Progetto di Ingegneria del Software 2025/26

Università Ca' Foscari Venezia

D4: Piano di Testing

0.5

V.M.F

17/11/2025

Document Informations

NomeProgetto	Acronimo	
Deliverable	Piano di Testing	
Data di Consegna	18/11/2025	
Team Leader	Francesco Pezzuto	902320@stud.unive.it
Team members	Matteo Tarushi 900554@stud.unive.it Victor Hortopan 900555@stud.unive.it	

Document History

Version	Issue Date	Stage	Changes	Contributors
0.5	17/11/2025	Draft	Definizione del documento di Piano di Testing	PF,HV,TM

Indice

1.Introduzione

1.1 Overview del progetto

Il presente documento descrive il **Piano di Testing** del sistema, il progetto consiste nello sviluppo di un'applicazione mobile, **GreenZone**, il cui obiettivo è quello di facilitare la scoperta e la promozione di luoghi sostenibili nella propria area (come punti di ricarica elettrica, fontanelle, aree verdi, negozi bio).

L'app offre una mappa interattiva che consente di:

- Visualizzare le diverse categorie di luoghi sostenibili;
- Calcolare percorsi a piedi o in bici verso tali punti;
- Filtrare i risultati per categoria o distanza;
- Segnalare nuovi luoghi o correggere informazioni errate;
- Salvare i luoghi preferiti per un accesso rapido

L'obiettivo principale è favorire comportamenti sostenibili nella vita quotidiana, rendendo più semplice per gli utenti contribuire al benessere ambientale della propria comunità.

L'obiettivo del **Piano di Testing** è definire in modo chiaro e strutturato le attività necessarie per la verifica e validazione del software, garantendo che esso rispetti i requisiti funzionali e non funzionali indicati nel **Documento dei Requisiti (D3)** e le pianificazioni del **Piano di Progetto (D2).**
Il documento specifica:

- Le tecniche di testing adottate nel processo di sviluppo;
- Le modalità di verifica dei requisiti;
- La definizione dettagliata dei casi di test;
- La pianificazione delle attività di testing rispetto alle milestone di progetto;
- Le procedure per registrazione, monitoraggio e aggiornamento dei test;
- I requisiti hardware e software necessari per condurre le attività;
- I vincoli che condizionano le operazioni di testing;
- Eventuali test ulteriori (prestazioni, sicurezza, usabilità).

Il **Piano di Testing** è un documento **dinamico** che può essere aggiornato durante l'evoluzione del progetto in caso di modifiche ai requisiti o/e alla progettazione.

2.Glossario

- **Test Case**

  Insieme strutturato di precondizioni, input, azioni da eseguire, output attesi e post-condizioni utilizzato per verificare uno specifico requisito.

- **Test di Regressione**

  Test rieseguiti dopo modifiche al codice per assicurarsi che le funzionalità già implementate continuino a comportarsi correttamente.

- **Black-Box Testing**

  Tecnica di testing basata sulle specifiche funzionali senza considerare l'implementazione interna.

- **White-Box Testing**

  Tecnica di testing che utilizza la conoscenza della struttura interna del codice per massimizzare la copertura (rami, condizioni, cicli).

- **Google Maps API**

  Servizi forniti da Google per l'integrazione di mappe interattive e funzionalità di geolocalizzazione.

- **Firebase**

  Piattaforma Google utilizzata per la gestione del back-end, del dataabse in tempo reale e dell'autenticazione degli utenti.

- **Bug**

  Qualsiasi comportamento del sistema che non rispetta i requisiti, causa errori o produce risultati inattesi.

- **Ambiente di Test**

  Configurazione hardware, software e dati necessari per eseguire i test.

- **GitHub**

  Piattaforma online per il versionamento del codice e la collaborazione tra sviluppatori.

- **Google Drive**

Servizio cloud di Google utilizzato per la condivisione e l'archiviazione della documentazione di progetto.

- **Excel**

  Software di Microsoft Office utilizzato per l'analisi dei dati raccolti dai questionari e la creazione di grafici e tabelle di sintesi.

- **Word**

  Software di Microsoft Office utilizzato per creare, modificare e formattare documenti come lettere, relazioni, libri, ecc.

- **Whatsapp**

  Applicazione di messaggistica utilizzata dal team per la comunicazione e il coordinamento rapido delle attività.

- **Discord**

  Applicazione social di messagistica istantanea tramite chiamate vocali, videochiamate, messaggi di testo e contenuti multimediali

- **Google form**

  Strumento di Google per la creazione e la distribuzione di questionari, utilizzato per la raccolta dei feedback dagli utenti e stakeholder.

3.Tecniche di Testing

Per garantire un controllo completo sulla qualità del software, il progetto GreenZone utilizza un insieme mirato di tecniche di testing che coprono sia il comportamento esterno del sistema sia la sua struttura interna.

Le tecniche adottate sono complementari tra loro e permettono di individuare errori logici, malfunzionamenti, difetti di implementazione e incoerenze rispetto ai requisiti.

In particolare, verranno applicati congiuntamente il **Black-box Testing** e il **White-box Testing**, poiché la loro combinazione consente sia di verificare la correttezza funzionale del sistema dal punto di vista dell'utente, sia di controllare la logica interna del codice e la gestione dei casi limite.

A queste attività si affiancano i **Test di Regressione**, necessari per assicurare che l'introduzione di nuove funzionalità o modifiche non comprometta il comportamento delle parti già validate.

3.1 Black-Box Testing

Il **Black-box Testing** verifica il comportamento esterno del sistema sulla base dei requisiti funzionali.

In questo approccio il codice non viene analizzato: i test si concentrano esclusivamente sugli input e sugli output attesi.

Le principali strategie utilizzate sono:

- **Equivalence Partitioning**: suddivisione degli input in classi di equivalenza per ridurre il numero di test mantenendo una buona copertura.
- **Boundary Value Analysis**: verifica del comportamento ai limiti degli intervalli di input, dove più spesso emergono errori.
- **Pairwise Testing**: selezione di combinazioni significative di valori d'ingresso quando sono presenti molte variabili.

Il Black-box Testing consente di valutare la correttezza funzionale del sistema e l'aderenza alle specifiche del D3.

3.2 White-Box Testing

Il **White-box Testing** analizza la struttura interna del codice e permette di verificare la correttezza logica delle implementazioni.

Le tecniche principali includono:

- **Copertura delle istruzioni**
- **Copertura dei rami decisionali**
- **Copertura delle condizioni**
- **Copertura dei cammini**

Questo approccio consente di individuare difetti logici, condizioni non gestite, errori nei cicli o nei percorsi alternativi.

3.3 Test di Regressione

Il Test di Regressione verifica che, a seguito di modifiche, refactoring o nuove funzionalità, il sistema continui a funzionare correttamente.

Questa tecnica consiste nel rieseguire periodicamente i casi di test già superati, garantendo che:

- Nessuna funzionalità precedentemente valida venga compromessa;
- L'introduzione di nuovi moduli non generi anomalie nelle parti esistenti.

Il test di regressione viene eseguito al termine di ogni incremento dello sviluppo.

3.4 Verifica Statica

La verifica statica consente di individuare difetti nel codice o nella documentazione **prima dell'esecuzione del software**.

Le tecniche utilizzate sono:

- **Inspection**:

  Analisi sistematica del codice tramite checklist, con ruoli specifici (autore, lettore, ispettori, moderatore).
  Consente di rilevare errori, omissioni, incoerenze e violazioni delle specifiche.

- **Walkthrough**:

  Lettura critica e collaborativa del codice, simulandone il flusso logico per individuare errori e migliorare la chiarezza e la manutenibilità.

4. Criteri per la scelta dei test case

La scelta dei casi di test avviene adottando criteri mirati che assicurano una copertura efficace delle funzionalità e dei comportamenti interni del sistema, evitando ridondanze e garantendo la capacità dei test di individuare difetti significativi.

4.1 Criteri basati sui requisiti

Ogni requisito funzionale viene trasformato in uno o più casi di test che coprono:

- il comportamento atteso nelle condizioni normali;
- gli scenari alternativi o di errore;
- i casi limite individuati nella specifica del requisito.

La selezione si concentra sugli aspetti che hanno maggiore impatto sulla correttezza del sistema.

4.2 Selezione dei dati di test

Gli input vengono suddivisi in insiemi che rappresentano comportamenti analoghi, da ciascuna classe si seleziona un solo valore rappresentativo, riducendo i test inutili.

Vengono privilegiati gli input ai confini degli intervalli validi e non validi, dove è più probabile che emergano malfunzionamenti.

Nel caso di funzionalità con più parametri, si selezionano solo le combinazioni di input più significative, evitando l'esplosione di test non necessari.

4.3 Criteri basati sulla struttura del codice

Per i moduli che richiedono un controllo interno più approfondito, i test vengono scelti in modo da attraversare:

- I principali rami decisionali;
- Le condizioni critiche;
- I percorsi alternativi di esecuzione.

Non vengono analizzati tutti i percorsi possibili, ma solo quelli che hanno rilevanza per individuare errori logici o condizioni non gestite.

4.4 Criteri per la regressione

Per il test di regressione vengono selezionati:

- I test relativi alle funzionalità più critiche;
- I test delle aree modificate nello sprint corrente;
- I test associati a bug risolti;
- Un insieme minimo di test che copre i flussi principali dell'applicazione.

Questo garantisce stabilità dopo ogni aggiornamento del sistema.

5.Modalità di verifica dei singoli requisiti

La verifica dei requisiti ha l'obiettivo di accertare che tutte le funzionalità descritte nel Documento dei Requisiti (D3) siano state implementate correttamente, siano coerenti con le aspettative e non presentino malfunzionamenti.

Il processo di verifica si articola in due fasi principali: verifica statica e verifica dinamica, applicate in modo complementare per coprire sia la correttezza formale sia il comportamento operativo del sistema.

5.1 Verifica statica dei requisiti

Nella verifica statica vengono esaminati i documenti e il codice dell'app prima di eseguire i test veri e propri.

L'obiettivo è individuare incoerenze, omissioni o scelte implementative rischiose che potrebbero generare errori runtime.

Nel progetto **GreenZone** questa fase riguarda:

- Coerenza tra requisiti e architettura dell'app, ad esempio verificando che ogni requisito (RF-01 Login, RF-03 Mappa, RF-06 Segnalazioni, ecc.) abbia un corrispondente modulo nell'architettura Android (Activity, ViewModel, Repository);
- Controllo dei flussi critici, come registrazione/login con Firebase Authentication, caricamento e caching dei luoghi sostenibili e gestione dei permessi di localizzazione;
- Inspection del codice dei moduli più delicati, come la logica per ottenere il percorso tramite Google Maps Directions API o la validazione dei campi di input per segnalazioni e profilo;
- Walkthrough dei flussi principali, simulando manualmente cosa accade, ad esempio, quando un utente crea una nuova segnalazione, la modifica o la elimina.

Questa verifica permette di individuare problemi già a livello di progettazione e implementazione, riducendo i difetti da correggere nella fase di testing dinamico.

5.2 Verifica dinamica dei requisiti

La verifica dinamica consiste nell'eseguire l'app GreenZone su emulatori e dispositivi Android reali per controllare che ogni requisito del D3 sia rispettato.

Ogni requisito funzionale è associato a uno o più test case, che verificano:

- Precondizioni (ex: utente autenticato, connessione attiva, permessi GPS concessi);
- Comportamento dell'interfaccia (UI) nelle varie schermate;
- Corretta comunicazione con Firebase e con le API esterne (Google Maps, geolocalizzazione);
- Gestione degli errori (assenza di rete, input non validi, coordinate mancanti, ecc.).

Nella verifica dinamica del progetto **GreenZone**, saranno coinvolti:

- **RF-01 / RF-02 – Registrazione e Login**

  Verifica che Firebase risponda correttamente a credenziali valide/non valide e che la UI guidi l'utente in modo corretto.

- **RF-03 – Visualizzazione mappa**

  Controllo che la posizione venga ottenuta e che i marker caricati dal database appaiano correttamente sulla mappa.

- **RF-04 – Filtri**

  Verifica che selezionando una categoria (es. fontanelle, punti di ricarica) la mappa si aggiorni immediatamente mostrando solo i marker rilevanti.

- **RF-05 – Percorso sostenibile**

  Controllo della richiesta alla Maps Directions API, della corretta visualizzazione del percorso e del comportamento in caso di mancanza del GPS.

- **RF-06 – Segnalazioni**

  Verifica che l'inserimento avvenga correttamente, che la posizione venga rilevata dall'utente e che il nuovo luogo compaia o entri nello stato "in attesa".

- **RF-08 / RF-09 – Preferiti e profilo**

  Controllo che l'aggiunta/rimozione dei preferiti aggiorni Firebase e che la schermata profilo mostri correttamente i dati dell'utente.

Oltre ai requisiti funzionali, la verifica dinamica include il controllo dei requisiti non funzionali:

- Tempi di risposta (caricamento mappa, query Firebase);

- Correttezza della gestione permessi Android (localizzazione, rete);
- Usabilità dell'interfaccia, verificando che la navigazione sia chiara e coerente.

Infine, viene eseguito il **test di regressione**: ogni volta che viene introdotta una modifica, vengono ripetuti i test delle funzionalità già implementate (login, mappa, preferiti, segnalazioni) per verificare che continuino a funzionare senza regressioni.

6.Schedule del Testing

Il testing di GreenZone viene pianificato in coerenza con lo sviluppo incrementale del progetto.

Ogni incremento funzionale prevede una fase di verifica statica (inspection) seguita da una fase di verifica dinamica (esecuzione dei test case).

La pianificazione tiene conto delle scadenze dei deliverable del corso e della disponibilità dei membri del team.

6.1 Pianificazione temporale del testing

Lo svolgimento del testing viene distribuito lungo tutto il ciclo di vita del progetto.

La seguente tabella riassume la suddivisione temporale delle attività principali:

Periodo	Attività	Dettagli	Responsabili
Settimana 1-2	Analisi e definizione dei test case	Stesura, analisi e definizione dei criteri	Tutto il team
Settimana 3-4	Verifica statica	Inspection del codice iniziale, walktrough dei flussi principali	Tutto il team
Settimana 4-6	Testing dinamico - Primo ciclo	Esecuzione test su RF-01, RF-02, RF-03, RF-04	Victor Hortopan (Database), Matteo Tarushi (UI), Francesco Pezzuto (Coordinamento)
Settimana 6-8	Testing dinamico - Secondo ciclo	Verifica Maps API e segnalazioni	Tutto il team
Settimana 9	Test di regressione	Riesecuzione test precedenti dopo le modifiche attuate	Tutto il team

Settimana 10	Testing finale e validazione	Test non funzionali: prestazioni, rete usabilità	Tutto il team
Consegna finale	Consolidamento risultati e allineamento finale	Compilazione report dei test e correzioni finali	Francesco Pezzuto

Nota: la tabella 6.1 rappresenta la pianificazione complessiva del testing, che potrebbe entrare in conflitto con lo schema Agile adottato nel D2 (Piano di progetto) nel caso in cui non venissero rispettate le scadenze di ogni sprint organizzativo.

Durante gli sprint, i test verranno eseguiti e adattati man mano che le funzionalità vengono sviluppate.

Se dovessero verificarsi differenze tra pianificazione e tempi reali degli sprint, la schedule sarà aggiornata per mantenere coerenza con l'approccio Agile.

6.2 Risorse allocate al testing

Le seguenti risorse vengono dedicate al testing:

Ruolo	Membro	Responsabilità nel testing
Testing Lead	Francesco Pezzuto	Coordinamento dei test, revisione dei documenti, regressione
Sviluppo + Testing Firebase	Victor Hortopan	Test di autenticazione, database e segnalazioni
UI/UX + Testing dinamico	Matteo Tarushi	Test grafico, flussi utente e usabilità
Tutti	Team completo	Esecuzione dei test case, walktrough e inspection

7.Procedure di registrazione dei test

La registrazione dei test ha lo scopo di documentare l'esito delle prove eseguite e garantire la tracciabilità tra i requisiti del sistema e i test case progettati.
Ogni test case viene eseguito e registrato all'interno del registro dei test del progetto, che raccoglie in maniera ordinata le informazioni necessarie al monitoraggio dell'avanzamento del testing.

Ad ogni esecuzione vengono annotati il codice del test case (ad esempio TC-03), il requisito verificato, la versione dell'app testata, la data, il nome del tester e l'esito dell'esecuzione.
In caso di mancato superamento del test, il tester descrive l'anomalia riscontrata e apre un ticket nel sistema di gestione dei problemi del progetto, così da permettere allo sviluppatore responsabile di intervenire e correggere il difetto.
Una volta effettuata la correzione, il test case viene rieseguito e il registro aggiornato con il nuovo esito.

7.1. Test Case

Di seguito sono riportati i test case progettati per verificare i requisiti funzionali dell'applicazione GreenZone:

Campo	Descrizione
ID Test	TC-01
Requisito associato	RF-01 – Registrazione
Obiettivo del test	Verificare che l'utente possa registrare correttamente un nuovo account
Pre-Condizione	App installata; Connessione attiva; email non già registrata
Input	Email valida, password validati, eventuali dati aggiuntivi
Procedura	L'utente compila i campi e conferma la registrazione

Output atteso	Messaggio di conferma o accesso automatico nell'app
Post-Condizione	Utente presente nel database su Firebase
Esito	-
Priorità	Alta

Campo	Descrizione
ID Test	TC-02
Requisito associato	RF-02-Login
Obiettivo del test	Verificare l'accesso con credenziali corrette e gestione degli errori
Pre-Condizione	Account registrato nel database
Input	E-mail e password
Procedura	Inserire le credenziali e premere "Accedi"
Output atteso	Accesso alla schermata principale o errore se credenziali errate
Post-Condizione	Sessione utente attiva
Esito	-
Priorità	Alta

Campo	Descrizione
ID Test	TC-03
Requisito associato	RF-03 – Mappa dei luoghi sostenibili
Obiettivo del test	Verificare che la mappa venga caricata correttamente con i marker.
Pre-Condizione	Connessione attiva; permesso GPS concesso; luoghi presenti nel database.
Input	Apertura schermata “Mappa”.
Procedura	Accedere alla schermata mappa.
Output atteso	Mappa visibile, marker correttamente posizionati.
Post-Condizione	Mappa utilizzabile.
Esito	-
Priorità	Alta

Campo	Descrizione
ID Test	TC-04
Requisito associato	RF-04 – Filtri
Obiettivo del test	Verificare la corretta applicazione dei filtri di categoria e distanza.

Pre-Condizione	Mappa e marker caricati.
Input	Categoria selezionata o range distanza.
Procedura	Applicare il filtro nel pannello dedicato.
Output atteso	Marker aggiornati in base ai filtri.
Post-Condizione	Filtri attivi finché non rimossi.
Esito	-
Priorità	Media

Campo	Descrizione
ID Test	TC-05
Requisito associato	RF-05 – Percorso
Obiettivo del test	Verificare il calcolo del percorso a piedi/in bici tramite Maps API.
Pre-Condizione	GPS attivo; luogo selezionato sulla mappa.
Input	Richiesta percorso.
Procedura	Aprire dettagli luogo → “Percorso” → scegliere modalità.
Output atteso	Visualizzazione percorso, distanza e tempo stimato.

Post-Condizione	Nessuna modifica persistente.
Esito	-
Priorità	Alta

Campo	Descrizione
ID Test	TC-06
Requisito associato	RF-06 – Segnalazione luogo
Obiettivo del test	Verificare l'inserimento corretto di una nuova segnalazione.
Pre-Condizione	Utente autenticato; connessione attiva.
Input	Nome luogo, categoria, posizione, descrizione.
Procedura	Aprire "Segnala luogo", compilare e inviare.
Output atteso	Conferma inserimento o messaggio di errore.
Post-Condizione	Nuovo luogo registrato in Firebase.
Esito	-
Priorità	Alta

Campo	Descrizione
ID Test	TC-07
Requisito associato	RF-07 – Modifica/Eliminazione
Obiettivo del test	Verificare la modifica e/o eliminazione di un luogo esistente.
Pre-Condizione	Luogo esistente creato dall'utente.
Input	Dati aggiornati o conferma eliminazione.
Procedura	Accedere ai dettagli del luogo → " Modifica/Elimina".
Output atteso	Luogo aggiornato o rimosso correttamente.
Post-Condizione	Database aggiornato.
Esito	-
Priorità	Media

Campo	Descrizione
ID Test	TC-08
Requisito associato	RF-08 – Preferiti
Obiettivo del test	Verificare l'aggiunta e rimozione di un luogo dai preferiti.

Pre-Condizione	Utente autenticato.
Input	Tap su icona preferito.
Procedura	Aprire luogo → premere icona “Preferito”.
Output atteso	Lista preferiti aggiornata.
Post-Condizione	Stato preferito sincronizzato su Firebase.
Esito	-
Priorità	Bassa

Campo	Descrizione
ID Test	TC-09
Requisito associato	RF-09-Profilo
Obiettivo del test	Verificare la corretta visualizzazione del profilo utente
Pre-Condizione	Utente autenticato
Input	Apertura schermata profilo
Procedura	Accedere al profilo dal menu
Output atteso	Visualizzazione dati utente, preferiti e segnalazioni
Post-Condizione	Nessuna

Esito	-
Priorità	Bassa

Campo	Descrizione
ID Test	TC-10
Requisito associato	RF-10-Logout
Obiettivo del test	Verificare la corretta terminazione della sessione
Pre-Condizione	Utente autenticato
Input	Comando di logout
Procedura	Premere “Logout” dal menu o profilo
Output atteso	Ritornare alla schermata iniziale/login
Post-Condizione	Sessione invalidata
Esito	-
Priorità	Alta

Nota: gli esiti riportati nelle tabelle dei test case sono attualmente lasciati vuoti, in quanto la loro compilazione avverrà durante l’effettiva esecuzione del testing.
Al momento dell’esecuzione, per ogni test case verrà indicato se l’esito è Positivo o Negativo, insieme ad eventuali osservazioni o anomalie riscontrate.
In caso di esito negativo, verrà aperta un’apposita segnalazione nel sistema di gestione dei problemi del progetto, descrivendo il comportamento osservato e i passi necessari per

riprodurre il difetto. Dopo la correzione, il test verrà rieseguito e il nuovo esito sarà aggiornato nel registro.
Questo approccio consente di mantenere una tracciabilità completa tra requisiti, esecuzioni, problemi identificati e correzioni applicate, garantendo un controllo accurato sulla qualità finale del software.

8.Requisiti hardware e software

Le attività di testing dell'app GreenZone richiedono un insieme minimo di risorse hardware e software che garantiscano la corretta esecuzione dei test sia su emulatori sia su dispositivi fisici.

Dal punto di vista hardware, il testing viene condotto sia su dispositivi reali sia su emulatori Android.

L'utilizzo di smartphone fisici, con versioni del sistema operativo diverse (Android 11 e Android 13), permette di osservare il comportamento dell'app in condizioni reali, verificando componenti fondamentali come GPS, rete dati, sensori integrati e reattività dell'interfaccia.

Parallelamente, i test su emulatori consentono di effettuare verifiche ripetibili, controllate e rapide, soprattutto durante le prime fasi dello sviluppo. Questi emulatori vengono eseguiti su PC dotati di CPU multicore e almeno 8 GB di RAM, in modo da supportare agevolmente compilazione, debugging e simulazioni.

Per quanto riguarda le risorse software, l'ambiente principale utilizzato è Android Studio, insieme alle versioni appropriate dell'Android SDK e agli emulatori con Google Play Services attivi.

L'applicazione richiede inoltre l'accesso ai servizi offerti da Firebase (autenticazione utenti e database) e alle API di Google Maps, essenziali per la visualizzazione della mappa, dei marker e del calcolo dei percorsi.

Per la gestione del progetto viene utilizzato GitHub, che consente il controllo delle versioni, la revisione del codice e il tracciamento dei bug tramite issue.

9.Vincoli che condizionano il testing

Le attività di testing dell'app **GreenZone** sono influenzate da una serie di vincoli tecnici, organizzativi e operativi che possono incidere sulla pianificazione e sull'esecuzione dei test.

Un primo limite significativo è rappresentato dalla **dipendenza da servizi esterni**, come Firebase e le API di Google Maps. Poiché l'app fa uso della rete per localizzazione, autenticazione e caricamento dei luoghi sostenibili, eventuali problemi di connettività, rallentamenti del server o limitazioni temporanee delle API possono compromettere l'esecuzione dei test o alterarne i risultati. Questo rende necessario effettuare verifiche in condizioni di rete diverse, incluse situazioni di connessione instabile.

Un altro vincolo riguarda la **variabilità dei dispositivi Android**. Differenze tra versioni del sistema operativo, risoluzioni dello schermo, capacità hardware e sensibilità dei sensori possono generare comportamenti non uniformi. Per questo motivo è importante eseguire test sia su emulatori sia su dispositivi fisici, anche se il numero di questi ultimi è limitato.

Dal punto di vista organizzativo, il progetto segue una metodologia **Agile**, con sviluppo suddiviso in sprint. Questo comporta che i test devono essere adattati continuamente all'avanzamento delle implementazioni. Se una funzionalità viene completata più tardi del previsto o necessita di modifiche impreviste, la pianificazione dei test potrebbe richiedere aggiornamenti per mantenere coerenza con lo stato reale del progetto.

Un ulteriore vincolo deriva dal tempo complessivo a disposizione. Le scadenze imposte dal corso richiedono una gestione rigorosa delle attività di testing, evitando ritardi che potrebbero compromettere le successive fasi di sviluppo o integrazione. È quindi fondamentale assicurarsi che i test essenziali vengano eseguiti nei tempi stabiliti e che i test di regressione siano pianificati in modo da non rallentare il rilascio di nuove versioni.

Inoltre, nella gestione del tempo complessivo disponibile, è necessario considerare anche i vincoli accademici dei singoli membri del gruppo. Le sessioni d'esame e gli impegni universitari personali possono infatti influire sulla continuità delle attività e ridurre, in alcuni periodi, il tempo effettivamente dedicabile al progetto. Per questo motivo, la pianificazione temporale prevista nel Piano di Progetto potrebbe richiedere aggiustamenti per mantenere un equilibrio realistico tra sviluppo, testing e carichi accademici.

Infine, esistono vincoli legati agli strumenti software utilizzati: le versioni delle API, gli aggiornamenti di Android Studio o eventuali incompatibilità temporanee possono impedire la corretta esecuzione dei test fino alla risoluzione del problema.

10.Riferimenti

Per la stesura di questo documento, sono state utilizzate le slide presentate a lezione riguardo il Piano di Testing.
Inoltre, si è fatto riferimento al **D2** (Piano di progetto) e al **D3** (Documento dei requisiti).