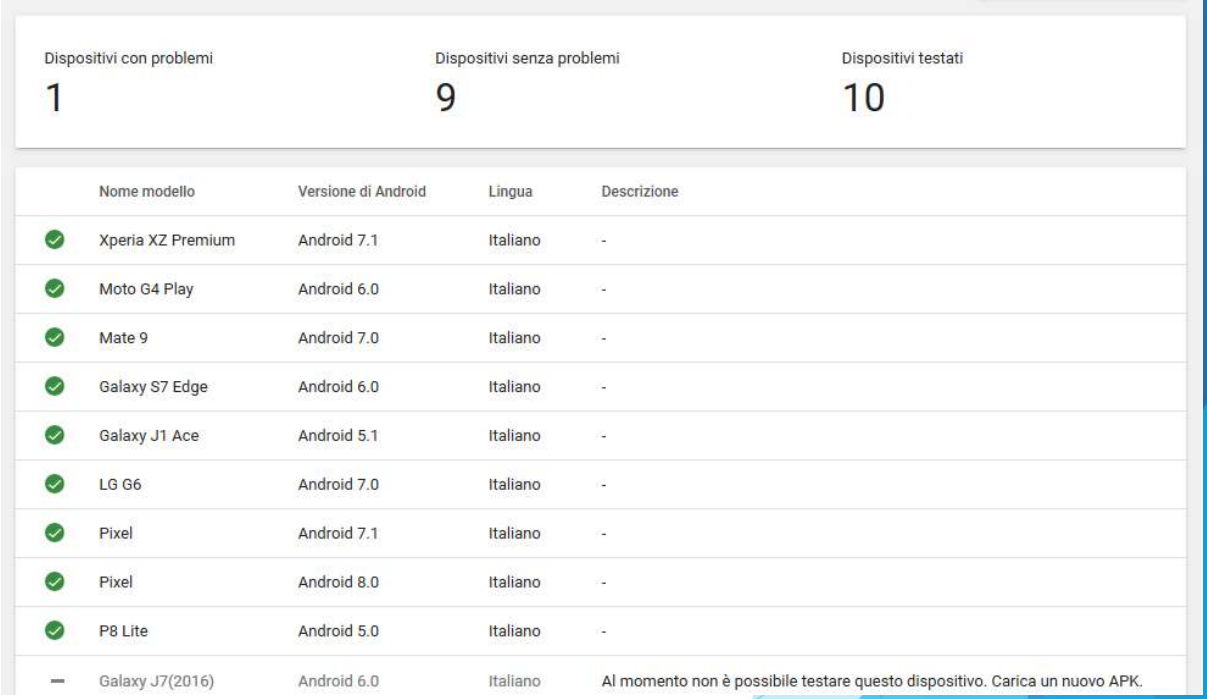
Fase 4

La quarta fase del modello a cascata è il collaudo.

Google mette a disposizione molti tools per testare e controllare l'App prima del rilascio sul Play Store

Il collaudo è avvenuto attraverso 2 fasi:

- La fase **Alpha** dove è stata rilasciata ad un piccolo gruppo di programmatori
- La fase **Beta** dove il gruppo di persone coinvolte era più eterogeneo e l'attenzione è stata rivolta all'esperienza utente

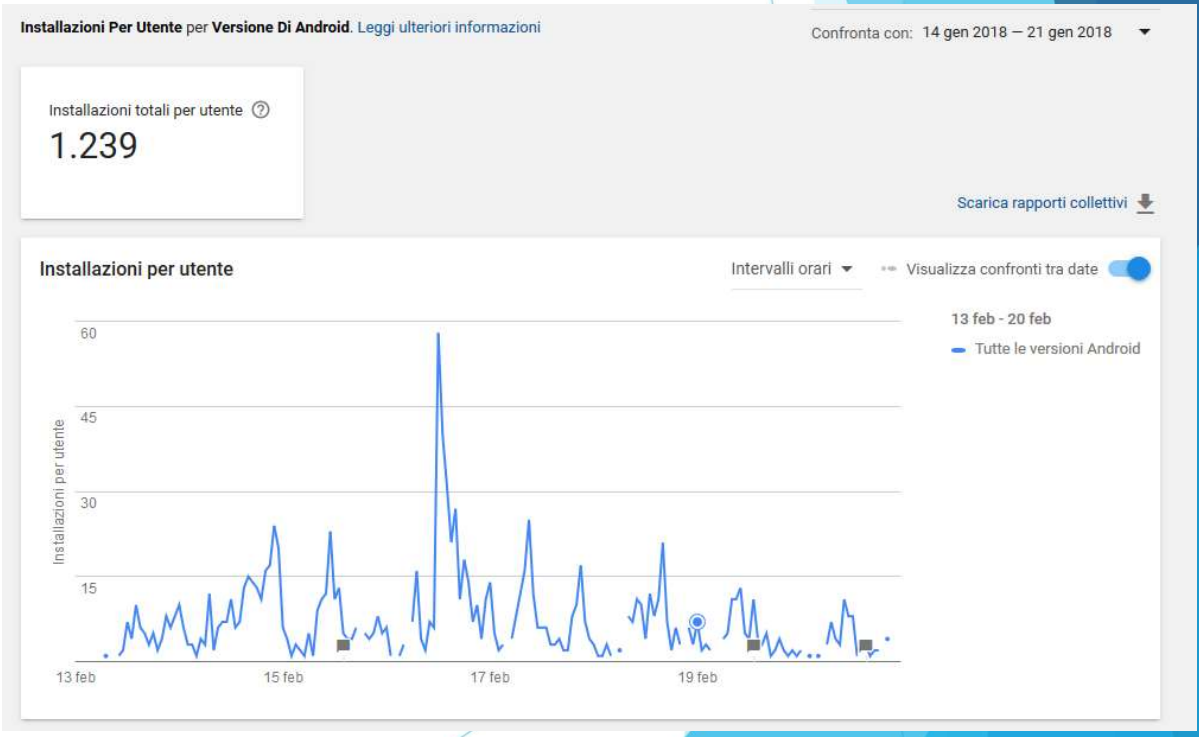


Dispositivi con problemi	Dispositivi senza problemi	Dispositivi testati
1	9	10

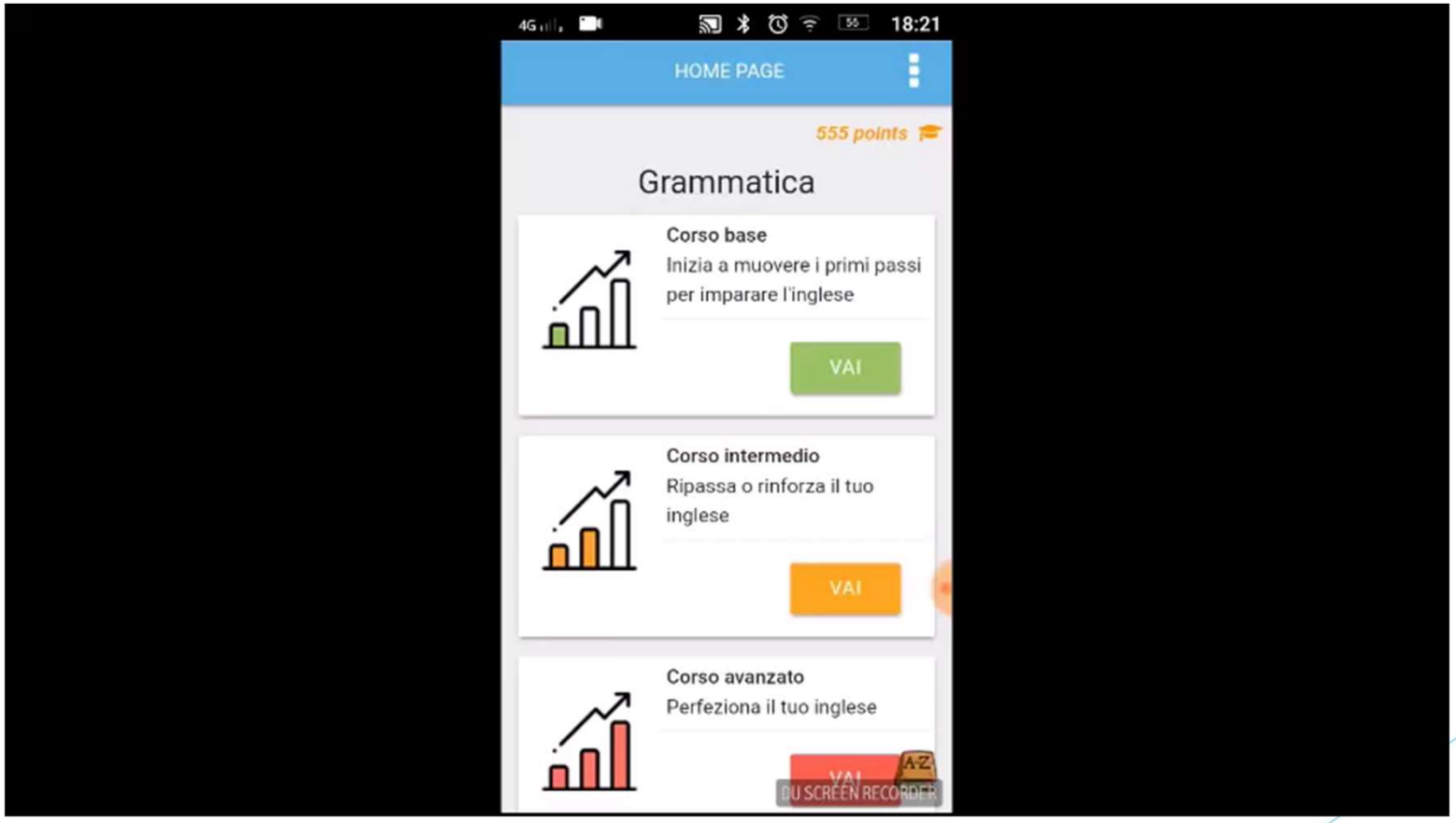
	Nome modello	Versione di Android	Lingua	Descrizione
✓	Xperia XZ Premium	Android 7.1	Italiano	-
✓	Moto G4 Play	Android 6.0	Italiano	-
✓	Mate 9	Android 7.0	Italiano	-
✓	Galaxy S7 Edge	Android 6.0	Italiano	-
✓	Galaxy J1 Ace	Android 5.1	Italiano	-
✓	LG G6	Android 7.0	Italiano	-
✓	Pixel	Android 7.1	Italiano	-
✓	Pixel	Android 8.0	Italiano	-
✓	P8 Lite	Android 5.0	Italiano	-
–	Galaxy J7(2016)	Android 6.0	Italiano	Al momento non è possibile testare questo dispositivo. Carica un nuovo APK.

Rilascio sul play store

- Una volta corretti tutti i bug e implementati i suggerimenti ricevuti la App è stata finalmente pubblicata il 13/02/2017 sul Play Store.



Breve (30s) video della App



5.1 Collaudo

Per il testare le API è stato creato un piccolo programma Windows che richiama tutti gli URL e controlla che:

1. l'url sia raggiungibile
2. venga restituita una risposta
3. il csv sia ben formattato
4. la sequenza dei tipi dei vari campi del csv (es. int, string, int) sia corretta