

**SOFTWARE ENGINEERING PROJECT**

**“Fitness App”**

Anno accademico 2022-2023

Roberto Bertocchi 1060957

Mirko Carrara 1059971

Stefano Cosseddu 1067656

Francesco Peracchi 1057616

Indice

* **Il progetto**
* **1.Project plan**
  + 1.1 Introduzione
  + 1.2 Modello di processo
  + 1.3 Organizzazione del progetto
  + 1.4 Standards, linee guida e procedure
  + 1.5 Attività di gestione
  + 1.6 Rischi
  + 1.7 staff & skills richieste
  + 1.8 Metodi e tecniche
  + 1.9 Garanzie di funzionamento e qualità
  + 1.10 Pacchetti di lavoro
  + 1.11 Requisiti minimi e risorse
  + 1.12 Budget e pianificazione
  + 1.13 Cambiamenti
  + 1.14 Rilascio applicazione
* **2. Software lifecycle**
* **3. Configuration management**
  + 3.1 Struttura del progetto
  + 3.2 Issues
* **4. People management**
* **5. Software quality**
  + 5.1 Operatività del software
  + 5.2 Revisione del software
  + 5.3 Transizione verso un nuovo ambiente
* **6. Requirement engineering**
* **7. Modelling**
  + 7.1 Grafica
  + 7.2 Funzionamento del back-end
* **8. Software architecture**
* **9. Software testing e manutenibilità**

**Il progetto**

L’applicazione Fitness app viene sviluppata in java tramite l’utilizzo del software Android Studio.

Questa applicazione permette di creare una scheda di allenamento personalizzata da visualizzare su qualsiasi dispositivo tramite l’uso di un database.

Grazie a questa applicazione l’atleta potrà tener traccia di tutti i suoi allenamenti svolti e potrà monitorare i miglioramenti nel lungo periodo dei vari esercizi.

**1.Introduzione**

Il nostro progetto d’esame di Ingegneria del software consiste nello sviluppo di un applicativo mobile, ”**Fitness app**”, il cui obiettivo è quello di supportare gli sportivi, offrendo una piattaforma intuitiva e funzionale che permetta di tenere traccia delle varie sedute di allenamento, la loro programmazione e i progressi ottenuti.

In particolare questa applicazione creata per il sistema operativo Android prevede la creazione di schede relative ad ogni specifico allenamento, dove in ogni scheda vengono inseriti i vari esercizi svolti

Le schede poi possono essere visualizzate per tipologia di allenamento in modo da poter monitorare i progressi relativi ad uno specifico allenamento

# **2.Modello di processo**

Un’ applicazione mobile può variare molto dal punto di vista della complessità; dal tipo di funzionalità che deve implementare, dall’esistenza o meno di una base di dati, dall’accesso ad internet fino al poter sfruttare determinati comportati hardware del dispositivo.

Nel nostro caso il modello di processo che più si adatta ad un applicativo come il nostro, è sicuramente un modello lightwork agile, in quanto pensiamo che i feedback degli utenti saranno fondamentali per migliorare il servizio offerto, e che sicuramente Immagine che contiene diagramma

Descrizione generata automaticamentecomporteranno dei cambiamenti sia lato UI che lato funzionalità.

Infatti il modello agile presenta alcuni valori chiave che si adattano perfettamente al software che vogliamo creare, quali

* + Collaborazioni con il cliente, più importanti di aspetti contrattuali
  + Risposta al cambiamento, più importante rispetto al planning
  + individui e interazioni, più importanti rispetto a processi e strument

Main goal

Gestione, creazione e visualizzazione piani di allenamento



Requirements specification

SW requirements:L’applicazione verrá sviluppata per android 8.0 (api 26) siccome ci permette di coprire il 90.7% dei dispositivi

Documentation: developer.android.com

Implementation

Pseudo codice e progettazione: StarUML IDE sviluppo: Android studio

Linguaggi utilizzati: java e XML

Testing: Prove demo tramite apk android



Maintenance

# **3.Organizzazione progetto**

Un progetto simile sicuramente richiede vari campi da coprire: dalla progettazione dell’interfaccia grafica, la gestione dell’eventuale base di dati e ovviamente il codice dell’applicazione che comprende una parte dedicata all’interfaccia e una orientata verso il codice java, puramente algoritmica, in cui si definiscono classi, oggetti ecc.

I ruoli come Project manager, programmatore, analista, vengono definiti in fasi successive di sviluppo del progetto, in quanto non abbiamo schemi rigidi in base anche al modello di processo agile. Ci sembrano appropriati approcci come XP o Extreme Programming, metodo agile per eccellenza.

**4.Standard, linee guida, procedure**

Nel nostro caso per aiutare l’utente in caso di difficoltà nell’utilizzo dell’app abbiamo inserito una sezione di “domande frequenti” dove saranno descritti i problemi piú comuni che si possono riscontrare nell’utilizzo dell’applicazione.

Per poter rendere l’applicazione più “user friendly” possibile, è buona norma fare riferimento alle Google Guide Lines, presenti nel sito Material.io, in questo modo l’utente è già “istruito” sulla posizione di molti elementi di UI; questo perché molte delle applicazioni che si usano nella vita di tutti i giorni, vengono sviluppate con le medesime meccaniche sia di visualizzazione che di interazione. <https://material.io/resources/tutorials#android-java>

# **5.Attività di gestione**

Il repository presente su github, ci permette, oltre che a lavorare al progetto, di poter rapidamente segnalare problemi, eventuali parti mancanti e l’accesso contemporaneo ai file con il salvataggio dei progressi.

# **6.Rischi**

I rischi principali a cui si va incontro, nello sviluppo di un software tale, sono da rapportare alla concorrenza e alla sfruttabilità dell’applicazione, in quanto possono già essere presenti altre applicazioni che presentano funzioni simili, per questo motivo è importante implementare le stesse funzioni, ma in maniera più semplice e rapida. Oltre a questo, più nello specifico, è importante evidenziare quali possono essere gli aspetti più critici dell’applicazione, quali funzioni possono dare problemi e la difficoltà di risoluzione di essi.

# **7.Staff e skills richieste**

Per poter portare avanti questo progetto, sono necessarie determinate competenze di Progettazione, quali l’utilizzo del software StarUML per la realizzazione di Casi d’uso, Activity diagrams ecc. L’applicazione Android, in quanto tale, richiede anche la conoscenza dell’IDE Android Studio, in cui si sfrutterà il linguaggio ad oggetti Java, per la parte di codice riguardante la logica, e XML per la parte di interfaccia e componenti grafici e realizzazione di template di schermate e icone (ad esempio qualche stilizzata icona relativa al fitness)

**8.Metodi e tecniche**

In questa sezione utilizziamo un ambiente object oriented(java) per lo sviluppo del codice e tramite l’utilizzo di Github viene utilizzato un metodo agile in cui ogni componente del gruppo condivide una parte del codice con relativa documentazione. Successivamente viene utilizzato Android Studio per la creazione e le specifiche dell’applicazione android ”Fitness app”.

# **9.Garanzia di funzionamento e qualità**

Per poter garantire il corretto funzionamento dell’applicazione, è necessario che la versione del sistema operativo su cui girerà il software, sia android.

Si dovrà quindi indicare nella creazione del progetto questa caratteristica. Inoltre è importante garantire il corretto funzionamento anche in base ai diversi DPI (densità di pixel per inch) dei display in quanti possono esserci problemi di visualizzazione dell’interfaccia.

Si potrebbero anche implementare delle impostazione relative alla lingua di sistema, oppure alle unità di misura utilizzata negli esercizi (come Kg o Libbre)

# **10.Pacchetti di lavoro (workpackages)**

Immagine che contiene diagramma

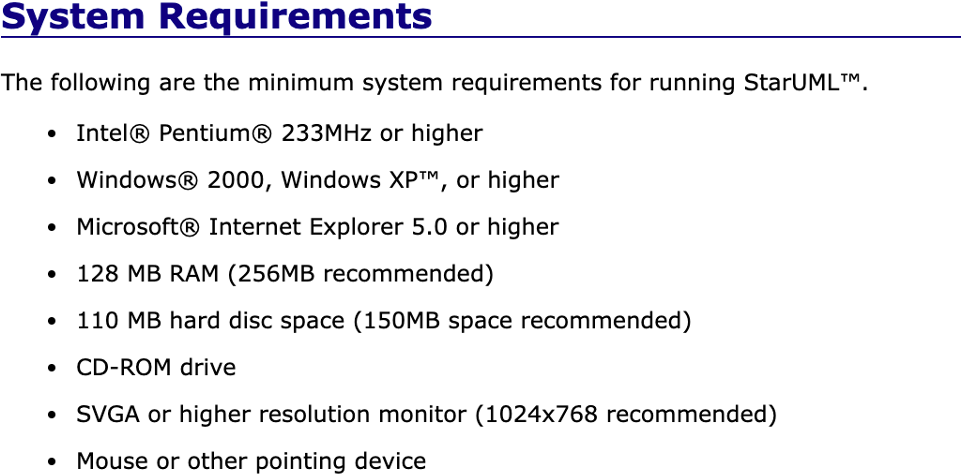
Descrizione generata automaticamente

**11/12. Requisiti minimi e risorse + costi e budget**

Requisiti minimi e consigliati per l’IDE Android Studio e la macchina virtuale per emulare android ed eseguire l’applicazione e fare debug



Requisiti minimi per il software StarUML.

Il software in questione richiederebbe una licenza commerciale, dal costo di 129$/utente

Fonti:

<https://developer.android.com/studio> <http://staruml.sourceforge.net/docs/user-guide(en)/ch01.html>

Costi delle relative macchine per poter far girare i software (1000€ CA.). Licenza account Google Developer (25$)

La licenza di Android Studio è open source.

# **13.Cambiamenti**

Google, attraverso la sua piattaforma di pubblicazione, permette agli utenti di recensire e commentare l’app, in base alla loro esperienza utente, sottolineando criticità e possibili funzionalità.

In seguito, grazie a questi commenti, valuteremo quali modifiche apportare all’applicazione.

# **14.Rilascio applicazione.**

L’applicazione verrà rilasciata su Google Play Store in maniera gratuita.

Per farlo, occorre un account developer per poter fare accesso a Google Play Console, dal costo di 25$.

Si può valutare in base ai refactoring, l'implementazione di alcune funzionalità nuove, utilizzabili solo acquistando una “versione completa”, implementando un “in-app purchase” attraverso una libreria messa a disposizione direttamente da Google, chiamata Billing Library version 3.

**Software lifecycle**

Per lo sviluppo del progetto, il team ha scelto un approccio **agile ,** questo perché la metodologia agile si adatta meglio alla metodologia di lavoro desiderata, ovvero:

1. Si considera importante il gruppo, le abilità dei suoi membri e le loro interazioni.

I lavori vengono assegnati in base alle capacità personali riducendo il tempo necessario allo studio di una competenza non conosciuta.

Nel caso in cui un componente della squadra non sia in grado di portare a termine un task, il team si riunisce per trovare una soluzione.

1. Nel team non c'è una struttura di tipo gerarchico, tutti i membri hanno la stessa importanza e possono esprimere le loro opinioni.
2. Si considera più importante un prodotto software funzionante piuttosto che una documentazione estremamente elaborata. Ciò consente di dedicare più tempo allo sviluppo e ottenere un prodotto pubblicabile in minore tempo con un risparmio di risorse. Nel caso sia necessario, la documentazione potrà essere espansa o modificata in seguito.
3. team è organizzato secondo la filosofia dell'extreme programming e durante lo sviluppo si è sfruttato spesso il pair programming come metodo di verifica.

I programmatori in coppia hanno lavorato sullo stesso frammento di codice in modo tale da avere una verifica in tempo reale sul lavoro svolto.

1. Si è anche sfruttata la tecnica del timeboxing per la suddivisione dello sviluppo a intervalli temporali entro i quali determinate funzionalità devono essere implementate ciò è avvenuto tramite l’utilizzo di branch e di pull request(funzioni contenute all’interno di github).
2. Si è scelto di dare molta importanza alla collaborazione con gli utenti, infatti dopo aver messo appunto le funzionalità base dell'applicazione, si è deciso di dare la possibilità agli utenti di collaborare mediante suggerimenti e feedback o anche attivamente allo sviluppo degli aggiornamenti.
3. Il team è favorevole al cambiamento e si impegna a dedicare il giusto tempo alla pianificazione del futuro del software.

Per la progettazione del software è stato utilizzato il model-driven architecture, ovvero prima di procedere alla scrittura del codice sono stati costruiti dei modelli in UML che rappresentano gli obiettivi e il funzionamento dell’applicazione fitness app.

**Configuration managment**

Il lavoro svolto, che si tratti di documentazione o di codice, viene regolarmente salvato nel repository di GitHub in condivisione con tutti i membri del team.

**Utilizzo di github**

Il repository è strutturato nel seguente modo:

utilizziamo 2 diversi branch, i quali ci permettono di sviluppare il progetto senza modificare oggetti essenziali all’interno del main del progettp

**branches**:

* **main**: Contiene le versioni stabili del codice e la documentazione
* **develop**: Contiene le versioni in via di sviluppo del codice

Successivamente suddividiamo il progetto in cartelle per aver una gestione migliore

**cartelle**:

* **code**: Contiene il codice sorgente del progetto.
* **docs**: Contiene la documentazione del progetto

Nei vari incontri tra i componenti del progetto agile vengono creati i temi che dovranno essere sviluppati.

Le varie attività sono create come issue, aggiungendo una breve descrizione del lavoro da svolgere.

Durante la settimana i membri del team sono liberi di creare nuove issue in base alle necessità.

Le issue sono suddivise in 3 categorie principali:

**task**: attività da svolgere

**bug**: segnalazione di un bug

**enhancement**: proposta di miglioramento

Gli stati che può assumere il progetto possono essere:

* Not started: L'attività non è ancora stata iniziata
* Working on it: L'attività è in fase di sviluppo
* To test: L'attività è stata completata e deve essere testata
* Testing: L'attività è in fase di testing che comprende la scrittura di test JUnit o la verifica manuale
* To fix: L'attività è stata testata ma non funziona correttamente
* Done: L'attività è stata completata e funziona correttamente

**People Management**

Per lo sviluppo di questa applicazione è stato utilizzato una organizzazione con metodo agile, in questo modo i componenti del team hanno potuto lavorare in parallelo per quanto riguarda la creazione del codice che sulla produzione della documentazione.

L’utilizzo di github ha aiutato notevolmente in questa fase, permettendo ai componenti del team di conoscere lo stato di avanzamento del progetto e di standardizzare la piattaforma su cui operare e condividere i file di progetto.

Nell’ambito di questo progetto sono stati applicati i concetti della soluzione divisionale relativi ai meccanismi di coordinamento i quali hanno permesso di ottenere una buona autonomia su come raggiungere gli obiettivi dichiarati.

**Software quality**

Il team ha deciso di sviluppare un'applicazione che rispetti i parametri e gli attributi di qualità definiti da McCall-Richards-Walters come da specifica 1977

**Funzionamento del prodotto**

* **Correttezza**: L'applicazione soddisfa i requisiti e le specifiche stabilite.
* **Affidabilità**: L'applicazione è stata testata mediante strumenti e metodologie diverse e non presenta anomalie durante l’utilizzo in domini diversi.
* **Efficienza**: In generale l’applicazione deve permettere di utilizzare nel miglior modo possibile le varie risorse, cercando di utilizzarle solo quando sono realmente necessarie.
* **Integrità**: Il software prima di essere condiviso viene testato e si controlla la sicurezza dei dati presenti all’interno dell’applicazione e si controlla la loro integrità in ogni condizione di utilizzo.
* **Usabilità**: Il prodotto deve essere il più semplice possibile da utilizzare, apprendere e operare in tutti i vari domini nei quali può essere utilizzata l’applicazione.Inoltre l’interfaccia grafica dell’applicazione deve essere accattivante, in modo tale da invogliare l’utente all’utilizzo.

I requisiti base per l’esecuzione dell’applicazione fitness app sono:

* Avere un dispositivo android.
* Avere una connessione a internet
* Avere una scheda video (anche integrata)
* Avere una conoscenza minima della palestra e dei vari esercizi correlati.

**Revisione del software**

* **Manutenibilità**: La fase di individuazione degli errori è semplificata dalla scelta di mantenere la struttura del programma semplice, la parte più delicata è la corretta interconnessione tra il database dell’applicazione e l’applicativo stesso, il quale deve rimanere consistente e affidabile.
* **Testabilità**: Tutte le funzionalità incluse sono testabili in qualunque momento tramite test manuali.
* **Flessibilità** la scelta della struttura del progetto semplifica la fase di modifica o implementazione di nuove funzionalità rendendolo molto flessibile.

**Transizione verso un nuovo ambiente**

**Portabilità**: Molto difficile, in quanto sviluppato specificatamente per Android, mal si presta alla transizione verso sistemi terzi (Apple) che richiedono un linguaggio di sviluppo totalmente diverso. Tuttavia il design del sistema può essere riutilizzato per implementarlo ex novo su altre piattaforme, ma questo esula dal codice scritto.

**Requirement Engineering**

La fase iniziale dello sviluppo è stata dedicata al project plan ovvero alla definizione dei requisiti e degli obiettivi da raggiugere dal committente.

Per la gestione del processo di sviluppo dell’applicazione sono stati definiti i requisiti utilizzando il criterio di MoSCoW:

* **Must have**:
  + Sviluppo di una versione dell’applicazione fitness app funzionante
  + Interfaccia grafica semplice e intuitiva
  + Stabilità e sicurezza del software
* **Should have**:
  + Ampia selezione dei vari esercizi
  + Facilità nella comprensione delle varie schede di allenamento
  + Possibilità di modifica di una scheda già esistente
  + Selezione del tipo di allenamento(cardio/pesistica)
* **Could have**:
  + Guida all’uso dell’applicazione
  + Timeout della durata delle schede di allenamento
  + Grafica diversificata per le fasi giorno e notte
* **Won't have**:
  + Impostazioni per modificare la grafica
  + Condivisione dei dati con altri utenti/personal trainer
  + Autenticazione tramite impronta digitale
  + Traduzione in altre lingue
  + Grafica diversificata giorno notte (tema chiaro tema scuro)

**MODELLING**

La fase di modellazione del software è una fase importante del processo di sviluppo del software. In questa fase i requisiti del software vengono tradotti in un modello di software che descrive l’architettura, i componenti, le funzionalità e le relazioni tra le diverse parti del sistema.

La fase di modellizzazione del software comprende diverse attività, tra cui:

* **Identificazione dei requisiti:** definire i requisiti del software, ovvero le funzionalità e le caratteristiche che il software deve avere.
* **Progettazione dell'architettura**: definire l'architettura del software, ovvero come i componenti del sistema interagiscono tra loro.
* **Progettazione dei componenti**: definire i componenti del software e come si integrano nell'architettura.
* **Definizione dei dati:** definire i dati che il software deve gestire, inclusi i formati dei dati, le strutture dei dati e le relazioni tra i dati.
* **Definizione del flusso di lavoro**: definire come il software deve gestire i processi e i flussi di lavoro, inclusi i processi aziendali, i processi di autorizzazione e i flussi di lavoro specifici del software.
* **Progettazione dell'interfaccia utente**: definire l'aspetto e il comportamento dell'interfaccia utente del software.

Durante la fase di modellazione sono stati utilizzati degli strumenti come UML(Unified Modelling Language), utilizzato per rappresentare il software e per comunicare le idee di progettazione in maniera chiara. Partendo dallo studio dei casi d’uso si va poi a modellare l’intera idea del nostro software, rendendolo conforme a ciò che l’utente intende fare con il sistema.

Immagine che contiene diagramma

Descrizione generata automaticamenteDi seguito **l’use case diagram:**

Sempre durante questa cruciale fase nell’ambito del processo di sviluppo, contemporaneamente a varie prove e studi effettuati si è giunti ad un class diagram definitivo, uno schema che ci permette di rappresentare in maniera chiara e sintetica tutte le classi che sono state create per lo sviluppo dell’applicativo. Nello schema sono state messe tutte quelle classi che hanno il solo obiettivo di implementare funzionalità “tecniche” come le stampe e la creazione di certi tipi non primitivi e metodi funzionali ai soli programmatori. Per quanto riguarda la logica dell’applicazione, insieme al suo funzionamento sono stati seguiti gli schemi UML

Il class diagram è utile per comprendere quali classi degli oggetti devono essere implementate all’interno del programma, le loro proprietà, i loro metodi e le relazioni tra di essi.

Come si può notare dal diagramma UML, l’applicazione ha una struttura gerarchica nella quale sono presenti i vari allenamenti nei diversi giorni della settimana, inoltre all’interno del workout abbiamo presenti i vari esercizi che si suddividono in Esercizio\_Cardio e Esercizio\_Pesistica.Immagine che contiene diagramma

Descrizione generata automaticamente

Durante lo sviluppo dell’applicazione fitness-app una parte essenziale era quella di rendere l’applicazione il più facile possibile da utilizzare per i vari utenti.

Durante l’avvio, l’utente dovrà andare a creare gli esercizi da inserire all’interno della scheda, in questo caso si può notare che sono già suddivisi i gruppi muscolari per facilitare l’utente nella creazione del workout e dell’esercizio per la scheda.

L’utente dopo aver selezionato il gruppo muscolare andrà a prendere uno degli esercizi generali presenti all’interno dell’elenco, nel caso l’esercizio voluto dall’utente non fosse presente all’interno dell’applicazione, l’utente avrà la possibilità di creare un nuovo esercizio da inserire all’interno degli esercizi presenti nell’elenco.

Infine dopo aver inserito il nome dell’esercizio, si andranno a completare gli altri campi(numero ripetizioni, tempo di recupero, numero serie, carico utilizzato) sempre mostrando all’utente un’interfaccia molto semplice e intuitiva.

Immagine che contiene diagramma

Descrizione generata automaticamenteDi seguito **l’activity diagram**

**Sequence diagram**

**Immagine che contiene diagramma

Descrizione generata automaticamente** L’activity diagram e il sequence diagram riportano in maniera schematica l’idea alla base dell’interfaccia utente.

Infine lo **state chart diagram** può essere un utile strumento per comprendere il comportamento dinamico di un sistema, facilitando la comprensione del flusso di lavoro e le possibili transizioni. Inoltre, può aiutare a identificare eventuali errori o problemi nel design del sistema, migliorando la qualità e l'efficacia del software o del processo modellato. Per rappresentare quindi gli stati del sistema durante la sua esecuzione abbiamo pensato un UML state chart diagram:

**Immagine che contiene diagramma

Descrizione generata automaticamente**

**Grafica**

Durante lo studio dell’interfaccia grafica, il punto di maggiore interesse era quello di ottenere un’applicazione il più facile e intuitiva possibile per ogni esigenza dell’utente.

**Software architecture**

Durante lo sviluppo del software, si è deciso di mantenere l’architettura e il design il più semplici possibili per l’utilizzo da parte dell’utente per creare, gestire e visualizzare i vari dati di una scheda di allenamento.

Il pattern architetturale utilizzato durante la programmazione del software è stato il Model-View-Controller(MVC).

In questo pattern il modello rappresenta la parte del sistema che gestisce i dati e la logica del programma, la vista si occupa di far visualizzare i dati al cliente in modo che possano essere compresi nel migliore dei modi e infine il controller gestisce l’interazione tra il modello e la vista.

Il team utilizzando un modello Agile ha trovato più comodo utilizzare il pattern MVC, questo perché il model-view-controller aiuta a separare i compiti all’interno del sistema, in modo che ciascuna parte possa essere gestita separatamente.

Ciò rende più facile mantenere il sistema nel tempo e consente di poter effettuare modifiche senza dover riprogettare l’intero sistema.

**Testing**

Durante la creazione dell’applicazione Fitness-App il software è stato continuamente testato per accertare il corretto funzionamento e per controllare che rispettasse tutti i requisiti definiti durante la fase di progettazione.

La fase di Testing del software è composta da diverse attività:

1. **Pianificazione dei test**: definire gli obiettivi dei test, i casi di test e le procedure da seguire durante i test.
2. **Progettazione dei test**: definire i casi di test, che includono input, output e comportamenti attesi.
3. **Esecuzione dei test:** eseguire i casi di test e verificare che il software si comporti come previsto.
4. **Registrazione dei risultati dei test**: registrare i risultati dei test, che possono includere successi, fallimenti e problemi riscontrati.
5. **Valutazione dei risultati dei test**: valutare i risultati dei test e determinare se il software rispetta i requisiti e le aspettative.
6. **Debugging:** correggere eventuali problemi riscontrati durante i test.

Durante la fase di test, ci siamo soffermati a implementare casi di test automatici mediante JUnit, dove utilizzando il metodo assertEquals() andiamo a controllare se i vari esercizi all’interno di una scheda sono effettevamente quelli che sono stati creati e inseriti all’interno del database.

Se il caso di test individuasse un errore, la scheda dovrà essere modificata e successivamente salvata nel database. Se invece il caso di test non ritorna errori, la scheda è pronta per essere utilizzata.

**Nello specifico:**

Durante lo sviluppo il corretto funzionamento del codice è stato testato costantemente con test di esecuzione del programma per provare ad implementare le sue funzionalità.

Questo processo è stato aiutato particolarmente dalla scelta di inserire all'interno del codice un gran numero di messaggi di debug, i quali ci hanno avvisato in tempo reale nel momento in cui avviene una situazione di errore.

Una volta ottenuta una versione definitiva del codice,il team ha deciso di svolgere dei test automatici mediante l’utilizzo di Junit, per verificare il corretto funzionamento del codice anche in presenza di casi limite.

Lo sviluppo di applicazioni Android si avvale di 2 macro categorie di test che è possibile implementare:

* Test sulle classi java, sui metodi, eseguiti sulla macchina di sviluppo
* Test sulla UI, il life cycle ed i suoi cambiamenti, eseguiti dalla macchina di sviluppo sul dispositivo Android emulato o fisico, creando un eseguibile .apk parallelo all’app.

Immagine che contiene testo

Descrizione generata automaticamente

Package di test del secondo tipo

Package di test del primo tipo

Per quanto riguarda la prima categoria, abbiamo utilizzato Junit 4.13.2 (era una versione di default proposta dall’IDE Android studio Flamingo 2022.2.1), e abbiamo testato i metodi Set, relativi agli oggetti esercizi, workout e alla classe Tempo, in modo da capire se effettivamente davano il risultato aspettato.

Immagine che contiene testo

Descrizione generata automaticamenteAd esempio, due metodi della classe TestTempo, con annotazione @Test, che si occupano di richiamare due metodi della classe Tempo e tramite assertEquals(), confrontare un risultato sperato con quello che gli viene restituito dopo le due conversioni, rispettivamente in stringa e in intero.

Immagine che contiene testo

Descrizione generata automaticamente

Eseguendo il primo metodo di test, ma “4:11” invece che “4:10”, l’IDE ci restituisce una chiara visione sul risultato del test, del perché è stato fallito e cosa ha ottenuto come risultato rispetto a quello aspettato.

Immagine che contiene testo

Descrizione generata automaticamente

Nella classe TestWorkout, abbiamo testato l’inserimento della lista di esercizi all’interno di Workout, usando assertArrayEquals(), creando quindi una lista di prova con 1 esercizio e confrontandola con quello che l’oggetto workout restituisce con il suo metodo getList\_esercizi().

Abbiamo tentato di coprire anche la seconda categoria di test tipicamente effettuati nei progetti Android, utilizzando un framework di test, già implementato di default nelle dipendenze del progetto, chiamato Espresso.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Nel nostro metodo di test, presente nella classe TestInAppUi, abbiamo testato la navigazione all’interno della UI dell’applicazione, in particolare simulando 2 click su due View: il pulsante nella BottomNavigationView per portarci sulla schermata degli esercizi, ed il pulsante per andare nella activity di creazione di un nuovo esercizio.

Il passaggio tra activity è gestito dall’oggetto Intent e dal metodo startActivity(intent). Quindi nel test, richiamando il metodo init(), ci mettiamo “all’ascolto” di un intent, facciamo eseguire i due click in sequenza e tramite intended, andiamo ad impostare il nome della Activity che ci aspettiamo dal test, infine si rilascia “l’ascolto” dell’intent tramite release().

Se premendo sul tasto, effettivamente si riscontra un intent per il passaggio alla schermata NuovoEsercizioActivity.class, allora il test sarà superato.

**Implementazione**

Per la fase di codice legato a classi di funzionalità e oggetti, abbiamo usato java dividendo in vari package la struttura del progetto.

Come alternativa avremmo potuto utilizzare il linguaggio kotlin(anche più supportato rispetto a java, ma avrebbe richiesto una formazione ad hoc nel team).

Per l’implementazione della classe relativa al database è stata scritta sempre in java ma includendo righe di linguaggio SQLite per la creazione di tabelle, uso di join e update.

La parte relativa a UI è stata implementata con il metalinguaggio xml.