Forgia Paolo

SSSE  SIG

Manuale di progetto

Indice

[1 Abstract 4](#_Toc118822813)

[2 Sommario 4](#_Toc118822814)

[2.1 Descrizione progetto 4](#_Toc118822815)

[2.2 Per chi 4](#_Toc118822816)

[2.3 Benefici 4](#_Toc118822817)

[2.4 Risultato ottenuto 4](#_Toc118822818)

[3 Introduzione 4](#_Toc118822819)

[3.1 Descrizione azienda 4](#_Toc118822820)

[3.1.1 Organigramma 4](#_Toc118822821)

[3.2 Scopo 5](#_Toc118822822)

[3.3 Obiettivi 5](#_Toc118822823)

[3.4 Benefici attesi 5](#_Toc118822824)

[3.5 Limitazioni 5](#_Toc118822825)

[4 Vision 6](#_Toc118822826)

[5 Situazione attuale 6](#_Toc118822827)

[5.1 Come viene fatto il lavoro ora 6](#_Toc118822828)

[5.2 Mappa processi aziendali 7](#_Toc118822829)

[5.3 Analisi documenti interni 7](#_Toc118822830)

[5.4 Confronto vecchio-nuovo 7](#_Toc118822831)

[6 Analisi 8](#_Toc118822832)

[6.1 Use case 8](#_Toc118822833)

[6.1.1 Use Case di contesto 8](#_Toc118822834)

[6.1.2 Controlli 8](#_Toc118822835)

[6.1.3 Impostazioni 9](#_Toc118822836)

[6.2 State diagram 9](#_Toc118822837)

[6.2.1 Connessione Bluetooth 9](#_Toc118822838)

[6.3 Class diagram 10](#_Toc118822839)

[6.4 Requisiti 10](#_Toc118822840)

[6.5 App ibrida vs nativa 10](#_Toc118822841)

[6.5.1 Nativa 10](#_Toc118822842)

[6.5.2 Ibrida 11](#_Toc118822843)

[6.6 Analisi tecnologica 11](#_Toc118822844)

[6.6.1 Xamarin 11](#_Toc118822845)

[6.6.2 Flutter 11](#_Toc118822846)

[6.6.3 Ionic 11](#_Toc118822847)

[6.6.4 Cordova 11](#_Toc118822848)

[6.6.5 React Native 12](#_Toc118822849)

[6.6.6 Confronto 12](#_Toc118822850)

[6.7 Analisi tecnologica approfondita 12](#_Toc118822851)

[6.7.1 Xamarin 12](#_Toc118822852)

[6.7.2 React Native 13](#_Toc118822853)

[6.7.3 Scelta finale: Xamarin 14](#_Toc118822854)

[6.7.4 Considerazioni a posteriori 14](#_Toc118822855)

[6.7.5 Fonti 16](#_Toc118822856)

[7 Personas 17](#_Toc118822857)

[7.1 Strategie aziendali 17](#_Toc118822858)

[7.2 Ricerche di mercato 17](#_Toc118822859)

[8 Qualità 17](#_Toc118822860)

[8.1 Norme aziendali 17](#_Toc118822861)

[8.1.1 Regolamento documentazione 17](#_Toc118822862)

[8.1.2 Rilascio software 17](#_Toc118822863)

[8.2 ISO 17](#_Toc118822864)

[8.3 Standard 17](#_Toc118822865)

[9 Pianificazione 17](#_Toc118822866)

[9.1 Gantt 17](#_Toc118822867)

[9.1.1 Iniziale 17](#_Toc118822868)

[9.1.2 Finale 17](#_Toc118822869)

[9.2 Trello 17](#_Toc118822870)

[10 Mappa dei processi 18](#_Toc118822871)

[11 Metodologia 18](#_Toc118822872)

[11.1 Fonte di ispirazione 18](#_Toc118822873)

[11.2 Adattamento della metodologia teorica 18](#_Toc118822874)

[12 Analisi rischi 18](#_Toc118822875)

[12.1 Strumenti 18](#_Toc118822876)

[12.2 What-if 19](#_Toc118822877)

[12.3 Rischi accaduti 19](#_Toc118822878)

[13 Volumi / Quantità / Flussi 19](#_Toc118822879)

[13.1 Scalabilità 19](#_Toc118822880)

[13.2 Volumi futuri 19](#_Toc118822881)

[13.3 Stress test 20](#_Toc118822882)

[13.4 Performance test 20](#_Toc118822883)

[14 Comunicazione 20](#_Toc118822884)

[14.1 Modalità e contenuto incontri 20](#_Toc118822885)

[15 Reportistica 20](#_Toc118822886)

[16 Migrazione dati 20](#_Toc118822887)

[17 Macro requisiti 20](#_Toc118822888)

[18 Analisi costi e benefici 20](#_Toc118822889)

[19 Mockup 21](#_Toc118822890)

[19.1 Iniziale 21](#_Toc118822891)

[19.2 Finale 22](#_Toc118822892)

[19.3 Prodotto finale 23](#_Toc118822893)

[19.4 Confronto mockup iniziale – finale 23](#_Toc118822894)

[19.5 Confronto mockup finale – prodotto realizzato 23](#_Toc118822895)

[20 Gestione sicurezza 24](#_Toc118822896)

[21 Migrazione dati 24](#_Toc118822897)

[22 Interfaccia dati 24](#_Toc118822898)

[23 Analisi di mercato 25](#_Toc118822899)

[24 Gestione della comunicazione 25](#_Toc118822900)

[24.1 Eventi ricorrenti 25](#_Toc118822901)

[24.2 Verbali previsti e tempistiche 25](#_Toc118822902)

[24.3 Incontri formali 25](#_Toc118822903)

[24.4 Distanza/presenza 25](#_Toc118822904)

[24.5 Gestione materiali 25](#_Toc118822905)

[25 Processo di sviluppo 25](#_Toc118822906)

[25.1 Ciclo di vita del software 25](#_Toc118822907)

[25.2 Differenze tra processi di sviluppo 25](#_Toc118822908)

[26 Sviluppi futuri 26](#_Toc118822909)

[26.1 Funzionalità mancanti 26](#_Toc118822910)

[Lista bug esistenti 26](#_Toc118822911)

[26.2 Nice to have 26](#_Toc118822912)

[27 Conclusioni 26](#_Toc118822913)

[27.1 Progettuali 26](#_Toc118822914)

[27.2 Personali 26](#_Toc118822915)

[27.3 Scolastici 26](#_Toc118822916)

# 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 - 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.

## 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.

# Vision

Graphical user interface, text, application, email

Description automatically generated

# Situazione attuale

## Come viene fatto il lavoro ora

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 – Radiocomando

## Mappa processi aziendali

## Analisi documenti interni

All’inizio del progetto mi sono stati consegnati i seguenti documenti (trovabili sotto ‘Allegati’):

* Annuncio di progetto CPT Cingolato.docx
* Quaderno dei compiti.pdf
* Progetto Rover.zip

‘Progetto Rover.zip’ contiene tutti i files e i documenti creati dallo studente che ha realizzato il rover e il radiocomando. Leggere questi documenti è stato importante soprattutto nella fase iniziale per comprendere il funzionamento di varie parti della comunicazione tra radiocomando e rover.

## Confronto vecchio-nuovo

Il bitrate è cambiato da 9’600 a 115’200 bit/secondo.

# Analisi

## Use case

### Use Case di contesto

Diagram

Description automatically generated

Diagramma di contesto delle sezioni principali dell’applicazione.

### Controlli

Diagram

Description automatically generated

Diagramma dei controlli che vengono scambiati tra applicazione e rover.

### Impostazioni

Diagram

Description automatically generated

Diagramma delle impostazioni che l’utente può impostare.

## State diagram

### Connessione Bluetooth

Diagram

Description automatically generated

State diagram della gestione della connettività Bluetooth all’interno dell’applicazione.

## Class diagram

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Classe | Descrizione |
| Bluetooth | Gestione di Bluetooth sul dispositivo.   * Accendere Bluetooth * Ricercare dispositivi * Collegarsi ad un dispositivo * Inviare dati al dispositivo * Ecc. |
| Commands | Genera le stringhe di testo conformi alle regole prestabilite. Si occupa anche di convertire stringhe di testo contenente comandi, validare se sono conformi alle regole e convertirle in classi ‘Data’. |
| Data | Una classe che rappresenta uno dei possibili dati in entrata. |
| RepeatableAction | Classe astratta che definisce la logica per delle azioni ripetibili fino a che non viene cancellata l’operazione. |
| BuzzerAction | Estende ‘RepeatableAction’ e definisce l’azione da eseguire in ripetizione per attivare e disattivare il buzzer. |
| LeftEngineAction | Estende ‘RepeatableAction’ e definisce l’azione da eseguire in ripetizione per definire la velocità del motore sinistro. |
| RightEngineAction | Estende ‘RepeatableAction’ e definisce l’azione da eseguire in ripetizione per definire la velocità del motore destro. |

## 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 tecnologica 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:

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

Figure - 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

Graphical user interface, text, application, email

Description automatically generated

Figure - Creato un render specifico per Android

A picture containing timeline

Description automatically generated

Figure - 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

// Intervistare studente -> cosa sarebbe bello avere in future, considerazioni varie (applicativo, funzionalità, usabilità)

// Cambio titolo

// Cosa c’è sul mercato? (Rover da mettere insieme e pilotare)

# Qualità

## Norme aziendali

### Regolamento documentazione

### Rilascio software

// Pubblicazione su Play Store

## ISO

## Standard

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

// Regole personali

# Pianificazione

## Gantt

### Iniziale

Chart, timeline

Description automatically generated

### Finale

## 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 – Trello (Sprint 1 – 3)

# Mappa dei processi

# Metodologia

// Processo di sviluppo

## Fonte di ispirazione

Per la metodologia mi sono ispirato a Scrum siccome l’ho già utilizzata diverse volte in ambito lavorativo e siccome la suddivisione del lavoro in tanti piccoli task e pianificare il lavoro per un periodo relativamente corto è una cosa che mi aiuta.

## Adattamento della metodologia teorica

Lavorando da solo invece che in un team ho dovuto tagliare alcuni elementi classici di Scrum, come ad esempio la daily standup, la sprint review, ecc.

Ho invece definito lo sprint da 4 settimane, e inserivo le storie che pianificavo di fare in quel periodo, che fosse il periodo corrente o uno futuro.

Alla fine dello sprint, nel caso una storia fosse stata completata solamente a metà, la dividevo in 2 storie più piccole, così da segnare cosa era stato realmente fatto in quello sprint.

Non avevo una gestione dello stato della storia (ToDo, In Progress, Done) ma consideravo l’insieme delle storie come completato una volta che passavo allo sprint seguente, se non fosse così veniva spostata in quello successivo o suddivisa.

Il backlog è stato rimosso e al suo posto inserivo le storie in uno sprint futuro. Siccome io avevo sia il ruolo di developer che di Scrum master, quando creavo delle storie, veniva subito pianificato quando era necessario finirla.

## Adattamento nel corso del progetto

Avevo inizialmente definito gli sprint da 2 settimane, in modo da avere una sorta di Sprint review ogni volta che vedevo il cliente e lo aggiornavo sui progressi fatti.

Questo è però risultato troppo limitante, alcune storie, nonostante piccole, prendevano più di 2 settimane. Invece di ridurre ulteriormente le storie per adattarle agli sprint, ho deciso di passare a sprint più lunghi.

Questo ha ridotto notevolmente il tempo speso di lavoro impiegato all’aggiustamento delle storie e pianificazione degli sprint futuri.

# Analisi rischi

## Strumenti

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

## What-if

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | What if | Manifestazione | Conseguenze | Contromisure |
| 1 | Risorse del pc troppo ridotte | L’emulazione richiede troppe risorse. | Impossibilità testare ciò è stato sviluppato. | Ricercare altri metodi per testare l’applicativo. |
| 2 | Smarrimento o rottura di componenti del rover | Durante il trasporto a casa di uno o più componenti, questi vengono danneggiati o persi. | Impossibilità di lavorare per un periodo di tempo.  Nuovi costi per comprare il componente. | Acquistare in anticipo dei pezzi di ricambio.  Portare via solo il minimo indispensabile. |
| 3 | Comunicazione tra telefono e rover non funziona | La libreria non comunica correttamente con il modulo Bluetooth del rover. | Necessità di cambiare la libreria utilizzata e dover riscrivere parte del codice. | Appena aggiunta la libreria provare ad inviare dei comandi al rover per accertarsi il corretto funzionamento. |
| 4 | Modulo Bluetooth incompatibile con alcuni modelli di telefoni/sistemi operativi | Il modulo Bluetooth HC-05 utilizzato è considerato legacy e questo potrebbe creare dei problemi con telefoni più recenti. | Impossibilità di scambiare comandi tra telefono e rover. | Avere un modulo Bluetooth più recente come alternativa.  Ridurre il target di dispositivi supportati. |

## Rischi accaduti

Rischio (4) è accaduto in modo parziale. Il dispositivo necessita della certificazione MFi per comunicare con i dispositivi Apple e il modulo HC-05 non dispone di quest’ultima, per questo motivo iOS è stato scartato dai sistemi operativi da supportare.

# Volumi / Quantità / Flussi

## Scalabilità

// Più rover nella stessa stanza ma controllare solo uno (quello corretto)

// Limite bluetooth se ci sono troppi rover vicini

## Volumi futuri

// Logging (logging 1 anno?)

// Pulisci dati (cache) ?

## Stress test

// Come funziona con altri dispositivi Bluetooth collegati

## Performance test

// Latenza dei comandi, interferenze

// Latenza su varie distanze

# Comunicazione

## Modalità e contenuto incontri

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

// Descrivere cosa avviene

// Non viene effettuato nessun verbale.

# Reportistica

# Migrazione dati

// Export

# Macro requisiti

# Analisi costi e benefici

// Costi -> telefoni Android, tempo di sviluppo

// Benefici, risparmio su costi per costruire radiocomando

# Mockup

## Iniziale

Graphical user interface

Description automatically generated

Figure - Mockup iniziale (telefono)

Graphical user interface, application

Description automatically generated

Figure - Mockup iniziale (tablet)

## Finale

Graphical user interface, application

Description automatically generated

Figure - Mockup finale (telefono)

Il cambiamento maggiore è stata l’eliminazione della versione tablet, la quale è stata rimandata.

Per quanto riguarda i mockup del telefono sono rimasti quasi invariati, l’unico cambiamento è stato relativo al controllo direzionale del rover.

Al seguito di un confronto con il cliente, il quale ha evidenziato che i motori sinistro e destro sono distinti, e che possono essere controllati solamente in avanti e indietro con un range di velocità, si è allora deciso di cambiare la croce direzionale in due slider.

## Prodotto finale

Graphical user interface, application

Description automatically generated

Figure - Prodotto finale (telefono)

Il prodotto realizzato rispecchia molto fedelmente il mockup finale, sono stati effettuati alcuni accorgimenti, quali: una migliore gestione degli spazi e delle dimensioni di bottoni e testi; ed è stata aggiunta una variazione di colore agli slider a dipendenza del tipo di input che mandano: blu – fermo, rosa – indietro e verde – avanti.

Questa aggiunta del colore degli slider migliora l’usabilità. Un altro miglioramento all’usabilità è stato aumentare la lunghezza degli slider in modo da consentire un controllo più preciso.

# Gestione sicurezza

Al primo avvio dell’applicativo, all’utente apparirà un popup tramite il quale potrà consentire o rifiutare l’accesso all’applicazione a Bluetooth, questo processo è interamente gestito dal sistema operativo.

Il modulo Bluetooth sul rover sarà configurato con un nome e un PIN.

Nell’applicativo ci sarà una pagina dedicata tramite la quale l’utente potrà inserire il nome con il quale il dispositivo Bluetooth è configurato, così da non collegarsi accidentalmente ad un altro dispositivo.

Graphical user interface, application

Description automatically generated

Figure - Schermata per configurare il nome del dispositivo

Nel caso il dispositivo con il nome definito non venisse trovato verrà visualizzato un popup che avviserà l’utente.

Graphical user interface, text, application

Description automatically generated

# Migrazione dati

Per questo progetto non è stata necessaria nessuna migrazione di dati.

# Interfaccia dati

# Analisi di mercato

# Gestione della comunicazione

## Eventi ricorrenti

Ci sono stati incontri regolari con il cliente, i quali avvenivano il venerdì pomeriggio ogni due settimane.

Questi incontri servivano ad aggiornare il cliente sullo stato di avanzamento, ad avere uno scambio di pareri ed a chiarire alcuni dubbi sorti nei giorni precedenti.

## Verbali previsti e tempistiche

## Incontri formali

Una volta al mese sono stati pianificati degli incontri formali nei quali sono stati scambiati dei feedback con il cliente.

// TODO: completare

## Distanza/presenza

Il progetto è stato portato avanti per la maggior parte a distanza, la parte effettuata in presenza equivale circa ad un 10% del tempo complessivo.

## Gestione materiali

// Modulo Bluetooth, telefono

# Processo di sviluppo

## Ciclo di vita del software

## Differenze tra processi di sviluppo

# Sviluppi futuri

## Funzionalità mancanti

// Programs non finito ?

## Lista bug esistenti

### Switch in Dark Mode

Il componente Switch non viene visualizzato correttamente se il telefono ha la Dark Mode attivata.

Graphical user interface

Description automatically generated with medium confidence

Figure - Switch con Dark Mode attiva

Graphical user interface

Description automatically generated with low confidence

Figure - Switch con Dark Mode disattivata

## Nice to have

### Tablet

Il supporto per i tablet è stato inizialmente menzionata come possibilità, ma infine scartata per dare spazio a funzionalità più importanti.

Il dispositivo principale in cui sarebbe stato utilizzato è comunque lo smartphone.

### Portrait Mode

La possibilità di utilizzare il dispositivo in modalità portrait (in orientamento orizzontale) era anche stata menzionata ma anch’essa scartata perché meno importante.

### Modulo Bluetooth con certificazione MFi

Vista la limitazione legata ai dispositivi Apple per il modulo Bluetooth corrente, sarebbe interessante avere un nuovo modulo certificato MFi in modo da poter sviluppare l’applicativo anche per iOS.

# 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)