Forgia Paolo

SSSE  SIG

Manuale di progetto

Indice

[1 Abstract 4](#_Toc118472273)

[2 Sommario 4](#_Toc118472274)

[2.1 Descrizione progetto 4](#_Toc118472275)

[2.2 Per chi 4](#_Toc118472276)

[2.3 Benefici 4](#_Toc118472277)

[2.4 Risultato ottenuto 4](#_Toc118472278)

[3 Introduzione 4](#_Toc118472279)

[3.1 Descrizione azienda 4](#_Toc118472280)

[3.1.1 Organigramma 4](#_Toc118472281)

[3.2 Scopo 5](#_Toc118472282)

[3.3 Obiettivi 5](#_Toc118472283)

[3.4 Situazione attuale 5](#_Toc118472284)

[3.5 Benefici attesi 6](#_Toc118472285)

[3.6 Limitazioni 6](#_Toc118472286)

[4 Introduzione 7](#_Toc118472287)

[4.1 Descrizione azienda 7](#_Toc118472288)

[4.1.1 Organigamma 7](#_Toc118472289)

[4.2 Scopo 7](#_Toc118472290)

[4.3 Obiettivi 7](#_Toc118472291)

[4.4 Benefici attesi 7](#_Toc118472292)

[5 Vision 7](#_Toc118472293)

[6 Situazione attuale 7](#_Toc118472294)

[6.1 Come viene fatto il lavoro ora 7](#_Toc118472295)

[6.2 Mappa processi aziendali 7](#_Toc118472296)

[6.3 Analisi documenti interni 7](#_Toc118472297)

[6.4 Confronto vecchio-nuovo 7](#_Toc118472298)

[7 Analisi 7](#_Toc118472299)

[7.1 Use case di contesto 7](#_Toc118472300)

[7.2 Activity concettuale 7](#_Toc118472301)

[7.3 Requisiti 7](#_Toc118472302)

[7.4 App ibrida vs nativa 8](#_Toc118472303)

[7.4.1 Nativa 8](#_Toc118472304)

[7.4.2 Ibrida 8](#_Toc118472305)

[7.5 Analisi tecnologica 8](#_Toc118472306)

[7.5.1 Xamarin 8](#_Toc118472307)

[7.5.2 Flutter 8](#_Toc118472308)

[7.5.3 Ionic 8](#_Toc118472309)

[7.5.4 Cordova 9](#_Toc118472310)

[7.5.5 React Native 9](#_Toc118472311)

[7.5.6 Confronto 9](#_Toc118472312)

[7.6 Analisi approfondita 9](#_Toc118472313)

[7.6.1 Xamarin 9](#_Toc118472314)

[7.6.2 React Native 10](#_Toc118472315)

[7.6.3 Scelta finale: Xamarin 11](#_Toc118472316)

[7.6.4 Considerazioni a posteriori 11](#_Toc118472317)

[7.7 Fonti 13](#_Toc118472318)

[8 Personas 14](#_Toc118472319)

[8.1 Strategie aziendali 14](#_Toc118472320)

[8.2 Ricerche di mercato 14](#_Toc118472321)

[9 Qualità 14](#_Toc118472322)

[9.1 Norme aziendali 14](#_Toc118472323)

[9.1.1 Regolamento documentazione 14](#_Toc118472324)

[9.1.2 Rilascio software 14](#_Toc118472325)

[9.2 ISO 14](#_Toc118472326)

[9.3 Standard 14](#_Toc118472327)

[10 Pianificazione 14](#_Toc118472328)

[10.1 Gantt 14](#_Toc118472329)

[10.2 Trello 14](#_Toc118472330)

[11 Mappa dei processi 15](#_Toc118472331)

[12 Metodologia 15](#_Toc118472332)

[12.1 Fonte di ispirazione 15](#_Toc118472333)

[12.2 Adattamento della metodologia teorica 15](#_Toc118472334)

[13 Analisi rischi 15](#_Toc118472335)

[13.1 Strumenti 15](#_Toc118472336)

[13.2 Rischi accaduti 16](#_Toc118472337)

[14 Volumi / Quantità / Flussi 16](#_Toc118472338)

[14.1 Scalabilità 16](#_Toc118472339)

[14.2 Volumi futuri 16](#_Toc118472340)

[14.3 Stress test 16](#_Toc118472341)

[14.4 Performance test 16](#_Toc118472342)

[15 Comunicazione 16](#_Toc118472343)

[15.1 Frequenza riunioni 16](#_Toc118472344)

[15.2 Modello verbale 16](#_Toc118472345)

[15.3 Modalità e contenuto incontri 16](#_Toc118472346)

[16 Reportistica 16](#_Toc118472347)

[17 Migrazione dati 16](#_Toc118472348)

[18 Sviluppi futuri 16](#_Toc118472349)

[18.1 Funzionalità mancanti 16](#_Toc118472350)

[18.2 Lista bug esistenti 16](#_Toc118472351)

[18.3 Nice to have 16](#_Toc118472352)

[19 Analisi dei rischi 17](#_Toc118472353)

[19.1 Uso strumenti 17](#_Toc118472354)

[19.2 Catalogo rischi 17](#_Toc118472355)

[19.3 Rischi accaduti 17](#_Toc118472356)

[20 Macro requisiti 17](#_Toc118472357)

[21 Analisi costi e benefici 17](#_Toc118472358)

[22 Mockup 17](#_Toc118472359)

[22.1 Confronto iniziale – finale 17](#_Toc118472360)

[23 Diagrammi 17](#_Toc118472361)

[24 Gestione sicurezza 17](#_Toc118472362)

[25 Migrazione dati 17](#_Toc118472363)

[26 Interfaccia dati 17](#_Toc118472364)

[27 Analisi di mercato 18](#_Toc118472365)

[28 Gestione della comunicazione 18](#_Toc118472366)

[28.1 Eventi ricorrenti 18](#_Toc118472367)

[28.2 Verbali previsti e tempistiche 18](#_Toc118472368)

[28.3 Incontri formali 18](#_Toc118472369)

[28.4 Distanza/presenza 18](#_Toc118472370)

[28.5 Gestione materiali 18](#_Toc118472371)

[29 Processo di sviluppo 18](#_Toc118472372)

[29.1 Ciclo di vita del software 18](#_Toc118472373)

[29.2 Differenze tra processi di sviluppo 18](#_Toc118472374)

[30 Conclusioni 18](#_Toc118472375)

[30.1 Progettuali 18](#_Toc118472376)

[30.2 Personali 18](#_Toc118472377)

[30.3 Scolastici 18](#_Toc118472378)

# Abstract

// Obiettivi, risultati, sviluppi futuri e conclusioni

// Riassunto

# Sommario

## Descrizione progetto

Il progetto consiste nello sviluppare un’applicazione mobile per pilotare un rover attraverso dei comandi inviati tramite Bluetooth.

A picture containing wall, indoor, projector

Description automatically generated

Figure 1 - Rover da controllare

## Per chi

## Benefici

## Risultato ottenuto

// Prodotto finito? Tutto ok?

# Introduzione

## Descrizione azienda

### Organigramma

https://www.cptbellinzona.ti.ch/doc/M\_1-12\_Organigramma\_funzionale.pdf

## Scopo

La Scuola d’arti e mestieri di Bellinzona sta sviluppando un piccolo rover pilotato da un radiocomando che comunica mediante una connessione Bluetooth. Nell’anno scolastico 2020/21 uno studente in elettronica del quarto corso ha sviluppato un primo prototipo funzionante del rover e del radiocomando.

Visto l’interesse suscitato da questo progetto, si è deciso di dar seguito allo sviluppo di un nuovo prototipo dove anche l’intera parte meccanica sarà sviluppata e realizzata in sede. In questa seconda fase l’idea è quella di pilotare il rover mediante uno smartphone e non più utilizzare un radiocomando dedicato.

## Obiettivi

Sviluppare un applicativo per Android e iOS che vada a rimpiazzare il telecomando utilizzato al momento.

Questo applicativo dovrà connettersi al dispositivo e inviare i comandi al rover per pilotarlo. Allo stesso momento riceverà costantemente dati dal rover riguardo la percentuale della batteria e distanza da un ostacolo.

Le informazioni ricevute verranno poi mostrate nell’applicazione come informazione per l’utente.

Attraverso l’interfaccia dell’applicazione si dovrà essere in grado di gestire il rover in modo esaustivo.

## Situazione attuale

Al momento entrambe le schede elettroniche imbarcano un microcontrollore e un modulo bluetooth per la comunicazione. Sulla scheda del rover sono presenti inoltre: un buzzer, un sensore di distanza ad ultrasuoni, quattro LED indipendenti e l’elettronica necessaria all’azionamento dei motori. Sul radiocomando, invece, sono stati previsti: due joystick a due assi (X/Y), quattro pulsanti, quattro LED e un display LCD.

La comunicazione tra i due dispositivi avviene mediante lo scambio di stringhe di testo debitamente formattate. I due moduli Bluetooth, accoppiati tra loro in precedenza, gestiscono la comunicazione in modo trasparente per i microcontrollori che, dalla loro prospettiva, scambiano semplici caratteri ASCII attraverso la propria interfaccia seriale.

A picture containing electronics, circuit

Description automatically generated

Figure 2 – Radiocomando

## Benefici attesi

Tra i benefici attesi c’è un miglioramento all’usabilità da parte dell’utente finale, il quale tramite un’interfaccia chiara e funzionale, potrà interagire con il rover più comodamente.

Tutte le funzioni saranno disponibili su un’unica schermata e verranno visualizzate anche le informazioni provenienti dal rover, quali stato della batteria e distanza da un ostacolo.

Gli utenti potranno poi scaricare l’applicativo e installarlo sul proprio telefono così da poter controllare il rover. Con questo sistema di andrà ad eliminare il problema di un unico telecomando, il quale se dovesse scaricarsi o rompersi renderebbe impossibile interfacciarsi con il rover.

Inoltre, si avrebbe la possibilità di aggiungere funzionalità o migliorare quelle presenti, molto più facile tramite un semplice aggiornamento dell’applicazione.

## Limitazioni

Una limitazione che è sopraggiunta subito dopo un’analisi delle tecnologie è legata supporto per l’applicazione ad iOS. Siccome per comunicare tramite vecchie versioni di Bluetooth con dispositivi Apple è necessaria la certificazione MFi[[1]](#footnote-1), si è deciso di scartare il supporto ai dispositivi Apple.

Nonostante il supporto ad iPhone è stato rimosso si è deciso comunque di rimanere su tecnologie multipiattaforma, ovvero che hanno la possibilità di creare applicazioni sia per Android che per iOS. Questo soprattutto perché i vantaggi dello sviluppo nativi sono soprattutto legati alle performance, e l’applicazione in questione è molto leggera.

Inoltre se in futuro la situazione dovesse evolvere e l’implementazione su iOS dovesse diventare una possibilità concreta, avendo sviluppato con tecnologie come Xamarin o React Native, il passaggio ad iOS è molto rapido.

# Introduzione

## Descrizione azienda

// Cosa fa l’azienda

### Organigamma

## Scopo

## Obiettivi

## Benefici attesi

# Vision

// + confronto tra prodotti esistenti

# Situazione attuale

## Come viene fatto il lavoro ora

## Mappa processi aziendali

## Analisi documenti interni

## Confronto vecchio-nuovo

# Analisi

## Use case di contesto

## Activity concettuale

## Requisiti

// generali e riassuntivi

## App ibrida vs nativa

### Nativa

Un app nativa è compilata in uno specifico linguaggio per una specifica piattaforma. Queste applicazioni non sono veramente native, perché questo vorrebbe dire sviluppare un’applicazione diversa per ogni sistema, ma ci si avvicinano molto.

Le applicazioni native possono avere accesso a funzionalità legate all’hardware del dispositivo, come fotocamera, GPS, contatti, ecc.

### Ibrida

Un’applicazione ibrida può girare

Un app ibrida è usa HTML, CSS e JavaScript per creare una versione web dell’applicativo e poi viene visualizzato all’interno del dispositivo.

Al contrario delle applicazioni native, quelle ibride non hanno accesso ad alcune funzionalità del dispositivo. Questo non vuol dire necessariamente che non sia sempre possibile accedergli: Ionic, ad esempio, usa dei plugin di Cordova per integrare le funzionalità native richieste per il funzionamento dei componenti hardware.

## Analisi tecnologica

Sono stati scelti esclusivamente frameworks che supportano lo sviluppo cross-platform, siccome l’applicativo deve funzionare sia su dispositive Android che iOS, e la possibilità di sviluppare due sistemi distinti è stata scartata.

### Xamarin

|  |  |
| --- | --- |
|  |  |
| Sviluppatore | Microsoft |
| Rilascio | 2011 |
| Linguaggio | C#, F# |
| Tipologia | Nativo |
| Open source | Si |
| IDE | Visual Studio |

### Flutter

|  |  |
| --- | --- |
|  |  |
| Sviluppatore | Google |
| Rilascio | 2017 |
| Linguaggio | Dart |
| Tipologia | Nativo |
| Open Source | Si |
| IDE | Indipendente |

### Ionic

|  |  |
| --- | --- |
|  |  |
| Sviluppatore | Drifty |
| Rilascio | 2013 |
| Linguaggio | JavaScript con Angular, React o Vue |
| Tipologia | Ibrido |
| Open Source | Si |
| IDE | Indipendente |

### Cordova

|  |  |
| --- | --- |
|  |  |
| Sviluppatore | Adobe |
| Rilascio | 2019 |
| Linguaggio | HTML5, CSS3, and JavaScript |
| Tipologia | Ibrido |
| Open Source | Si |
| IDE | Indipendente |

### React Native

|  |  |
| --- | --- |
|  |  |
| Sviluppatore | Meta (Facebook) |
| Rilascio | 2015 |
| Linguaggio | JavaScript con React |
| Tipologia | Nativo |
| Open Source | Si |
| IDE | Indipendente |

### Confronto

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Xamarin | | Flutter | Ionic | Cordova | React Native |
| Sviluppatore | Microsoft | Google | | Drifty | Adobe | Facebook |
| Rilascio | 2011 | 2017 | | 2013 | 2009 | 2015 |
| Linguaggio | C#, F# | Dart | | JavaScript con Angular, React o Vue | HTML5, CSS3, and JavaScript | JavaScript con React |
| Nativo/Ibrido | Nativo | Nativo | | Ibrido | Ibrido | Nativo |
| Open source | Si | Si | | Si | Si | Si |
| IDE | Visual Studio | Indipendente | | Indipendente | Indipendente | Indipendente |

## Analisi approfondita

Tutte le tecnologie sono delle valide possibilità per questo progetto ma vorrei evitare le tecnologie ibride siccome andiamo a lavorare con Bluetooth, quindi ad interagire in modo diretto con l’hardware del telefono.

Inoltre ho deciso di scartare Flutter per via del nuovo linguaggio da dover imparare, il quale rallenterebbe notevolmente lo sviluppo iniziale.

### Xamarin

#### Sviluppo

Gli strumenti all’interno di Visual Studio velocizzano il processo di sviluppo con controlli e layout predefiniti. La funzione di “Hot Reloading”, ovvero la possibilità di vedere l’applicazione aggiornata immediatamente senza bisogno di ricompilarla, accelera molto la parte di sviluppo.

#### Performance

A livello di performance è quasi alla pari con le applicazioni sviluppate nativamente per iOS o Android.

#### Orientamento dispositivo

La possibilità di avere schermate diverse a dipendenza se il dispositivo è in orizzontale o in verticale è direttamente integrata nel framework.

#### Dimensioni dispositivo

Supporta vari tipi di dispositivo e si possono configurare facilmente delle viste per determinati dispositivi.

#### Bluetooth

Per connettersi ad un dispositivo che usa Bluetooth Classic, si può aggiungere un plugin chiamato “Plugin.BluetoothClassic[[2]](#footnote-2)”.

#### Emulazione

Visual Studio racchiude un emulatore Android tramite il quale si può vedere l’applicativo e viene aggiornato in tempo reale. L’emulatore gira senza problemi su una macchina dotata di almeno 16GB di RAM[[3]](#footnote-3) e con installato Hyper-V o altri software che assistono alla virtualizzazione, come Intel HAXM.

### React Native

#### Sviluppo

Come per Xamarin, anche React Native supporta la funzione di ‘Hot Reloading’. Performance

A livello di performance è quasi alla pari con le applicazioni sviluppate nativamente per iOS o Android.

#### Orientamento dispositivo

Non è predisposto ad avere la gestione comoda dell’interfaccia a dipendenza dell’orientamento del dispositivo. Per fare ciò è necessario avvalersi di una libreria esterna (react-native-orientation), la quale richiede una configurazione manuale scritta nel linguaggio nativo per il dispositivo (Objective C per iOS o Java per Android).

Tutto questo rende il tutto impossibile da testare senza compilare il tutto e installarlo su un dispositivo.

#### Dimensioni dispositivo

React Native usa un sistema senza unità per definire le dimensioni dei vari elementi, il quale si basa sulla densità di pixel del dispositivo. Questa è una soluzione che può creare dei problemi quando si passa su dispositivi con uno schermo e una densità di pixel molto alti. Ci sono però delle librerie che danno più flessibilità e aiutano a dare un risultato migliore (ad esempio “react-native-size-matters”).

È anche possibile implementare due schermate diverse a dipendenza della dimensione del dispositivo.

#### Bluetooth

Al contrario di Xamarin non c’è una soluzione per la connessione Bluetooth direttamente integrate ma esistono delle soluzioni sviluppate da terzi.

Un esempio è la libreria “react-native-bluetooth-classic[[4]](#footnote-4)” offre un’implementazione altrettanto semplice e veloce quanto quella di Xamarin.

#### Emulazione

React Native non necessita però di un emulatore ma l’applicativo può essere visualizzato su un qualsiasi browser moderno.

### Scelta finale: Xamarin

Xamarin è sembrata la scelta più solida fin dall’inizio, l’unica cosa ache mi frenava inizialmente era l’emulazione in fase di sviluppo che richiede molte risorse e il computer che mi era stato assegnato per lavorare era limitato sotto quel punto di vista.

Dopo aver creato un prototipo ho però notato la possibilità di eseguire il deploy direttamente su un reale telefono Android. Questo fattore non solo ha rimosso la necessità di avere un computer con hardware più performante, ma ha anche dato la possibilità di testare più comodamente le funzioni.

### Considerazioni a posteriori

Xamarin si è rivelato uno strumento molto completo e facile ma poco flessibile quando ho provato ad uscire dalle classiche customizzazioni od operazioni da eseguire.

I componenti che fornisce Xamarin sono molti e hanno incluse molte funzionalità, questo rende lavorare con Xamarin molto intuitivo e rapido.

Per quanto riguarda la personalizzazione alcune volte vengono fornite delle funzioni direttamente sul componente, mentre altre volte bisogna andare a toccare molti componenti per una semplice customizzazione.

Una di queste customizzazioni è stato cambiare il colore del bordo del input di testo da rosa a blu.

Graphical user interface, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

Per eseguire ciò ho dovuto applicare le modifiche e creare le classi seguenti:

Text

Description automatically generated

Figure 3 - Applicato un Effects, il quale permette di customizzare per ogni piattaforma i controlli

Text

Description automatically generated

Figure 4 - Creato l’Effects base, ovvero indipendente dalla piattaforma

Text

Description automatically generated

Figure 5 - Creato l’Effect legato alla piattaforma, in questo caso Android

Un’operazione simile è stata utilizzata per cambiare il colore del cursore.

Graphical user interface, text, application, Word

Description automatically generated

Text

Description automatically generated

Figure 6 - Creato un render specifico per Android

Text

Description automatically generated

Figure 7 - XML contentente le istruzioni per disegnare il nuovo cursore

Nel complesso devo dire che sono soddisfatto della scelta di Xamarin nonostante quanto appena menzionato siccome il tempo speso per customizzare questi dettagli, è stato ampiamente compensato da molte altre funzionalità che hanno accelerato lo sviluppo.

Nel caso si volesse creare un applicativo con un design molto specifico sconsiglierei l’utilizzo di Xamarin.

## Fonti

Qui si possono trovare le fonti utilizzate per l’analisi delle tecnologie.‌

Ravichandran, Adhithi. 2019. “React Native vs. Ionic: Which One Is Right for You?” LogRocket Blog. July 26, 2019. https://blog.logrocket.com/react-native-vs-ionic/.

“Xamarin vs Ionic - Which Is Better?” 2021. August 22, 2021. https://blog.back4app.com/xamarin-vs-ionic/.

Gray, Lerma. 2022. “Xamarin vs Flutter: 3 Critical Lessons Learnt after Using Xamarin.” DevCount.com. May 1, 2022. https://devcount.com/xamarin-vs-flutter/.

Solanki, Jignesh. 2021. “Xamarin vs Ionic for Mobile Application Development in 2022.” Simform - Product Engineering Company. January 22, 2021. https://www.simform.com/blog/xamarin-vs-ionic/.

Ravichandran, Adhithi. 2019. “React Native vs. Ionic: Which One Is Right for You?” LogRocket Blog. July 26, 2019. https://blog.logrocket.com/react-native-vs-ionic/.

“Cordova vs. React Native: A Detailed Analysis | Waldo Blog.” n.d. Www.waldo.com. Accessed November 4, 2022. https://www.waldo.com/blog/cordova-vs-react-native.

“Xamarin vs React Native - Make a Choice in 5 Steps [2022 Guide].” n.d. Brainhub.eu. Accessed November 4, 2022. https://brainhub.eu/library/react-native-vs-xamarin.

Alferd, Sten. 2019. “Flutter vs Xamarin vs React Native — Let the Battle Begin!” Medium. December 3, 2019. https://medium.com/@stenalferd/flutter-vs-xamarin-vs-react-native-let-the-battle-begin-d3e783bb4bf1.

Vallecillo, Yamill. 2022. “Yamill/React-Native-Orientation.” GitHub. October 26, 2022. https://github.com/yamill/react-native-orientation.

Diaz, Jorge Perales. 2021. “How to Integrate Bluetooth LE in Xamarin Forms.” EnigmaMx. April 19, 2021. https://medium.com/enigmamx/how-to-integrate-bluetooth-le-in-xamarin-forms-2dcdf974703a.

“Different Mobile and Desktop Layouts with React.” 2016. Gosha Arinich. December 20, 2016. https://goshacmd.com/different-mobile-desktop-tablet-layouts-react/.

Kadam, Akshay. 2018. “Scaling React Native Apps for Tablets.” React Native Training. May 1, 2018. https://medium.com/react-native-training/scaling-react-native-apps-for-tablets-211de8399cf1.

# Personas

## Strategie aziendali

## Ricerche di mercato

# Qualità

## Norme aziendali

### Regolamento documentazione

### Rilascio software

## ISO

## Standard

// Codica (nomi variabili, attributi db,,...)

// Regole personali

# Pianificazione

## Gantt

## Trello

Ho seguito un approccio Agile, definendo degli sprint di 4 settimane e definendo delle storie (o task) i quali ho raggruppato sono le seguenti macro-categorie:

* Analisi
* Interfaccia
* Funzionalità
* Documentazione

Graphical user interface, text, application, email

Description automatically generated

Figure 8 - Trello

# Mappa dei processi

# Metodologia

// Processo di sviluppo

## Fonte di ispirazione

## Adattamento della metodologia teorica

# Analisi rischi

## Strumenti

// what-if, causa-effetto, matrice rischi, ...

## Rischi accaduti

# Volumi / Quantità / Flussi

## Scalabilità

## Volumi futuri

## Stress test

## Performance test

# Comunicazione

## Frequenza riunioni

La frequenza delle riunioni è ogni due settimane e avvenivano effettuate di persona.

## Modello verbale

## Modalità e contenuto incontri

# Reportistica

# Migrazione dati

# Sviluppi futuri

## Funzionalità mancanti

## Lista bug esistenti

## Nice to have

# Analisi dei rischi

## Uso strumenti

// (what-if, causa-effetto, ...)

## Catalogo rischi

## Rischi accaduti

# Macro requisiti

# Analisi costi e benefici

# Mockup

## Confronto iniziale – finale

# Diagrammi

# Gestione sicurezza

# Migrazione dati

# Interfaccia dati

# Analisi di mercato

# Gestione della comunicazione

## Eventi ricorrenti

// Venerdì pomeriggio

## Verbali previsti e tempistiche

## Incontri formali

## Distanza/presenza

## Gestione materiali

# Processo di sviluppo

## Ciclo di vita del software

## Differenze tra processi di sviluppo

# Conclusioni

## Progettuali

## Personali

## Scolastici

1. Vedi approfondimento nel Manuale Tecnico, sezione relativa a Bluetooth [↑](#footnote-ref-1)
2. https://github.com/rostislav-nikitin/Plugin.BluetoothClassic [↑](#footnote-ref-2)
3. Riferimento personale in base alle risorse utilizzate avendo l’ambiente di sviluppo e la macchina virtuale avviata. [↑](#footnote-ref-3)
4. https://www.npmjs.com/package/react-native-bluetooth-classic [↑](#footnote-ref-4)