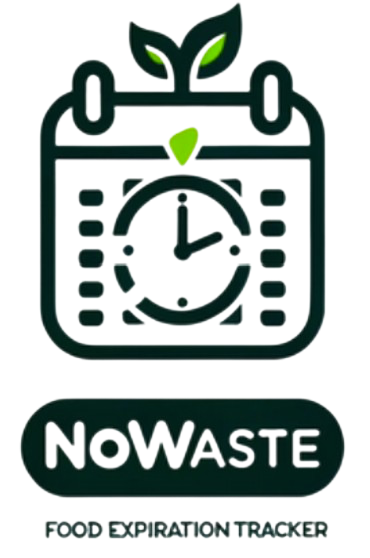
**Progetto di Ingegneria del Software 2023/24 Università Ca’ Foscari Venezia**

****

**Documento di Testing**

**2.0**

**No Waste**

29/01/2024

*Document Informations*

| NomeProgetto | Acronimo |  |
| --- | --- | --- |
| Deliverable | Piano di testing |  |
| Data di Consegna | 29 / 01 / 24 |  |
| Team Leader | Martina Ragusa | 885113@stud.unive.it |
| Team members | Daniela Spac [885386@stud.unive.it](mailto:885386@stud.unive.it)  Diego Marigo [853421@stud.unive.it](mailto:853421@stud.unive.it) |  |

*Document History*

| Version | Issue Date | Stage | Changes | Contributors |
| --- | --- | --- | --- | --- |
| 1.0 | 10/11/23 | Draft | Piano di Testing | MR, DS, DM |
| 1.1 | – | Draft | – | – |
| 2.0 | 29/01/24 | Final | – | MR, DS, DM |

**Indice**

[***1.INTRODUZIONE 3***](#_heading=h.gjdgxs)

[***1.1. Obbiettivo 3***](#_heading=h.30j0zll)

[***1.2. Ambito 3***](#_heading=h.1fob9te)

[***2. GLOSSARIO 3***](#_heading=h.3znysh7)

[***3. RIEPILOGO DEI TEST 4***](#_heading=h.2et92p0)

[***3.1. In ambito di test 4***](#_heading=h.tyjcwt)

[***3.2. Fuori dall’ambito di test 4***](#_heading=h.3dy6vkm)

[***4. ANALISI DELLE AREE DI TEST E DEL CAMPO DI APPLICAZIONE 5***](#_heading=h.1t3h5sf)

[***4.1. Rilascio del prodotto 5***](#_heading=h.4d34og8)

[***4.2. Piattaforma di testing 5***](#_heading=h.17dp8vu)

[***5. OBIETTIVI DEL TEST DI PROGRESSIONE 5***](#_heading=h.3rdcrjn)

[***6. ALTRI TEST 6***](#_heading=h.26in1rg)

[***6.1. Privacy 6***](#_heading=h.lnxbz9)

[***6.2. Stress e volume 7***](#_heading=h.1ksv4uv)

[***6.3. Ripristino di emergenza / Backup 7***](#_heading=h.44sinio)

[***6.4. Test sulle unità 8***](#_heading=h.2jxsxqh)

[***6.5. Test di integrazione 8***](#_heading=h.z337ya)

[***7. TEST SCHEDULING 8***](#_heading=h.3j2qqm3)

[***7.1. Livelli di responsabilità 8***](#_heading=h.1y810tw)

[***7.2. Tecniche di testing 8***](#_heading=h.2s8eyo1)

[***7.3. Pianificazione dell’esecuzione dei test 9***](#_heading=h.1ci93xb)

[***7.4. Strumenti di test 9***](#_heading=h.2bn6wsx)

[***7.5. Metriche 10***](#_heading=h.3as4poj)

[***8. PIANO DELL’AMBIENTE DI TEST 10***](#_heading=h.1pxezwc)

[***8.1. Test sulle interazioni 10***](#_heading=h.49x2ik5)

[***8.2. Dettagli sull’ambiente di test 10***](#_heading=h.2p2csry)

[***8.3. Scelta dell’ambiente 11***](#_heading=h.147n2zr)

[***8.4. Strumenti per il controllo dell'ambiente 11***](#_heading=h.3o7alnk)

[***9. CRITERI DI INIZIO E FINE TEST 11***](#_heading=h.32hioqz)

[***10. PIANO AMMINISTRATIVO 12***](#_heading=h.1hmsyys)

[***10.1. Test milestones 12***](#_heading=h.41mghml)

[***10.2. Gestione dei difetti 12***](#_heading=h.2grqrue)

[***11. RIFERIMENTI 13***](#_heading=h.vx1227)

# 1.INTRODUZIONE

### 1.1. Obbiettivo

Il Piano di Testing ha il compito di definire in che modo le attività di testing accompagneranno lo sviluppo delle componenti del sistema e la loro integrazione.

Grazie a questo documento, il team sarà in grado di individuare eventuali malfunzionamenti che potrebbero compromettere la soddisfazione di tutti i punti presentati nel documento dei requisiti redatto in precedenza. Il Piano di Testing ha dunque il compito di definire, oltre alle tecniche di testing che si vogliono utilizzare e la modalità utilizzata per verificare i singoli requisiti, anche i casi di test che verranno utilizzati, in che modo sarà realizzata la schedulazione di tali test (sia in termini di tempo che di risorse), le procedure per registrare tali test e i vincoli che potrebbero condizionarli.

### 1.2. Ambito

In questo documento verranno specificate le metodologie di testing dell’applicazione effettuate dagli sviluppatori.

Di seguito verranno definiti i seguenti ambiti:

1. Le sezioni 5 e 6 tratteranno cosa verrà analizzato e testato
2. La sezione 7 tratterà come verrà effettuato il test, quali e quando le risorse verranno richieste

# 2. GLOSSARIO

* **Android**: sistema operativo basato su kernel Linux per dispositivi mobili, quali smartphone e tablet, nel nostro caso verranno considerate solamente versioni di Android superiori alla 8.1;
* **API**: acronimo di Application Programming Interface, si indica un insieme di procedure atte a risolvere uno specifico problema di comunicazione tra diversi computer o tra diversi software o tra diversi componenti di software. [Fonte](https://it.wikipedia.org/wiki/Application_programming_interface)
* **App**: è un'applicazione software dedicata ai dispositivi di tipo mobile, tipicamente progettata e realizzata in maniera più leggera in termini di risorse hardware. [Fonte](https://it.wikipedia.org/wiki/Applicazione_mobile)
* **Bug**: nell'ambito della programmazione informatica, è un'anomalia che porta al malfunzionamento di un software. [Fonte](https://it.wikipedia.org/wiki/Bug)
* **Database**: rappresenta la locazione in cui vengono memorizzati tutti i dati riguardanti le partite nel database, i dati verranno strutturati in modo tale da facilitare l’inserimento e l’aggiornamento degli stessi in modo dinamico.
* **Firebase**: Firebase è una piattaforma serverless per lo sviluppo di applicazioni mobili e web. Open source ma supportata da Google. [Fonte](https://www.geekandjob.com/wiki/firebase)
* **GDPR**: Acronimo inglese di “General Data Protection Regulation” ed è un regolamento dell’UE riguardante il trattamento dei dati personali e di privacy.
* **GPS**: sistema di geolocalizzazione situato all’interno del dispositivo per il tracciamento dei movimenti.
* **Java**: linguaggio di programmazione orientato agli oggetti, progettato per essere indipendente dalla piattaforma di esecuzione.
* **Smartphone**: un telefono cellulare con capacità di calcolo, memoria e connessione dati molto più avanzate rispetto ai normali telefoni cellulari, basato su un sistema operativo per dispositivi mobili. [Fonte](https://it.wikipedia.org/wiki/Smartphone)
* **Google Sheets**: Fogli Google è un programma per fogli di lavoro incluso come parte della suite gratuita Google Docs basata sul Web offerta da Google. [Fonte](https://en.wikipedia.org/wiki/Google_Sheets)
* **Exceptions (programmazione):** il termine eccezione (exception) è usato per descrivere l'occorrenza di diversi tipi di condizioni o eventi che alterano il normale flusso di esecuzione di un programma. Possono essere considerati simili a degli errori a tempo di esecuzione da parte del programma eseguito.

## 3. RIEPILOGO DEI TEST

​​Il piano di testing prevede le seguenti fasi:

* Preparazione: definizione delle tecniche di testing, dei casi di test e delle procedure dei test;
* Esecuzione: esecuzione dei test e registrazione dei risultati;
* Analisi: valutazione dei risultati dei test e identificazione degli errori;
* Correzione: correzione degli errori e verifica delle soluzioni apportate.

Le risorse e il tempo allocati per il testing dipenderanno dalla complessità dell’app e dalla disponibilità delle risorse umane e tecniche.

Di seguito faremo una distinzione tra il piano di testing all’interno dell’ambito di test e fuori dall’ambito di test. La differenza principale tra i due riguarda gli obiettivi e le attività pianificate. All’interno dell’ambito di test, il piano di testing è focalizzato sugli obiettivi specifici dell’applicazione, del sistema o del processo che si sta testando. Fuori dall’ambito di test, invece, il piano di testing può avere obiettivi più generali e può coprire una varietà di attività di test che non sono strettamente legate all'applicazione o al sistema specifico.

### 3.1. In ambito di test

L’ambiente di testing sarà costituito da ogni singolo utente che dovrà aggiungere alimenti alla propria lista. In questo modo potremmo osservare il corretto funzionamento di ciascun requisito descritto in precedenza nel “Documento dei Requisiti”. Questo ci permetterà inoltre di scoprire eventuali miglioramenti e ulteriori funzionalità che possono arricchire l’applicazione.

### 3.2. Fuori dall’ambito di test

Fuori dall'ambito di test si considererà anche l'usabilità generale dell'app, incluse eventuali opzioni di accessibilità e la facilità di navigazione, per far sì che l'esperienza utente sia positiva indipendentemente dalla competenza tecnica degli utenti.

# 4. ANALISI DELLE AREE DI TEST E DEL CAMPO DI APPLICAZIONE

### 4.1. Rilascio del prodotto

Lo sviluppo dell’applicazione avverrà all’interno di Android Studio, usufruendo delle librerie che offre Android (è richiesta una versione di Android 8.1, o più recenti, per la compatibilità con tali librerie) e Java. Per la comunicazione con il database esterno, Firebase, verranno sfruttate le API di quest’ultimo.

L’applicazione si basa su un software progettato per funzionare su uno smartphone Android e richiede una connessione ad internet per comunicare con il database esterno. Affinché l’applicazione possa operare correttamente, deve essere in grado di stabilire tale connessione con il database e interagire con esso tramite le operazioni di lettura e scrittura dei dati.

### 4.2. Piattaforma di testing

I test delle diverse componenti saranno inizialmente condotti sui nostri dispositivi Android per ottimizzare le prestazioni su di essi. In seguito, saranno eseguiti anche sugli emulatori forniti dallo stesso Android Studio. La procedura di testing proseguirà in parallelo con l’avanzamento del lavoro. In questo modo, man mano che verrà sviluppato il codice, potremmo scovare più facilmente la presenza di bug, che saranno inoltre più semplici da risolvere.

# 5. OBIETTIVI DEL TEST DI PROGRESSIONE

In questa sezione verrà descritta nel dettaglio la progressione degli obiettivi dei test coperti dal gruppo di progetto, che verranno semplificati in tabelle per una maggiore leggibilità e gestione del tracciamento.

| Funzionalità | Rif. | Obiettivo del test | Criteri di valutazione | Priorità |
| --- | --- | --- | --- | --- |
| Creazione utente | S-01 | Verificare se viene creato correttamente l’utente con  tutti i dati inseriti dall’app.  Verranno creati 5 utenti utilizzati per il testing. | Verificare se l’utente viene aggiunto al db | Alta |
| Aggiunta / eliminazione / modifica alimento | S-02 | Verificare che l’operazione riguardante l’alimento venga effettuata correttamente | Verificare se l’alimento viene aggiunto (rimosso o modificato) correttamente, modificando  effettivamente anche il db | Alta |
| Visualizzazione degli alimenti | S-03 | Verificare la presenza degli alimenti | Esistenza dei vari alimenti all’interno del db | Alta |
| Modifica delle impostazioni | S-04 | Verificare la possibilità di modificare le impostazioni | Verificare che le modifiche che effettua l’utente vengano effettivamente fatte | Media |
| Ricezione notifiche | S-05 | Verificare che l’utente riceva le notifiche.  Verranno inviate 3 notifiche: la prima dopo un’ora, la seconda dopo due ore e l’ultima dopo tre ore. | Verificare che l’utente riceva le notifiche secondo la cadenza da lui selezionata | Alta |
| Condivisione della lista degli alimenti | S-06 | Verificare che la condivisione con altri utenti avvenga correttamente | Verificare che gli utenti con cui è stata condivisa la lista possano visualizzarla correttamente e modificarla | Alta |
| Login | S-07 | Verificare che l’utente possa accedere con le proprie  credenziali all’app | Verificare la comunicazione con il server e l’effettivo accesso dell'utente | Alta |

# 6. ALTRI TEST

### 6.1. Privacy

La sicurezza è uno degli aspetti chiave della nostra applicazione, poiché si verificano sempre più frequentemente attacchi da parte di utenti malintenzionati. Di conseguenza, verranno sicuramente inclusi test che hanno il compito di garantire il rispetto delle regole fondamentali del GDPR e l'adozione di pratiche di programmazione e gestione dei dati sicure al fine di mitigare questo problema.

Per garantire la connessione sicura, Firebase ci viene in aiuto. Infatti, esso mette a disposizione Firebase Cloud Messaging (FCM) che gioca un ruolo importante nella gestione della connessione. Esso utilizza canali sicuri che permettono di garantire che i dati siano inviati in modo protetto.

| Funzionalità | Rif. | Obiettivo del test |  | Criteri di valutazione | Priorità |
| --- | --- | --- | --- | --- | --- |
| Privacy | S-01 /  S-02 /  S-03 /  S-04 /  S-05 /  S-06 /  S-07 | Verificare  comunicazione sicura  client e db | la  tra | Verificare la sicurezza del canale di comunicazione tra  client e db | Alta |

### 

### 6.2. Stress e volume

Al fine di assicurare una robusta prestazione del sistema, condurremo test sulla capacità di gestione dei dati del database. Valuteremo la capacità del database di gestire volumi di dati e verificheremo quanto velocemente possono variare e essere letti, garantendo così un'esperienza utente fluida e priva di interruzioni.

Per effettuare queste operazioni di stress testing, verranno utilizzati tre dispositivi contemporaneamente. In questo modo potremo testare la scrittura e lettura dei dati dal database.

| Funzionalità | Rif. | Obiettivo del test | Criteri di valutazione | Priorità |
| --- | --- | --- | --- | --- |
| Traffico in scrittura nel database | S-01 /  S-02 /  S-04/  S-06/  S-07 | Verificare la corretta scrittura in un contesto dove più dispositivi scrivono nel database simultaneamente. | Verrà testata la scrittura simultanea da parte di più simulatori, i quali aggiornano le proprie entry nel database ripetutamente per comprendere se possono generare conflitti. | Alta |
| Lettura dal database | S-03 /  S-04/ S-07 | Verificare la lettura simultanea da parte di più dispositivi dei dati all’interno del database esterno. | Lettura simultanea da più simulatori di entry diverse, se il test avrà un esito positivo, si testerà la lettura simultanea di una entry soltanto da parte degli stessi simulatori. | Alta |

### 

### 6.3. Ripristino di emergenza / Backup

Al fine di prevenire possibili problemi come la perdita di dati, specialmente per quanto riguarda il codice sorgente durante le fasi di sviluppo e completamento, verranno eseguiti backup regolari su GitHub. Questa precauzione è adottata per mitigare il rischio di perdita di dati, poiché un evento del genere potrebbe compromettere l'intero progetto.

### 6.4. Test sulle unità

Ogni singola unità, una volta concluso il suo sviluppo, verrà testata sia sugli emulatori forniti da Android Studio, sia sui nostri dispositivi Android fisici con versione 12.

### 6.5. Test di integrazione

Dopo il superamento dei test descritti in precedenza, procederemo con i test in gruppi, replicando la configurazione che dovrebbero avere nel prodotto finale. Questo approccio ci consentirà di verificare l'integrazione corretta tra le unità senza eventuali conflitti.

# 7. TEST SCHEDULING

### 7.1. Livelli di responsabilità

Di seguito verranno dichiarati i livelli di responsabilità dei test che verranno svolti.

| Livello di test | Team di progetto | Professore | Esterno |
| --- | --- | --- | --- |
| Privacy | P | ― | ― |
| Connettività | P | S | S |
| Stress e volume | P | ― | ― |
| Ripristino di emergenza / backup | P | ― | S |
| Test sulle unità | P | ― | ― |
| Test di integrazione | P | ― | ― |

Legenda:

* P: Responsabilità primaria
* S: Responsabilità secondaria

### 7.2. Tecniche di testing

Per il testing stiamo considerando di utilizzare:

* Black-box testing: per testare le funzionalità dell’app indipendentemente dai dettagli dell’implementazione;
* White-box testing: per testare il flusso di controllo del programma e coprire tutte le istruzioni;
* Top-down testing: per testare le singole componenti dell’app e le loro interazioni.

Per ogni test la procedura che seguiremo sarà la seguente:



### 7.3. Pianificazione dell’esecuzione dei test

I test saranno condotti seguendo le linee guida descritte in precedenza, in particolare nelle sezioni 5 e 6.   
In questo modo, sarà possibile ottimizzare i tempi nello sviluppo dell’applicazione, iniziando con lo sviluppo delle funzionalità che riteniamo essenziali per la buona riuscita del progetto e, successivamente, anche a seguito dei vari test degli utenti, penseremo alle funzionalità secondarie.

### 7.4. Strumenti di test

Per permetterci di tenere traccia di ciascun test e del suo andamento utilizzeremo i seguenti strumenti:

| Processo | Strumento/i |
| --- | --- |
| Creazione casi di test | Documenti Google - Documento del Piano di Testing |
| Tracciamento casi di test | Google Sheets / Excel |
| Esecuzione casi di test | Manualmente (uso dei dispositivi) |
| Gestione casi di test | Google Sheets / Excel |
| Gestione difetti | Google Sheets / Excel |

Di seguito viene mostrato un esempio dell’utilizzo di Excel per registrare l’andamento dei test.

| Descrizione test | Quando si verifica il problema? | Tipo di test | Risolto | Data |
| --- | --- | --- | --- | --- |
| Crash dell’applicazione | Ad un primo avvio dell’app | Risolvere il crash | SI | 28/12/2023 |
| … | … | … | … | … |
|  |  |  |  |  |

### 7.5. Metriche

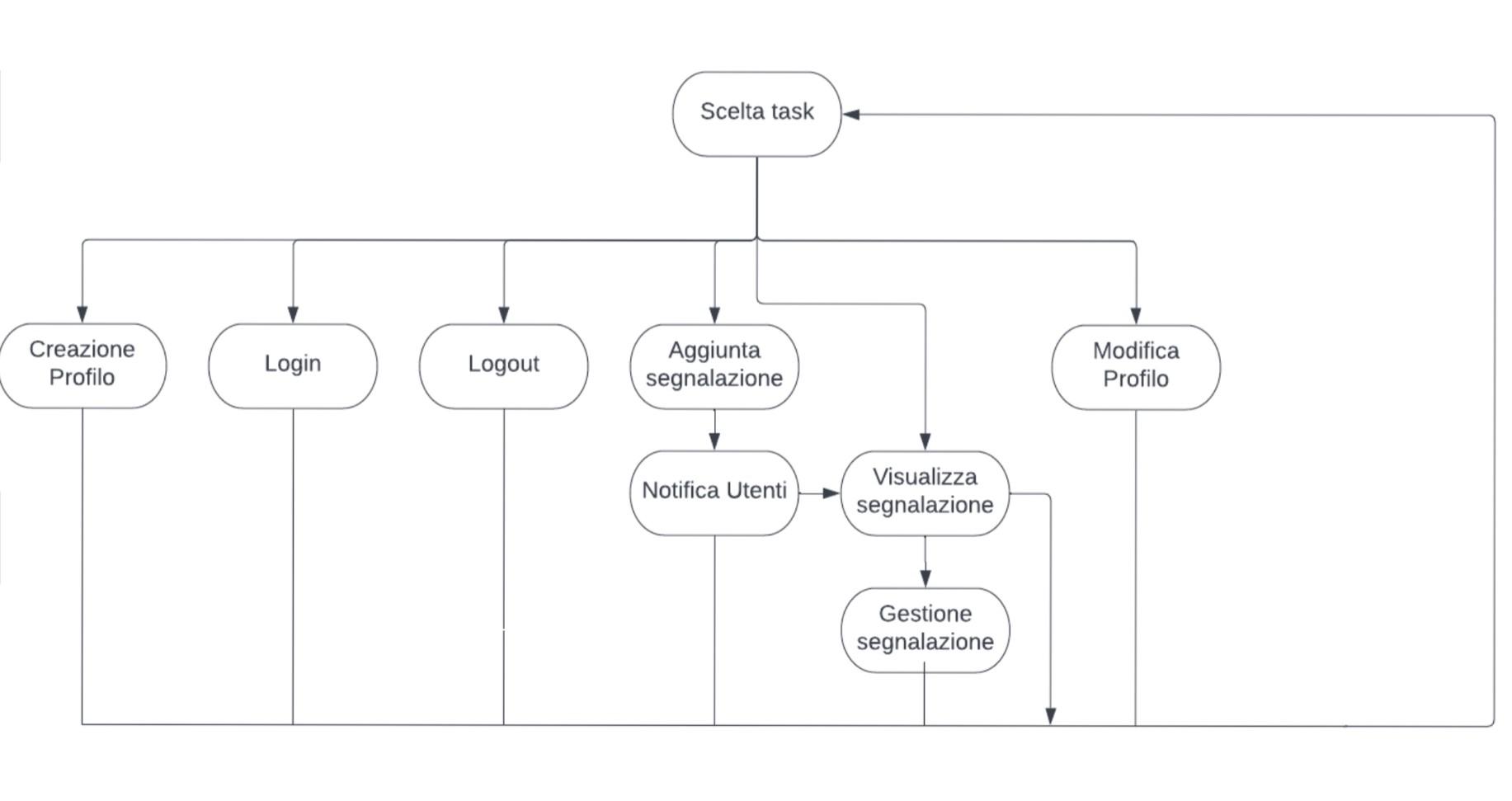
Facendo riferimento alla tabella di progressione dei test presente nella sezione 5, il gruppo procederà alla verifica prima dei singoli componenti e, in seguito, di quelli composti da due o più componenti che interagiscono tra loro.

Per quanto riguarda l’avanzamento generale, l’implementazione di un nuovo componente non avverrà fino a quando non saremo sicuri che quello precedente funzioni in modo adeguato.

# 8. PIANO DELL’AMBIENTE DI TEST

### 8.1. Test sulle interazioni

Sotto abbiamo raffigurato le possibili task da verificare:



Le seguenti task saranno verificate per garantire interazioni corrette tra l'utente e l'applicazione:

- Aggiunta alimenti con scadenza

- Modifica dati alimenti

- Notifiche di prossima scadenza

- Sincronizzazione con il cloud/database

- Esportazione e importazione di dati

### 8.2. Dettagli sull’ambiente di test

Testare la gestione delle notifiche in tempo reale e la sincronizzazione richiederà l'uso di più dispositivi per simulare scenari reali di utilizzo. I test di base possono essere eseguiti su un singolo dispositivo. Per le funzionalità che richiedono interattività o sincronizzazione, si utilizzeranno più dispositivi mobili per testare le funzionalità in rete.

### 8.3. Scelta dell’ambiente

Requisiti da rispettare durante il testing per ogni funzionalità

| Funzionalità | Requisiti |
| --- | --- |
| Creazione Profilo | Profilo non ancora creato |
| Login | Profilo esistente nel database |
| Logout | L'utente deve essere connesso ed autenticato |
| Aggiungi Alimenti | L'utente deve essere connesso ed autenticato |
| Notifiche di Scadenza | Scadenza degli alimenti definita |
| Visualizza Alimenti | Lista alimenti non vuota |
| Gestione Scadenze | Scadenza degli alimenti definita |
| Modifica Profilo | L'utente deve essere connesso ed autenticato |

### 8.4. Strumenti per il controllo dell'ambiente

L'ambiente di test richiederà l'accesso a dispositivi con Android 8.1 o versioni successive, e presume una connessione ad internet. Sarà inoltre utilizzata una piattaforma di virtualizzazione o emulazione (come Android Studio Emulator) per simulare diverse condizioni di rete e hardware.

# 9. CRITERI DI INIZIO E FINE TEST

Qui sono definiti in dettaglio i criteri per i quali un test può cominciare o può essere definito concluso.

Questo aiuta a capire a quale fase di test il team è arrivato.

| Fase di testing | Quando può iniziare | Quando è finito |
| --- | --- | --- |
| Creazione utente | ― | Ricerca va a buon fine, creazione va a buon fine, scrittura va a buon fine. |
| Modifica della lista degli alimenti | Dopo “Creazione utente”. | Scrittura va a buon fine e legge i dati corretti; lettura, modifica ed eliminazione vanno a buon fine e i dati vengono aggiornati adeguatamente. |
| Visualizzazione della lista degli alimenti | Dopo “Creazione utente” e “Modifica della lista degli alimenti”. | Lettura va a buon fine e legge i dati corretti. |
| Modifica delle impostazioni | Dopo “Creazione utente” | Scrittura va a buon fine. |
| Ricezione delle notifiche | Dopo “Creazione utente” e “Modifica della lista degli alimenti”. | Lettura va a buon fine e legge i dati corretti. |
| Condivisione della lista degli alimenti | Dopo “Creazione utente” e “Modifica della lista degli alimenti”. | Scrittura va a buon fine e legge i dati corretti. |
| Log out | Dopo “Creazione utente”. | Viene tagliata la connessione con l’applicazione e si presenta la schermata home senza accesso. |
| Log in | Dopo “Creazione utente”. | Lettura va a buon fine e legge i dati corretti, concesso l’accesso con recupero dati dell’utente desiderato dal database. |

# 

# 10. PIANO AMMINISTRATIVO

### 10.1. Test milestones

Qui vediamo una struttura che identifica le milestones, per quanto riguarda i test (alcuni dei test back-to-back), a cui viene associata un’ipotetica data di raggiungimento pianificata alla quale dovrà essere affiancata la data di raggiungimento veritiera.

| Milestone | Data di fine pianificata | Data di fine effettiva |
| --- | --- | --- |
| Versione 1.0 | 15 Dicembre 2023 | - |
| Versione completa | 15 Gennaio 2024 | 29 Gennaio 2024 |

### 

### 10.2. Gestione dei difetti

In caso di problemi di ogni genere ci si troverà tutti assieme per pensare una possibile soluzione e successivamente implementarla. Nel caso non esista una soluzione vera e propria si penserà ad uno stravolgimento del tutto così da creare una strada secondaria per arrivare all’obiettivo desiderato.

# 11. RIFERIMENTI

Per creare questo documento sono stati utilizzati come riferimento:

* ”Piano di testing” di alcuni gruppi degli anni passati;
* Documenti già stilati dal team (“Piano di progetto” e “Documento di analisi e specifica”)
* Template del “piano di testing” messo a disposizione dal professore