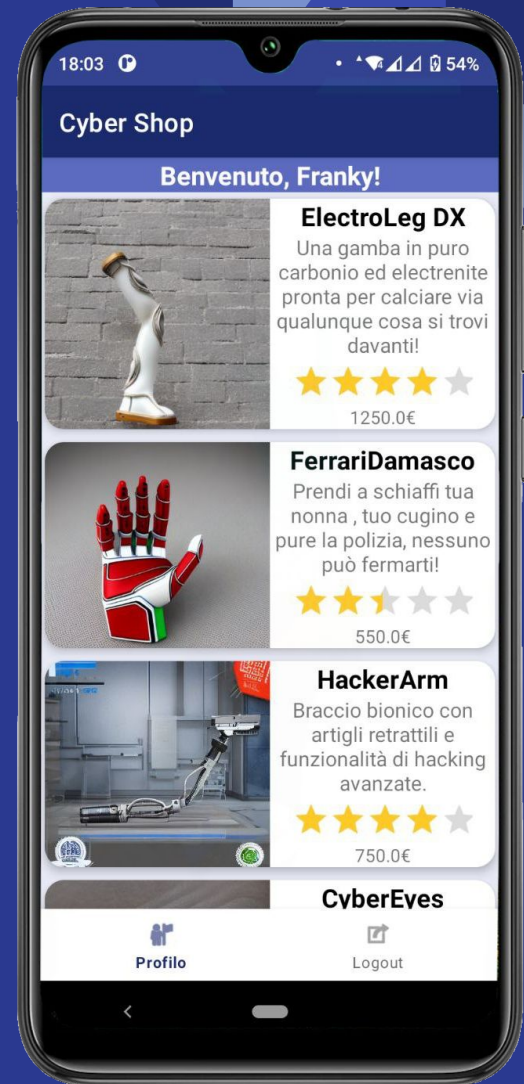


CyberShop

Siena Lorenzo 1000015814

Ingegneria informatica



Cos'è CyberShop?

Cybershop è un'app per la gestione di una fittizia attività commerciale del 2079 a tema cyberpunk.

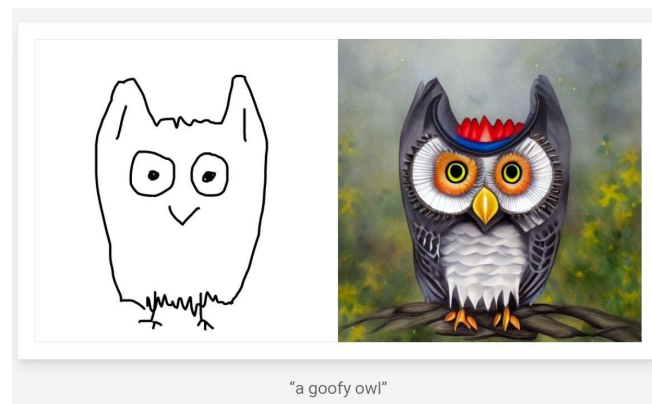
Il suo scopo è quello di vendere protesi cibernetiche da visualizzare nell'app attraverso la realtà aumentata o in 3D sotto forma di file STL, un formato di CAD, che una volta acquistato può essere stampato a casa usando la propria stampante 3D abilitata per la stampa di impianti e circuiti elettronici biocompatibili.

Tutte le immagini sono state generate con una IA **Scribble Diffusion**

(derivata da ControlNet) in grado di trasformare prompt

e semplici disegni fatti a mano in immagini generate dalla rete neurale.

Il progetto è Open Source e con licenza MIT.



Cos'è CyberShop?

L'app in sé soddisfa le specifiche del progetto e permette:

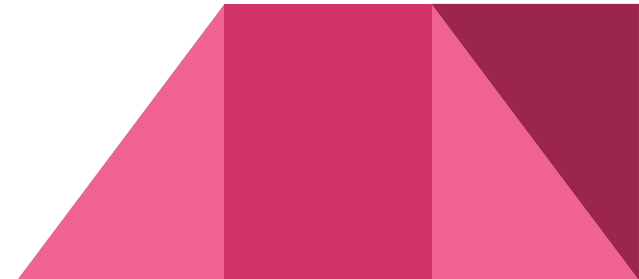
Registrazione e autenticazione degli utenti.

Accesso a un home con i prodotti presenti nello store.

La possibilità di aggiungere un prodotto alla wishlist dell'utente e/o recensirlo dando da 1 a 5 stelle.

Accesso al profilo dell'user autenticato per visualizzare la propria wishlist.

Accesso a una dashboard per l'amministratore che può aggiungere, modificare ed eliminare un prodotto direttamente dall'app, visualizzando i propri dati e statistiche base dello store.



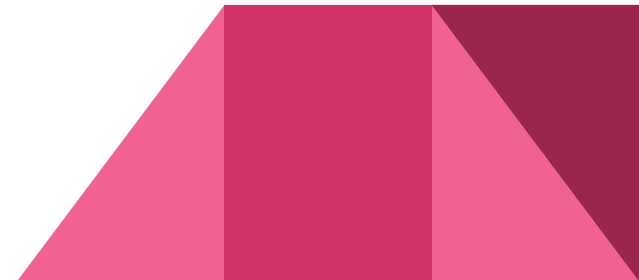
L'architettura, il paradigma e lo sviluppo

Sono stati valutati più approcci possibili allo sviluppo dell'applicazione:

Activity e fragment con riutilizzo dei layout.

Architettura a singola Activity e molti fragment da raggiungere con un Navigator.

Ma a causa della complessità della progettazione e della generalità del paradigma a oggetti dove le possibili implementazioni sono virtualmente infinite, si è scelto un approccio KISS (Keep it simple, stupid!) ,utilizzando un activity e un layout per schermata, mantenendo lo stato dell'applicazione attraverso Intent in modo da visualizzare informazioni necessarie.



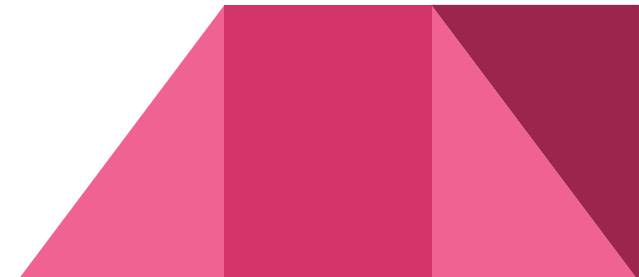
Il database

Per salvare le informazioni da caricare e visualizzare dinamicamente col client sono stati scelti i seguenti servizi di FireBase:

Authentication: per gestire email , UID utente e l'autenticazione.

Realtime Database: lista dei prodotti ,utenti e wishlist.

Storage: dove sono salvate immagine e STL del prodotto



Il database Authentication

Qui sono salvati dei profili da utilizzare nell'applicazione.
A scopo di test ogni user ha come password :**"password"**
tranne l'amministratore
dello shop che accede con

user: **admin@admin.com**
password: **administrator**

E' possibile registrarsi
da `SignupActivity`
e loggare da `LoginActivity`

Cerca per indirizzo email, numero di telefono o UID utente					Aggiungi utente	
Identificatore	Provider	Data di creazione	↓	Accesso eseguito	UID utente	
test4@test.com	✉	6 mar 2023		18 mar 2023	[REDACTED]	
test3@test.com	✉	23 feb 2023			[REDACTED]	
test2@test.com	✉	23 feb 2023			[REDACTED]	
test@test.com	✉	20 feb 2023		23 mar 2023	[REDACTED]	
admin@admin.com	✉	20 feb 2023		23 mar 2023	6rCsNsSUQjXCpX5UTo2yFRobEW...	

Righe per pagina: 50 1 - 5 of 5

Il database Realtime

“prodotti”

Qui è presente la lista dei prodotti nello shop.

Il ramo **prodotti** ha come figli chiavi **uid** randomiche generate durante la creazione del prodotto con gli attributi:


nome: stringa

descrizione: stringa

prezzo: float

urlImage: Path a Firebase che identifica la risorsa da utilizzare come immagine per il prodotto

urlStl: Path a Firebase che identifica la risorsa da utilizzare come file in 3d in formato STL per il prodotto.

 <https://cybershop-b79a9-default-rtdb.europe-west1.firebaseio.com>

<https://cybershop-b79a9-default-rtdb.europe-west1.firebaseio.com/>

▼ — prodotti

▼ — -NPrJtbhU8UUr8G2Ihk_

— descrizione: "Una gamba in puro carbonio ed electrenite pronta per calciare via qualunque cosa si trovi davanti!"

— nome: "ElectroLeg DX"

— prezzo: 1250

▼ — reviews

— 4Re2T5ZJuVQsAID0yS6q0R7SE1j2: 2

— CNL1q5XJhRhQROF72nih0cYpnbs2: 5

— kgx0HuTVX1WGuVohxrVZMZ7cuio2: 4

— zto1ZsfzTUPmkNNXWiPgWHli8b33: 5

— urlImage: "gs://cybershop-b79a9.appspot.com/IMAGE_STL/ceramic_leg.png"

— urlStl: "gs://cybershop-b79a9.appspot.com/FILE_STL/d20.stl"

▶ — -NPrJtbiTrmMLqj4tb4

▶ — -NPrJtbj00j9UDJsIKM7

Il database Realtime

“prodotti”

Qui è presente la lista dei prodotti nello shop.

Il ramo **prodotti** ha come figli chiavi **uid** randomiche generate durante la creazione del prodotto con gli attributi:


reviews:

qui sono presenti le valutazioni da 1 a 5 degli user secondo la struttura

UID utente:rating

Il rating medio del prodotto viene calcolato dalla classe

RecyclerViewAdapterLong quando il prodotto viene visualizzato sotto forma di card.

 <https://cybershop-b79a9-default-rtdb.europe-west1.firebaseio.com>

<https://cybershop-b79a9-default-rtdb.europe-west1.firebaseio.com/>

▼ — prodotti

▼ — -NPrJtbhU8UUr8G2Ihk_

— descrizione: "Una gamba in puro carbonio ed electrenite pronta per calciare via qualunque cosa si trovi davanti!"

— nome: "ElectroLeg DX"

— prezzo: 1250

▼ — reviews

— 4Re2T5ZJuVQsAID0yS6q0R7SE1j2: 2

— CNL1q5XJhRhQROF72nih0cYpnbs2: 5

— kgx0HuTVX1WGuVohxrVZMZ7cuio2: 4

— zto1ZsfzTUPmkNNXWiPgWH1i8b33: 5

— urlImage: "gs://cybershop-b79a9.appspot.com/IMAGE_STL/ceramic_leg.png"

— urlStl: "gs://cybershop-b79a9.appspot.com/FILE_STL/d20.stl"

▶ — -NPrJtbiTrmMLqj4tb4

▶ — -NPrJtbj00j9UDJsIKM7

Il database Realtime

“users”

Qui è presente la lista degli user iscritti allo shop.

Il ramo **users** ha come figli chiavi **uid** randomiche generate durante la creazione dell'user con gli attributi:

admin: **true** solo per l'admin

e **false** di default durante la creazione di un nuovo user

mail: che coincide con la mail salvata in Authentication

name: nome user

surname: cognome user

wishlist: contenente la chiave del prodotto e l'intero prodotto duplicato dal ramo “prodotti”.

N.B. questo tipo di ridondanza è un design pattern noto come

Denormalization e una best practices per Firebase

sia per i metodi della classe `FirestoreRecyclerOptions` che appartiene a `FirestoreUI` e che accetta interi oggetti quando fa la build e non permette di usare una lista di chiavi.

A questo si aggiunge che scalando l'applicazione i tempi di accesso per il database in caso di join diventano una latenza evidente e che la natura `Nosql` di Realtime non permette join.

<https://cybershop-b79a9-default-rtdb.europe-west1.firebaseio.com>

▼ users

▶ 4Re2T5ZJuVQsAID0yS6q0R7SE1j2

▼ 6rCsNsSUQjXCpX5Uto2yFRobEWe2

— admin: true

— mail: "admin@admin.com "

— name: "kill"

— surname: "bill"

▼ CNL1q5XJhRhQR0F72nih0cYpns2

— admin: false

— mail: "test@test.com"

— name: "Franky"

— surname: "Breakneck"

▼ wishlist

▶ -NPrJtbhU8UUr8G2Ihk_

▶ -NPrJtbiTrrmMLqj4tb4

▶ -NPrJtbmWU1dNG1PCzaL

Il database Realtime

“user_wish”

Per risolvere il problema di più path prodotto e quindi più rating che possano compromettere la consistenza dei dati e il calcolo medio della valutazione, è stata implementata una look up table che tiene conto dei prodotti in wishlist per utente.

Il ramo **user_wish** ha come figli chiavi **UIDuser** che ha come figli le chiavi dei **prodotti** e il valore true. Quando un user aggiunge un prodotto alla wishlist viene creata una voce, quando viene eliminato si elimina il nodo.

Questa look up table viene consultata quando avviene l'assegnazione del rating di un prodotto che va propagata in tutti i path necessari



Il database Realtime

“multipath update”

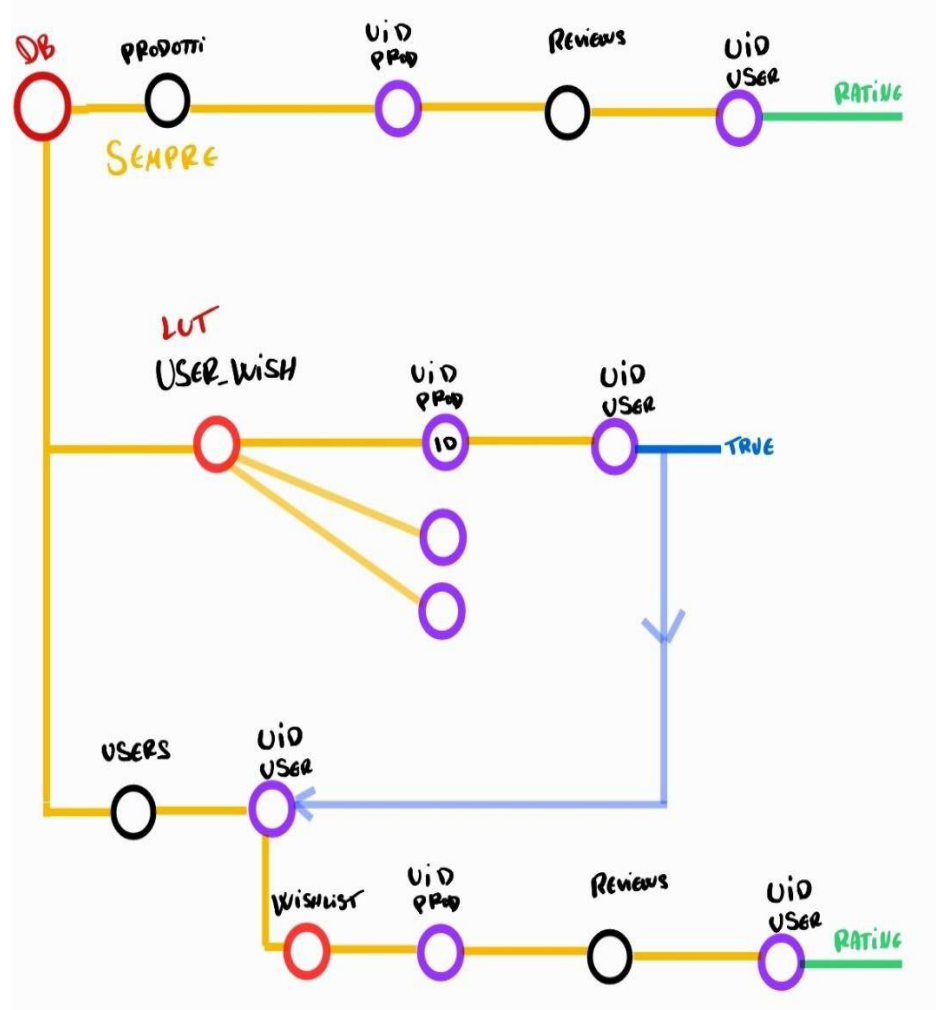
PATH:

“prodotti/prodotto”

“users/(UID_USER)/wishlist/Prodotto”

“user_wish/(UID_USER)/(UID_Prodotto:true)”

A destra è possibile visualizzare i percorsi per il “*multipath update*” che avviene in maniera **atomica** in FullProductActivity sincronizzando i prodotti ed il rating.



Il database Storage

Qui sono salvate le immagini e gli stl da caricare nelle anteprime e da scaricare



<input type="checkbox"/>	Nome	Dimensioni	Tipo	Ultima modifica
<input type="checkbox"/>	FILE_STL/	—	Cartella	—
<input type="checkbox"/>	IMAGE_STL/	—	Cartella	—

Le posizioni dei prodotti in RealTime **urlImage** e **urlStl** rispecchiano il formato

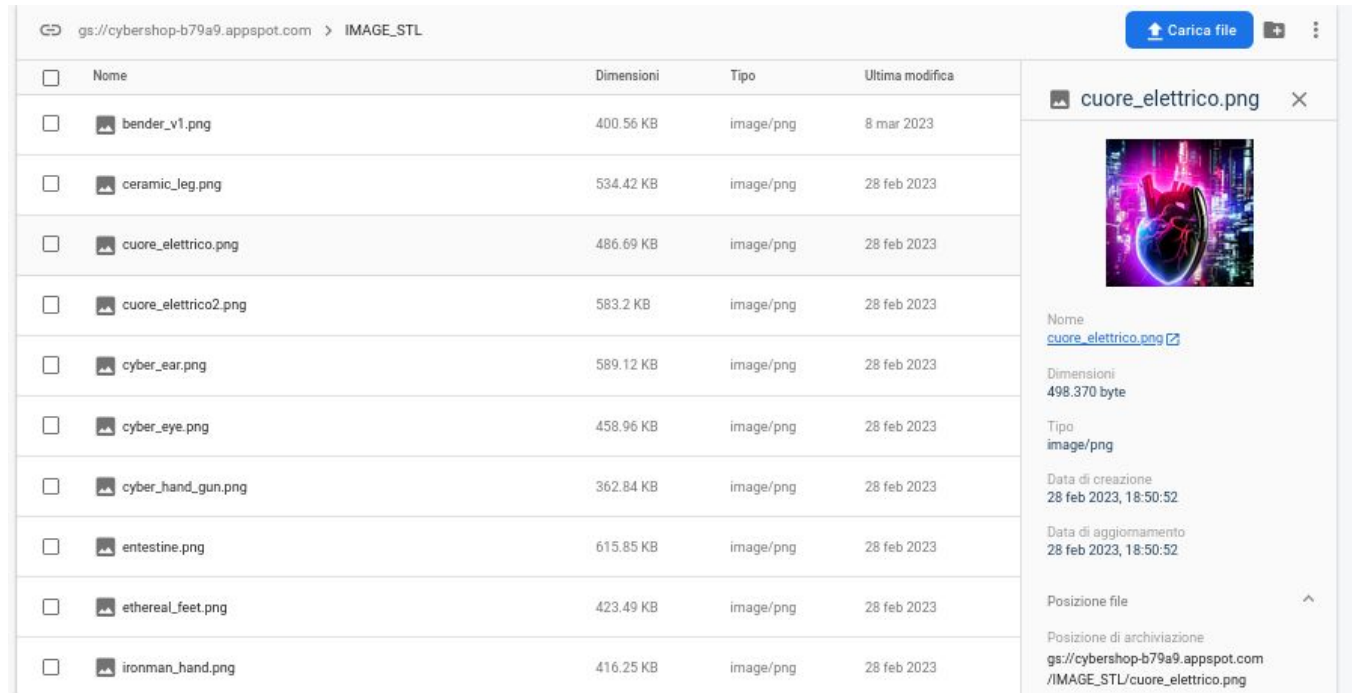
gs://xxxxxxx.appspot.com/IMAGE_STL/nomefile.png

gs://xxxxxxx.appspot.com/IMAGE_STL/nomefile.png











Il database Storage

"IMAGE_STL"


Le immagini in IMAGE_STL vengono caricate dall'applicazione grazie al plugin Glide che prende la posizione di archiviazione e ne setta il riferimento alla ImageView che fa richiesta.




gs://cybershop-b79a9.appspot.com > IMAGE_STL

<input type="checkbox"/>	Nome	Dimensioni	Tipo	Ultima modifica
<input type="checkbox"/>	 bender_v1.png	400.56 KB	image/png	8 mar 2023
<input type="checkbox"/>	 ceramic_leg.png	534.42 KB	image/png	28 feb 2023
<input type="checkbox"/>	 cuore_elettrico.png	486.69 KB	image/png	28 feb 2023
<input type="checkbox"/>	 cuore_elettrico2.png	583.2 KB	image/png	28 feb 2023
<input type="checkbox"/>	 cyber_ear.png	589.12 KB	image/png	28 feb 2023
<input type="checkbox"/>	 cyber_eye.png	458.96 KB	image/png	28 feb 2023
<input type="checkbox"/>	 cyber_hand_gun.png	362.84 KB	image/png	28 feb 2023
<input type="checkbox"/>	 entestine.png	615.85 KB	image/png	28 feb 2023
<input type="checkbox"/>	 ethereal_feet.png	423.49 KB	image/png	28 feb 2023
<input type="checkbox"/>	 ironman_hand.png	416.25 KB	image/png	28 feb 2023

[Carica file](#)

 cuore_elettrico.png



Nome
[cuore_elettrico.png](#)

Dimensioni
498.370 byte

Tipo
image/png

Data di creazione
28 feb 2023, 18:50:52

Data di aggiornamento
28 feb 2023, 18:50:52

Posizione file

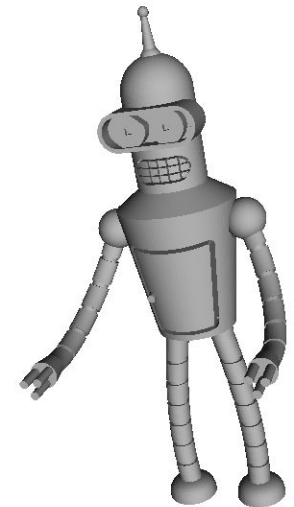
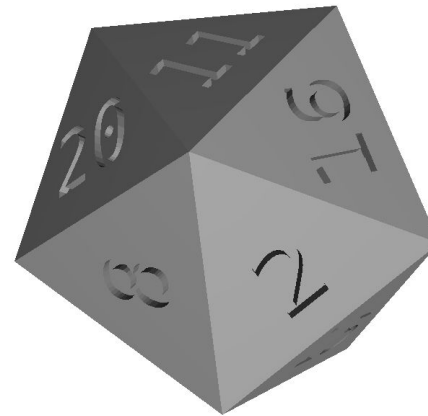
Posizione di archiviazione
gs://cybershop-b79a9.appspot.com /IMAGE_STL/cuore_elettrico.png

Il database Storage

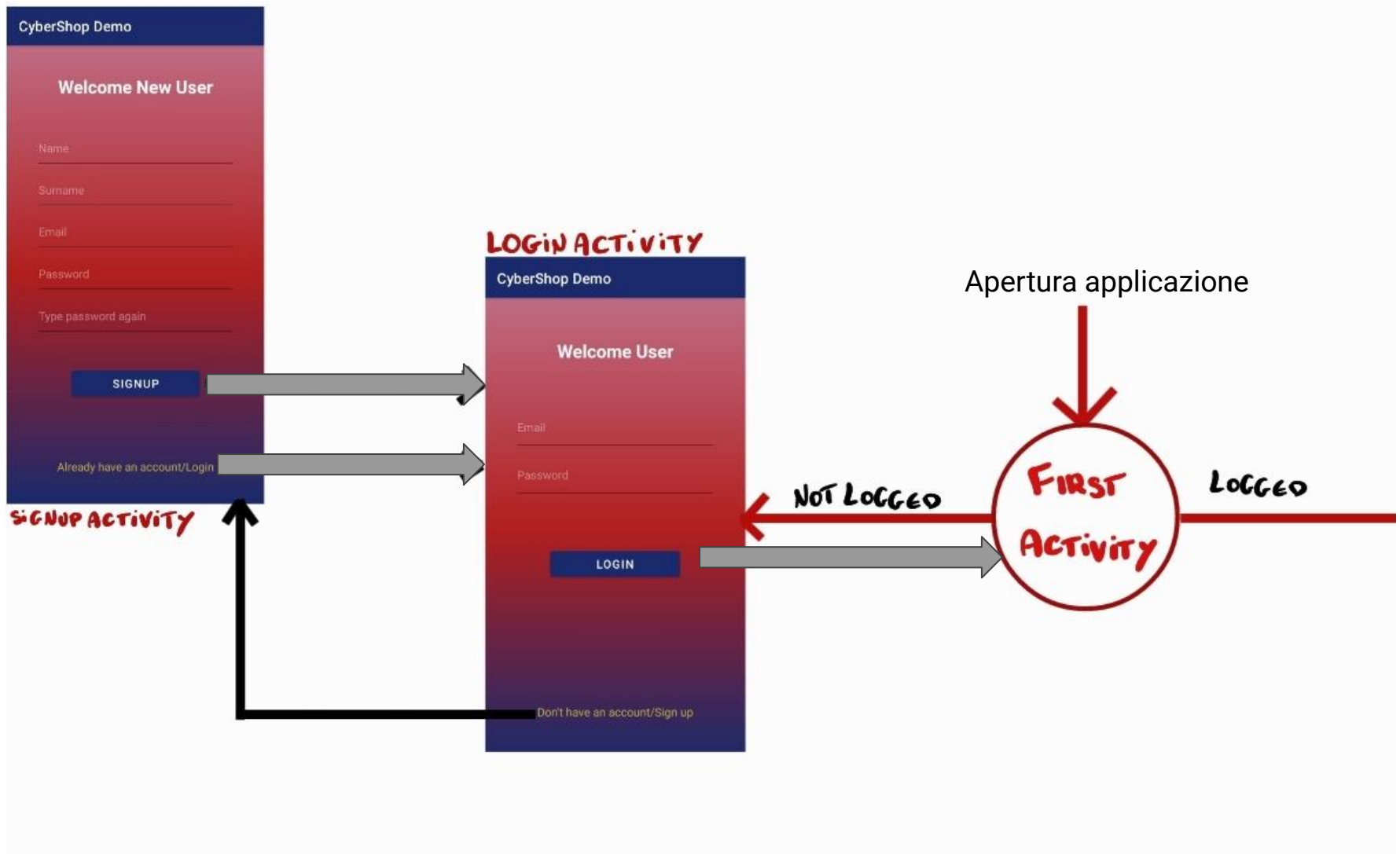
"FILE_STL"

gs://cybershop-b79a9.appspot.com > FILE_STL		Carica file + ⋮		
<input type="checkbox"/>	Nome	Dimensioni	Tipo	Ultima modifica
<input type="checkbox"/>	 bender_v1.stl	3.61 MB	model/stl	28 feb 2023
<input type="checkbox"/>	 d20.stl	226.06 KB	model/stl	28 feb 2023

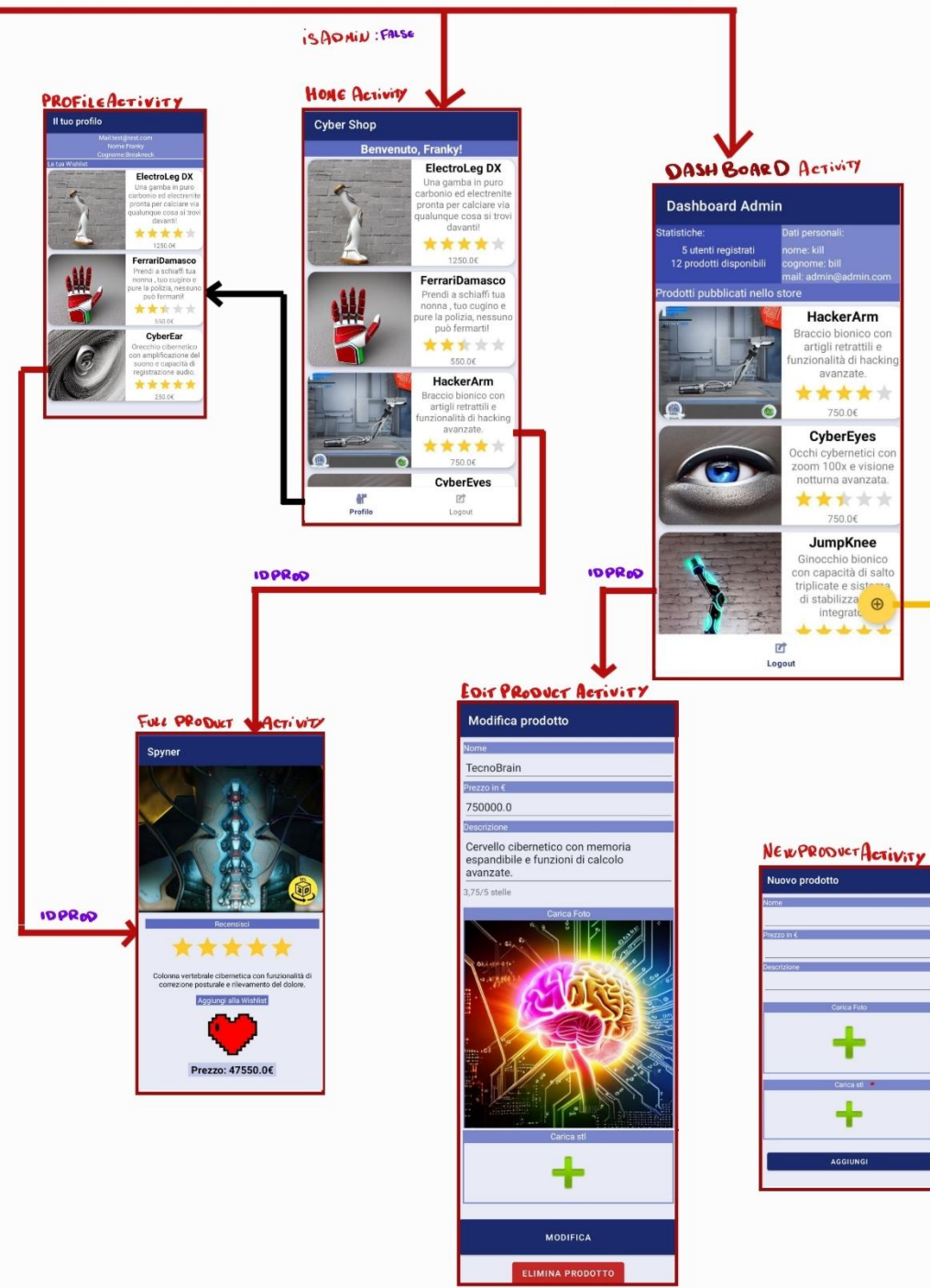
Qui sono presenti 2 file stl di test pronti per l'uso e per la stampa che lasciano spazio a future implementazioni non presenti in questa demo, in quanto la libreria ARCore **non** viene implementata come da specifiche.



Struttura di navigazione Not Logged



isAdmin: True



FirstActivity

Activity di ingresso che reindirizza l'utente in 3 stati possibili:

Loggato con diritti di User spedito alla `HomeActivity`

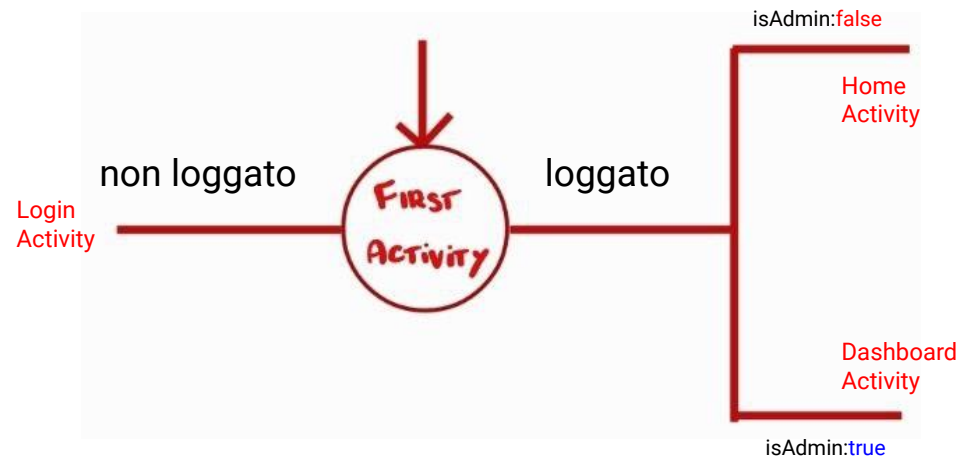
Loggato con diritti di Superuser spedito a `DashboardActivity`

Non loggato spedito a `LoginActivity`

Database coinvolti

[FirebaseAuth e Realtime Database]

Si controlla che l'user corrente sia autenticato ,
che esista nel database e se è un admin.



LoginActivity

Activity che setta 2 listener sul tasto di login e sulla scritta signup

se clicca **login** :

fa il check delle credenziali inserite e in base al tipo di utente manda un intent verso la home o la dashboard.

se clicca su **Don't have an account/Sign up** :

spedisce l'utente nella pagina di registrazione.

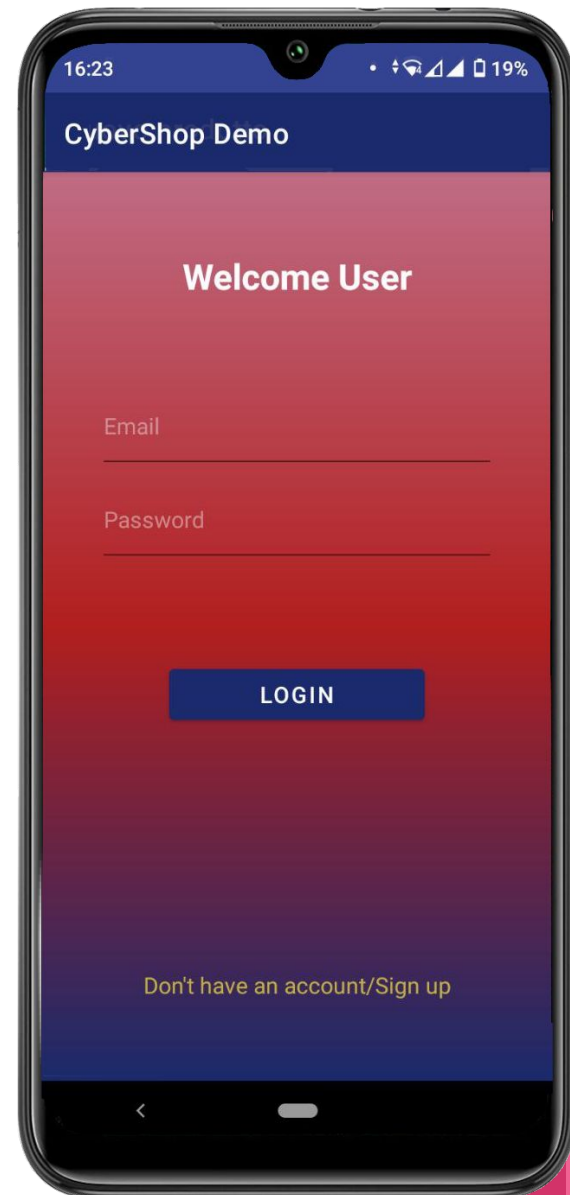
Database coinvolti

[FirebaseAuth]

Autenticazione User.

[Realtime Database]

Controlla che l'user sia regolarmente registrato.



SignupActivity

Activity di registrazione che setta 2 listener sul tasto di signup e sulla scritta login

Quando l'user clicca su **signup** :

controlla che i dati inseriti non siano vuoti e che siano validi in caso positivo reindirizza alla pagina di login.

Se clicca **Already have an account/Login** :

spedisce indietro l'utente nella pagina di login.

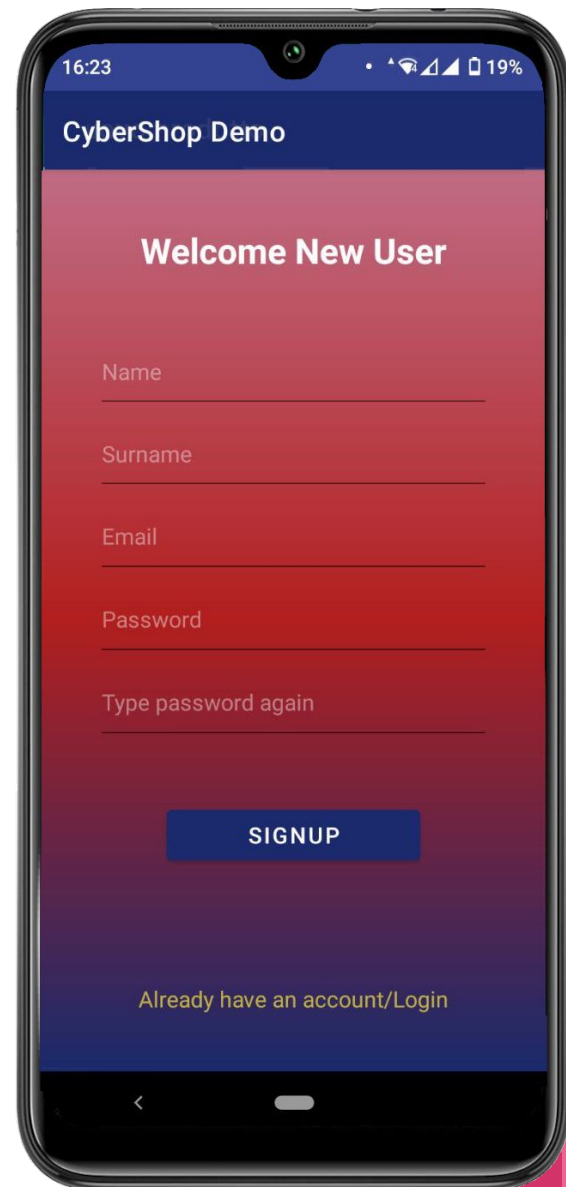
Database coinvolti

[*FirebaseAuth*]

Creazione dell'user

[*Realtime Database*]

Controlla che l'user sia stato creato



HomeActivity

Inizializza un menù da mettere sotto che porta a:
profile dell'user [ProfileActivity]
oppure al logout [LoginActivity]

Estrae dal database la lista dei prodotti e la inserisce nel
recyclerView dove mostra uno a uno i prodotti sotto forma di
card [long_card.xml]

Implementa e setta un listener per ogni card che spedisce
l'user e id del prodotto cliccato a [FullProductActivity]
in modo da visualizzarlo con tutti i dettagli.

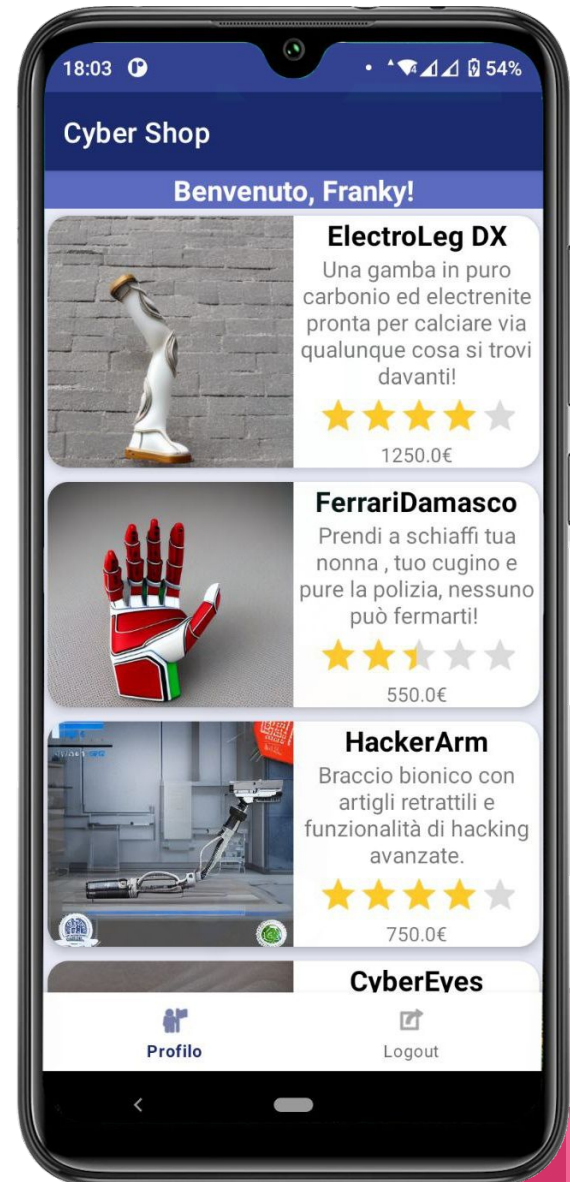
Database coinvolti

[FirebaseAuth]

Legge il nome user per salutare l'utente nella schermata in alto

[Realtime Database]

Legge la lista "prodotti" e li visualizza a schermo.



FullProductActivity

Activity che ha il compito di visualizzare un prodotto nello shop (se viene da HomeActivity) o in wishlist (se viene da ProfileActivity)

Riceve dall'Activity precedente un intent con l'id del prodotto da visualizzare

Da qui è possibile :

- Visualizzare **nome, descrizione, immagine, prezzo** e **recensione** precedente [1-5] stelle, 0 stelle se mai recensito.
- Visualizzare* in 3d l'stl del prodotto attraverso un **fab**.

[Feature non presente in questa demo in quanto ARCore non viene implementato come da specifiche, premendo il fab un messaggio di Snackbar avverte l'utente, Il file stl è in ogni caso disponibile e funzionale nel database [Storage] per una successiva implementazione]

- Aggiungere il prodotto alla propria wishlist
- Aggiungere una nuova recensione al prodotto [1-5] stelle

con

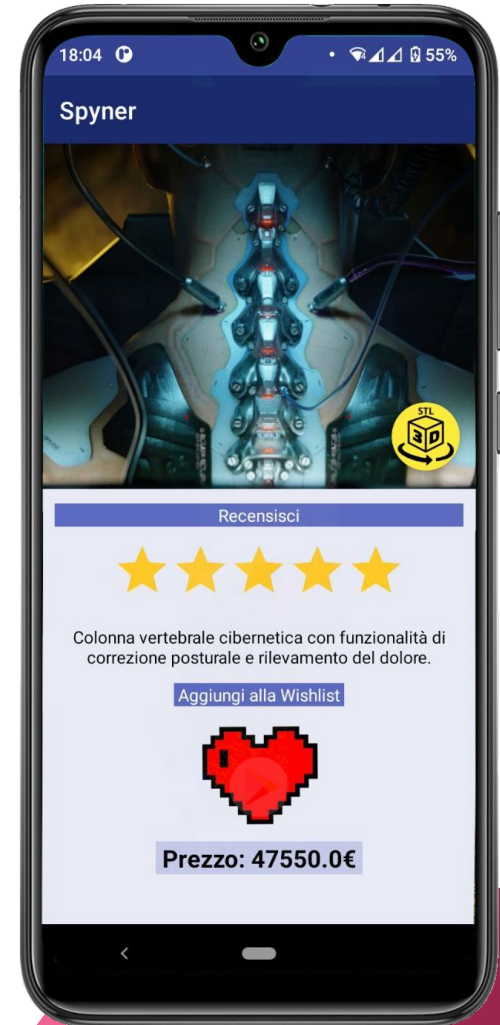


FullProductActivity

L'Activity imposta i listener per:

Aggiungere una recensione al prodotto da 1 a 5 stelle, attraverso ***onRatingChanged*** controllando e riempiendo la barra con la recensione precedente se presente.

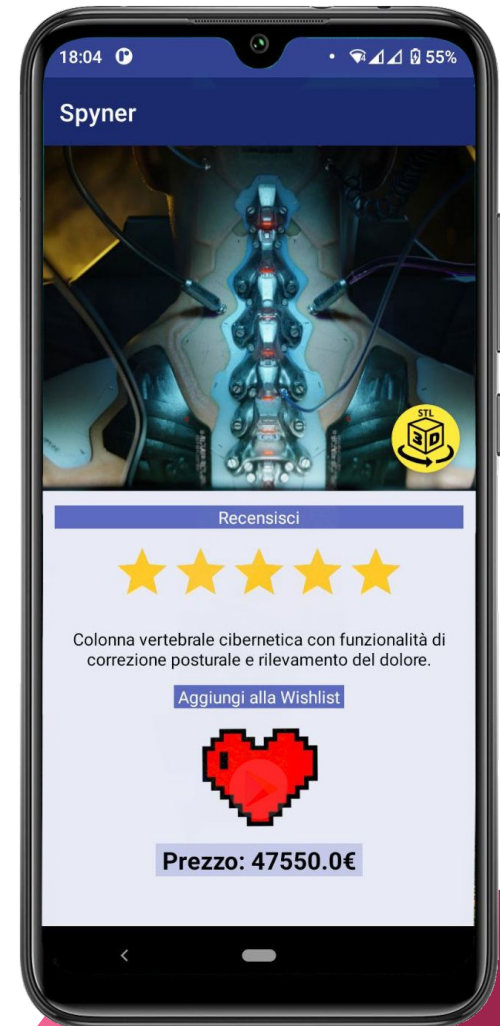
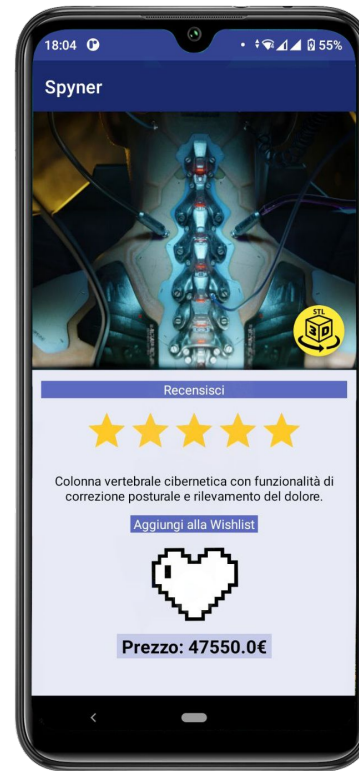
(implementazione completa nella documentazione della classe FullProductActivity)



FullProductActivity

L'Activity imposta i listener per:
Aggiungere il prodotto alla
wishlist attraverso
buttonWishlist.setOnClickListener
controllando se il prodotto è già in
wishlist e di conseguenza l'icona
del "cuore" associata.

(implementazione completa
nella documentazione della
classe FullProductActivity)



FullProductActivity

Database coinvolti

[*FirebaseAuth*]

Per prendere l'uid dell'user loggato.

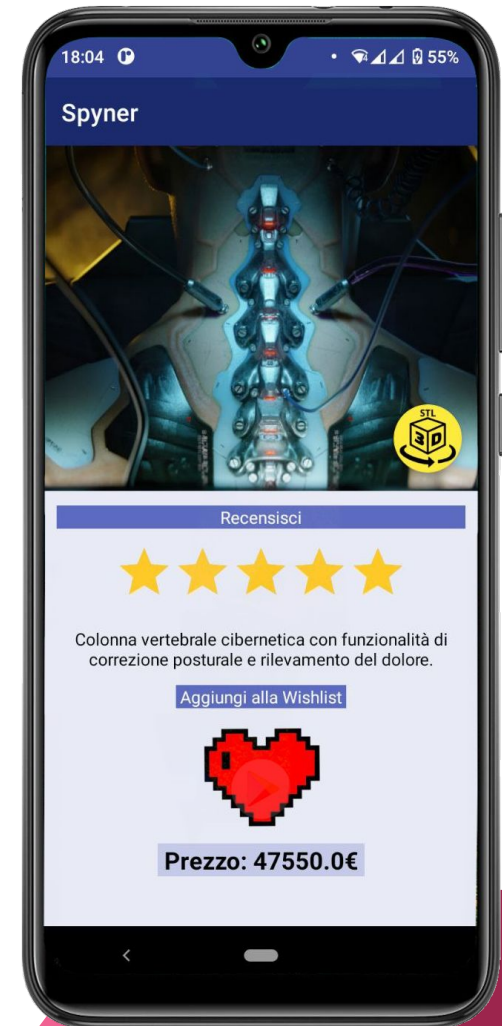
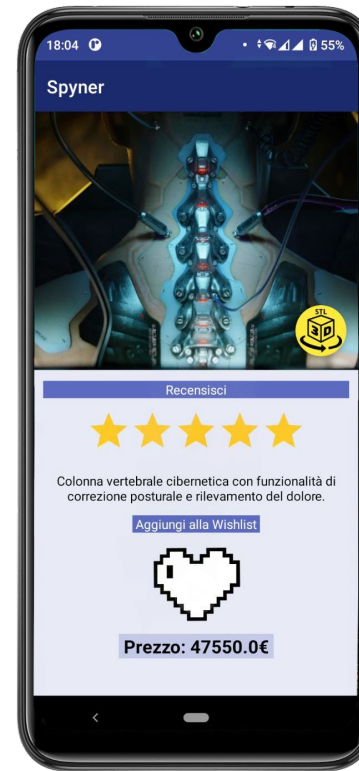
[*Realtime Database*]

Per visualizzare il prodotto e per aggiungere/rimuoverlo dalla wishlist dell'utente loggato.

Per mantenere la look up table nel path
“user_wish/(UID_PROD)/(UID_USER)”

[*Storage*]

Per visualizzare con Glide l'immagine del prodotto ed eventualmente l'stl da visualizzare.



ProfileActivity

Estrae dal database la lista dei prodotti in wishlist dell'user e la inserisce nel recyclerView dove mostra uno a uno i prodotti sotto forma di card (long_card.xml)

Implementa e setta un listener per ogni card che spedisce l'user e id del prodotto cliccato a [FullProductActivity] in modo da visualizzarlo con tutti i dettagli.

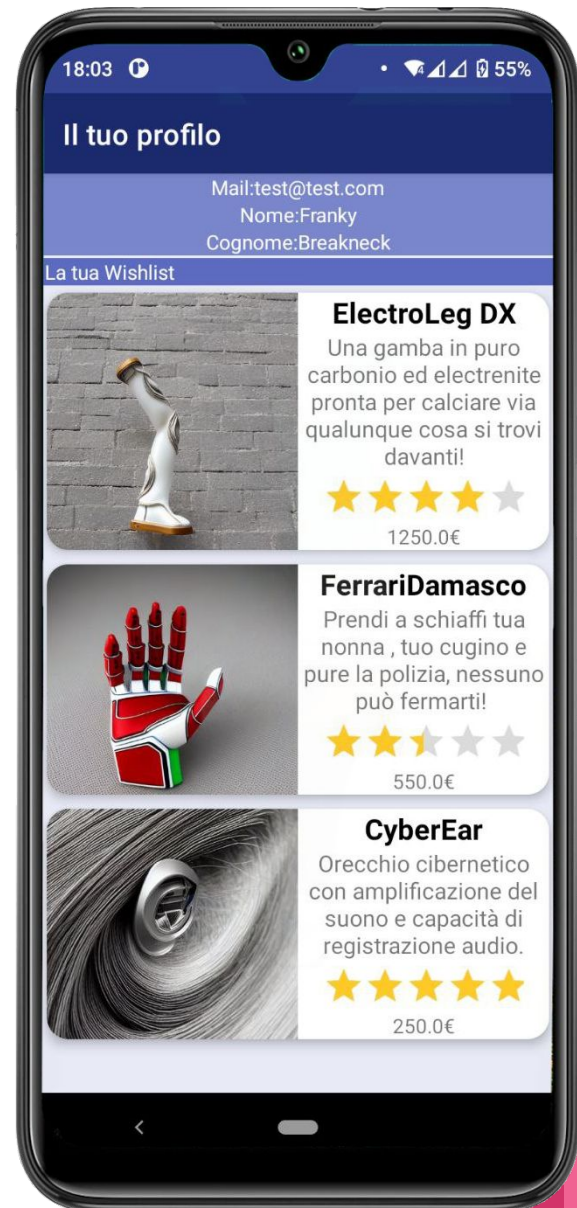
Database coinvolti

[FirebaseAuth]

Visualizza e stampa :nome ,cognome e mail.

[Realtime Database]

Legge la lista dei prodotti da "users/(UID_USER)/wishlist/prodotto" e li visualizza a schermo.



DashboardActivity

Questa è la home per l'admin che visualizza :

I dati dell'amministratore: nome,cognome e mail.

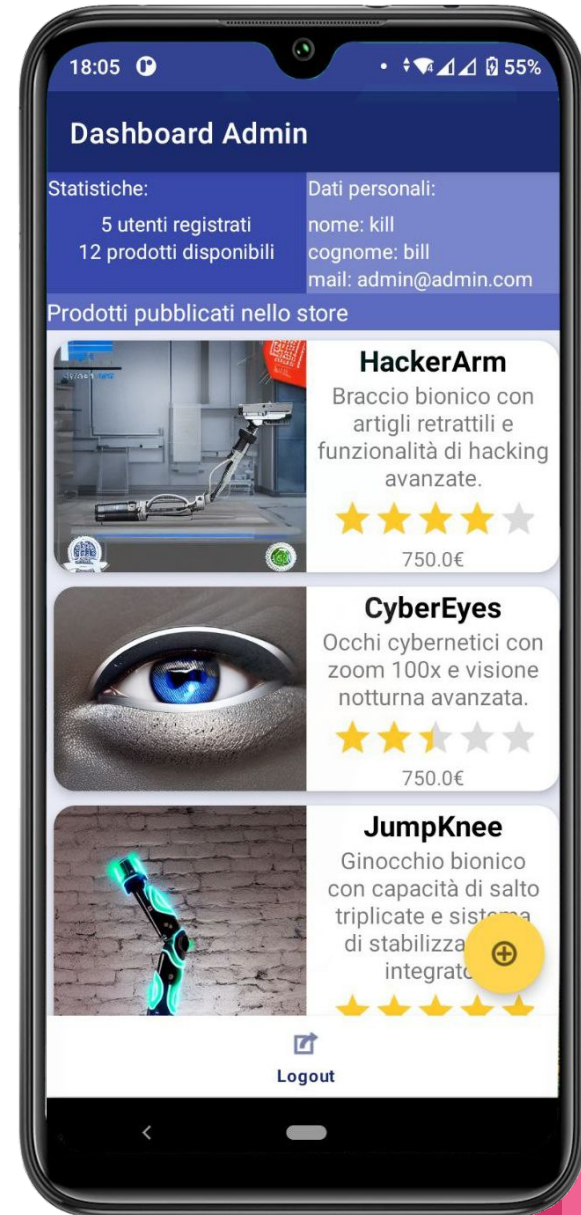
Statistiche del negozio: numero prodotti in negozio e numero user registrati.

Da qui è possibile manipolare il database e i prodotti dello shop per:

1) Modificare o cancellare un prodotto cliccando su un elemento della lista dei prodotti che una volta cliccati reindirizzano a `EditProductActivity` [con l'id del prodotto da editare]

2) aggiungere un prodotto nel negozio cliccando sul **fab (+)** flottuante sopra la lista dei prodotti del negozio , venendo spediti a `NewProductActivity`.

Cliccando la navigation bar in basso si esegue il logout.



DashboardActivity

Database coinvolti

[FirebaseAuth]

Vengono controllati i privilegi dell'user nella pagina

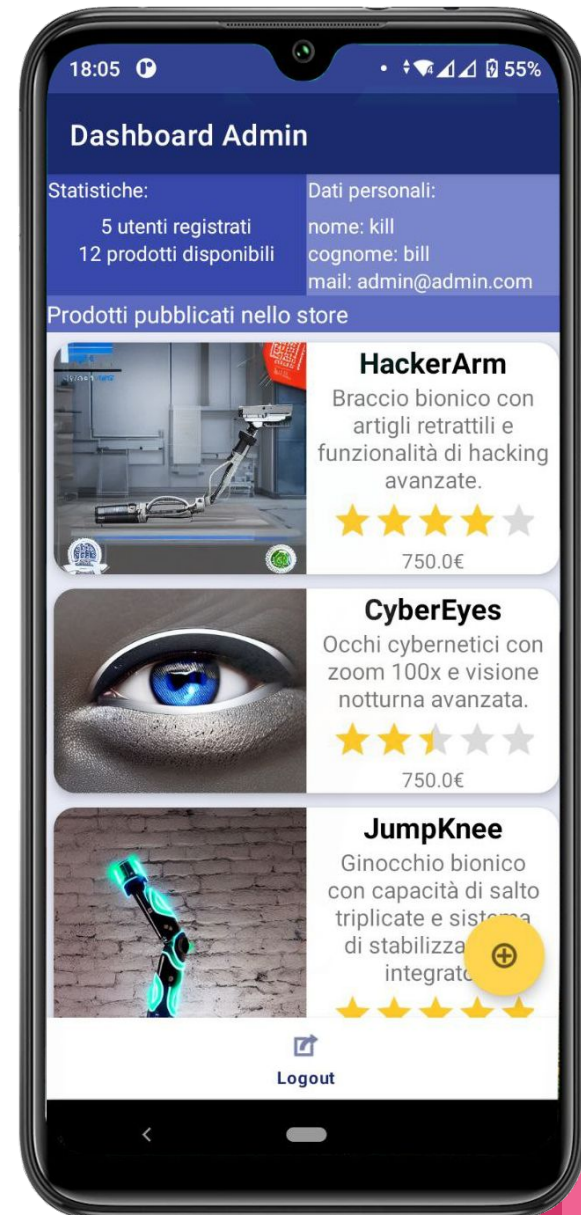
[Realtime Database]

Si visualizzano i prodotti

Si ricava nome,cognome e mail dell'admin.

Si conta il numero di utenti registrati.

Si conta il numero di prodotti nel negozio.



NewProductActivity

Dopo aver cliccato sul fab (+) nella dashboard activity si apre questa activity dove è possibile aggiungere :

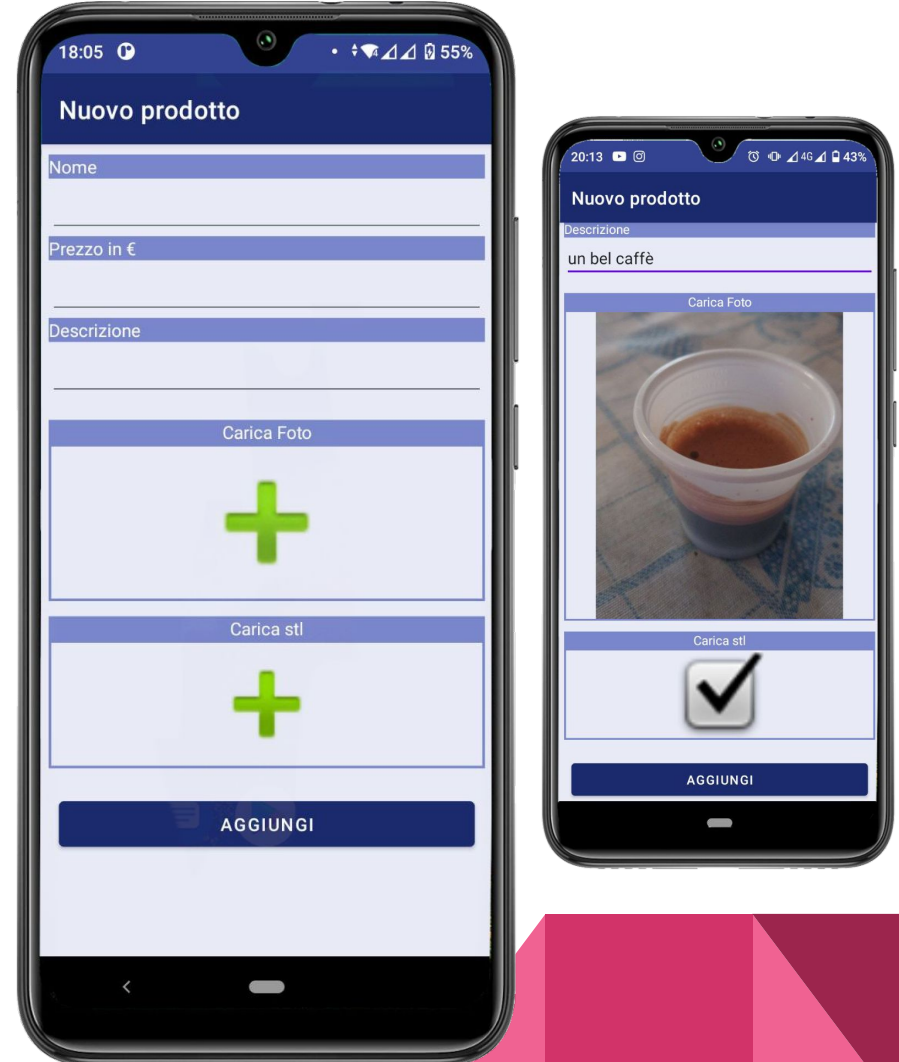
nome,
descrizione ,
prezzo ,
l'immagine del prodotto
e il file stl del prodotto [che andranno in Storage].

Sotto **Carica Foto** E' presente un ImageButton **previewImage** con un listener che invoca **selectImage()** per scegliere l'immagine dalla memoria del dispositivo che:

Imposta il path del file in locale in **mImageUri** attraverso un **onActivityResult** e che setta l'immagine nella **previewImage**

Sotto **Carica stl** è presente un ImageButton **previewStl** con un listener che invoca **fileChooser()** per scegliere il file stl dalla memoria del dispositivo che:

Imposta il path del file in locale in **mStlUri** attraverso un **onActivityResult** e che setta [✓] nella **previewStl**



NewProductActivity

In fondo è presente **AGGIUNGI** con un listener settato che una volta cliccato invoca **addNewItemToDatabase()** che :

Controlla che i dati inseriti siano tutti validi e non nulli rimandando il focus se qualche campo non è presente e infine contatta il db per inserire nel database [*Realtime Database*] il nuovo prodotto con i riferimenti **mStorageRef** e **mStorageRef2** per inserire l'immagine e l'stl in [*Storage*].

Se l'activity viene distrutta perché il telefono viene ruotato attraverso **onSaveInstanceState** vengono ricaricati e controllati **mStlUri** e **mImageUri** in modo da non doverli reinserire.

Se il prodotto viene inserito correttamente si viene rimandati a *DashboardActivity* con un toast di conferma.



NewProductActivity

Database coinvolti

[*Realtime Database*]

Viene contattato per aggiungere il nuovo prodotto

[*Storage*]

viene contattato per:

aggiungere l'immagine in

gs://cybershop.../IMG_STL/nome.png

e il file Stl in

gs://cybershop.../FILE_STL/nome.stl



EditProductActivity

Dopo aver cliccato su un prodotto della lista nella `DashboardActivity` si apre questa activity dove è possibile **modificare** il prodotto usando l'id del prodotto ricevuto :

nome, descrizione , prezzo ,l'immagine del prodotto e il **file stl** del prodotto [che andranno in Storage]

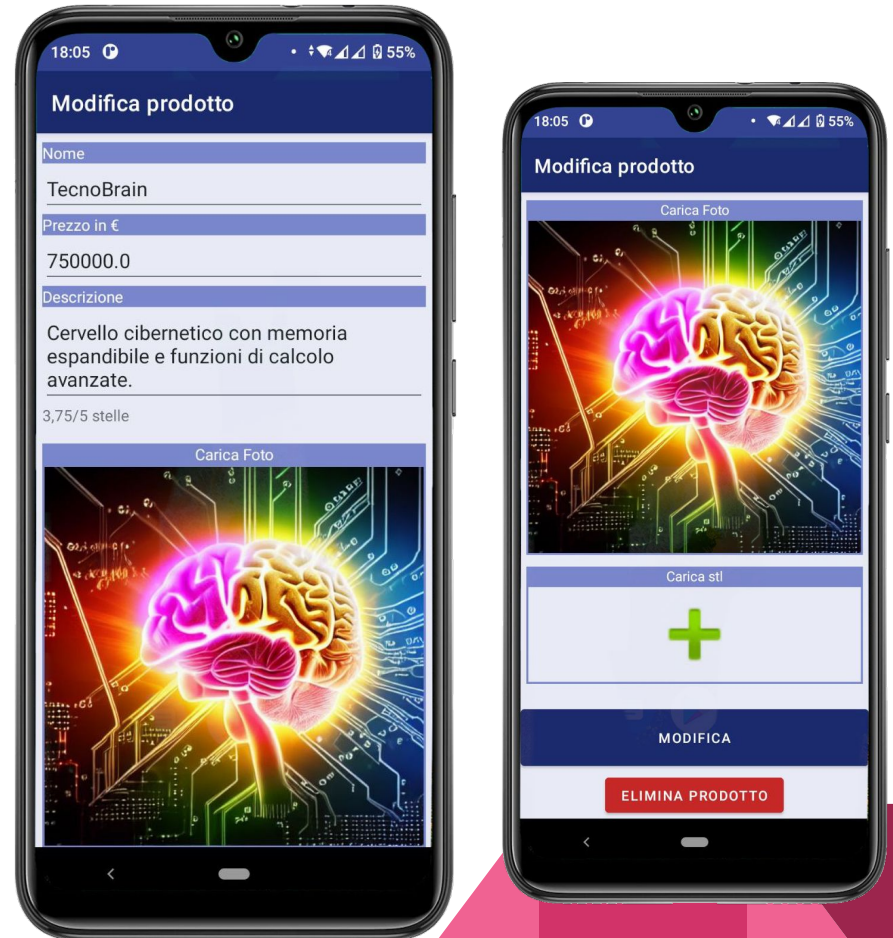
I campi sono **precompilati** con i dati del prodotto da modificare.

Sotto **Carica Foto** E' presente un ImageButton **previewImage** con un listener che invoca **selectImage()** per scegliere l'immagine dalla memoria del dispositivo che:

Imposta il path del file in locale in **mImageUri** attraverso un **onActivityResult** e che setta l'immagine nella **previewImage** sostituendo il riferimento precedente

Sotto **Carica stl** è presente un ImageButton **previewStl** con un listener che invoca **fileChooser()** per scegliere il file stl dalla memoria del dispositivo che:

Imposta il path del file in locale in **mStlUri** attraverso un **onActivityResult** e che setta [✓] nella **previewStl** sostituendo il riferimento precedente



EditProductActivity

E' presente **MODIFICA** con un listener settato che una volta cliccato invoca **modItemToDatabase()** che :

controlla che i dati inseriti siano tutti validi rimandando il focus se qualche campo non è presente

e infine contatta il db per inserire nel database[*Realtime Database*] il prodotto modificato mantenendo l'id con i riferimenti **mStorageRef** e **mStorageRef2** per inserire l'immagine e l'stl in [*Storage*] in caso di modifica.

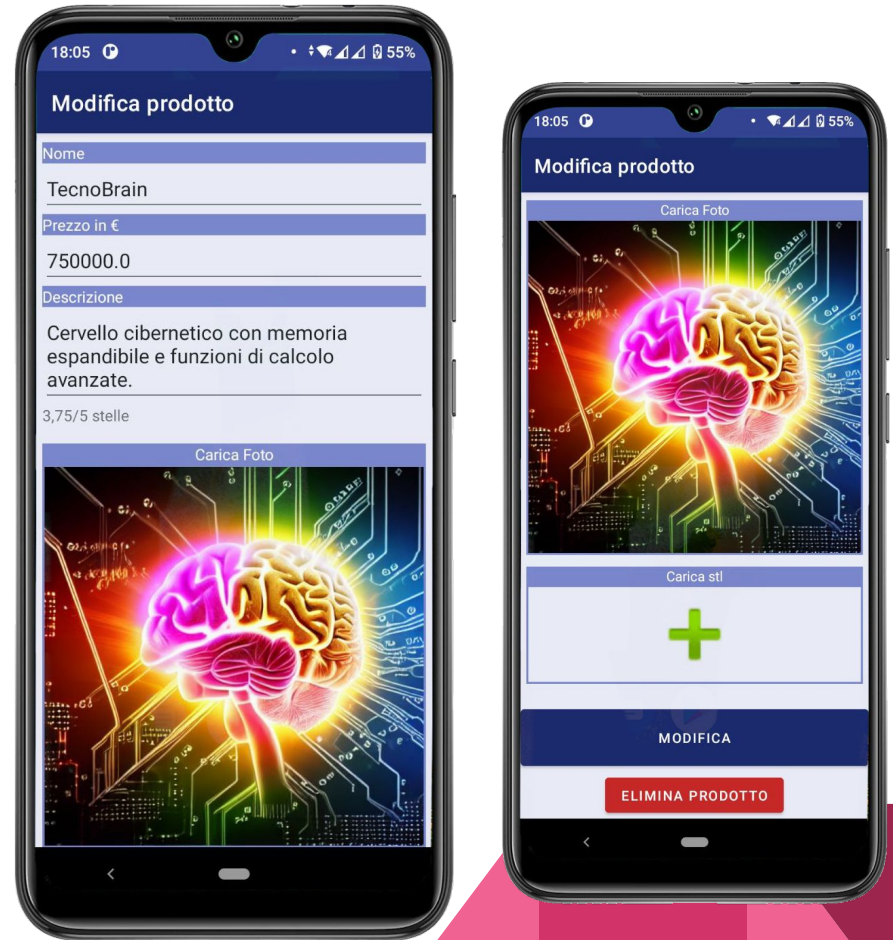
Ovviamente cambiando i file stl e/o immagine i vecchi file in storage saranno eliminati.

Per mantenere una consistenza rigida dei casi e ottimizzare gli accessi a [*Storage*] durante la modifica ci sono 4 casi possibili

1. non cambio nè stl nè png
2. cambio solo stl
3. cambio solo png
4. cambio entrambi

Se un dato non viene modificato si userà il dato del prodotto iniziale.

Se il prodotto viene modificato correttamente si viene rimandati a DashboardActivity con un toast di conferma.



EditProductActivity

E' presente **ELIMINA PRODOTTO** con un listener settato che una volta cliccato invoca **deleteItemToDatabase(id)** che :

interrogherà il database per l'id del prodotto e lo eliminerà dalla voce presente in [Realtime Database] e i relativi file associati in [Storage]

Database coinvolti

[Realtime Database]

Interroga il database per la lettura, modifica e eliminazione del prodotto

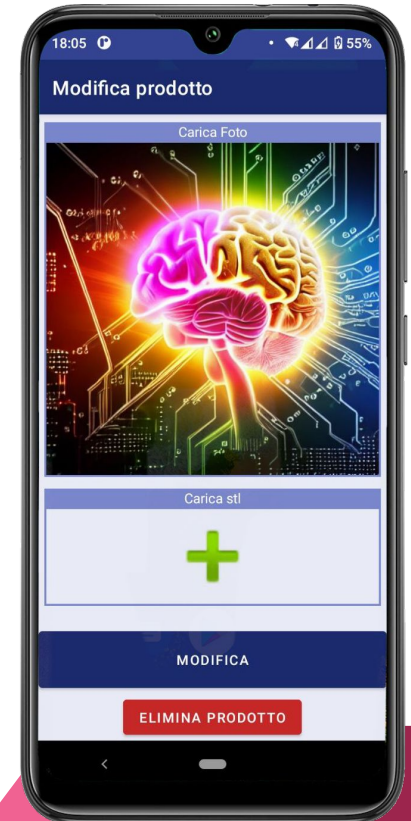
[Storage]

viene contattato per:

caricare l'immagine del prodotto da modificare

eliminare il file stl e png da sostituire

aggiungere l'immagine in **gs://cybershop.../IMG_STL/nome.png**
e il file Stl in **gs://cybershop.../FILE_STL/nome.stl**



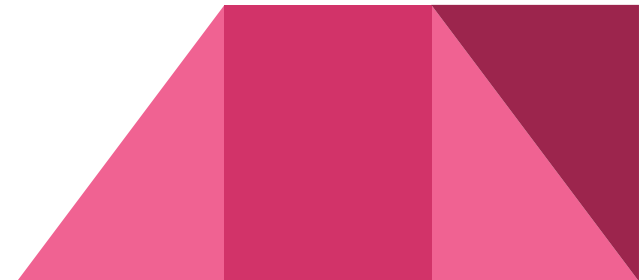
Stile e colori

Per i colori è stata scelta la palette material 50,100,200 di una sfumatura di viola.

E' stata seguita la convenzione di nomi:

NameActivity -> activity_name.xml

Ogni Activity funziona sia in verticale che in orizzontale senza overflow del contenuto grazie all'attenta progettazione del layout.



Grazie dell'attenzione

Tutto il codice, le risorse, il database
Realtime e il relativo apk sono presenti
all'interno del progetto android studio in
formato zip.

