Diego Mazzalovo
Matricola 1236592

E-Mail : diego.mazzalovo@studenti.unipd.it

Mobile Programming e Multimedia

Cyberpunk Characters Manager

Indice

1 Introduzione

1.1 Descrizione

Cyberpunk 2020, è un gioco di ruolo creato dalla *Wizards of the Coast*[1] negli anni 90, nel quale gli autori cercano di descrivere come sarebbe stato il mondo nel 2020. Essi, si immaginavano una società molto diversa rispetto a quella in cui viviamo, dove sarebbero stati presenti molti esseri umani "cybernetizzati", ovvero dei mezzi robot, con chips in grado di fornire particolari abilità e arti cybernetici. Inoltre, pensavano che il mondo sarebbe interamente stato dominato da grandi multinazionali chiamate *Corporazioni*(che non si distoglie troppo dalla realtà). Come ultima grande differenza, gli autori immaginavano l'utilizzo di internet tramite collegamenti neurali direttamente al cervello della persona.

1.2 Finalità

Cyberpunk Characters Manager è un'applicazione per mobile, sviluppata con l'utilizzo del framework *Flutter*[2], creata al fine di consentire agli utilizzatori una comoda gestione dei propri personaggi per l'omonimo gioco ed avere sempre accesso al manuale di gioco per visualizzarne le regole per il suo utilizzo. Inoltre, fornisce la possibilità di utilizzare un lanciatore di dadi, conformemente a quanto richiesto dal gioco.
In particolare, offre:

- Creazione dei personaggi;
- Accesso al manuale di gioco con ricerca;
- Lancia dadi;
- Modifica caratteristiche e abilità personaggi;
- Gestione dell'inventario del personaggio;
- Creazione automatica del background del personaggio con eventuale modifica.

L'applicazione si prefigge quindi di essere completa al fine di sostituire le principali componenti fisiche richieste per giocare.
L'applicazione contiene quindi termini familiari per utenti familiari con il gioco ma che di norma potrebbero non esserlo.

1.3 Utilizzo Flutter

Per lo sviluppo dell'applicazione, ho deciso di utilizzare il framework *Flutter* per vari motivi:

- Risulta essere un framework relativamente nuovo e quindi, per dargli supporto;
- Il linguaggio *Dart*, mi è sembrato ad un primo sguardo familiare, in quanto conoscevo già *Java* e *C++* che ne sono simili.

[1] https://company.wizards.com/
[2] https://flutter.dev/

2 Funzionamento di Flutter

2.1 Hot Reload

Grazie all'utilizzo di una macchina virtuale di Dart, Flutter consente la compilazione del codice just-in-time, ovvero, consente di rendere effettivi i cambiamenti durante l'esecuzione dell'applicazione per consentire un più veloce debug della stessa, senza doverla riavviare o perdere dei dati. Questo prende il nome di Hot Reload.
Questa funzionalità mi è risultata molto comoda durante lo sviluppo, anche se non è sempre funzionata nel modo corretto. Ad esempio, certe volte ho perso molto tempo a cercare quale fosse l'errore anche se lo avevo già corretto, perché era comunque necessario un riavvio dell'applicazione.

2.2 Flutter Engine

L'engine di Flutter, è scritto in C++, ed utilizza la libreria Skia[3] di Google, per il rendering delle parti grafiche. L'Engine, si interfaccia inoltre con gli SDK specifici delle varie piattaforme(Android e iOS), e contiene le librerie principali scritte in Dart, tra cui tutte le librerie grafiche e quelle per l'utilizzo dei dispositivi di I/O.

2.3 Libreria Foundation

Consente di interfacciarsi comodamente con l'Engine di Flutter, attraverso la definizione di API specifiche.

2.4 Tutto è Widget

In Flutter, tutto è considerato un Widget. Quindi, la creazione di una singola schermata, parte dal metodo `build()`, il quale a sua volta è uno widget. Per creare widget più complessi basta semplicemente comporre vari widget, dei quali ognuno a sua volta avrà un suo metodo `build()` per la rispettiva creazione.
In alternativa, è possibile agire direttamente tramite i metodi della libreria *Foundation* per modificare direttamente i singoli pixel, pratica consigliata solo a chi ha grande esperienza o specifiche necessità.
Spesso, l'app stessa, risulta essere semplicemente il più grande widget presente, il quale contiene a sua volta tutti gli altri widget, il quale viene comunemente chiamato MyApp.

[3] *https://skia.org/*

3 Mobile Design

3.1 Colori

Per garantire una migliore esperienza per l'utente(sia a livello di affaticamento visivo che di apprezzamento), nello sviluppo dell'applicazione è stato usato il Dark Theme(in particolare in quanto l'ambientazione del gioco è un po catastrofica e mi sembrava più inerente). Ho scelto di utilizzare il Material Dark Theme[4], come da figura 1.

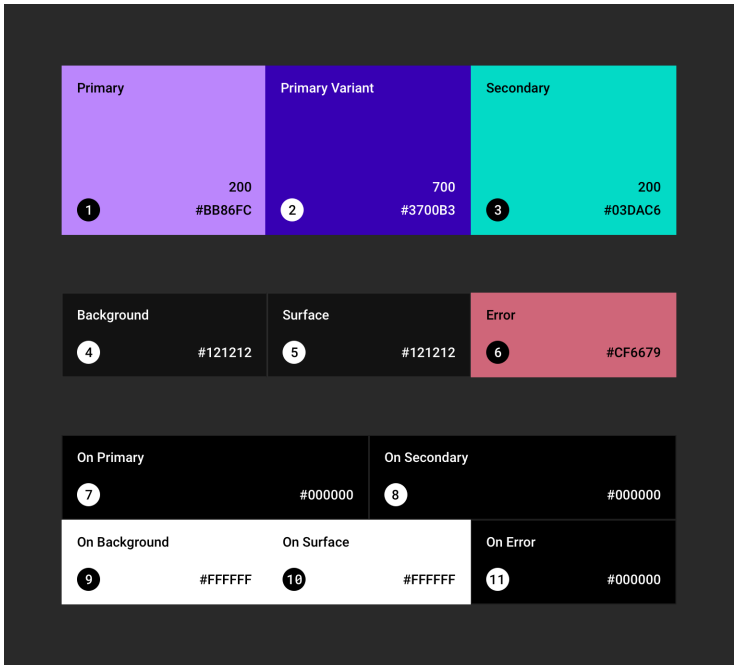


Figura 1: Material Dark Theme

Per le scritte è stato quindi utilizzato il bianco variato di opacità in base alla rilevanza della scritta, mentre, per scritte come titoli, campi di selezione e icone nell'app bar ho utilizzato il colore primario. Per i bottoni, ho generalmente utilizzato il primario variante, esclusi i bottoni si e no(per i quali ho usato colore Error per il rosso e verde) e per pulsanti con il fine di cancellare qualcosa, utilizzando così colori che richiamassero l'azione.
Per le scritte in bianco, ho utilizzato i livelli di opacità sempre suggeriti alla stessa pagina per testo primario, secondario e inattivo.

3.1.1 Utilizzo dei ThemeData

Flutter, consente di utilizzare una classe specifica per definire gli stili dell'applicazione, chiamata *ThemeData*[5] per avere degli stili unici in tutta l'applicazione.
Inizialmente, ho provato ad utilizzarla, ma poi mi sono trovato meglio a crearmi delle classi statiche a parte per colori e stili, in quanto le volevo utilizzare in contesti statici, ad esempio per la creazione di parti sempre uguali.

[4] *https://material.io/design/color/dark-theme.html#properties*
[5] *https://api.flutter.dev/flutter/material/ThemeData-class.html*

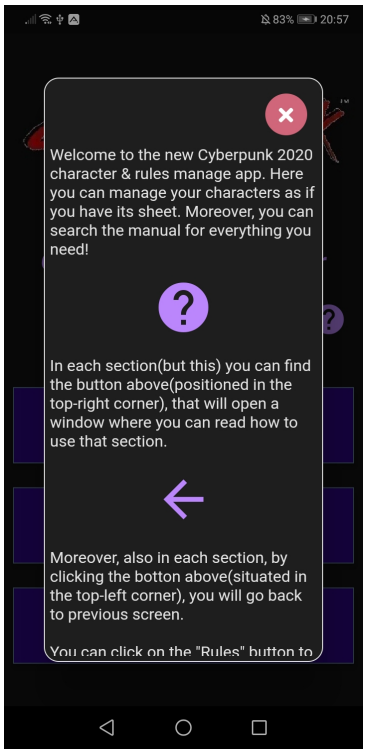
3.2 Safe Zone e Scalatura

Come si potrà vedere dalle immagini successive, generalmente tutti i pulsanti di interesse o comunque le aree cliccabili sono interne alla safe zone, eccetto per quanto descriverò nelle rispettive sezioni.
Le azioni che possono causare perdita di dati, ovvero la cancellazione di un personaggio(Fig.3a), la generazione casuale di un nuovo background(Fig.4, il quale pulsante cancella il background precedente se presente. Per background si intende la storia del personaggio) e la rimozione di un oggetto(Fig.4), sono accompagnate da un popup di conferma, per evitarne il click casuale.
Inoltre, l'applicazione è creata in modo da scalare su dispositivi di diverse dimensioni(la ho provata solo su simulatori di tablet android e non su dispositivi reali), mantenendo le zone di interesse all'interno della safe zone.

3.3 AppBar

In ogni sezione, esclusa la Homepage, è presente una AppBar in alto, la quale contiene il titolo della scheda, il tasto per tornare indietro(al di fuori della comfort zone) e il tasto per accedere al tutorial della pagina relativa.
Per questo scopo, Flutter inserisce direttamente all'interno della definizione del corpo della scheda un attributo per poter inserire una AppBar. Il suo scopo principale è di consentire all'utente di conoscere sempre in quale scheda si trovi al momento.

3.4 Tutorial

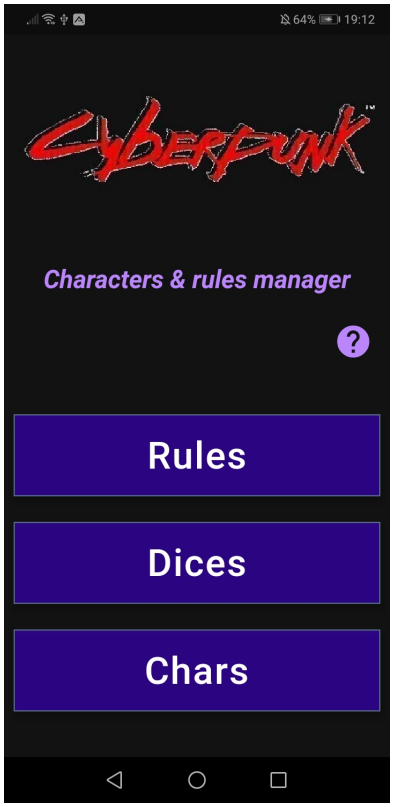


In ogni pagina, è presente un tasto con un punto di domanda:

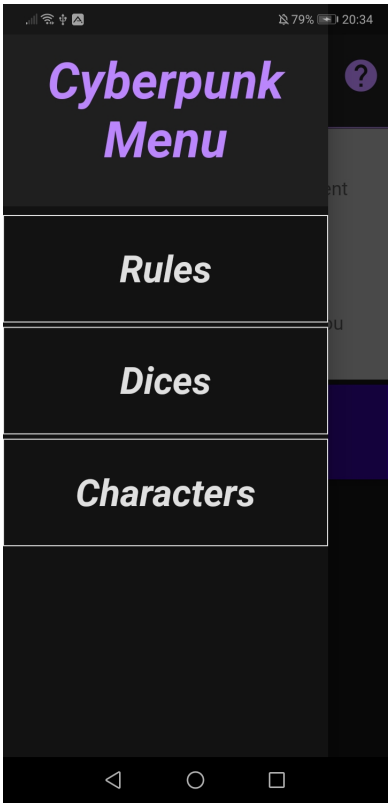
- Nella Homepage, è appena sotto il nome dell'applicazione;
- In ogni scheda, si trova nella App Bar presente in alto, nella parte destra.

Con questo tasto, è possibile accedere al tutorial specifico della pagina in cui ci si trova, nel quale vengono spiegati tutti i tasti e le azioni presenti nella pagina nel dettaglio. Al primo avvio dell'applicazione, appare in automatico il tutorial della Homepage che spiega il funzionamento generale dell'applicazione e come poter accedere ai vari tutorial.

3.5 Homepage



(a) Figura 2a: HomePage Android



(b) Figura 2b: HomePage Menu Laterale

Nella Home Page Fig.2a, sono presenti il menù per poter navigare nelle altre schede dell'applicazione e il tasto con il punto di domanda, che apre il tutorial di cui ho parlato nella sezione 3.4. Inoltre, per garantire una migliore navigabilità all'utente, ho dato la possibilità di accedere al menù semplicemente eseguendo una gesture di swipe da sinistra a destra(che viene spiegata nel tutorial iniziale), la quale fa apparire un menù laterale(Fig.2b). I tasti, sono di larghezza pari allo schermo e di altezza sufficientemente grandi per evitare click errati.
In Flutter, la navigazione avviene tramite il *Navigator*, che consente, attraverso una semplice riga di codice:

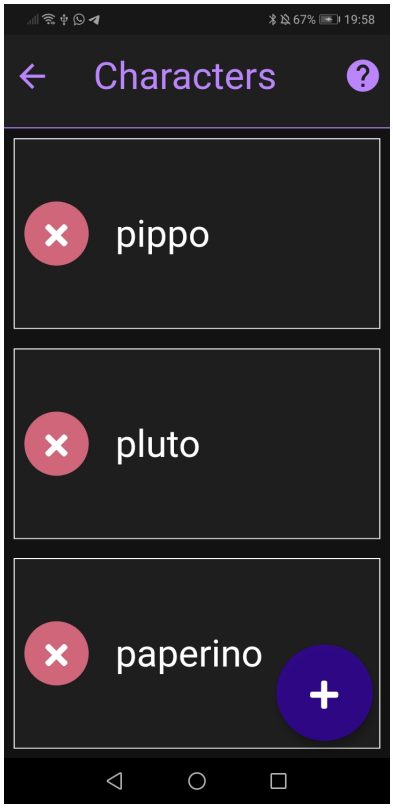
```
1 Navigator.push(context, MaterialPageRoute(builder: (context) => SecondRoute()));
```

In questo modo, si possono anche trasferire dati alla schermata successiva. Conseguentemente, per fare ritorno alla schermata precedente, basta la seguente riga, la quale consente anche di restituire un risultato(come ad esempio la selezione in un popup):

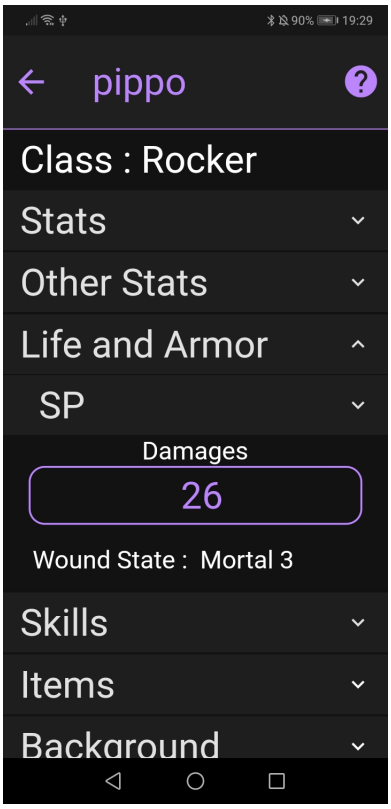
```
1 Navigator.of(context).pop(true);
```

3.6 Personaggi

3.6.1 Scelte



(a) Figura 3a: Selezione Personaggi



(b) Figura 3b: Schermo Gestione Personaggio

L'applicazione, fornisce la possibilità di aggiungere nuovi personaggi(immagine 3a), tramite l'utilizzo di un floating button sempre presente nella schermata di selezione degli stessi, sufficientemente grande, che si trova a cavallo tra la safe zone e la not safe zone per evitarne click indesiderati. Una volta creato un personaggio, è possibile accedere alla sua scheda e modificarlo. Per evitare troppi scroll, la ho organizzata in maniera gerarchica, inserendole in una lista espandibile all'occorrenza, dividendo nelle varie sezioni principali come nella scheda cartacea(vedere scheda personaggio. Immagine 7 in appendice).

Al fine di evitare continui cambiamenti con il manuale di gioco e quindi tap inutili, ho riproposto le descrizioni degli oggetti direttamente nella schermata attuale(Fig. 4). Inoltre, ho utilizzato pulsanti per la modifica che possano ricordare l'azione che si andrebbe ad eseguire, come il + per l'inserimento di un nuovo oggetto(che rimanda ad una schermata di selezione di oggetti di quel tipo ordinati in ordine alfabetico) e il - per la cancellazione.

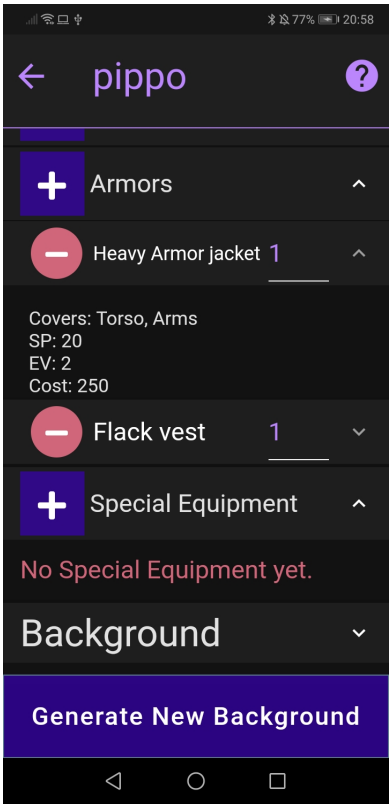


Figura 4: Equipaggiamento Personaggi

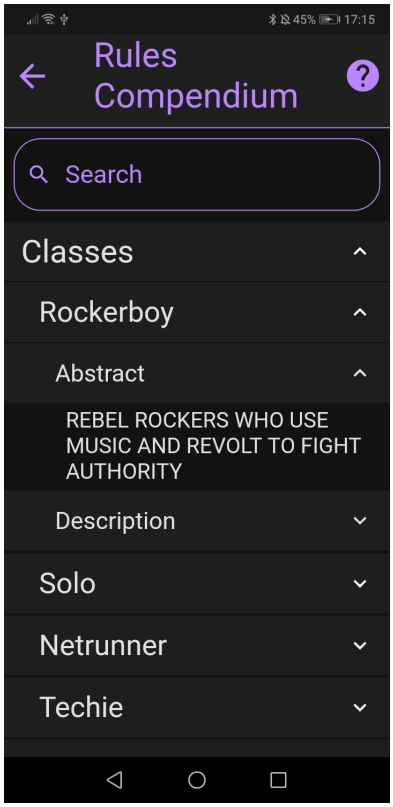
3.6.2 Salvataggio dei Dati

Flutter, garantisce 3 metodi principali per l'accesso a dati esterni:

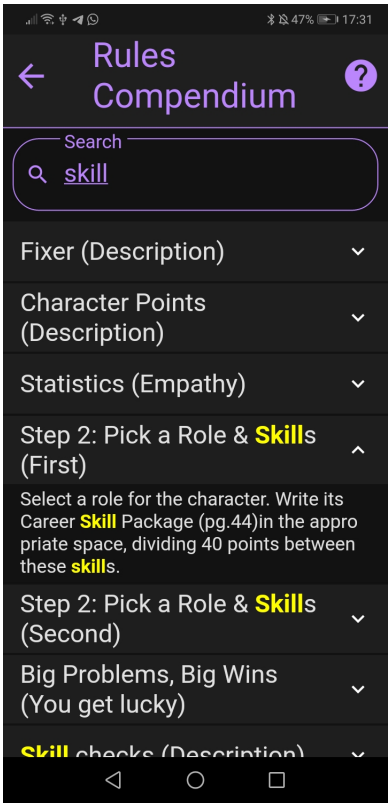
- **Salvataggio su file**: ho utilizzato la libreria `path_provider` per ottenere il percorso delle cartelle dell'applicazione all'interno delle quali salvare il manuale di gioco;
- **Salvataggio su database**: all'inizio avevo valutato la possibilità di salvare i dati dei personaggi su un database tramite `sqlite`, ma questa opzione si è rivelata meno efficiente del salvataggio su file;
- **Shared Preferences**: è possibile il salvataggio di semplici coppie di dati chiave-valore, al fine di salvare impostazioni dell'applicazione.

Sotto questo punto di vista, Flutter garantisce quindi una buona scelta, al fine di poter soddisfare ogni necessità in maniera relativamente semplice, funzionante sia su Android che iOS.
I personaggi, sono stati salvati su dei file *.json* sempre utilizzando la libreria `path_provider`.

3.7 Regolamento



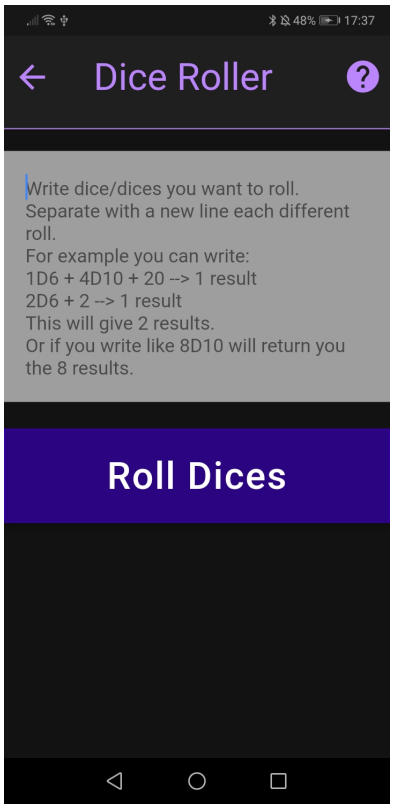
(a) Figura 5a: Regolamento



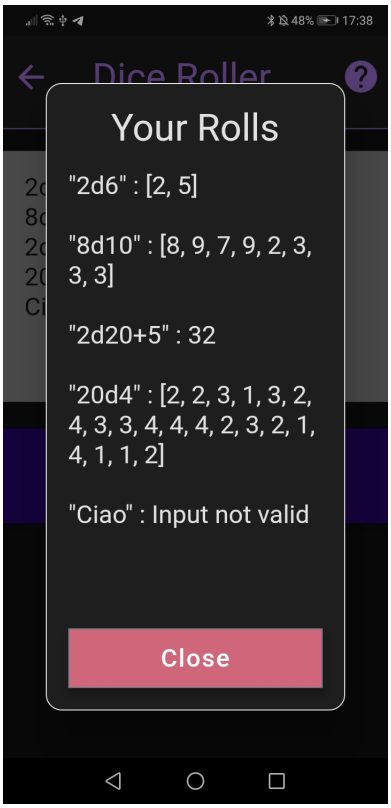
(b) Figura 5b: Ricerca

Per evitare troppi scroll all'utente, anche il regolamento di gioco è suddiviso in maniera gerarchica(Fig.5a). Inoltre, per una migliore comprensione, i sottocapitoli hanno una piccola tabulazione per una migliore suddivisione dello stesso. Arrivati invece ad una sezione senza ulteriori paragrafi, ho messo un background diverso in modo da differenziare e far capire meglio all'utente che non può più espandere.
Ho anche dotato il tutto di una ricerca per velocizzarne l'utilizzo, la quale risulta sempre in primo piano e va a sostituire l'intero manuale in caso di utilizzo con i soli paragrafi che contengono le parole cercate(in ordine di apparizione nel manuale), evidenziando anche la sezione di testo che le contiene(Fig. 5b).

3.8 Lancia Dadi



(a) Figura 6a: Lancia Dadi



(b) Figura 6b: Esempio di Lancio di Dadi

Il gioco si svolge principalmente, lanciando dadi quando si desidera che il proprio personaggio compia certe azioni e per vedere quanto bene gli riescono. Per questo, ho creato una sezione per consentire agli utenti di avere un lancia dadi personalizzato(Fig. 6a).
Il linguaggio utilizzato per scrivere quali dadi si desidera tirare, è quello comunemente usato anche nel parlato, risulta quindi essere molto familiare per l'utilizzatore. In fig. 6b si può vedere un esempio di lancio e cosa si ottiene nel caso si inserisca valori non validi.
Il funzionamento del lancia dadi è spiegato in maniera molto veloce come testo all'interno dell'area di scrittura dei lanci da voler effettuare come hint.
Alternativamente, nel tutorial della pagina risulta descritto in maniera più completa, con tutta la sintassi accettata.

4 Conclusioni

4.1 Pregi

4.1.1 Documentazione

Flutter ha una documentazione accedibile all'indirizzo *https://flutter.dev/docs*, la quale risulta essere ben costruita, grazie anche all'utilizzo di immagini che per meglio comprendere quello che stiamo leggendo e completa.

4.1.2 Widget

Il comparto widget, nel complesso risulta ben sviluppato, garantendone una vasta gamma utilizzabile. Presentano solo certi problemi come discusso in 4.2.3.

4.1.3 Sviluppo Framework

Durante lo sviluppo, ho ricevuto molti aggiornamenti di Flutter e del linguaggio sottostante, facendo così capire che si stanno impegnando molto in questo momento al suo miglioramento. In particolare, ho visto più cambiamenti della schermata di errore restituita quando se ne verificano, partendo da una generica schermata rossa con molte scritte senza significato, fino ad arrivare a qualcosa di sensato.

4.1.4 Community

Dietro a tutto, è presente una community molto attiva e pronta ad aiutare quando ce ne sia il bisogno. Dallo sviluppo di widget molto personalizzabili alla semplice risoluzione di errori che normalmente non sarebbero molto comprensibili.

4.1.5 Linguaggio Dart

Mi sono trovato molto bene a programmare in *Dart*[6], in quanto il linguaggio assomiglia molto a Java/C++ che conosco già abbastanza bene. Inoltre, il modo di creare gli widget, assegnargli proprietà come la centratura o simili risulta essere molto semplice e intuitivo(gerarchicamente come figli).

4.2 Difetti

4.2.1 Async/Await

Flutter, consente di caricare dati dall'esterno nel metodo build, attraverso l'utilizzo dei *FutureBuilder*, i quali si preoccupano di caricare il dato e costruire la schermata senza bloccare il tutto. Qualora sia necessario caricare dati da una sola fonte, questo metodo risulta effettivamente efficace. Mi sono trovato male in un'occasione, in quanto dovevo caricare dati da due fonti diverse e, non è stato possibile usare due *FutureBuilder* direttamente nel metodo build. Inoltre, non è possibile rendere tale metodo asincrono tramite la keyword *async* al fine di poter utilizzare l'*await* al suo interno per richiamare metodi asincroni e attenderne il risultato. Ho dovuto quindi trovare alter soluzioni meno eleganti.

[6] *https://dart.dev/*

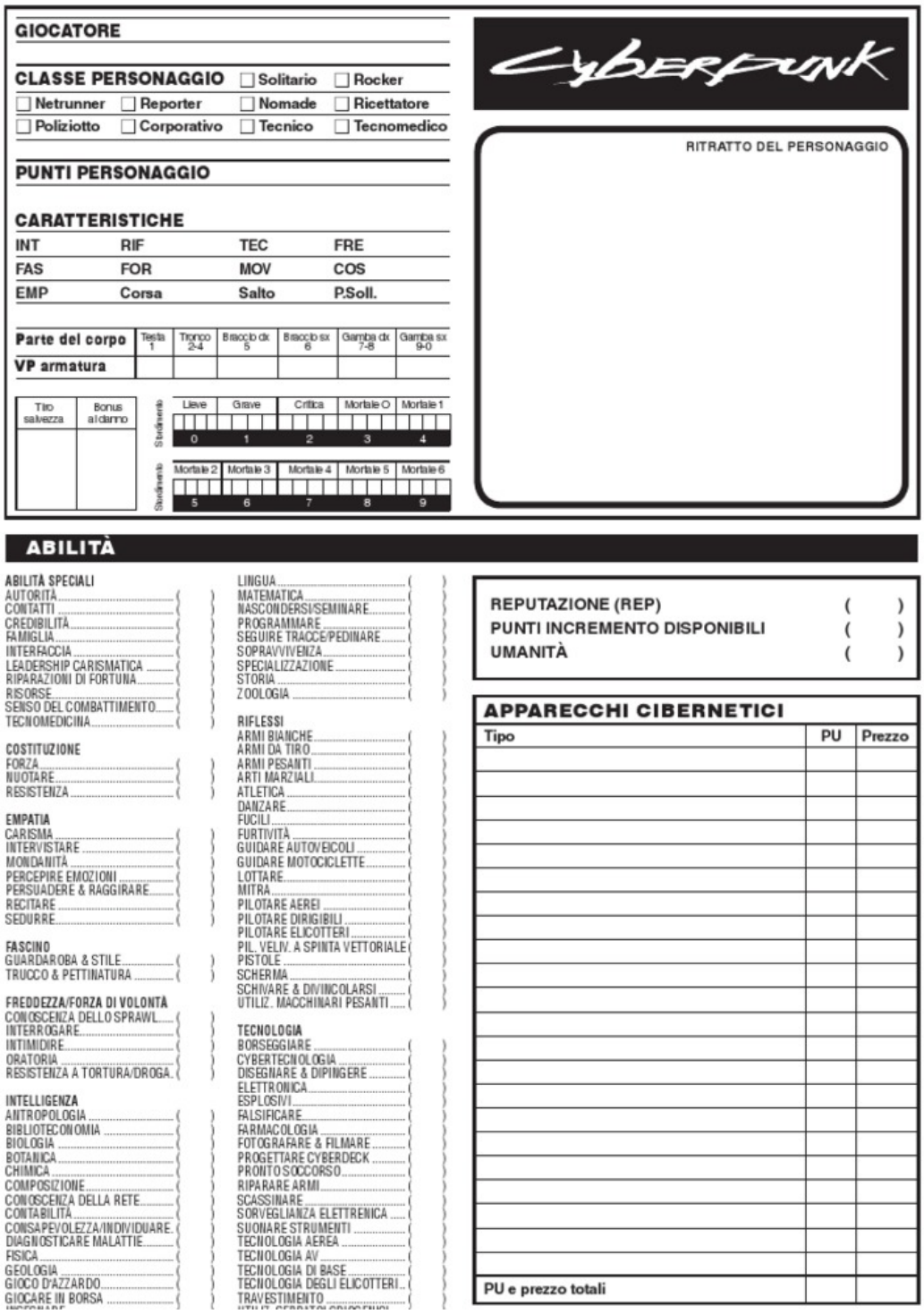
4.2.2 Errori

Ho avuto molte difficoltà in vari passaggi durante lo sviluppo, in quanto gli errori che Flutter restituisce molte volte non sono comprensibili o, ancora peggio non si riesce nemmeno a risalire all'errore dalla pila di errori nell'editor. Inoltre, qualche volta è capitato che cercando come risolvere un errore, nessuno fosse in grado di aiutare, vedendomi così costretto a risolvere il problema cercando altri metodi. Tutto questo probabilmente è dovuto al fatto che è un framework relativamente nuovo, non avendo così soluzioni a problemi noti come potrebbe essere per altri framework più consolidati.

4.2.3 Widget

Mi sono meravigliato di trovarmi qualche volta a dover utilizzare widget per i quali mancavano certe funzionalità, le quali ad esempio semplici attributi per cambiarne il colore o la grandezza.
In certi casi, ad esempio, mi è toccato copiare classi home made da alcuni utenti o copiare e incollare il codice della classe per poterlo modificare.

A Scheda del Personaggio Cartacea



GIOCATORE

CLASSE PERSONAGGIO ☐ Solitario ☐ Rocker
☐ Netrunner ☐ Reporter ☐ Nomade ☐ Ricettatore
☐ Poliziotto ☐ Corporativo ☐ Tecnico ☐ Tecnomedico

PUNTI PERSONAGGIO

CARATTERISTICHE

INT	RIF	TEC	FRE
FAS	FOR	MOV	COS
EMP	Corsa	Salto	P.Soll.

Parte del corpo	Testa 1	Tronco 2-4	Braccio dx 5	Braccio sx 6	Gamba dx 7-8	Gamba sx 9-0
VP armatura						

Tiro salvezza	Bonus al danno

Stordimento: Lieve 0 | Grave 1 | Critica 2 | Mortale 0 3 | Mortale 1 4
Stordimento: Mortale 2 5 | Mortale 3 6 | Mortale 4 7 | Mortale 5 8 | Mortale 6 9

CYBERPUNK

RITRATTO DEL PERSONAGGIO

ABILITÀ

ABILITÀ SPECIALI
AUTORITÀ ( )
CONTATTI ( )
CREDIBILITÀ ( )
FAMIGLIA ( )
INTERFACCIA ( )
LEADERSHIP CARISMATICA ( )
RIPARAZIONI DI FORTUNA ( )
RISORSE ( )
SENSO DEL COMBATTIMENTO ( )
TECNOMEDICINA ( )

COSTITUZIONE
FORZA ( )
NUOTARE ( )
RESISTENZA ( )

EMPATIA
CARISMA ( )
INTERVISTARE ( )
MONDANITÀ ( )
PERCEPIRE EMOZIONI ( )
PERSUADERE & RAGGIRARE ( )
RECITARE ( )
SEDURRE ( )

FASCINO
GUARDAROBA & STILE ( )
TRUCCO & PETTINATURA ( )

FREDDEZZA/FORZA DI VOLONTÀ
CONOSCENZA DELLO SPRAWL ( )
INTERROGARE ( )
INTIMIDIRE ( )
ORATORIA ( )
RESISTENZA A TORTURA/DROGA ( )

INTELLIGENZA
ANTROPOLOGIA ( )
BIBLIOTECONOMIA ( )
BIOLOGIA ( )
BOTANICA ( )
CHIMICA ( )
COMPOSIZIONE ( )
CONOSCENZA DELLA RETE ( )
CONTABILITÀ ( )
CONSAPEVOLEZZA/INDIVIDUARE ( )
DIAGNOSTICARE MALATTIE ( )
FISICA ( )
GEOLOGIA ( )
GIOCO D'AZZARDO ( )
GIOCARE IN BORSA ( )

LINGUA ( )
MATEMATICA ( )
NASCONDERSI/SEMINARE ( )
PROGRAMMARE ( )
SEGUIRE TRACCE/PEDINARE ( )
SOPRAVVIVENZA ( )
SPECIALIZZAZIONE ( )
STORIA ( )
ZOOLOGIA ( )

RIFLESSI
ARMI BIANCHE ( )
ARMI DA TIRO ( )
ARMI PESANTI ( )
ARTI MARZIALI ( )
ATLETICA ( )
DANZARE ( )
FUCILI ( )
FURTIVITÀ ( )
GUIDARE AUTOVEICOLI ( )
GUIDARE MOTOCICLETTE ( )
LOTTARE ( )
MITRA ( )
PILOTARE AEREI ( )
PILOTARE DIRIGIBILI ( )
PILOTARE ELICOTTERI ( )
PIL. VELIV. A SPINTA VETTORIALE ( )
PISTOLE ( )
SCHERMA ( )
SCHIVARE & DIVINCOLARSI ( )
UTILIZ. MACCHINARI PESANTI ( )

TECNOLOGIA
BORSEGGIARE ( )
CYBERTECNOLOGIA ( )
DISEGNARE & DIPINGERE ( )
ELETTRONICA ( )
ESPLOSIVI ( )
FALSIFICARE ( )
FARMACOLOGIA ( )
FOTOGRAFARE & FILMARE ( )
PROGETTARE CYBERDECK ( )
PRONTO SOCCORSO ( )
RIPARARE ARMI ( )
SCASSINARE ( )
SORVEGLIANZA ELETTRONICA ( )
SUONARE STRUMENTI ( )
TECNOLOGIA AEREA ( )
TECNOLOGIA AV ( )
TECNOLOGIA DI BASE ( )
TECNOLOGIA DEGLI ELICOTTERI ( )
TRAVESTIMENTO ( )

REPUTAZIONE (REP) ( )
PUNTI INCREMENTO DISPONIBILI ( )
UMANITÀ ( )

APPARECCHI CIBERNETICI

Tipo	PU	Prezzo
PU e prezzo totali		

Figura 7: Scheda Personaggio