



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

Una soluzione mobile based a supporto della terapia cognitivo comportamentale

Relatore: Chiari.ma Prof.ssa Daniela Micucci

Correlatore: Chiari.mo Dott. Davide Ginelli

Relazione della prova finale di:

Ionut Daniel Fagadau

Matricola 845279

Anno Accademico 2020-2021

*Alla mia famiglia per avermi sempre supportato
e spronato a dare il meglio di me.*

Alla Prof.ssa Micucci e al Dott. Ginelli per la fiducia riposta in me, per avermi dato la possibilità di svolgere questo progetto, per l'infinita pazienza e disponibilità e per l'incredibile passione dimostrata per il proprio lavoro.

Ad Andrei, Cosmin, Andrea e a tutti i miei amici di Roma che mi sono sempre rimasti vicini nonostante la distanza che ci separa.

A Giada, per avermi sopportato ogni giorno e per i suoi preziosi consigli.

Ad Elettra, Sofia, Alessio, Daniele e tutti gli altri compagni di corso che hanno vissuto insieme a me quest'esperienza e hanno contribuito a renderla speciale.

Ai miei coach, per avermi insegnato a non mollare mai.

ABSTRACT

Gli interventi a distanza sono spesso di grande aiuto per chi soffre di sintomatologie come depressione e ansia, difatti in molti paesi del mondo sono molto diffusi. In Italia tuttavia, almeno in ambito universitario, non esiste ancora una soluzione di questo tipo.

Questo elaborato si propone di descrivere il lavoro svolto durante lo stage trimestrale eseguito in remoto presso l'***Università degli Studi di Milano - Bicocca*** durante il quale è stata creata, assieme al **dipartimento di Psicologia**, una prima versione di una soluzione mobile sviluppata in un ambiente multi-piattaforma per eseguire interventi di terapia cognitivo comportamentale in maniera anonima utilizzando la piattaforma *Qualtrics*.

INDICE

Introduzione	1
1 MindBlooming	3
1.1 Interventi Internet Based	3
1.2 Dispositivi Mobile	3
1.3 mHealth	4
1.4 Strategie Principali	5
1.5 Funzionalità Offerte	5
2 Progettazione	6
2.1 Glossario	7
2.2 Struttura applicazione	8
2.3 Prima di Qualtrics	11
2.4 Dopo Qualtrics	12
2.4.1 Modello dati: domande	12
2.4.2 Modello dati: blocchi	13
2.4.3 Modello dati: risposte	13
2.5 Architettura adottata	15
2.5.1 MVC in Flutter	15
2.6 Interfaccia utente	17
2.6.1 Material Design	17
3 Implementazione	19
3.1 Endpoint utilizzati	19
3.1.1 Autenticazione	21
3.1.2 Sondaggi	22
3.1.3 Sondaggio: domande, blocchi e flow	23
3.1.4 Response	25

3.2	Gestione blocchi	26
3.2.1	Ordine blocchi	26
3.2.2	Domande appartenenti al blocco	27
3.2.3	Trash	27
3.3	Memorizzazione dati	28
3.3.1	Struttura dati	28
3.3.2	Organizzazione dati	28
3.4	Domande	29
3.4.1	Text Entry - "QuestionType": "TE"	29
3.4.2	Multiple Choice - "QuestionType": "MC"	31
3.4.3	NPS - "QuestionType": "MC"	35
3.4.4	Matrix Table - "QuestionType": "Matrix"	37
3.4.5	Side By Side - "QuestionType": "SBS"	45
3.4.6	Slider - "QuestionType": "Slider"	47
3.4.7	Rank Order - "QuestionType": "RO"	50
3.4.8	Pick, Group and Rank - "QuestionType": "PGR"	52
3.4.9	Description Box - "QuestionType": "DB"	53
3.5	Question Text	54
3.5.1	Media	54
3.5.2	YouTube support	55
3.5.3	Rich Text Editor	55
3.6	Validation	56
3.6.1	Struttura dati e traduzione	57
3.6.2	Risposta in app	59
3.7	Screening	60
3.7.1	Depressione	60
3.7.2	Ansia	61
3.7.3	Problemi di sonno	62
3.7.4	Dolore cronico	62
3.7.5	Burnout	63
3.7.6	Pensieri autodistruttivi	64
3.7.7	Difficoltà relazionali	64
3.7.8	Priorità	64
3.8	Esercizi	65
3.8.1	Suddivisione questionari	65
3.8.2	Identificazione domande	66
4	Manuale Utente	67
4.1	Prerequisiti	67
4.2	On-Boarding	68
4.3	Presentazione tutor	69

4.4	Screening	70
4.5	Risultati	71
4.6	Selezione patologie	72
4.7	Impostazioni iniziali	73
4.8	Homepage	74
4.9	Esercizi	75
4.10	Profilo	76
5	Conclusioni e sviluppi futuri	77
5.1	Idee future	77
5.1.1	Backend proprio	78
5.1.2	E-Coaches	78
5.1.3	Integrazione in altri ambienti	78
Appendice		80
A	Note per gli sviluppatori	81
A.1	Ambiente di sviluppo	81
A.2	Pacchetti utilizzati	82
Bibliografia		84

ELENCO DELLE FIGURE

1.1	Numero di utilizzatori di smartphone nel mondo [3]	4
2.1	Timeline Applicazione	9
2.2	Modello dati prima di Qualtrics	11
2.3	Modello dati Qualtrics semplificato	14
2.4	Esempio di widget tree	15
2.5	Archittettura MVC adottata in Flutter	16
2.6	Esempio di interazione tra diversi componenti Material	18
3.1	Text Entry	30
3.2	Multiple Choices	33
3.3	Domanda con media e text entry nelle opzioni	34
3.4	Domanda Net Promoter Score	36
3.5	Matrix Likert	39
3.6	Matrix Bipolar	40
3.7	Matrix MaxDiff	41
3.8	Matrix Text Entry	42
3.9	Matrix Rank Order	43
3.10	Matrix Constant Sum	44
3.11	Side By Side	46
3.12	Slider	49
3.13	Rank Order	51
3.14	Pick, Group and Rank	53
3.15	Vari media nel question text	54
3.16	Esempio di testo editato	55
3.17	Esempio di validazione errata	59
4.1	Schermata on-boarding	68

4.2	Schermata presentazione tutor	69
4.3	Schermata Screening	70
4.4	Schermata Risultati	71
4.5	Schermata Selezione patologie	72
4.6	Schermata impostazioni iniziali	73
4.7	Schermata Homepage	74
4.8	Schermata Esercizi	75
4.9	Schermata Profilo	76

INTRODUZIONE

Depressione e **ansia** sono i problemi più comuni sperimentati dagli studenti universitari; questo è dovuto al fatto che quello universitario è un periodo decisivo della vita di ogni persona che si ritrova a dover prendere molte decisioni importanti per il proprio futuro e a doversi adattare ad ambienti nuovi, spesso anche a città nuove. Tutti questi fattori possono portare a grandi quantità di **stress** che può determinare la nascita di problematiche maggiori.

È ormai risaputo che questo genere di disturbi è spesso più diffuso tra gli studenti universitari (*circa il 30% degli studenti universitari soffre di depressione*) [1] e spesso tutto questo sfocia in problematiche ancora più gravi (istinti suicidi, abuso di sostanze, autolesionismo, ...). È chiaro dunque che esiste un problema legato alla salute mentale degli studenti che va affrontato il prima possibile per evitare ripercussioni più gravi; molte istituzioni spesso mettono a disposizione del personale con cui trattare questi argomenti ma, purtroppo, non tutti sono disposti a parlare dal vivo dei propri problemi, che sia per vergogna o per mancanza di disponibilità. Non solo, in un periodo come questo appena vissuto, in cui una pandemia ha colpito il mondo intero, è ancor più difficile per chi soffre di questo tipo di problemi potersi mettere in contatto con chi di dovere.

Per questo motivo è utile e necessario avere una via alternativa, che possa essere percorsa anche da chi non ha disponibilità agli incontri dal vivo, all'identificazione e potenziale trattamento di questo genere di patologie. L'idea è utilizzare la tecnologia che usiamo già giorno per giorno per svolgere innumerevoli altri compiti. Lo scopo di questo progetto è dunque la realizzazione di un'applicazione a supporto della salute mentale che permetta, tramite degli **screening**, di identificare potenziali patologie accusate da uno studente e di affrontare un percorso di guarigione costituito da **video-lezioni** ed **esercizi**, composti da questionari in diversa forma,

Introduzione

mirati ad aumentare la consapevolezza del paziente sulle sue patologie e su come trattarle giorno per giorno. Le lezioni e gli esercizi saranno erogati settimanalmente, proprio come se fossero delle vere sedute dallo psicologo, in modo da permettere un trattamento graduale e il tutto sarà gestito in maniera anonima proprio per permettere la partecipazione anche a chi non è disposto a parlare apertamente. L'applicazione inoltre si occuperà di mantenere attivo il rapporto con l'utente tramite dei semplici questionari quotidiani che verranno notificati ad un orario a scelta arbitraria.

L'applicazione è stata sviluppata utilizzando il framework di Google **Flutter**, che utilizza il linguaggio di programmazione **Dart**. La scelta di utilizzare questa tecnologia è supportata dal fatto che Flutter permette di avere un'unica code base da cui poter compilare applicazioni sia per **iOS** che per **Android**, i due sistemi operativi leader nel mercato degli smartphone. Non solo, con la recente pubblicazione di Flutter 2, è possibile utilizzare la stessa code base per compilare anche applicazioni **web** stabili e applicazioni per Google **Fuchsia**, il nuovo sistema operativo di Google. Possiamo quindi affermare con sicurezza che la scelta di utilizzare Flutter mira, oltre che alla semplicità di sviluppo per diverse piattaforme, al supporto di tecnologie future.

Nel capitolo *MindBlooming* sono spiegati in dettaglio gli obiettivi dell'applicazione e le funzionalità che essa offre, mentre il capitolo *Progettazione* entra nel dettaglio di come sono effettivamente gestite queste funzionalità e con quali strumenti. Nel capitolo *Implementazione* invece sono descritti i componenti utilizzati per costruire l'applicazione, come sono stati costruiti e come sono stati risolti i problemi incontrati durante lo sviluppo. Infine, nel capitolo *Manuale Utente* è presente una guida all'utilizzo dell'applicazione dal punto di vista degli utenti.

CAPITOLO

1

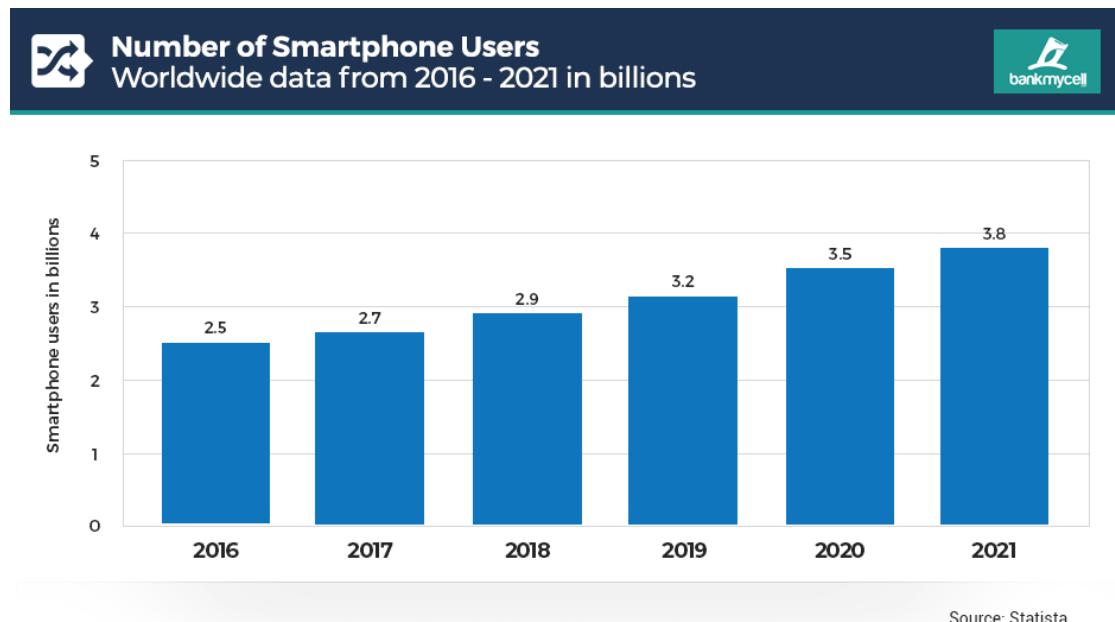
MINDBLOOMING

1.1 Interventi Internet Based

Diversi studi hanno dimostrato come degli interventi internet-based, ovvero virtuali, aiutassero i pazienti che soffrono di disturbi psichici[2]; in particolare alcuni studi riguardavano l'utilizzo di messaggi di testo sotto forma di promemoria, messaggi di supporto e/o procedure di auto-monitoraggio come terapia, con risultati positivi. In un mondo in cui la tecnologia è sempre più presente nelle nostre vite, ad esempio grazie agli **smartphone** la cui crescita in continuo aumento è osservabile nella Figura 1.1, questi studi rappresentano un'opportunità per permettere sempre a più persone di disporre di una terapia che altrimenti potrebbero rigettare (*per mantenere l'anonymato, per questioni di tempo o per qualsiasi altro motivo*).

1.2 Dispositivi Mobile

Secondo l'articolo in costante aggiornamento di **bankmycell**[3], sono circa 3.8 miliardi le persone che utilizzano uno smartphone al giorno d'oggi con oltre 10 miliardi di connessioni mobile attive; si stima che per il 2023 gli utilizzatori di smartphone saliranno a 7.33 miliardi. Sempre nello stesso articolo viene mostrato come **Apple** abbia la più grossa fetta del mercato mobile col suo **iOS**, seguita dalle case produttrici di smartphone che implementano **Android** asserendo quindi il dominio di questi due sistemi operativi nel mondo mobile.



Source: Statista

Figura 1.1: Numero di utilizzatori di smartphone nel mondo [3]

1.3 mHealth

Lo sviluppo di questi studi e la diffusione degli smartphone nel mondo ha portato a quella che è chiamata **mHealth** (*o mobile health*); la messa in pratica della medicina e della salute pubblica supportata dall'utilizzo di dispositivi mobile[4]. La mHealth è una parte di quella che viene chiamata **eHealth**, ovvero l'utilizzo di dispositivi informatici e tecnologici (*dai computer ai monitor dei pazienti*) per il supporto dei servizi salutari. Questo tipo di tecnologie permette oggi anche a paesi sottosviluppati di avere accesso ad una assistenza sanitaria di qualche tipo per permettere di identificare e curare in tempo eventuali disturbi.

Tra i vari servizi resi disponibili dalla mHealth ci sono anche quelli **educativi** del paziente sui disturbi di cui soffre, quelli di **helpline** che forniscono al paziente un contatto con esperti del settore e quelli di **remote disease surveillance** che permettono al paziente di diventare consapevole dei propri disturbi ed eventualmente sostenere un percorso di guarigione (*che sia online o meno*), che è esattamente quello a cui miriamo con la nostra applicazione.

1.4 Strategie Principali

Come abbiamo detto, il vantaggio principale delle pratiche come mHealth sono l'accesso in maniera anonima e la libera scelta di un orario, fattori sicuramente positivi per il target d'udienza dell'applicazione, ovvero gli studenti universitari. Per quanto riguarda l'identificazione e lo sviluppo di cura dei disturbi psichici, alcune strategie chiave sono il **monitoraggio dell'umore** del soggetto, la raccomandazione di **attività che scoraggino i pensieri negativi**, l'**educazione** del soggetto sui propri disturbi e l'**accesso alle reti di supporto** da parte di esso; tutte queste strategie fanno parte della **Terapia Cognitivo-Comportamentale**[5] che è un metodo di cura adatto al trattamento individuale ed è finalizzata a trattare i pensieri negativi, le emozioni disfunzionali e i comportamenti disadattivi del paziente in modo da facilitare l'eliminazione dei disturbi psicologici.

1.5 Funzionalità Offerte

Dunque, scopo finale dello stage è stato quello di realizzare un'applicazione di mHealth che permetta agli utenti, nel nostro caso agli studenti universitari, di auto-educarsi sui disturbi di cui potrebbero soffrire tramite un'attività iniziale di **screening** atta ad identificare segnali d'allarme che possano suggerire la presenza di tali disturbi, e l'eventuale **educazione e mitigazione** di essi tramite **attività di monitoraggio dell'umore**, **lezioni didattiche** sui propri disturbi e su come combatterli, **esercizi attivi** per il recupero. Questo sarà eseguito utilizzando dei questionari accompagnati da contenuti multimediali da completare settimanalmente, uniti a una parte di domande giornaliere per il monitoraggio dell'utente. I questionari sono creati sulla piattaforma **Qualtrics**, nata inizialmente per gestire dei questionari di tipo aziendale, che permette di definire diversi tipi di domande e di inserire vari media nelle domande stesse, offrendo l'accesso a tutto ciò tramite API. Verrà inoltre tenuta traccia dei progressi dell'utente e, utilizzando un calendario, verranno segnalati nuovi esercizi ogni volta che questi sono disponibili. Per permettere un monitoraggio del soggetto sono inoltre disponibili delle domande giornaliere che verranno notificate all'utente ad un orario a sua scelta.

CAPITOLO

2

PROGETTAZIONE

Prima di poter mettere mano all'applicazione è stato necessario un iniziale periodo di studio dell'ambiente, in particolare del linguaggio **Dart** e del framework **Flutter**; è stato piacevole scoprire come Flutter supporti nativamente sia il **Material Design**, stile adottato per lo sviluppo dell'interfaccia grafica (*almeno per quanto riguarda Android*), sia lo stile **Cupertino**, ovvero lo stile nativo di iOS, impostando di default alcune regole di tali stili quando vengono creati determinati widget.

Successivamente a questo periodo di studio è iniziata la vera e propria progettazione dell'applicazione, fase durante la quale sono stati stabiliti il modello dei dati e l'architettura adottata per lo sviluppo. Una nota importante riguardo i dati è che questi sono stati creati da un altro gruppo di persone specializzate in psicologia, che quindi ha ideato, durante lo sviluppo del progetto, le domande necessarie: era dunque necessario che il backend dell'applicazione, qualunque esso fosse stato, permettesse l'accesso sia all'applicazione stessa in lettura che al gruppo di psicologia in scrittura.

2.1 Glossario

Prima di procedere e parlare dell'effettiva organizzazione dei dati è necessario e utile definire un glossario di termini ricorrenti utilizzati durante lo sviluppo. Possiamo osservare tale glossario nella Tabella 2.1.

TERMINE	SINONIMI	SIGNIFICATO
Survey	<i>Sondaggio, Questionario</i>	La survey è l' <i>involtucco</i> più esterno che viene presentato all'utente; contiene uno o più <i>blocchi</i> , ognuno dei quali contiene delle domande. Le risposte che vengono inviate sono legate ad essa (<i>vengono inviate all'endpoint corrispondete al surveyID</i>).
Block	<i>Blocco, Sezione</i>	Un blocco è una parte di survey, che permette di mostrare le domande in un certo ordine piuttosto che in un altro; se ci sono più blocchi verranno visualizzate prima tutte le domande del primo blocco, e poi quelle del secondo e così via.
Question	<i>Domanda</i>	La parte atomica della survey, quella che viene effettivamente legata alla risposta (<i>tramite il suo ID</i>).
Response	<i>Risposta</i>	La risposta vera e propria relativa ad una certa domanda.
Choice	<i>Scelta</i>	Una possibile risposta ad una domanda; se scelta diventa una risposta.
Object	<i>Oggetto, Mappa, Map</i>	Sono gli oggetti JSON presenti nelle risposte alle API. Vengono chiamati anche mappe perché in flutter vengono tradotti in <code>Map<key: value></code> .
Validation	<i>Validazione</i>	Una serie di regole (<i>logiche</i>) che le risposte devono rispettare per essere valide. Il controllo avviene client-side, prima dell'invio.

Tabella 2.1: Glossario

2.2 Struttura applicazione

I dati ricevuti prima dell'inizio stesso del progetto riguardavano la struttura dell'applicazione in termini di sezioni e di come queste devono essere organizzate. In particolare, le patologie da identificare e su cui lavorare sono sette: Depressione, Ansia, Pensieri Autodistruttivi, Problemi di Sonno, Burnout, Dolore Cronico e Difficoltà Relazionali. Ognuna di queste problematiche prevede uno specifico **screening** con una specifica scala che determinerà se l'utente ne è affetto o meno e uno specifico **percorso terapeutico** della durata di sette settimane. Un'eccezione è fatta per il modulo relativo alle difficoltà relazionali che è **trasversale** rispetto agli altri: alla sesta settimana tutti gli utenti affronteranno una psico-educazione e degli esercizi relativi alle difficoltà relazionali. Le altre settimane sono caratterizzate da un'iniziale **psico-educazione** sulle patologie di cui soffre l'utente seguite da una serie di **esercizi** e da un **weekly screening** utile per monitorare l'andamento dell'utente. Durante tutto ciò, ogni giorno verrà somministrato all'utente un piccolo questionario giornaliero, notificato ad un orario scelto dall'utente. Infine, dopo il percorso e dopo 12 e 24 settimane verranno somministrati all'utente gli screening iniziali, in modo da poter avere un'idea del progresso fatto.

È possibile osservare la timeline delle diverse sezioni nella Figura 2.1.

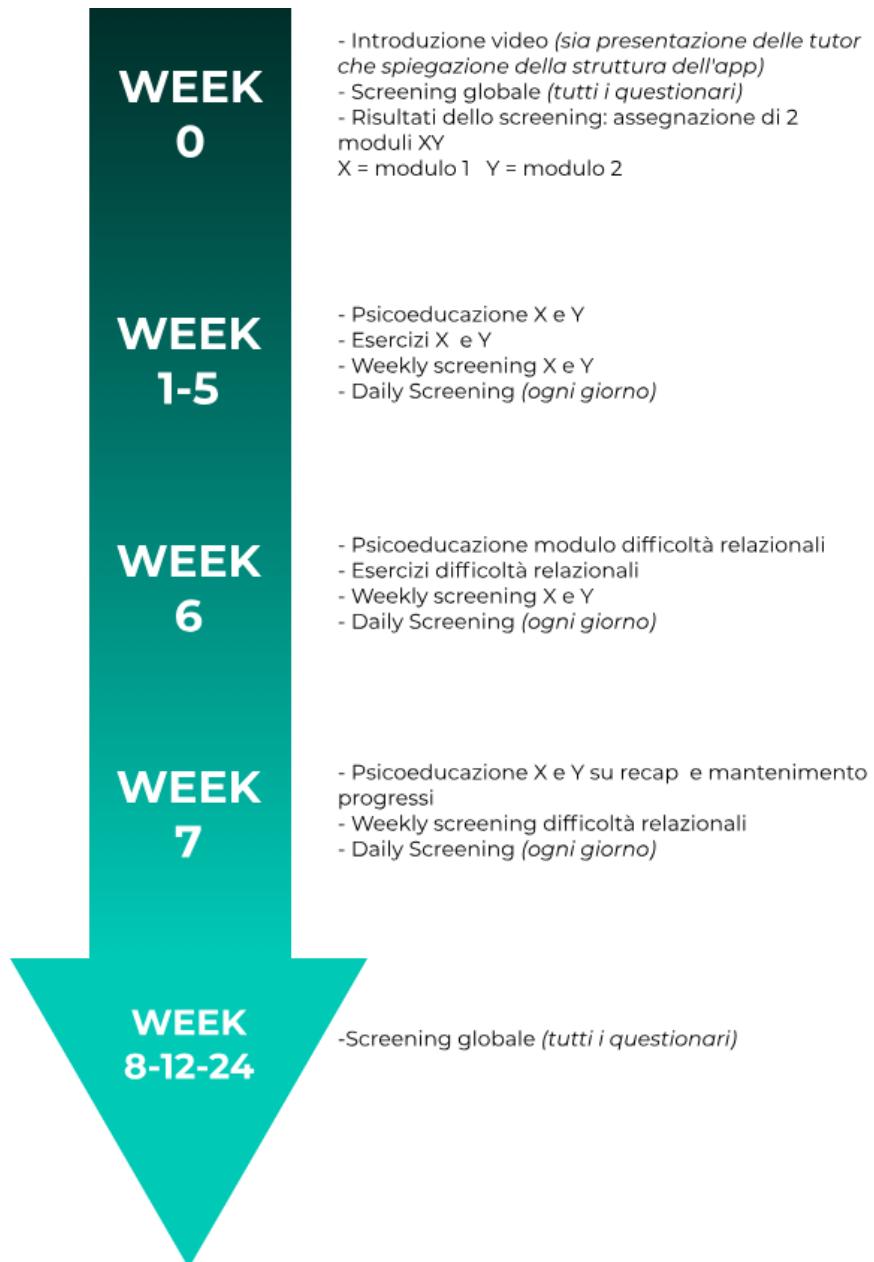


Figura 2.1: Timeline Applicazione

In particolare, le diverse sezioni sono:

- **Introduzione:** la prima cosa presentata all’utente è una introduzione video in cui le tutor di psicologia si presentano e spiegano all’utente cosa affronterà nelle successive settimane.
- **Screening:** subito dopo la presentazione iniziale inizia lo screening dell’utente, che consiste in un processo di identificazione e selezione delle patologie di cui (*presumibilmente*) l’utente soffre. Questo processo è realizzato tramite domande mirate che permettano di ottenere un punteggio associato ad ogni patologia secondo diversi algoritmi che dipendono dalla patologia stessa e dalla scala utilizzata per misurarla. Alla fine del processo di screening verranno scelte le due patologie con un punteggio abbastanza alto da essere considerate preoccupanti come patologie su cui lavorare se ce ne sono, altrimenti se solo una patologia ha un punteggio alto verrà scelta, lasciando libera scelta all’utente sulla seconda. Se nessuna patologia ha un punteggio abbastanza alto, l’utente può comunque scegliere due patologie su cui lavorare.
- **Esercizi:** una volta identificate le patologie da curare, l’utente affronterà settimanalmente degli esercizi interattivi accompagnati da una lezione di psico-educazione riguardanti le sue patologie; questa è dunque la fase di educazione e di mitigazione dei propri disturbi. Gli esercizi possono essere costituiti da immagini, video e/o audio corredati di qualche domanda sull’argomento in modo da mantenere l’utente concentrato.
- **Weekly & Daily Screenings:** durante il percorso terapeutico vengono somministrati ulteriori screening giornalieri e settimanali sotto forma di domande concise utili a mantenere l’utente concentrato e ad avere un monitoraggio più preciso sul suo progresso.

2.3 Prima di Qualtrics

Inizialmente non era stato scelto uno specifico backend, per cui lo sviluppo del modello dati è stato eseguito pensando ad un generale database (*come potrebbe essere Firebase*) e ad una generica applicazione che facesse uso di domande e risposte. Come possiamo osservare nella Figura 2.2, è stato pensato ad un modello estremamente semplice ma estendibile e che, per il momento, distingueva solo domande a risposta aperta, domande a risposta chiusa e domande "*speciali*" di screening (*che inizialmente pensavamo fossero solo a risposta chiusa*). Ad ogni domanda erano poi associate delle opzioni selezionabili (*o definibili nel caso di domande a risposta aperta*) e la risposta ad una certa domanda non era altro che l'opzione (*o le opzioni*) selezionata.

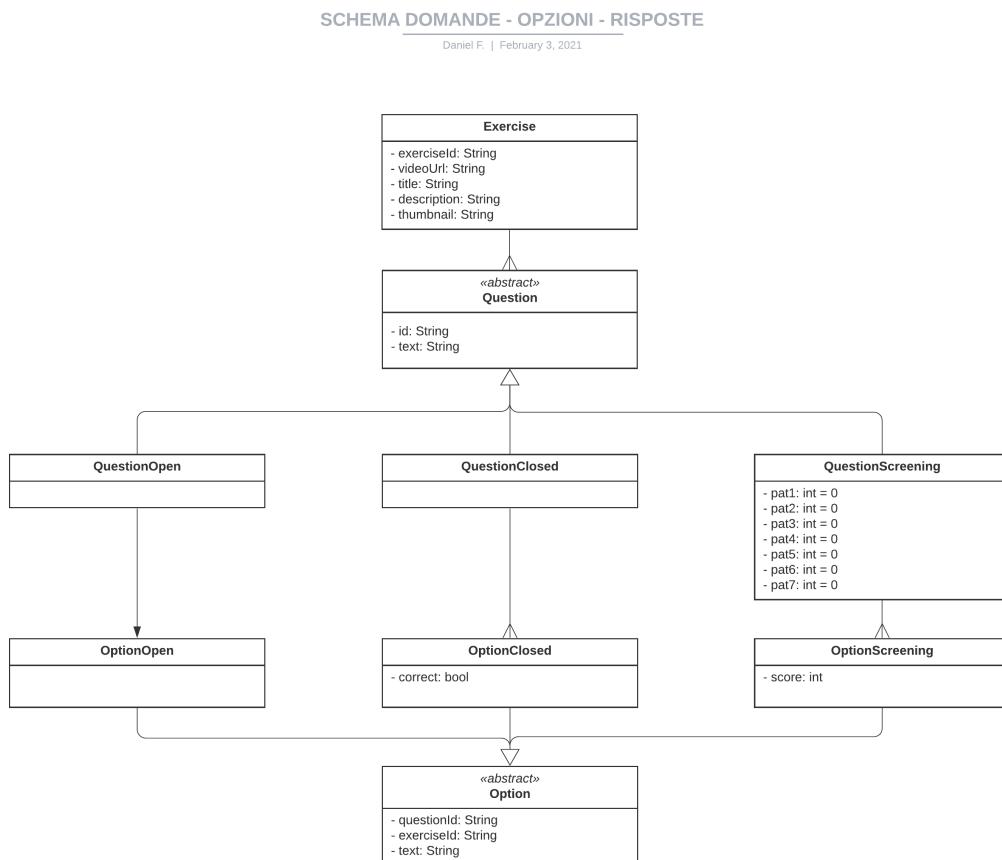


Figura 2.2: Modello dati prima di Qualtrics

2.4 Dopo Qualtrics

Non molto dopo i primi sviluppi è stato deciso di utilizzare **Qualtrics**¹ come effettivo backend dell'applicazione, che non è altro che una piattaforma per le indagini online che consente di creare sondaggi con vari gradi di complessità che vanno incontro a diverse esigenze di ricerca. Generalmente questa soluzione è orientata alle aziende per ottenere feedback sia dai propri clienti che dai propri impiegati: l'idea era adattare questo software già pronto e che presenta già diversi tipi di domande utili al nostro scopo per poter ottenere gli input necessari dai nostri utenti e presentarli a chi deve poi analizzarli. Non solo, essendo una piattaforma a sé stante, essa permette anche al gruppo di psicologia di creare i sondaggi necessari indipendentemente dallo sviluppo dell'applicazione, che si limita, tramite le API di Qualtrics, a ottenere e presentare questi sondaggi.

2.4.1 Modello dati: domande

In particolare, come possiamo vedere nel modello semplificato della Figura 2.3, il modello dati adottato da Qualtrics prevede un oggetto esterno, la **Survey** (*sondaggio*), che contiene, oltre a vari parametri di configurazione, una lista di **Questions** (*domande, che sono degli oggetti anch'esse*) che possono essere di vario tipo; le domande a loro volta contengono i loro parametri di configurazione ed una lista di possibili risposte (*Choices*) se a risposta chiusa. Tra i valori interessanti contenuti nei parametri delle domande ci sono il **QuestionText**, che rappresenta la domanda vera e propria in linguaggio naturale, il **Selector** che permette di avere domande simili ma differenziate per qualche dettaglio (*ad esempio possiamo avere una domanda a risposta chiusa in cui si possono selezionare più risposte piuttosto che solo una*) e il **ChoiceOrder**, che stabilisce in che ordine devono apparire le possibili risposte, contenute nell'oggetto **Choices**. Ogni entità è inoltre opportunamente identificata.

¹www.qualtrics.com/uk/core-xm/survey-software

2.4.2 Modello dati: blocchi

Un aspetto particolare riguardo l'organizzazione delle domande nell'ambiente Qualtrics è quello legato ai **Blocks**, che sono degli insiemi di domande presentati separatamente. Questo permette di non sovraccaricare una schermata di domande ma di suddividerle in parti visualizzate indipendentemente. Come si può vedere sempre nella Figura 2.3, i blocchi sono degli oggetti contenuti nell'oggetto **Survey** che dispongono di un **BlockID** e di una **Description**, una descrizione testuale utilizzata per dare un'idea del contesto delle domande presenti nel blocco. Inoltre, i blocchi contengono una lista di oggetti **BlockElements** che, seppur contenente oggetti, non contiene le domande vere e proprie, ma solo gli **QuestionID** delle domande facenti parte del rispettivo blocco; è necessario quindi un incrocio dei dati nell'applicazione per far sì che le domande rispettino l'ordine dei blocchi.

È presente inoltre, sempre nell'oggetto **Survey**, un ulteriore oggetto **SurveyFlow** che contiene l'ordine di visualizzazione dei blocchi in una lista contenente i **BlockID** corrispondenti (*in quanto potrebbero non essere nel giusto ordine all'interno dell'oggetto **Blocks***).

2.4.3 Modello dati: risposte

Sempre nella Figura 2.3 possiamo osservare anche come sono fatte le risposte nel backend; in particolare, essendo ogni entità opportunamente identificata direttamente da Qualtrics, le risposte non sono altro che degli oggetti contenenti delle coppie `{ID_Domanda: ID_Risposta}` oppure `{ID_Domanda: [ID_Risposta_1, ..., ID_Risposta_n]}`, nel caso di domande a risposta chiusa, mentre nel caso di domande a risposta aperta sono del tipo `{ID_Domanda: "Risposta"}`. Oltre a questo, possono contenere dei meta-dati che diano informazioni ad esempio sul tempo impiegato per rispondere o eventualmente sul browser utilizzato dagli utenti: di questi dati la maggior parte non verranno prodotti dall'applicazione, essendo opzionali (*ricordiamo che Qualtrics è stato pensato per essere implementato in siti web*).

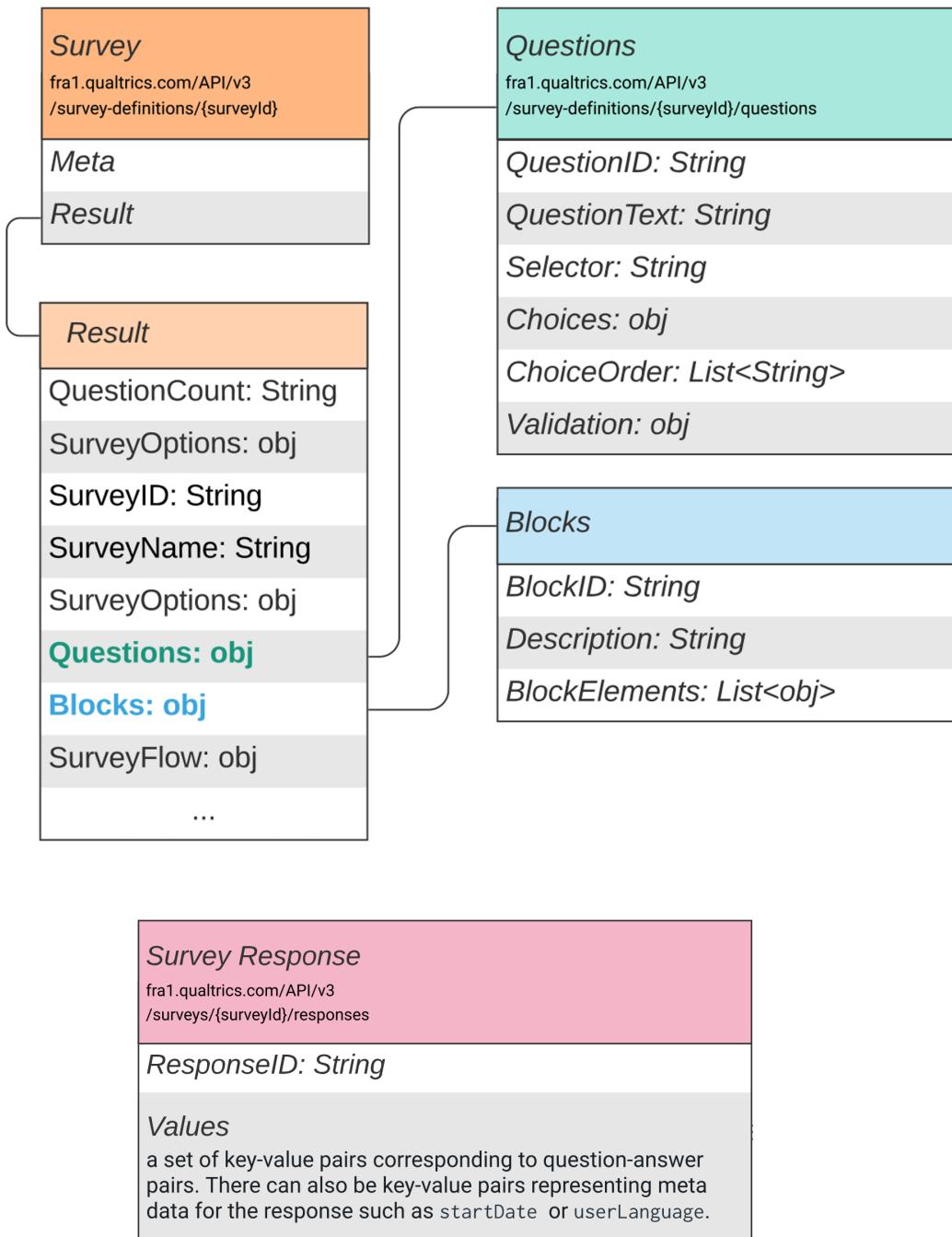


Figura 2.3: Modello dati Qualtrics semplificato

2.5 Architettura adottata

Per lo sviluppo vero e proprio dell'applicazione è stato adottato il pattern architettonico **Model-View-Controller (MVC)**[6], che è un'architettura che permette di slegare la **logica di presentazione** dalla **logica di business** tramite la separazione delle responsabilità tra tre principali componenti software:

- **Model**: questo componente fornisce i metodi necessari ad accedere ai dati interni dell'applicazione e aggiorna la *View*.
- **View**: i componenti di tipo *View* sono quelli che effettivamente l'utente vede a schermo e con cui interagisce. Questo tipo di componenti espone e presenta i dati all'utente.
- **Controller**: i *Controller* sono quei componenti che ricevono effettivamente i comandi dell'utente (*attraverso il View*) e li eseguono modificando lo stato del *Model*.

2.5.1 MVC in Flutter

Le componenti **View** dell'applicazione sono costituite dalla parte atomica di un'applicazione Flutter, i **widget**²: ogni elemento visibile a schermo è un qualche tipo di widget o una combinazione di widget. Flutter permette di avere dei widget all'interno di altri widget e di definire dei **custom widget**, costituiti da combinazioni di widget più basili, per definire delle UI complesse. Man mano che si definiscono diversi widget all'interno dell'applicazione, Flutter crea un **widget tree**, ovvero un albero che rappresenta la gerarchia tra widget mostrando i vari parent (*widget complessi di livello superiore*) e i children (*widget base definiti da Flutter*) e che verrà poi utilizzato per ottimizzare il rendering dell'UI. Possiamo osservare un esempio di widget tree nella Figura 2.4.

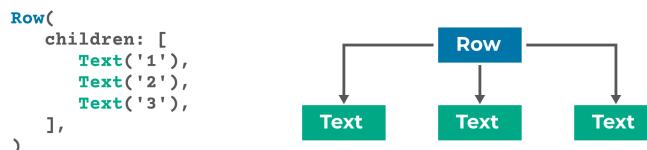


Figura 2.4: Esempio di widget tree

²<https://flutter.dev/docs/development/ui/widgets-intro>

Come è possibile osservare nella Figura 2.5, per realizzare le componenti **Model** e **Controller** in Flutter sono stati utilizzati i **Provider**: un Provider è una classe che viene estesa con la classe `ChangeNotifier`, che permette alla classe iniziale di ascoltare i cambiamenti nei dati e utilizzarli per aggiornare la **View** tramite il metodo `notifyListeners()` (*e che quindi implementa un design pattern di tipo Observer*). In particolare, i **Model** sono definiti dalle classi estese con `ChangeNotifier` che contengono effettivamente i dati e i metodi utilizzabili per modificarli, mentre i **Controller** sono definiti dall'accesso ai provider nelle classi **View** tramite la definizione `Provider.of<Classe>(context)`: questo perché quando viene definito un Provider ad un certo livello del widget tree, tutti gli elementi al di sotto di esso possono accedervi utilizzando il contesto fornito dal framework stesso, evitando così di dover importare ad ogni livello la classe **Model** per poter accedere ai suoi metodi.

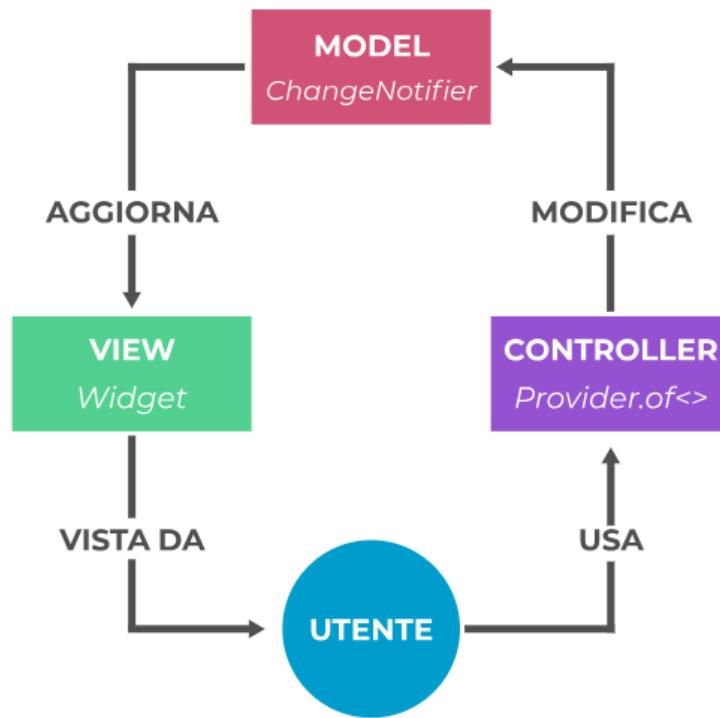


Figura 2.5: Architettura MVC adottata in Flutter

2.6 Interfaccia utente

Fin da subito è stato condotto uno studio anche su quale sarebbe stato lo stile grafico adottato per l'applicazione: visto il target e l'importanza di un'applicazione del genere è sicuramente molto importante che essa sia facile da utilizzare, intuitiva e gradevole. Vista la continua crescita e la predisposizione di Flutter verso il **Material Design**³, si è scelto questo come stile grafico da adottare in quanto fornisce delle linee guida per ottenere semplicità d'uso ed intuitività, lasciando però comunque spazio al designer per rendere il tutto anche visivamente piacevole e personale.

2.6.1 Material Design

Il *Material Design* si è evoluto molto nel tempo nel tentativo di Google di dare agli sviluppatori un sistema di design che aiutasse a creare delle esperienze di alta qualità che siano anche consistenti tra diverse piattaforme come Android, iOS e Web (*ed è anche per questo che Flutter adotta nativamente alcune linee guida di questo stile*) in un ambiente in cui Apple andava molto verso il realismo col suo *schemorismo* che però rischiava di distrarre troppo l'utente con i suoi troppi dettagli e dove Microsoft sosteneva il suo *Metro/Flat design* che invece creava confusione su cosa era effettivamente interattivo e cosa no. Google voleva quindi offrire i vantaggi di entrambe le soluzioni utilizzando sia riferimenti al realismo che la semplicità del minimalismo.

³<https://material.io/>

Difatti, il *Material Design* è molto ispirato al mondo fisico, soprattutto alle sue superfici, al modo in cui esse interagiscono con la luce (*riflettanza, ombreggiatura*) e a come interagiscono fra loro (*due oggetti fisici possono essere impilati, affiancati, ma non potranno mai compenetrarsi*) e cerca quindi di replicare questi comportamenti anche nel mondo digitale: il **materiale** è come un foglio di carta interattivo. Dunque, come è possibile notare nella Figura 2.6, i componenti *Material* interagiscono fra loro come se fossero in un ambiente 3D fatto di sovrapposizioni ed affiancamenti che generano le dovute ombreggiature e riflessi.

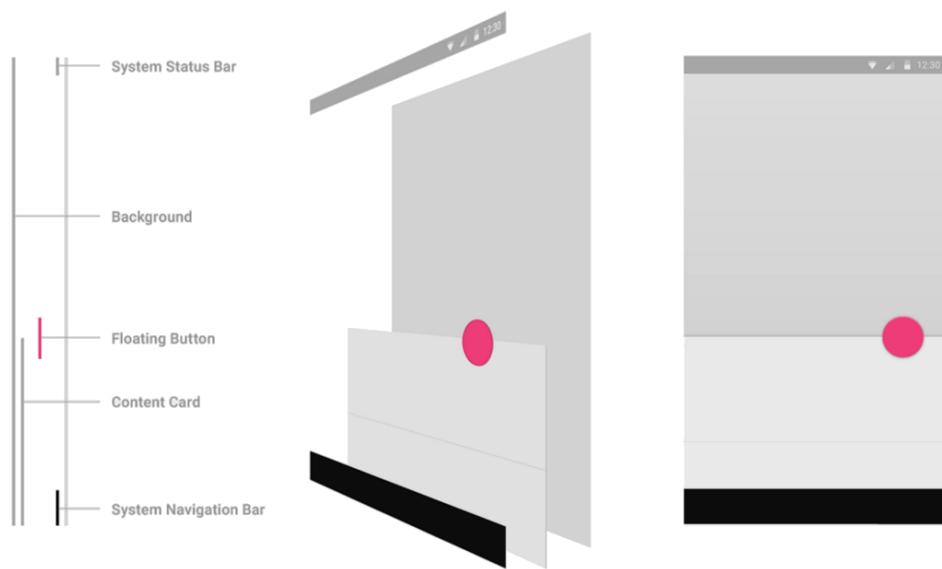


Figura 2.6: Esempio di interazione tra diversi componenti Material

Allo stesso tempo, questo linguaggio visivo implementa anche diversi metodi tradizionali della stampa e della tipografia atti a creare elementi gerarchici con il dovuto focus in modo da attirare l'attenzione dell'utente dove necessario in modo diretto e semplice evitando che quest'ultimo si senta spaesato, utilizzando quindi sempre elementi *flat* metaforizzati però nel mondo reale. È importante sottolineare che queste linee guida permettono di creare un design efficace che sia però comunque originale ed unico alle sensazioni che deve suscitare.

CAPITOLO

3

IMPLEMENTAZIONE

Dopo la fase di progettazione è stato possibile creare i primi componenti software ed interfacciarsi col backend di Qualtrics. In particolare è stata spesa buona parte del tempo per studiare i dati che fornisce Qualtrics tramite le sue API per poterli poi adattare ad un contesto mobile e alla creazione di custom widget che consumassero tali dati e li presentassero all'utente.

3.1 Endpoint utilizzati

La prima fase di studio si è concentrata nell'identificare quali chiamate sarebbero state necessarie per il funzionamento dell'applicazione; sono state necessarie varie chiamate per l'autenticazione al servizio di Qualtrics, per recuperare le domande e per inviare le risposte. Inoltre, alcune chiamate sono state utilizzate solo a scopo di test poiché Qualtrics restituisce già tutti i dati necessari per un determinato questionario in un'unica chiamata, seppur in una maniera poco leggibile; effettuare chiamate specifiche per reperire le domande e le informazioni di un determinato blocco permette di avere gli stessi dati in maniera più leggibile. Queste chiamate sono state necessarie (*seppur non implementate nell'applicazione*) poiché purtroppo la documentazione di Qualtrics non fornisce dettagli sulla struttura dei dati delle varie domande e delle risposte; tramite queste chiamate è stato possibile effettuare una sorta di reverse engineering di questi dati. Possiamo vedere un elenco riassuntivo delle chiamate interessanti nella Tabella 3.1.

TIPO	ENDPOINT	DESCRIZIONE
POST	/oauth2/token	Restituisce un token Oauth2 con determinati permessi che possa autenticare le chiamate future.
GET	/surveys	Restituisce tutti i questionari presenti nell'ambiente Qualtrics.
GET	/survey-definitions / {{surveyID}}	Restituisce i dati completi relativi al questionario {{surveyID}} come il numero di domande, i blocchi, meta-dati per la visualizzazione web, le domande e i blocchi (<i>seppur in maniera poco leggibile</i>).
GET	/survey-definitions /{{surveyID}}/questions	Restituisce tutte le domande del questionario {{surveyID}} (<i>senza distinzione fra blocchi</i>) sotto forma di lista di oggetti e senza ulteriori meta-dati legati alla survey, rendendo i dati molto più leggibili. Questa è una delle chiamate eseguite solo in fase di test ed è stata utilizzata unicamente per osservare e studiare il formato dei dati delle varie domande tramite Postman.
GET	/survey-definitions /{{surveyID}} /{{blockID}}	Restituisce i dati relativi al blocco {{blockID}} , tra i quali anche gli ID delle domande facenti parte di quel blocco. Anche questa chiamata è stata utilizzata unicamente a scopo di test.
POST	/survey-definitions /{{surveyID}}/responses	Invia le risposte al questionario {{surveyID}} al server.

Tabella 3.1: Elenco riassuntivo chiamate API

Nota: il punto di accesso alle API è `fra1.qualtrics.com/API/v3`, dove `fra1` è l'identificativo del data center europeo di Qualtrics.

3.1.1 Autenticazione

Prima di poter fare qualsiasi tipo di chiamata alle API di Qualtrics è necessario ottenere un token di autenticazione: in particolare Qualtrics permette di ottenere una chiave API direttamente dall'ambiente di creazione dei sondaggi (*meno sicura in quanto offre tutti i permessi a chi ne è in possesso*) oppure di richiedere, tramite la chiamata POST `fra1.qualtrics.com/oauth2/token`¹ un **bearer token**, molto più sicuro di una chiave in quanto è possibile definire per esso solo determinati permessi (*nel nostro caso serviranno solo i permessi per leggere i sondaggi e per scrivere le risposte*). Per richiedere tale token è necessario specificare nel campo **Authentication** della richiesta un **ClientID** ed un **ClientSecret**, entrambi generabili dall'ambiente Qualtrics ai quali verranno poi effettivamente assegnati i permessi necessari, mentre nel campo **Body** sarà necessario specificare il tipo di autenticazione e gli scope richiesti per il token tramite i parametri `grant_type=client_credentials&scope={scope}`. È interessante notare che è possibile richiedere solo un sottoinsieme di scope disponibili per la coppia `<ClientID, ClientSecret>`.

Possiamo vedere com'è fatta la risposta a questa chiamata nel JSON 3.1.

```
{
  "access_token": "xxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "token_type": "Bearer",
  "expires_in": 3599,
  "scope": "read:surveys"
}
```

JSON 3.1: Risposta a chiamata /oauth2/token

Una volta ottenuta tale risposta è possibile usare il bearer token presente nel campo `access_token` nel campo `Authentication` delle future richieste per autenticarle.

Nota: com'è possibile vedere anche nella risposta stessa, il bearer token ha una durata limitata nel tempo e scade in un'ora. Quando il token scade sarà necessario dunque richiederne uno nuovo.

¹api.qualtrics.com/guides/docs/Instructions/oauth-authentication.md

3.1.2 Sondaggi

A questo punto è possibile consumare effettivamente i dati forniti dalle API di Qualtrics e per far ciò un buon punto di partenza è recuperare i dati relativi ai sondaggi disponibili nello spazio di lavoro di Qualtrics tramite la chiamata `GET fra1.qualtrics.com/API/v3/surveys` che restituisce una lista di oggetti, ognuno dei quali rappresenta un sondaggio. Possiamo vedere un esempio semplificato di risposta a tale chiamata nel JSON 3.2 (*in cui è riportato solo un sondaggio senza ulteriori meta-dati per semplicità*), in particolare notiamo che i parametri che ci interessano maggiormente sono il campo `id`, tramite il quale potremmo ottenere ulteriori informazioni sul sondaggio (*come le sue domande*) e il campo `name`.

```
{  
  "result": {  
    "elements": [  
      {  
        "id": "SV_xxxxxxxxxxxxxxx",  
        "name": "nome_sondaggio",  
        "ownerId": "UR_xxxxxxxxxxxxxxx",  
        "lastModified": "2021-06-29T16:34:06Z",  
        "creationDate": "2021-04-08T19:40:33Z",  
        "isActive": true  
      },  
    ],  
  }  
}
```

JSON 3.2: Risposta a chiamata /surveys

3.1.3 Sondaggio: domande, blocchi e flow

Usando gli id restituiti dalla prima chiamata possiamo a questo punto ottenere ulteriori informazioni su ogni sondaggio tramite la chiamata

`GET fra1.qualtrics.com/API/v3/survey-definitions/{{surveyID}}`. In particolare, come è possibile osservare nel JSON 3.3 questa chiamata ci fornirà informazioni più dettagliate sul sondaggio come il numero di domande, i blocchi presenti nel sondaggio e il SurveyFlow, ovvero l'ordine dei blocchi.

Domande

Si può notare che questa chiamata ci fornisce già anche le domande; nell'applicazione, per questione di ottimizzazione di dati e chiamate, viene effettivamente fatta solo questa chiamata per impostare un determinato sondaggio. La particolarità però è che questa chiamata fornisce le domande come un oggetto con all'interno altri oggetti (*le domande stesse*), il tutto corredata con i dati specifici al questionario stesso: essendo le domande anch'esse degli oggetti relativamente complessi (*non sono mostrate in questo esempio perché ne parleremo dopo distinguendole per tipo*), può risultare difficile per una persona leggere questi dati, anche se un programma non ha questo problema.

Come detto prima, per risolvere questo inconveniente è stata usata la chiamata `GET /survey-definitions/{{surveyID}}/questions` che restituisce solo una lista con gli oggetti rappresentanti le domande, molto più facile da leggere per capire quali parametri servono per implementare la domanda stessa. Discorso analogo vale anche per i blocchi che sono già forniti in questa chiamata e che possono essere ottenuti in un formato più leggibile tramite la chiamata `GET /survey-definitions/{{surveyID}}/{{blockID}}`.

Anche in questo caso il JSON di esempio riportato è semplificato; sono stati omessi alcuni campi che non vengono utilizzati in ogni caso dall'applicazione.

```
{
  "result": {
    "QuestionCount": "246",
    "SurveyID": "SV_xxxxxxxxxxxxxxxx",
    "Questions": { "*oggetti domande*" }
    "Blocks": {
      "BL_xxxxxxxxxxxxxxxx": {
        "Type": "Default",
        "Description": "nome_blocco",
        "ID": "BL_xxxxxxxxxxxxxxxx",
        "BlockElements": [
          {
            "Type": "Question",
            "QuestionID": "QID235"
          }
        ],
      },
      "BL_xxxxxxxxxxxxxxxx": {
        "Type": "Trash",
        "Description": "Trash / Unused Questions",
        "ID": "BL_xxxxxxxxxxxxxxxx",
        "BlockElements": [],
      },
    "SurveyFlow": {
      "Flow": [
        {
          "ID": "BL_xxxxxxxxxxxxxx1",
        },
        {
          "ID": "BL_xxxxxxxxxxxxxx2",
        },
      ],
    },
  }
}
```

JSON 3.3: Risposta a chiamata /surveys

3.1.4 Response

La chiamata POST `/survey-definitions/{{surveyID}}/responses` è, escludendo la chiamata necessaria ad ottenere il bearer token, l'unica chiamata col metodo POST che l'applicazione andrà ad utilizzare per inviare al server di Qualtrics le risposte di un determinato questionario. Anche in questo caso la documentazione ufficiale di Qualtrics non era molto chiara a proposito di come dovevano effettivamente essere strutturati i dati da inviare, ma è stato tuttavia possibile creare delle risposte di testing e, utilizzando una richiesta GET, recuperare la struttura delle risposte per i diversi tipi di domanda. Possiamo osservare un esempio di risposta nel JSON 3.4. Non entriamo nei dettagli delle risposte vere e proprie adesso poiché dipendono dal tipo di domanda e ne parleremo dunque dopo. Notiamo però che ci sono molti meta-dati, tra i quali anche quelli legati alla localizzazione (*ottenuta tramite IP*) che però, al fine di mantenere l'anonimato dei nostri utenti, noi non forniremo considerando che sono opzionali. Quello che verrà effettivamente inviato al server sono delle coppie "QuestionID": "Response" inserite nel campo `values` (*similmente a quanto detto nella sottosezione 2.4.3 del capitolo precedente*) senza alcun meta-dato.

```
{
  "result": {
    "responseId": "R_xxxxxxxxxxxxxxxx",
    "values": {
      "startDate": "2021-07-02T17:59:35Z",
      "endDate": "2021-07-02T17:59:35Z",
      "status": 2,
      "progress": 100,
      "duration": 0,
      "finished": 1,
      "recordedDate": "2021-07-02T17:59:35.863Z",
      "locationLatitude": "45.3197937012",
      "locationLongitude": "8.42129516602",
      "distributionChannel": "test",
    }
  }
}
```

JSON 3.4: Esempio di oggetto response creato automaticamente da Qualtrics

3.2 Gestione blocchi

Abbiamo detto che i blocchi sono degli sotto-insiemi di domande da mostrare separatamente: per fare ciò tuttavia sono necessarie alcune operazioni preliminari atte a ordinare i blocchi e successivamente ad associare le domande che essi contengono ai relativi blocchi.

3.2.1 Ordine blocchi

Possiamo notare nel JSON 3.5 come l'oggetto **SurveyFlow** contenga, in ordine di presentazione, gli ID dei blocchi da mostrare. Per rispettare tale ordine è necessario un incrocio di dati: leggendo i dati del campo **SurveyFlow** possiamo accedere all'oggetto **Blocks**, di cui possiamo osservare la struttura nel JSON 3.6, nel giusto ordine, utilizzando il **BlockID** come chiave di accesso.

```
"SurveyFlow": {  
    "Flow": [  
        {  
            "ID": "BL_xxxxxxxxxxxxxx1",  
        },  
        {  
            "ID": "BL_xxxxxxxxxxxxxx2",  
        },  
    ],  
}
```

JSON 3.5: Oggetto SurveyFlow

3.2.2 Domande appartenenti al blocco

Allo stesso modo possiamo notare nel JSON 3.6 come i blocchi presentino solo l'ID delle domande che contengono ed è quindi nuovamente necessario, come detto anche nella sottosezione 2.4.2 del capitolo precedente, incrociare i dati per assegnare poi a livello di applicazione le domande ai relativi blocchi: leggendo il campo **BlockElements** si accede agli ID delle domande interessate e, utilizzandoli come chiave per accedere all'oggetto **Questions**, si possono ottenere solo le domande di un determinato blocco.

```
"Blocks": {
    "BL_xxxxxxxxxxxxxxxx": {
        "Type": "Default",
        "Description": "nome_blocco",
        "ID": "BL_xxxxxxxxxxxxxxxx",
        "BlockElements": [
            {
                "Type": "Question",
                "QuestionID": "QID235"
            }
        ],
    },
    "BL_xxxxxxxxxxxxxxxx": {
        "Type": "Trash",
        "Description": "Trash / Unused Questions",
        "ID": "BL_xxxxxxxxxxxxxxxx",
        "BlockElements": []
    }
}
```

JSON 3.6: Oggetto Blocks

3.2.3 Trash

Particolare attenzione va prestata ad uno specifico blocco, ovvero quello con il parametro **Type**: "Trash": questo particolare blocco è il cestino delle domande cancellate dal sondaggio e, se non svuotato, presenterà degli ID di domande che però non devono essere visualizzate. Il modo più semplice per gestire questo blocco è saltarlo direttamente controllando il campo **Type**.

3.3 Memorizzazione dati

In questa sezione saranno discusse le modalità in cui l'applicazione salva e ordina i dati ottenuti dalle API, in modo da poter poi costruire i widget necessari.

3.3.1 Struttura dati

Le strutture dati scelte per questa applicazione sono le mappe (*o Hash Tables, Hash Maps*), questo perché abbiamo osservato che ogni entità fornita da Qualtrics è provvista di un ID e, in particolare, generalmente gli ID non seguono un ordine preciso (*fanno eccezione i QuestionID che sono numerati in ordine di creazione, ma creando ed eliminando domande possono presentare comunque salti*). Essendo poi gli ID univoci conviene sicuramente utilizzare una mappa per salvare i dati, utilizzando gli ID come chiave di ogni entità: in questo modo è possibile accedere velocemente ad una qualsiasi entità salvata utilizzando il suo ID, questo perché l'accesso ai valori di una mappa, conoscendo la chiave, richiede un tempo **O(1)**, ovvero costante[7]. Inoltre, spesso i dati assumono strutture diverse (*ad esempio i diversi tipi di domanda*) e poiché Dart permette, tramite la keyword **dynamic**, di avere oggetti di tipo diverso come valore di una mappa è sicuramente più semplice utilizzare delle mappe piuttosto che degli array per salvare questi dati.

3.3.2 Organizzazione dati

Per quanto riguarda l'organizzazione dei dati, per semplicità e per come sarebbero poi stati visualizzati i dati, si è scelto di creare una mappa esterna le cui chiavi sono i SurveyID e i valori delle ulteriori mappe che rappresentano i blocchi di ogni sondaggio.

I blocchi, anch'essi delle mappe con chiave il BlockID, contengono a loro volta delle terze mappe, le più interne, che rappresentano le domande e che quindi hanno come chiave il QuestionID e come valore i vari parametri propri a ciascuna domanda. È stata scelta questa organizzazione dei dati anche perché le domande possono cambiare radicalmente in termini di parametri, quindi definire un tipo specifico per tutte le domande non sarebbe stato possibile. In questo modo inoltre ogni componente accede solo ad un livello dei dati: la pagina che contiene i vari sondaggi accede al primo livello, la pagina principale del sondaggio al secondo e la pagina effettiva delle domande al terzo.

3.4 Domande

In questa sezione saranno analizzati i vari tipi di domande che Qualtrics mette a disposizione, come sono fatte, come sono codificate dalle API e come vengono tradotte nell'applicazione mobile.

3.4.1 Text Entry - "QuestionType": "TE"

La più semplice domanda utilizzabile nel contesto di Qualtrics è la Text Entry: una domanda che mette a disposizione un campo di testo dove scrivere una risposta aperta. Possiamo osservare la struttura dati semplificata di questo tipo di domande nel JSON 3.7. Notiamo in particolare, oltre al `QuestionText` e al `QuestionType` che definisce il tipo di domanda, che c'è anche un oggetto `Validation`: questo oggetto è presente in ogni domanda e serve per definire delle regole secondo le quali una risposta è giusta o meno, ne parleremo in modo più dettagliato successivamente. Qualtrics permette anche di definire se la risposta dovrebbe essere corta, media o lunga, modificando la dimensione del campo di testo: questa distinzione viene fatta tramite il campo `Selector` che vale "`SL`" (*Single Line*) se la risposta è corta, "`ML`" (*Multi Line*) se è media e `ESTB` (*Essay Text Box*) se è lunga. Nell'applicazione viene letto questo campo e viene impostata una dimensione rispettivamente di una, 5 o 10 righe del campo di input. È importante notare tuttavia che questo limite non è fortemente restrittivo: seppur la dimensione del campo di input sia limitata, l'utente può comunque inserire una risposta più lunga, si tratta per lo più di una feature di quality of life. Possiamo osservare com'è renderizzata questa domanda rispettivamente nella webview di Qualtrics e nell'applicazione Flutter nella Figura 3.1.

Response

La risposta a questo tipo di domande è del tipo
`{"QuestionID": "testo risposta"}`, ovvero viene inserita come risposta la stringa stessa inserita dall'utente nel campo di input.

```
{  
    "QuestionText": "testo_domanda",  
    "QuestionID": "QID247",  
    "QuestionType": "TE",  
    "Selector": "SL",  
    "Validation": []  
}
```

JSON 3.7: Oggetto domanda Text Entry

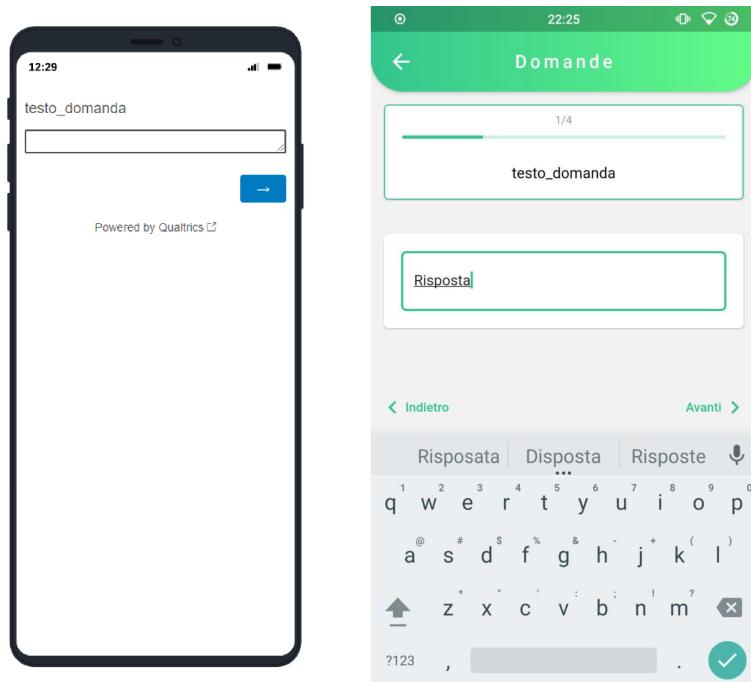


Figura 3.1: Text Entry

3.4.2 Multiple Choice - "QuestionType": "MC"

Un'altra domanda semplice e che ci si aspetta di trovare in qualsiasi applicazione che faccia uso di questionari è la Multiple Choice: la classica domanda a risposta chiusa che mette a disposizione dell'utente delle risposte predefinite da scegliere. Possiamo osservare la struttura semplificata di questo tipo di domanda nel JSON 3.8. Anche in questo caso Qualtrics mette a disposizione un **Selector** che però, a differenza di prima, cambia maggiormente la domanda. Questo selettore può essere **SAVR**, **SAHR** o **SACOL** per le domande a risposta singola (*SA = Single Answer*) che, sul web, possono essere visualizzate verticalmente, orizzontalmente o in colonna. Tuttavia nell'applicazione, per questioni di spazio disponibile, le risposte saranno visualizzate sempre verticalmente. Per le domande a risposta singola il selettore può assumere anche il valore **DL** (*Dropdown List*) che permette di scegliere la risposta tramite un menù a tendina.

Allo stesso modo il selettore può assumere anche i valori **MAVR**, **MAHR** e **MACOL** (*MA = Multiple Answers*) che permette alla domanda di avere più di una risposta scelta, influenzando così anche la risposta generata. Di conseguenza, come possiamo vedere nella Figura 3.2, in base al **Selector** l'applicazione costruirà diversi widget: uno per le risposte singole, uno col menù a tendina o uno per le risposte multiple.

Possiamo anche notare che tra i vari dati della domanda ci sono le **Choices**, ovvero le possibili risposte col proprio testo opportunamente identificate (*notiamo anche che gli ID sono in ordine di creazione, non di presentazione*). Osserviamo anche il campo **ChoiceOrder** che contiene gli ID delle choices in ordine di presentazione (*che può essere diverso dall'ordine nel quale sono presenti nel campo Choices*): sarà necessario anche in questo caso incrociare i dati come fatto precedentemente con i blocchi e le domande.

Media nelle opzioni

Come abbiamo visto prima, le opzioni di una domanda a risposta chiusa sono di fatto un oggetto: questo perché Qualtrics permette di inserire nelle scelte delle immagini o delle caselle di testo (*per esempio per implementare un'opzione "Altro"*) aggiungendo all'oggetto relativo alla Choice interessata rispettivamente i parametri "**Image**": `{}` (*che conterrà l'url dell'immagine*) e "**TextEntry**": `"true"`. Possiamo osservare un esempio di questo tipo di risposte nella Figura 3.3.

Response

La risposta a queste domande è diversa nel caso di risposta singola e nel caso di risposte multiple: se la domanda accetta solo una risposta è del tipo `{"QuestionID": ChoiceID}`, ovvero è l'ID della risposta scelta reperibile dal campo `Choices` (*notare l'assenza di apici*), mentre se la domanda accetta risposte multiple è del tipo `{"QuestionID": ["ChoiceID_1", "ChoiceID_2"]}`, ovvero una lista con gli ID delle risposte scelte. Se l'opzione scelta contiene una text entry inoltre, nella risposta sarà presente anche il campo `"QuestionID_TEXT": "testo risposta"`.

```
{
    "QuestionText": "testo_domanda",
    "QuestionID": "QID247",
    "QuestionType": "MC",
    "Selector": "SAVR",
    "Choices": {
        "4": {
            "Display": "Risposta 1"
        },
        "5": {
            "Display": "Risposta 2"
        },
        "6": {
            "Display": "Risposta 3"
        }
    },
    "ChoiceOrder": [
        "4",
        "5",
        "6"
    ],
    "Validation": {}
}
```

JSON 3.8: Oggetto domanda Multiple Choice

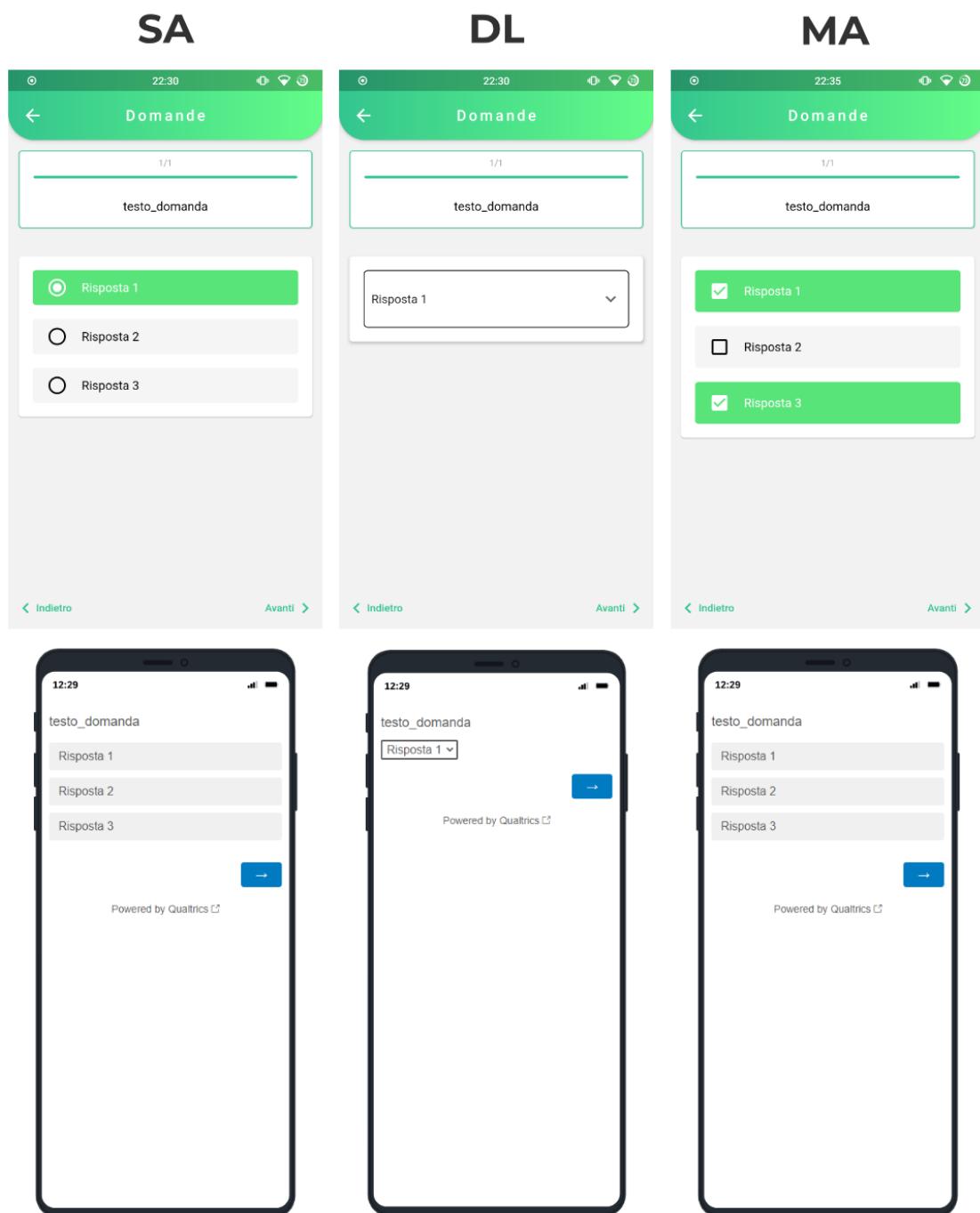


Figura 3.2: Multiple Choices

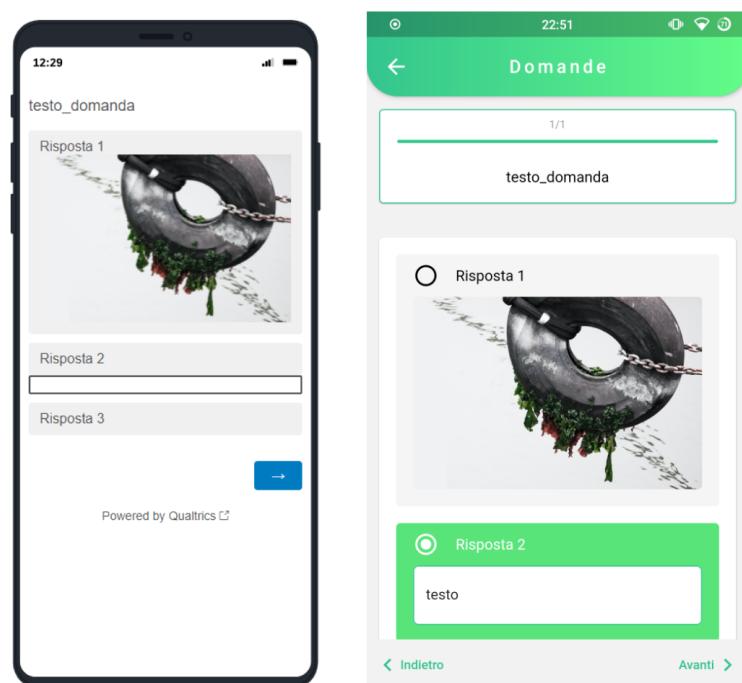


Figura 3.3: Domanda con media e text entry nelle opzioni

3.4.3 NPS - "QuestionType": "MC"

Un tipo particolare di domanda Multiple Choice che viene però trattato da Qualtrics come un tipo separato è il **Net Promoter Score**, una domanda che richiede di assegnare un punteggio da 0 a 10 a un qualcosa. Possiamo osservarne la struttura nel JSON 3.9 e notiamo che quello che la distingue dalle altre domande di tipo MC è il **Selector** di tipo NPS, e che contiene delle label nel campo **ColumnLabels** utili a dare un'idea del giudizio che si sta dando. Possiamo osservare un esempio di questa domanda nella Figura 3.4.

Response

La risposta a questo tipo di domanda è del tipo {"QuestionID": numero}.

```
{
  "QuestionText": "testo_domanda",
  "QuestionType": "MC",
  "Selector": "NPS",
  "Choices": [
    {
      "Display": "0"
    },
    ...
    {
      "Display": "10"
    }
  ],
  "ChoiceOrder": [],
  "ColumnLabels": [
    {
      "Display": "Not at all likely",
    },
    {
      "Display": "Extremely likely",
    }
  ],
}
```

JSON 3.9: Oggetto domanda NPS

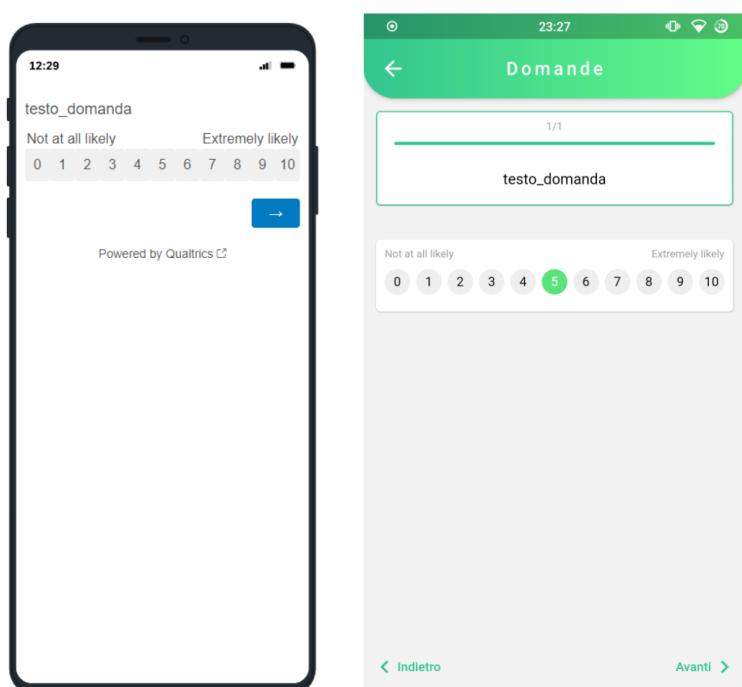


Figura 3.4: Domanda Net Promoter Score

3.4.4 Matrix Table - "QuestionType": "Matrix"

Uno dei tipi più complessi di domanda messi a disposizione da Qualtrics è il tipo Matrix: una serie di diverse domande che però hanno le stesse risposte (*o lo stesso tipo di risposta*). Questo tipo di domande sono utili per quando sono necessarie più domande che dovrebbero essere valutate su una stessa scala. Possiamo osservare la struttura di queste domande nel JSON 3.10. Notiamo che, almeno in parte, somiglia molto ad una domanda di tipo Multiple Choices: anche la matrice presenta delle **Choices** ordinate secondo un **ChoiceOrder**, ma presenta anche un secondo gruppo di elementi, le **Answers**, ordinate anch'esse secondo un **AnswerOrder**. Possiamo pensare alle **Choices** come alle risposte vere e proprie, uguali per tutte le domande, e alle **Answers** come alle domande da valutare (*la scelta dei nomi da parte di Qualtrics non è delle migliori in questo caso*). Un altro modo per visualizzare il tutto è pensare ad una tabella dove le **Choices** identificano le colonne e le **Answers** le righe (*che è anche il metodo in cui Qualtrics mostra, nella variante web, queste domande*). Le matrici inoltre, come le NPS, supportano le label.

Nel JSON possiamo anche notare, oltre al "classico" **Selector** già visto in precedenza, anche un **SubSelector** che svolge la funzione del **Selector** di prima, ovvero permette di scegliere il tipo di risposta che la domanda accetta. Mentre il **Selector** delle matrici serve a cambiare radicalmente il sotto-tipo di domanda: Qualtrics mette a disposizione diversi tipi di matrice, oltre che diversi tipi di risposte per esse.

Media

Come per le Multiple Choice, anche in queste domande è possibile inserire delle immagini e delle text entry nelle **Choices**, mentre per le **Answers** è possibile solo inserire delle immagini.

Response

In questo caso le response sono identificate, oltre che dal **QuestionID**, anche dal **ChoiceID**, la chiave difatti è **QuestionID_ChoiceID**. La risposta vera e propria tuttavia cambia, oltre che in base al tipo di risposta che accetta la matrice, anche in base al tipo stesso di matrice.

```
{  
    "QuestionText": "testo_domanda",  
    "QuestionType": "Matrix",  
    "Selector": "Likert",  
    "SubSelector": "SingleAnswer",  
    "Choices": {  
        "1": {  
            "Display": "Riga 1"  
        },  
        "2": {  
            "Display": "Riga 2"  
        },  
        "3": {  
            "Display": "Riga 3"  
        }  
    },  
    "ChoiceOrder": [ "1", "2", "3" ],  
    "Answers": {  
        "1": {  
            "Display": "Colonna 1"  
        },  
        "2": {  
            "Display": "Colonna 2"  
        },  
        "3": {  
            "Display": "Colonna 3"  
        }  
    },  
    "AnswerOrder": [  
        "1",  
        "2",  
        "3"  
    ],  
}
```

JSON 3.10: Oggetto domanda Matrix

Matrix: "Selector": "Likert"

Il primo tipo di matrice messo a disposizione da Qualtrics è la matrice likert, una matrice che contiene di fatto delle domande simili alle multiple choice. Tramite il Subselector è possibile stabilire anche se le risposte debbano essere singole (**SingleAnswer**), multiple (**MultipleAnswer**) o singole con menù a tendina (**DL**). Possiamo osservare un esempio di questo tipo di domande nella Figura 3.5; in particolare è possibile notare come è stato scelto di mantenere un design coerente con le multiple choice, in modo da richiamare quel tipo di domande anche dal punto di vista del design, oltre che per le funzionalità. Questo tipo di matrici supporta anche un formato **Profile** che permette di avere delle risposte diverse per ogni sotto-domanda (*ma devono tuttavia essere dello stesso tipo*).

Response: Le risposte sono del tipo `{"QuestionID_ChoiceID": "AnswerID"}` nel caso di risposta singola e del tipo `{"QuestionID_ChoiceID": ["AnswerID_1", "AnswerID_2"]}` nel caso di risposta multipla. Nel caso un'opzione prevedesse una text entry, la risposta a tale text entry è del tipo `{"QuestionID_ChoiceID_TEXT": "testo"}`.

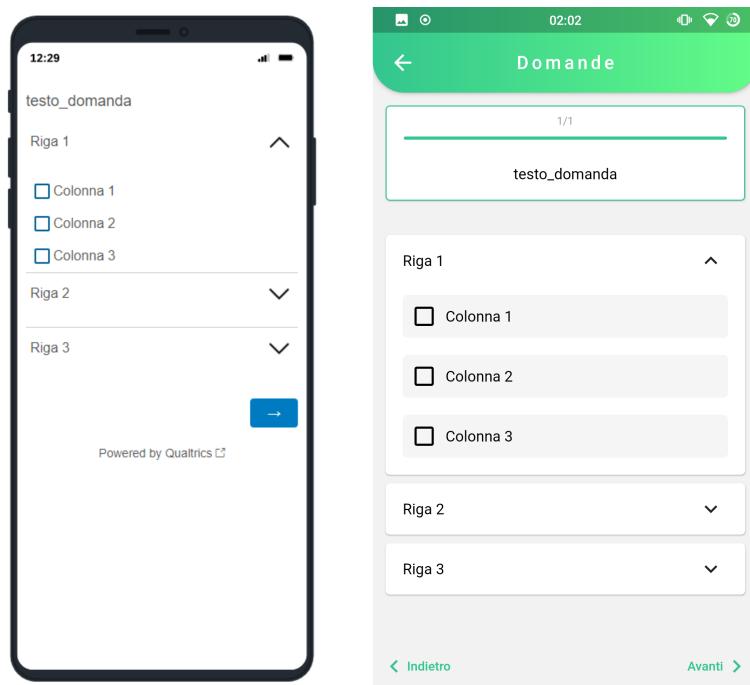


Figura 3.5: Matrix Likert

Matrix: "Selector": "Bipolar"

Le matrici di tipo Bipolar permettono di selezionare una risposta che si avvicini di più ad una di due possibili scelte. Permettono dunque solo risposte singole. Il testo delle `Choice` tuttavia è "separato" in due parti utilizzando il carattere ':'. Se viene utilizzato tale carattere nel testo stesso delle scelte tuttavia questo viene preceduto da una coppia di caratteri di escape da Qualtrics. È stato dunque necessario identificare e trattare nel modo desiderato tale escaping nell'applicazione, operazione eseguita tramite delle espressioni regolari che, nel caso trovassero una espressione '\:' rimuovessero i caratteri di escape e stampassero il ':', mentre nel caso trovassero un ':' senza escaping dividessero la stringa in due, in modo da avere parte destra e sinistra del testo della scelta. Possiamo osservare un esempio di questa domanda nella Figura 3.6 dove possiamo anche osservare come la webview mobile di Qualtrics non sia ottimale e non supporti nemmeno le label.

Response: Le risposte, essendo una matrice che accetta solo risposte singole, sono del tipo `{"QuestionID_ChoiceID": AnswerID}`.

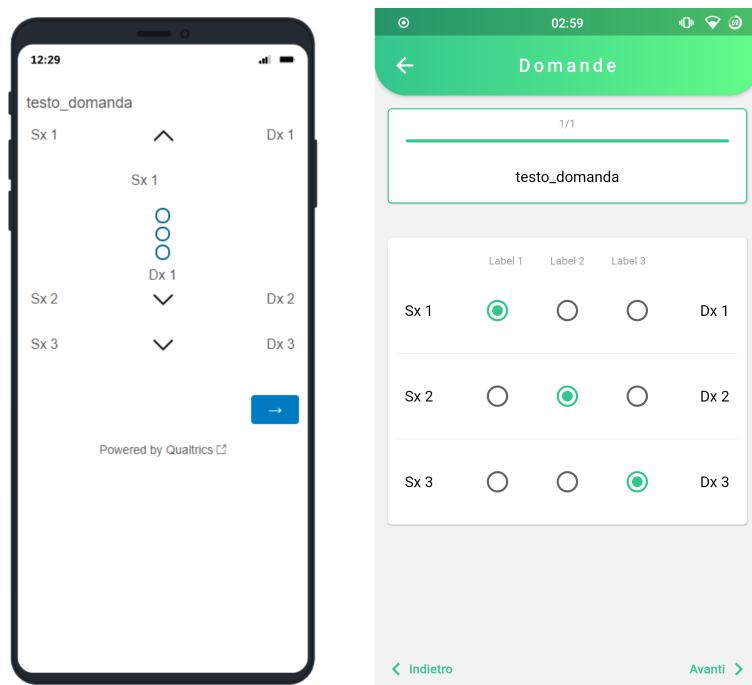


Figura 3.6: Matrix Bipolar

Matrix: "Selector": "MaxDiff"

Un altro tipo di matrice che supporta solo risposte singole è il MaxDiff che è il sotto-tipo duale delle Bipolar: permette di scegliere solo tra due risposte (*ad esempio Si/No*). Non hanno il problema della divisione del testo delle opzioni dunque. Possiamo osservare un esempio di questo tipo di domanda nella Figura 3.7.

Response: Anche in questo caso, essendo una domanda che accetta solo risposte singole, le risposte sono del tipo {"QuestionID_ChoiceID": AnswerID}.

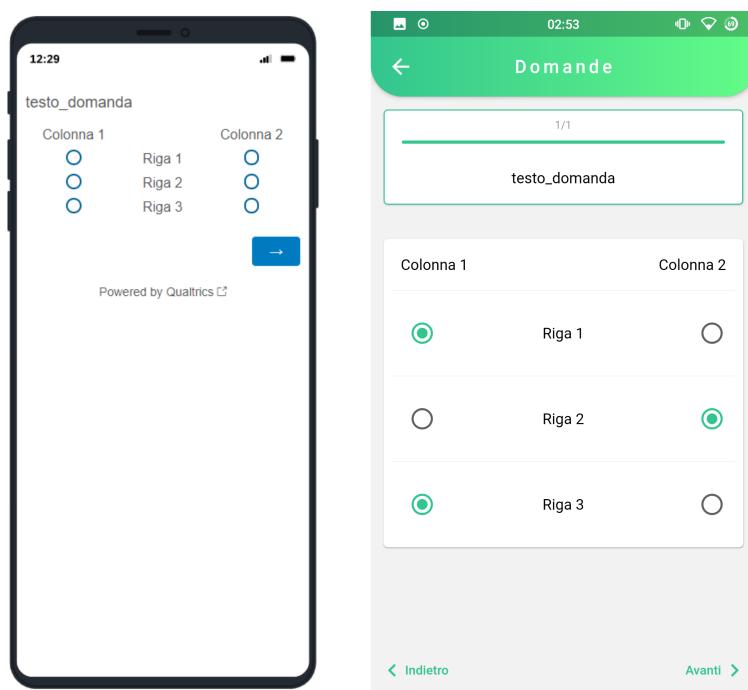


Figura 3.7: Matrix MaxDiff

Matrix: "Selector": "TextEntry"

Una matrice simile alle Likert è la matrice Text Entry che, invece che avere delle risposte chiuse, permette di scrivere una risposta aperta per ogni sotto-domanda. Possiamo notare in Figura 3.8 come anche lo stile adottato richiami molto le matrici Likert, proprio per creare un design coerente in tutta l'applicazione.

Response: Le risposte a questo tipo di domande sono del tipo

{"QuestionID_ChoiceID": "testo"}.

Nel caso un'opzione prevedesse una text entry aggiuntiva, la risposta a tale text entry è del tipo {"QuestionID_ChoiceID_TEXT": "testo"}.

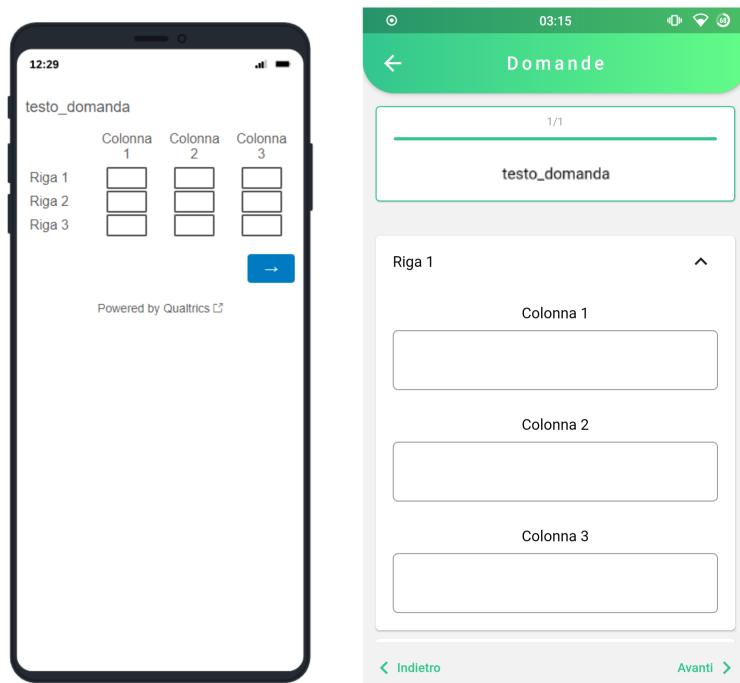


Figura 3.8: Matrix Text Entry

Matrix: "Selector": "RO"

Anche le matrici di tipo Rank Order somigliano molto alle matrici Likert, ma hanno un campo di input in cui l'utente può inserire un numero: queste domande richiedono all'utente di definire un ordine tra le varie opzioni assegnando dei numeri interi ad ogni opzione. Possiamo notare nella Figura 3.9 come lo stile adottato sia molto simile allo stile delle matrici Text Entry ma con un campo di input minore, essendo necessario meno spazio per inserire dei numeri.

Response: Le risposte a questo tipo di domande sono del tipo
{"QuestionID_ChoiceID": "numero"}.

Nel caso un'opzione prevedesse una text entry, la risposta a tale text entry è del tipo {"QuestionID_ChoiceID_TEXT": "testo"}.

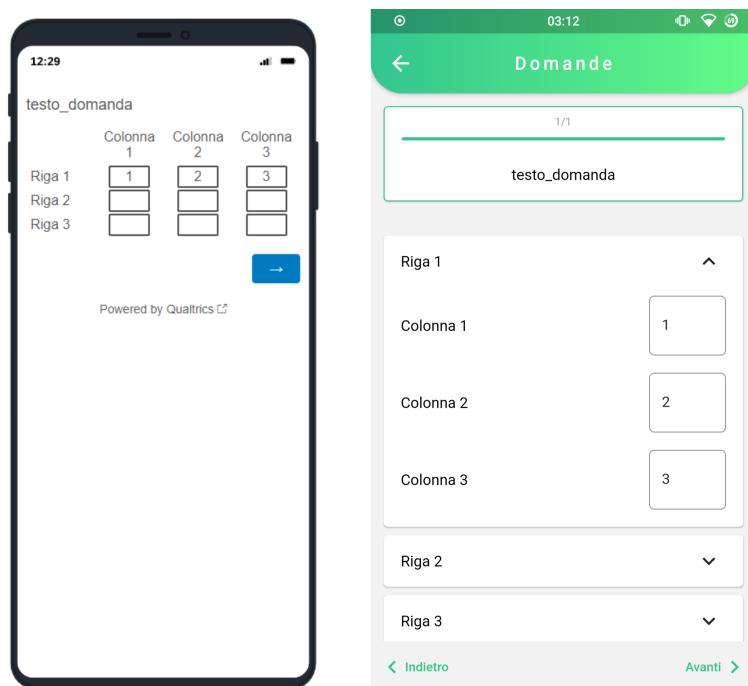


Figura 3.9: Matrix Rank Order

Matrix: "Selector": "CS"

Anche questo ultimo tipo di matrici è simile alle Text Entry: permette di inserire dei numeri come le Rank Order ma che non hanno lo scopo di ordinare le risposte. Queste matrici riportano infine la somma di tutti i numeri inseriti nelle varie opzioni. È possibile osservare nella Figura 3.10 come lo stile adottato sia molto simile alle matrici Rank Order, visto che accettano lo stesso tipo di input.

Response: Le risposte a questo tipo di domande sono del tipo
`{"QuestionID_ChoiceID": "numero"}`.

Nel caso un'opzione prevedesse una text entry, la risposta a tale text entry è del tipo `{"QuestionID_ChoiceID_TEXT": "testo"}`.

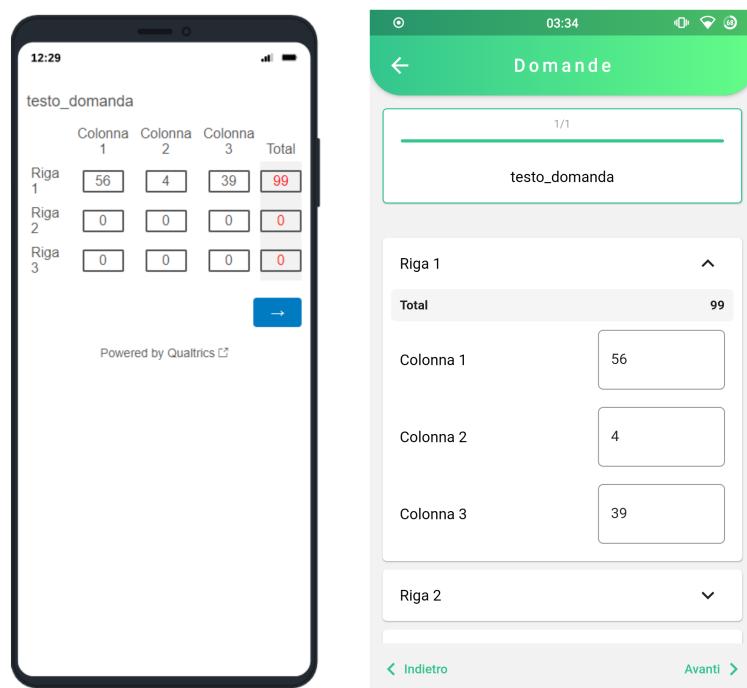


Figura 3.10: Matrix Costant Sum

3.4.5 Side By Side - "QuestionType": "SBS"

Il tipo di domanda Side By Side è un'evoluzione delle matrici precedenti: questo tipo di domanda permette di avere domande completamente diverse in un'unica tabella (*quindi che abbiano anche risposte di diverso tipo, oltre che diverse come nelle matrici profile*). In particolare, i tipi di risposta possono essere Single Answer, SA con menù a tendina, Multiple Answers e Text Entry. Inoltre, come possiamo vedere nella Figura 3.11, a differenza delle matrici semplici è possibile anche definire un numero di risposte variabile per ogni sotto-domanda. Possiamo osservare la struttura di questo tipo di domande nel JSON 3.11 ed è subito notabile una particolarità: nell'oggetto **AdditionalQuestions** vengono salvati effettivamente degli oggetti che descrivono delle matrici come visto precedentemente, ognuna con le proprie **Answers** (*le Choices sono le stesse dell'oggetto SBS stesso*) e soprattutto ognuna col proprio **QuestionID**, parametro molto importante per costruire poi le risposte.

Media

Come per le matrici, anche le Side By Side supportano immagini su **Choices** e **Answers** e text entry aggiuntive sulle **Choices**.

Response

Avendo la domanda più domande interne indipendenti, le risposte sono separate e sono identificate da **QuestionID#SubQuestionID_ChoiceID** (*il SubQuestionID è quello riportato nel campo QuestionID delle sotto-domande nel campo AdditionalQuestions*). Le risposte sono dunque del tipo

{ "QuestionID#SubQuestionID_ChoiceID": "AnswerID"} per una risposta singola,
{ "QuestionID#SubQuestionID_ChoiceID": ["AnswerID_1", "AnswerID_2"] }
per risposte multiple e { "QuestionID#SubQuestionID_ChoiceID": "testo" }
nel caso di text entry.

Nel caso fossero presenti text entry addizionali sulle **Choices**, la risposta a tali text entry è del tipo { "QuestionID#SubQuestionID_ChoiceID_TEXT": "testo" }.

```
{
  "QuestionText": "testo_domanda",
  "QuestionID": "QID257",
  "QuestionType": "SBS",
  "Choices": {},
  "AdditionalQuestions": {
    "1": {
      "Choices": {},
      "Answers": {},
      "QuestionText": "Colonna 1",
      "QuestionType": "Matrix",
      "Selector": "Likert",
      "SubSelector": "MultipleAnswer",
      "QuestionID": "QID257#1",
    },
  },
}
```

JSON 3.11: Oggetto domanda Side By Side

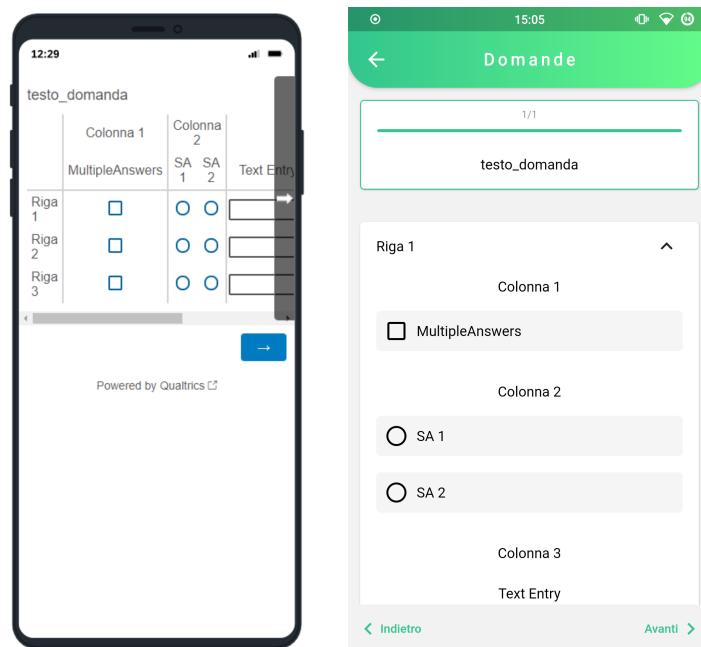


Figura 3.11: Side By Side

3.4.6 Slider - "QuestionType": "Slider"

Un tipo di domanda molto utile messo a disposizione da Qualtrics è lo Slider, ovvero una domanda la cui risposta è esprimibile attraverso dei cursori interattivi, utile ad esprimere ad esempio livelli di preferenza o di intensità. Possiamo osservare la struttura di queste domande nel JSON 3.12, in particolare possiamo osservare che una domanda di questo tipo può avere diversi slider, identificati dalle **Choices** e che, se impostato in modo che assuma solo determinati valori, questi sono presenti nel campo **Answers**. Di particolare interesse è il campo **Configuration** che permette di impostare alcuni parametri dello slider come potrebbero essere i valori minimi e massimi, determinati dai campi **CSSliderMin** e **CSSliderMax**. Se inoltre l'opzione **SnapToGrid** è attiva, lo slider assumerà solo i valori delle **GridLines** (quindi assumerà solo multipli di **CSSliderMax** / **GridLines**). Tramite la configurazione possiamo definire anche quali valori iniziali dovrà assumere ciascun slider, contenuti nel campo **SliderStartPosition**. Per lo sviluppo di questo widget è stato utilizzato come base lo slider generale messo a disposizione da Flutter modificato in termini di aspetto e di funzionalità per supportare i valori iniziali e lo snap to grid. Possiamo osservare anche come sono visibili le grid lines all'interno dello slider stesso nella Figura 3.12.

Media

Anche in questo caso è possibile associare ad ogni **Choice** (*quindi ad ogni slider in questo caso*) un'immagine e una text entry aggiuntiva.

Response

Le risposte a questo tipo di domande sono del tipo `{"QuestionID_ChoiceID": numero}`, dove numero è il valore intero assunto dallo slider. Nel caso ci fosse una text entry aggiuntiva, la risposta sarà del tipo `{"QuestionID_ChoiceID_TEXT": "testo"}`.

```
{  
    "QuestionText": "testo_domanda",  
    "QuestionID": "QID257",  
    "QuestionType": "Slider",  
    "Configuration": {  
        "CSSliderMin": 0,  
        "CSSliderMax": 100,  
        "GridLines": 10,  
        "SnapToGrid": true,  
        "SliderStartPositions": {  
            "1": 0,  
            "2": 0.5  
        }  
    },  
    "Choices": {  
        "1": {  
            "Display": "Slider 1"  
        },  
        "2": {  
            "Display": "Slider 2"  
        }  
    },  
    "ChoiceOrder": [],  
    "Labels": {  
        "1": {  
            "Display": "Label 1"  
        },  
    },  
    "Answers": {  
        "1": {  
            "Display": 0  
        },  
        "11": {  
            "Display": 100  
        }  
    },  
}
```

JSON 3.12: Oggetto domanda Slider

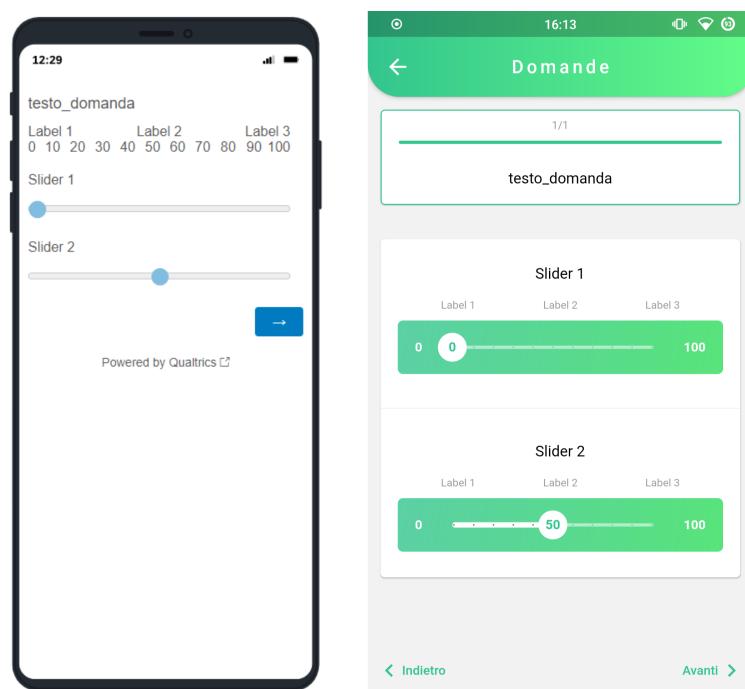


Figura 3.12: Slider

3.4.7 Rank Order - "QuestionType": "RO"

Uno dei tipi di domanda più creativi di Qualtrics è il tipo Rank Order che richiede all'utente, dati degli item, di ordinarli ad esempio per ordine di importanza. Questo ordinamento generalmente viene fatto trascinando gli item l'uno sopra l'altro in una lista: è stato dunque necessario implementare un tipo di lista che permettesse questa interazione. Come possiamo osservare nella Figura 3.13, per evitare ordinamenti accidentali è stato scelto, a differenza di come viene fatto nativamente da Qualtrics, di utilizzare un handler che ricevesse l'input di trascinamento di ogni item. Successivamente, per migliorare ancor di più l'usabilità, è stato scelto di aggiungere anche dei pulsanti che permettessero di scambiare una posizione all'interno della lista in maniera semplice ed efficace. Possiamo osservare la struttura di questa domanda nel JSON 3.13 e notiamo che di fatto gli item non sono altro che delle **Choices**.

Media

Come visto in precedenza, anche in questo caso è possibile assegnare alle **Choices** un'immagine e una text entry aggiuntiva.

Response

Le risposte a questo tipo di domande sono del tipo `{"QuestionID_ChoiceID": numero}`, dove numero è il valore intero che rappresenta la posizione nella lista della suddetta choice. Nel caso ci fosse una text entry aggiuntiva, la risposta sarà del tipo `{"QuestionID_ChoiceID_TEXT": "testo"}`.

```
{  
    "QuestionText": "testo_domanda",  
    "QuestionID": "QID257",  
    "QuestionType": "RO",  
    "Choices": {  
        "1": {  
            "Display": "Item 1"  
        },  
        "2": {  
            "Display": "Item 2"  
        },  
        "5": {  
            "Display": "Item 3"  
        }  
    },  
    "ChoiceOrder": []  
}
```

JSON 3.13: Oggetto domanda Rank Order

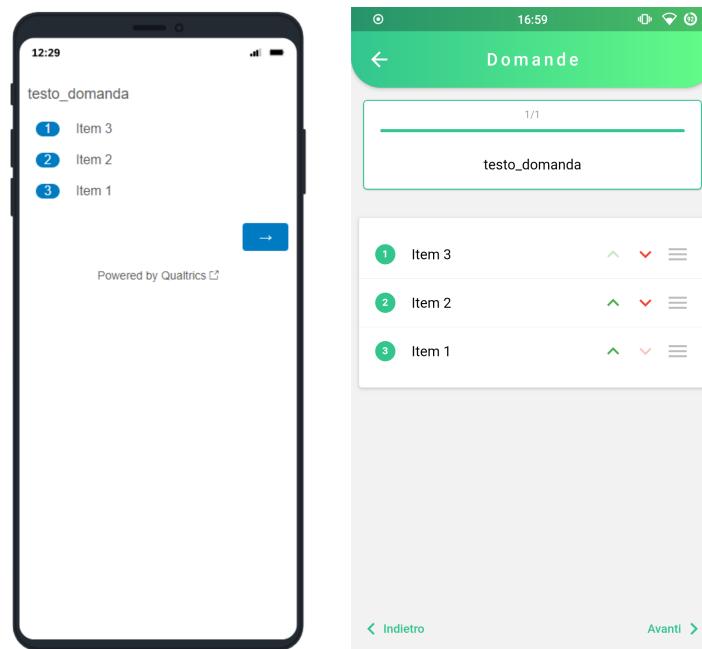


Figura 3.13: Rank Order

3.4.8 Pick, Group and Rank - "QuestionType": "PGR"

Un'evoluzione delle domande di tipo Rank Order sono le Pick, Group and Rank che sostanzialmente sono sempre delle Rank Order ma con più liste: la domanda richiede, oltre che di ordinare gli item di una lista, di assegnare gli item ad una delle liste disponibili. Come possiamo vedere nella Figura 3.14, anche qui sono stati fatti dei cambiamenti per migliorare l'usabilità su mobile: in particolare, per cambiare la lista di un item, non solo è possibile trascinarlo nella lista corrispondente, ma cliccandoci sopra apparirà una finestra di dialogo che permetterà il cambio di lista senza alcun trascinamento. Possiamo osservare la struttura di questo tipo di domanda nel JSON 3.14; possiamo vedere come sia molto simile alle domande Rank Order con l'aggiunta di un campo **Groups** che identifica le liste disponibili.

Media

Anche in questo caso è possibile assegnare alle **Choices** un'immagine e una text entry aggiuntiva.

Response

Le risposte a questo tipo di domande si suddividono in due parti: sono del tipo `{"QuestionID_index_GROUP": [ChoiceID_1, ChoiceID_2]}` per indicare quali item fanno parte di quale lista (*o gruppo*) e del tipo `{"QuestionID_Gindex_ChoiceID_RANK": numero}` per indicare l'ordinamento, dove *Gindex* indica il gruppo di cui fa parte l'item (*ad esempio G0 per il primo gruppo e G1 per il secondo*) e *numero* indica il rank di quell'item nel suddetto gruppo. Nel caso ci fosse una text entry aggiuntiva, la risposta sarà del tipo `{"QuestionID_ChoiceID_TEXT": "testo"}`.

```
{
  "QuestionText": "testo_domanda",
  "QuestionID": "QID257",
  "QuestionType": "PGR",
  "Choices": {},
  "ChoiceOrder": [],
  "Groups": [
    "Lista 1",
    "Lista 2"
  ],
}
```

JSON 3.14: Oggetto domanda Pick, Group and Rank

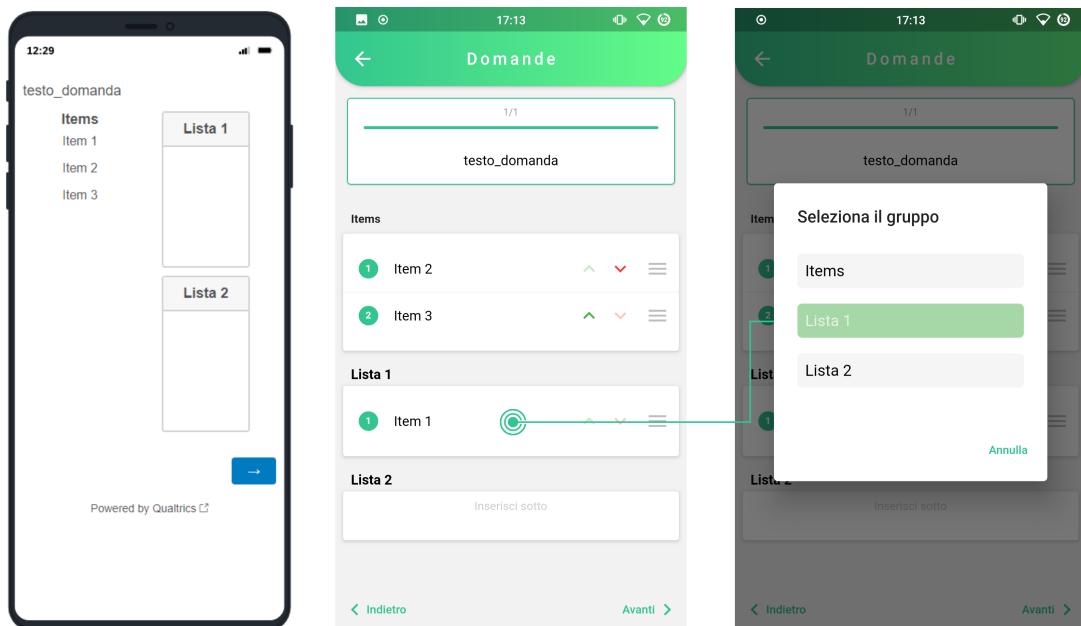


Figura 3.14: Pick, Group and Rank

3.4.9 Description Box - "QuestionType": "DB"

Infine, l'ultima entità messa a disposizione da Qualtrics è la Description Box, una "domanda" che contiene solo il testo ed eventualmente dei media (*di cui parleremo nella sezione successiva*), utile ad esempio per fornire delle spiegazioni all'utente prima che affronti un questionario.

3.5 Question Text

Abbiamo visto nella sezione precedente che c'è un elemento comune a tutte le domande di Qualtrics: tutte presentano un testo di domanda, che a prima vista può sembrare del semplice testo, ma che in realtà è altamente personalizzabile e ha quindi richiesto del codice specifico che lo mostrasse nel modo giusto.

3.5.1 Media

Un aspetto molto utile di questo testo è che Qualtrics permette di inserirvi dei media di accompagnamento come immagini, video e/o audio. A differenza delle domande però, quando viene inserito un media nel testo della domanda questo viene tradotto in un tag HTML adeguato (*in modo che il browser non deva fare nessun lavoro aggiuntivo per visualizzare i media*); quando viene inserita un'immagine viene creato un tag ``, per un video viene usato un tag `<video> <source src="" /></source> </video>` mentre per un audio viene usato un tag `<audio> <source src="" /></source> </audio>`. I tag vengono concatenati all'eventuale plain text.

L'applicazione dunque, prima di mostrare a schermo il testo fornito dalle API, lo analizza in cerca di questi tag e, ogni qual volta ne incontra uno, aggiunge il parametro `src` ad una lista di sorgenti in base al media da mostrare. Successivamente, utilizzando delle espressioni regolari, l'applicazione elimina completamente i tag media dal testo, in modo che non vengano poi mostrati a schermo. A questo punto, utilizzando le liste di sorgenti siamo in grado di renderizzare i media desiderati insieme al testo "pulito". Possiamo osservare alcuni media nella Figura 3.15.



Figura 3.15: Vari media nel question text

3.5.2 YouTube support

Purtroppo, per quanto riguarda i video, Qualtrics offre poco spazio di archiviazione e la soluzione è caricare i video in un file server dal quale ottenere il link allo stream diretto (*e creando i tag video di prima*), oppure caricare i vari video su YouTube, inserendo un iframe nella domanda. Sfortunatamente, di base i link di YouTube non sono link diretti allo stream video e di conseguenza non funzionano col lettore video utilizzato per quel tipo di link (*anche se sono ottenibili tramite procedure non ovvie, ma sono stream separati per audio e video e non ho la sicurezza che siano in linea con i termini d'uso di YouTube*). La soluzione è stata utilizzare un iframe vero e proprio anche nell'applicazione, seppur leggermente mascherato per avere un aspetto più nativo.

3.5.3 Rich Text Editor

Qualtrics permette inoltre di modificare il testo della domanda tramite un rich text editor, ovvero tramite un editor che permette di assegnare dello stile al testo che può essere grassetto, corsivo, sottolineato ma anche colorato, con lo sfondo colorato, in elenco puntato e così via. Questo è il motivo per il quale, in tutti gli screen visti fino ad ora, il testo appare "semplice": la scelta di modificarlo e dargli enfasi è stata lasciata a chi crea le domande, mentre l'applicazione ricrea tale stile. Per realizzare tutto ciò è stato seguito un approccio simile a quello per i media: per creare questi stili vengono utilizzati dei tag HTML che l'applicazione individua e, utilizzando un widget di tipo **RichText**, che permette di avere diversi widget **TextSpan** con diversi stili di testo ciascuno, ricrea lo stile desiderato eliminando dal testo poi i tag. Possiamo osservare un esempio di testo editato nella Figura 3.16.



Figura 3.16: Esempio di testo editato

3.6 Validation

Quando abbiamo parlato di domande abbiamo visto che un altro parametro comune a tutte è la cosiddetta *Validation*: questo oggetto contiene una serie di regole logiche (*definibili da chi crea le domande tramite l'ambiente Qualtrics*) che definiscono se una certa risposta è giusta o meno. Queste regole sono varie e possono definire delle risposte che devono essere scelte, risposte che non devono essere scelte, un numero minimo di scelte e così via. Un esempio di validation è visibile nel JSON 3.15.

```
"Validation": {
    "Logic": {
        "0": {
            "0": {
                "QuestionID": "QID162",
                "Operator": "Selected",
                "LeftOperand": "q://QID162/SelectableChoice/1",
            },
            "1": {
                "QuestionID": "QID162",
                "Operator": "EqualTo",
                "LeftOperand": "q://QID162/SelectedChoicesCount",
                "RightOperand": "1",
                "Conjunction": "And"
            },
            "2": {
                "QuestionID": "QID162",
                "Operator": "GreaterThanOrEqual",
                "QuestionIDFromLocator": "QID162",
                "LeftOperand": "q://QID162/SelectedChoicesCount",
                "RightOperand": "2",
                "Conjunction": "Or"
            },
        },
    },
}
```

JSON 3.15: Esempio di oggetto Validation

3.6.1 Struttura dati e traduzione

Possiamo notare come le regole logiche siano identificate da un **Operator**, che stabilisce cosa va controllato, e da degli operandi: il **LeftOperand** in particolare è un hyperlink ad un'opzione di una domanda piuttosto che al numero di opzioni selezionate e così via da cui però è deducibile a cosa si riferisce. L'applicazione, leggendo l'operatore e gli operandi potrà sapere cosa deve controllare per validare una risposta.

Notiamo però che le regole possono essere multiple e sono legate fra loro da una **Conjunction**, ovvero da un operatore logico che può essere AND oppure OR: è necessario che l'applicazione valuti l'intero set di regole per poter valutare una risposta. È necessaria dunque una struttura dati che venga valutata implicitamente dall'applicazione.

Ancora una volta sono tornate utili le mappe: ad ogni domanda munita di validation sarebbe stata associata una mappa che al suo interno avrebbe contenuto altre mappe. In particolare, al livello più esterno viene usato come chiave l'id della domanda alla quale è legata la regola e come valore è presente un'altra mappa che come chiave ha una **posizione**; la posizione determina quali regole sono da considerarsi insieme e quali no. La valutazione implicita viene eseguita qui: tutte le regole con la stessa posizione sono valutate come se fossero in AND logico, mentre tutte le regole con posizione diversa sono valutate in OR logico. Sotto alle posizioni ci sono poi delle ulteriori mappe che hanno come chiave l'operatore che identifica il tipo di controllo da eseguire e come valore un oggetto con due valori: un **access** che identifica il **LeftOperand** (*è la chiave d'accesso alla mappa delle risposte che ritorna il valore da controllare*) e un **RightOperand** che può essere un numero, del testo o una lista di id di opzioni, a seconda anche dell'operatore utilizzato.

Gli operatori disponibili (*e quindi le chiavi delle mappe più interne*) sono diversi e ognuno implementa un certo tipo di controllo.

Alcuni operatori sono:

- **whitelist** - contiene le scelte che devono essere selezionate
- **blacklist** - contiene le scelte che non devono essere selezionate
- **minChoices** - almeno n scelte selezionate
- **maxChoices** - al massimo n scelte selezionate

- **notEqualLength** - selezionate un numero $\neq n$ di scelte
- **equalLength** - selezionate esattamente n scelte
- **equal** - la risposta in corrispondenza della accessKey è uguale al rightOperand
- **greaterThanLength** - selezionate un numero $> n$ di scelte
- **greaterThanEqualLength** - selezionate un numero $\geq n$ di scelte
- **lessThanLength** - selezionate un numero $< n$ di scelte
- **lessThanEqualLength** - selezionate un numero $\leq n$ di scelte

In particolare, l'esempio visto nel JSON 3.15 viene tradotto nella struttura visibile nel JSON 3.16, ovvero considera in AND le regole **whitelist** (*must be selected*) e **equalLength** (*devono esserne selezionate esattamente n*) che hanno come posizione 0, il tutto considerato poi in OR con la regola **greaterThanEqualLength** (*devono esserne selezionate un numero $\geq n$*) che ha posizione 1.

```
"QID162": {
  0: {
    "whitelist": {
      "access": "QID162",
      "rightOperand": [1]
    },
    "equalLength": {
      "access": "QID162",
      "rightOperand": 1
    }
  },
  1: {
    "greaterThanEqualLength": {
      "access": "QID162",
      "rightOperand": 2
    },
  },
}
}
```

JSON 3.16: Esempio di mappa di regole logiche

3.6.2 Risposta in app

Per gestire la struttura dati di validazione, che è condivisa fra le varie domande, è stato creato un **provider** che permettesse così di aggiungere le regole in fase di lettura delle domande dalle API e di consultarle in fase di controllo. In particolare, se il controllo fallisce l'applicazione evidenzia la domanda con un alone rosso come possiamo osservare nella Figura 3.17.

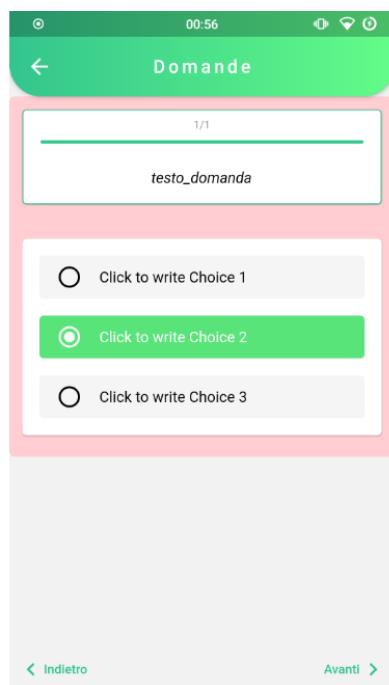


Figura 3.17: Esempio di validazione errata

3.7 Screening

Nel capitolo precedente abbiamo parlato di una fase iniziale di screening atta a individuare le patologie di cui potenzialmente soffre l'utente: questi screening sono composti sia da alcune domande generiche (*domande sociodemografiche*) che da specifiche domande misurabili che diano un'idea dei problemi del paziente. La particolarità di questo processo è che i tipi di domande e le scale utilizzate sono varie a seconda della patologia e sono basati su test psicologici ufficiali e ben noti e producono un risultato che collochi ogni patologia in una determinata fascia di rischio tra **minimo**, **lieve**, **moderato** e **alto**.

Per mantenere il livello di ogni patologia verrà utilizzato un provider apposito che fornisca al resto dell'applicazione le informazioni su quali sono i moduli da trattare. Inoltre, poiché gli screening vengono erogati tutti in una sola sessione, per la loro organizzazione si è scelto di creare un unico sondaggio che contenesse vari blocchi, ognuno dei quali si riferisce ad uno specifico screening: in questo modo l'applicazione può distinguere tra i vari screening e calcolarne il punteggio indipendentemente dagli altri. Inoltre, in questo modo verrà inviata una sola risposta al server di Qualtrics una volta che l'utente ha completato gli screening.

3.7.1 Depressione

Per misurare il livello di depressione viene utilizzata la scala **Beck Depression Inventory-II (BDI-II)**[8] che consiste di 21 domande (*anche dette items*), ognuna delle quali può assumere un punteggio che va da 0 a 3 in base alla risposta data. Il punteggio finale, ottenuto sommando i punteggi di tutte e 21 le domande, indicherà la fascia di rischio della suddetta patologia, in particolare:

- [0, 13] Indica un rischio minimo
- [14, 19] Indica un rischio lieve
- [20, 28] Indica un rischio moderato
- [29, 63] Indica un rischio alto

Fanno eccezione alcune domande trasversali, il quale punteggio, se alto, influenza anche il rilevamento di altre patologie come le problematiche del sonno e dell'ansia. Per semplificare l'ottenimento di tali punteggi, le risposte alle varie domande iniziano con "x. " dove x indica il punteggio relativo alla risposta.

3.7.2 Ansia

Per l'ansia, la scala utilizzata è la **State-Trait Anxiety Inventory (STAI)**[9] che consiste di 40 domande che accettano come risposta una scala Likert a 4 punti. La scala STAI si suddivide in due sotto-scale: **SAI** e **TAI** utili a misurare l'**ansia di stato** (*ansia legata ad un evento*) e l'**ansia di tratto** (*ansia legata alle caratteristiche personali*).

Di conseguenza, le domande si suddividono in 20 per la scala SAI e 20 per la scala TAI ed il punteggio minimo è di 20 per ciascuna scala (*la scala likert utilizzata va da 1 a 4*).

Occorre notare però che entrambe le sotto-scale presentano domande legate sia alla presenza che all'assenza di ansia. Di conseguenza alcuni item sono invertiti (*es: "Mi sento calmo: 4" vale 1*): vale a dire che il punteggio degli item legati all'assenza di ansia è l'inverso della risposta data.

Le rispettive fasce di rischio per le due scale sono:

- [0, 40] Indica un rischio minimo
- [41, 50] Indica un rischio lieve
- [51, 60] Indica un rischio moderato
- [61, 80] Indica un rischio alto

La sotto-scala col rischio maggiore determinerà quindi la fascia legata all'ansia nell'applicazione.

In questo caso la risposta stessa è il punteggio (*tranne per quando si tratta di domande invertite, risolvibile comunque con un'operazione di modulo*) e non sono necessarie ulteriori azioni per ricavarlo.

3.7.3 Problemi di sonno

Per calcolare le problematiche del sonno viene utilizzata la **Pittsburgh Sleep Quality Index (PSQI)**[10] che consiste di 19 item raggruppati in 7 componenti a cui viene assegnato un punteggio da 0 a 3. Per ottenere tale punteggio bisogna considerare i punteggi, sempre da 0 a 3, associati agli item facenti parte della relativa componente. Il punteggio massimo è quindi di 21 (*vengono considerati i punteggi delle componenti per la scala*) e le fasce di rischio sono:

- [0, 4] Indica un rischio minimo
- [5, 10] Indica un rischio lieve
- [11, 15] Indica un rischio moderato
- [16, 21] Indica un rischio alto

In questo caso le domande sono di vario tipo: alcune utilizzano una scala likert che fornisce direttamente il punteggio mentre altre richiedono all’utente di inserire, ad esempio, le ore di sonno dalle quali poi ricavare un punteggio.

3.7.4 Dolore cronico

Per calcolare la fascia di rischio del dolore cronico viene utilizzata la scala **Brief Pain Inventory (BPI)**[11] che consiste di due sotto-scale, la prima relativa all'**intensità del dolore** composta da 4 item e la seconda relativa all'**interferenza del dolore** composta da 11 item. Tutti gli item sono valutati tramite una scala likert a 11 punti (*da 0 a 10*). Ognuna delle due scale può assumere un punteggio da 0 a 10 e la fascia della patologia viene decisa dalla scala con punteggio maggiore.

In particolare, le fasce di rischio sono:

- [0, 4] Indica un rischio minimo
- [5, 6] Indica un rischio lieve
- [7, 8] Indica un rischio moderato
- [9, 10] Indica un rischio alto

3.7.5 Burnout

Il rischio di burnout viene calcolato tramite la scala **School Burnout Inventory (SBI)**[12] che si suddivide in tre sotto-scale: **Esaурimento a scuola**, **Cinismo verso la scuola** e **Senso di inadeguatezza a scuola**, ognuna delle quali contiene degli item valutati con una scala likert da 1 a 6 il cui punteggio totale va poi suddiviso per il numero di item della sotto-scala stessa. Il punteggio va poi confrontato con degli indici di un campione italiano di studenti in base al sesso dell'utente.

Le fasce di rischio sono quindi selezionate in base al sesso dell'utente e alla sotto-scala con punteggio maggiore come possiamo vedere nella Tabella 3.2.

MASCHI	MINIMO	LIEVE	MODERATO	ALTO
Esaурimento	1 - 2.91	2.92 - 3.612	3.613 - 4.306	4.307 - 5
Cinismo	1 - 3.16	3.17 - 3.77	3.78 - 4.39	4.40 - 5
Inadeguatezza	1 - 3.22	3.23 - 3.82	3.83 - 4.40	4.41 - 5

FEMMINE	MINIMO	LIEVE	MODERATO	ALTO
Esaурimento	1 - 3.04	3.05 - 3.6	3.7 - 4.34	4.35 - 5
Cinismo	1 - 3.05	3.06 - 3.706	3.707 - 4.352	4.353 - 5
Inadeguatezza	1 - 3.11	3.12 - 3.745	3.746 - 4.372	4.373 - 5

ALTRO	MINIMO	LIEVE	MODERATO	ALTO
Esaурimento	1 - 2.97	2.98 - 3.606	3.607 - 4.323	4.234 - 5
Cinismo	1 - 3.10	3.11 - 3.73	3.74 - 4.37	4.38 - 5
Inadeguatezza	1 - 3.16	3.17 - 3.78	3.79 - 4.38	4.39 - 5

Tabella 3.2: Tabella fasce di rischio SBI

3.7.6 Pensieri autodistruttivi

Per il calcolo della fascia dei pensieri autodistruttivi viene usata la **Columbia-Suicide Severity Rating Scale (C-SSRS)**[13] che prevede 6 domande la cui risposta è Sì/No in riferimento ad un breve periodo (*ad esempio un mese*) e in riferimento al corso di vita; in particolare un "No" ha peso 0, mentre un "Sì" ha peso 1 o 2 a seconda dell'item considerato.

Essendo questo modulo legato ad una tematica più grave rispetto alle altre non ha una vera e propria scala, ma se uno degli item con peso 2 viene selezionato, allora il modulo sarà somministrato all'utente. Se il peso massimo raggiunto è 0 o 1 allora il modulo non sarà somministrato; a primo impatto potrebbe sembrare sbilanciato come algoritmo, ma gli item con peso 1 sono pochi rispetto a quelli con peso 2.

3.7.7 Difficoltà relazionali

Infine, per le difficoltà relazionali è stato utilizzata la scala **Difficulties in Emotion Regulation Scale (DERS-20)**[14] che prevede 20 item che misurino la capacità dell'individuo a relazionarsi con gli altri tramite una scala likert.

Seppur anche questa scala abbia un suo metodo di calcolo come le altre, l'applicazione non ne calcolerà il punteggio poiché, come detto nella sezione 2.2 del capitolo precedente, il modulo delle difficoltà relazionali, visto il target di utenza, è trasversale agli altri e verrà comunque somministrato alla sesta settimana a tutti gli utenti.

3.7.8 Priorità

Nel caso due o più moduli cadano nella stessa fascia, in genere l'utente potrà selezionarne due su cui lavorare. Tuttavia, se la fascia considerata è Alta o Moderata e se uno dei moduli scelti dall'algoritmo è quello della depressione, esso avrà la priorità sugli altri, lasciando all'utente la scelta di un ulteriore modulo.

Inoltre, il modulo sui pensieri autodistruttivi, se selezionato, ha sempre la precedenza su tutti.

3.8 Esercizi

Abbiamo visto precedentemente come le risposte di un questionario su Qualtrics siano legate al questionario stesso (*sono quindi indipendenti dai blocchi*), di conseguenza è possibile inviare le risposte ad un dato questionario solo dopo aver risposto a tutte le sue domande. Questo impone una certa organizzazione dell'ambiente Qualtrics che permetta all'applicazione di usufruire in maniera separata di sottoinsiemi di domande e allo stesso tempo di fornire dei dati continuativi al gruppo di psicologia piuttosto che inviare i dati solo alla fine del percorso; difatti sarebbe stato possibile organizzare il tutto in un unico grande questionario, suddividendo screening ed esercizi vari in blocchi, ma questo avrebbe comportato un'impossibilità all'invio delle risposte man mano che l'utente completa i vari esercizi.

3.8.1 Suddivisione questionari

Dunque, per suddividere le varie parti del percorso curativo e per poter inviare dati regolarmente è stato scelto di creare diversi questionari: il primo conterrà dei blocchi, ognuno dei quali conterrà i vari screening e la presentazione iniziale. La suddivisione degli screening in blocchi permetterà inoltre all'utente di non doverli fare obbligatoriamente tutti in un'unica sessione (*l'applicazione salva comunque le risposte date tramite i provider implementati, ma avere dei blocchi separati permette all'utente anche di concentrarsi su un solo blocco alla volta*)

Sondaggi esercizi

Successivamente sono stati suddivisi gli esercizi e gli screening giornalieri e settimanali in ulteriori questionari, in particolare è stato creato un questionario per ogni settimana e per ogni patologia (*fatta eccezione per le difficoltà relazionali che coprono la sesta settimana per tutti*) che contenesse 2 blocchi, uno per la psico-eduzione (*che va somministrata prima degli esercizi veri e propri*) e uno per gli esercizi. Un altro motivo per il quale gli esercizi settimanali sono suddivisi in sondaggi diversi è che l'utente non può accedere ad un esercizio se prima non ha completato quello della settimana precedente.

Sondaggi daily screening

Per le domande giornaliere, essendo queste di carattere generale e non necessariamente associate ad una patologia, si è scelto di creare degli ulteriori sondaggi, uno per settimana, ognuno con sette domande da somministrare giorno per giorno all'utente. La scelta di avere sette sondaggi è dettata sempre dal poter inviare le risposte date dall'utente in una maniera più continuativa.

Sondaggi weekly screening

Le domande settimanali, che agiscono da screening settimanali, sono invece legate alle patologie, dunque è stato necessario creare dei sondaggi dedicati ad esse. In particolare i sondaggi sono uno per settimana per ogni patologia (*fatta eccezione per la sesta settimana che ha solo le domande settimanali delle difficoltà relazionali*).

3.8.2 Identificazione domande

Abbiamo detto che uno dei punti forte di questa applicazione è l'anonimato che offre. Però, a fini statistici e prescrittivi, è comunque necessario poter identificare quali risposte appartengono ad uno stesso utente (*indipendentemente dalla sua identità*): per far ciò, ogni sondaggio è provvisto di una domanda extra di tipo text entry, la cui risposta sarà compilata in automatico e corrisponderà ad un id univoco dell'utente.

Questo id non è in nessun modo associato alle informazioni personali dell'utente, difatti viene creato un **UUID** (*Universally Unique Identifier*) di versione 4, ovvero generato in modo casuale su 122 bit con i restanti 6 fissi che indicano la versione. Il numero di bit generati casualmente permette di essere certi che gli id generati siano univoci e allo stesso tempo anonimi: per poter avere una probabilità del 50% di generare lo stesso UUID v4 sarebbe necessario generare **2.71 quintiliioni** di id, equivalente a generare un miliardo di UUID v4 al secondo per 85 anni[15].

CAPITOLO

4

MANUALE UTENTE

Questo capitolo ha lo scopo di discutere e presentare le varie schermate dell'applicazione e di fornire una guida sull'utilizzo di quest'ultima, mettendo in evidenza cosa rappresenta ogni schermata. In particolare, saranno analizzati i vari elementi di ogni schermata e le varie rotte di navigazione che vengono offerte dalle varie parti dell'applicazione. Verrà inoltre presentato il flusso dell'applicazione, vale a dire cosa è stato progettato che l'utente veda, in che ordine dal primo avvio fino all'utilizzo quotidiano e il motivo per il quale è stato adottato tale flusso.

4.1 Prerequisiti

Per poter utilizzare l'applicazione, almeno per quanto riguarda Android, è necessario uno smartphone con una versione di Android con **API di livello almeno 19**¹, ovvero con almeno **Android 4.4 KitKat**, tuttavia è consigliato utilizzare almeno **Android 5.0 Lollipop** per via di alcune funzionalità che potrebbero non funzionare ugualmente bene su versioni anteriori. Considerando gli sviluppi attuali del sistema operativo è ragionevole pensare tutti i dispositivi Android supportino almeno Lollipop, che è stata la rivoluzione di casa Google in termini di sistema operativo, modificando aspetti legati all'esecuzione di applicazioni e mettendo in pratica il material design. Per quanto riguarda iOS, la versione minima supportata da Flutter è **iOS 9.0**².

¹source.android.com/setup/start/build-numbers

²flutter.dev/docs/development/tools/sdk/release-notes/supported-platforms

4.2 On-Boarding

Al primo avvio dell'applicazione l'utente verrà accolto da una brevissima introduzione su cosa fa l'applicazione e come si propone di farlo. Questa introduzione è estremamente riassuntiva e ha lo scopo di evitare che l'utente si trovi davanti da subito una schermata complicata e che potrebbe creare confusione a prima vista. Possiamo vedere nella Figura 4.1 tale schermata, composta da tre pagine opportunamente animate e curate esteticamente; questa è la prima impressione che l'applicazione darà all'utente ed è quindi importante che queste pagine siano curate nei minimi dettagli.



Figura 4.1: Schermata on-boarding

4.3 Presentazione tutor

Conclusa la schermata on-boarding, l'utente verrà accolto da una schermata di introduzione delle tutor di psicologia, accompagnata da video, nella quale ciascuna tutor si presenterà all'utente, spiegherà di che sintomatologia si occupa e come sarà costruito il proprio percorso. In questa fase c'è anche una spiegazione più dettagliata su come funziona l'applicazione, come sono erogati gli esercizi settimanali e le domande giornaliere e cosa comprendono. Possiamo vedere in dettaglio la schermata di presentazione nella Figura 4.2.

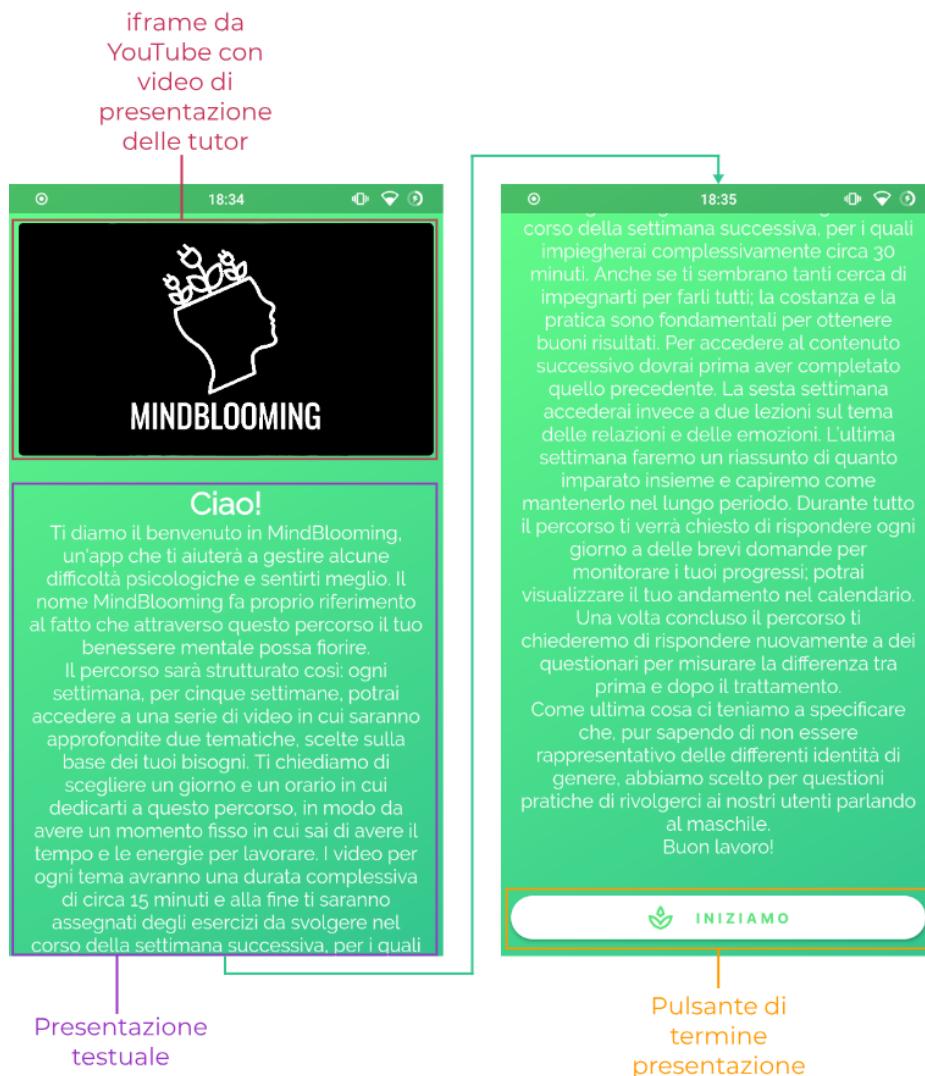


Figura 4.2: Schermata presentazione tutor

4.4 Screening

A questo punto l'utente è pronto ad eseguire gli screening iniziali per poter individuare le eventuali patologie di cui soffre. Come possiamo vedere nella Figura 4.3, all'utente verrà presentata una lista con le varie sezioni dello screening, che vanno da alcuni criteri di inclusione fino agli screening veri e propri di ogni patologia. Possiamo osservare inoltre come per poter accedere ad una sezione, l'utente deva prima completare la precedente. Inoltre, all'utente sarà visibile anche la presentazione generale di prima, consultabile in qualsiasi momento.

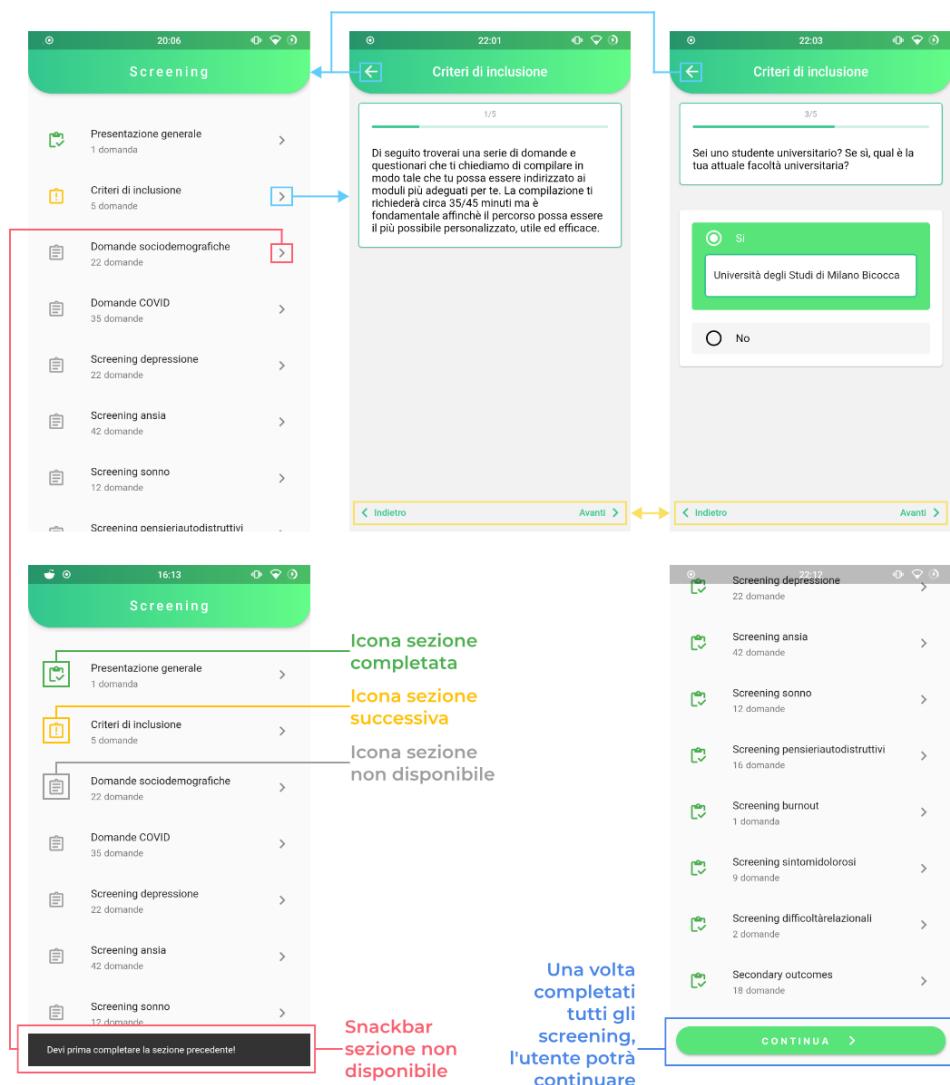


Figura 4.3: Schermata Screening

4.5 Risultati

Una volta completati tutti gli screenig, l'utente potrà procedere verso la schermata dei risultati, visibile in Figura 4.4, in cui verranno comunicati i disturbi rilevati, se ve ne sono, con un'eventuale descrizione di questi. Se l'algoritmo di screening individua esattamente due patologie su cui lavorare, l'utente da questa schermata potrà andare direttamente alla schermata finale delle impostazioni, mentre se l'algoritmo ne identifica solo una oppure ne identifica più di due senza poter scegliere una priorità, l'utente verrà condotto ad una schermata di scelta delle patologie. Se nessuna patologia viene rilevata dall'algoritmo, l'utente andrà direttamente alla schermata di scelta delle patologie, dove potrà comunque sceglierne due.

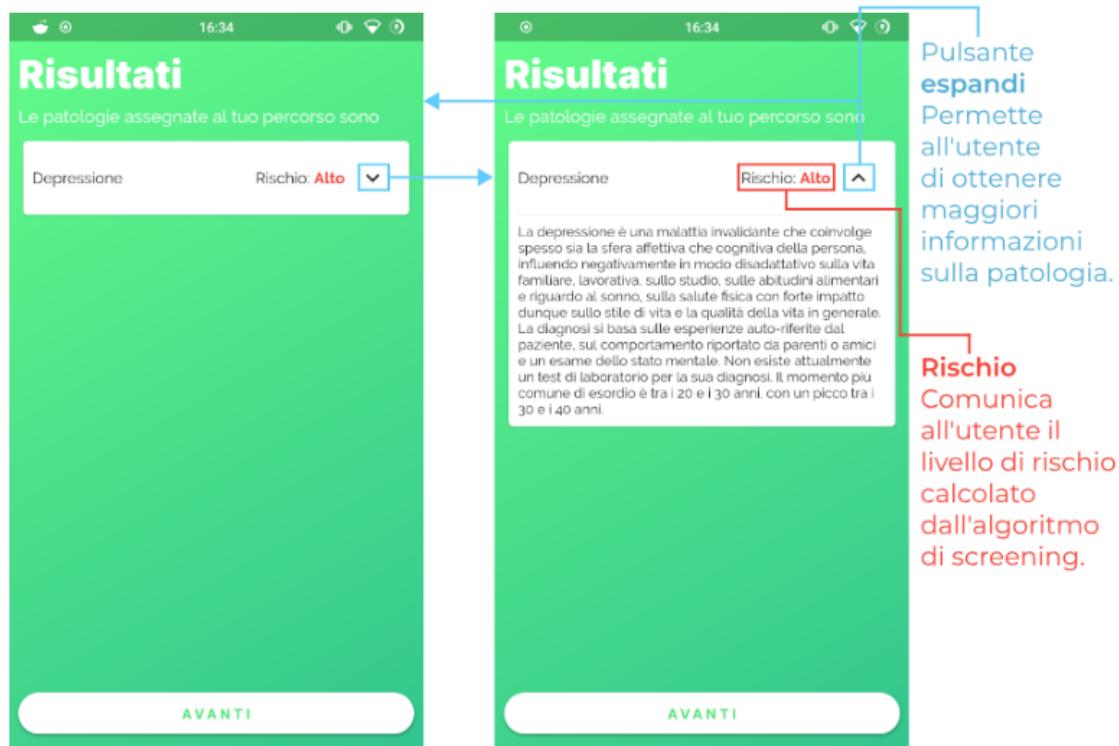


Figura 4.4: Schermata Risultati

4.6 Selezione patologie

Come detto prima, nel caso l'algoritmo non identifichi due patologie o nel caso non fosse possibile definire delle priorità tra quelle identificate, all'utente verrà presentata anche la schermata di scelta delle patologie che, come è visibile nella Figura 4.5, è molto simile alla precedente, ma permette di selezionare effettivamente le patologie (*fino ad arrivare a due patologie complessive tra scelte dall'algoritmo e scelte dall'utente*).

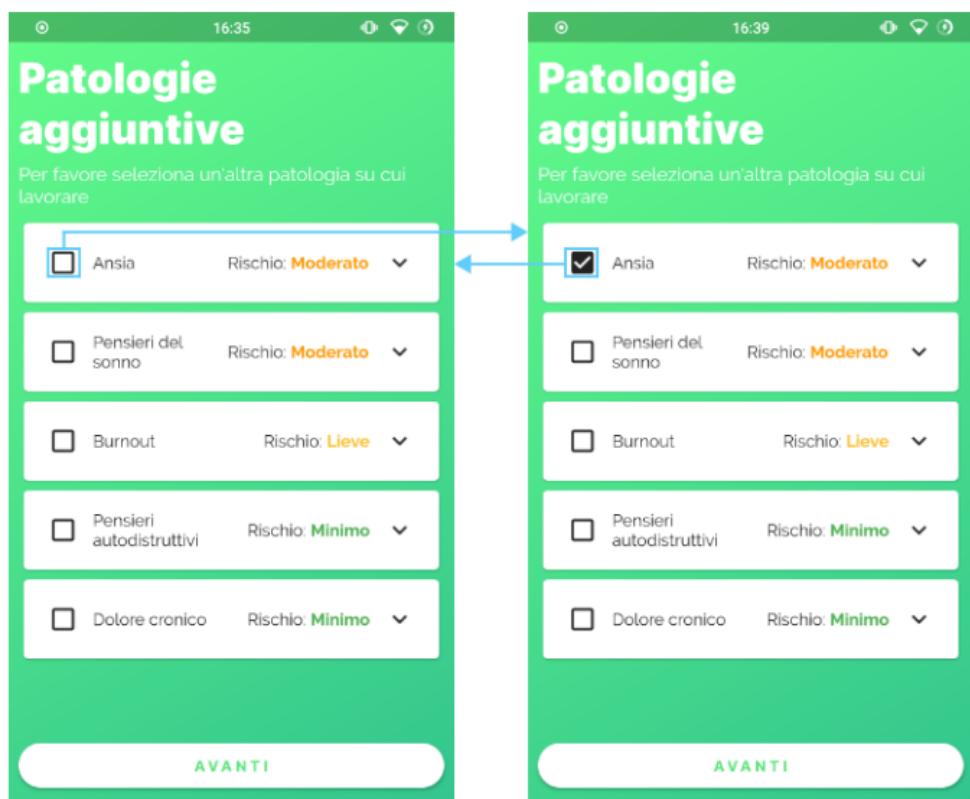


Figura 4.5: Schermata Selezione patologie

4.7 Impostazioni iniziali

Una volta scelte le patologie, come è osservabile nella Figura 4.6, l’utente potrà scegliere alcune impostazioni iniziali come l’orario al quale ricevere la notifica dello screening giornaliero (*daily screening*) e un eventuale ‘compagno di viaggio’, ovvero una piantina che mostri l’andamento del percorso curativo e che crescerà man mano che l’utente completa gli esercizi (*questa funzionalità fa parte degli sviluppi futuri tuttavia*). Conclusa questa schermata l’utente accederà alla homepage. È possibile osservare come i metodi di selezione dell’orario e della piantina siano molto basilari e semplici in modo che sia subito evidente come utilizzarli.

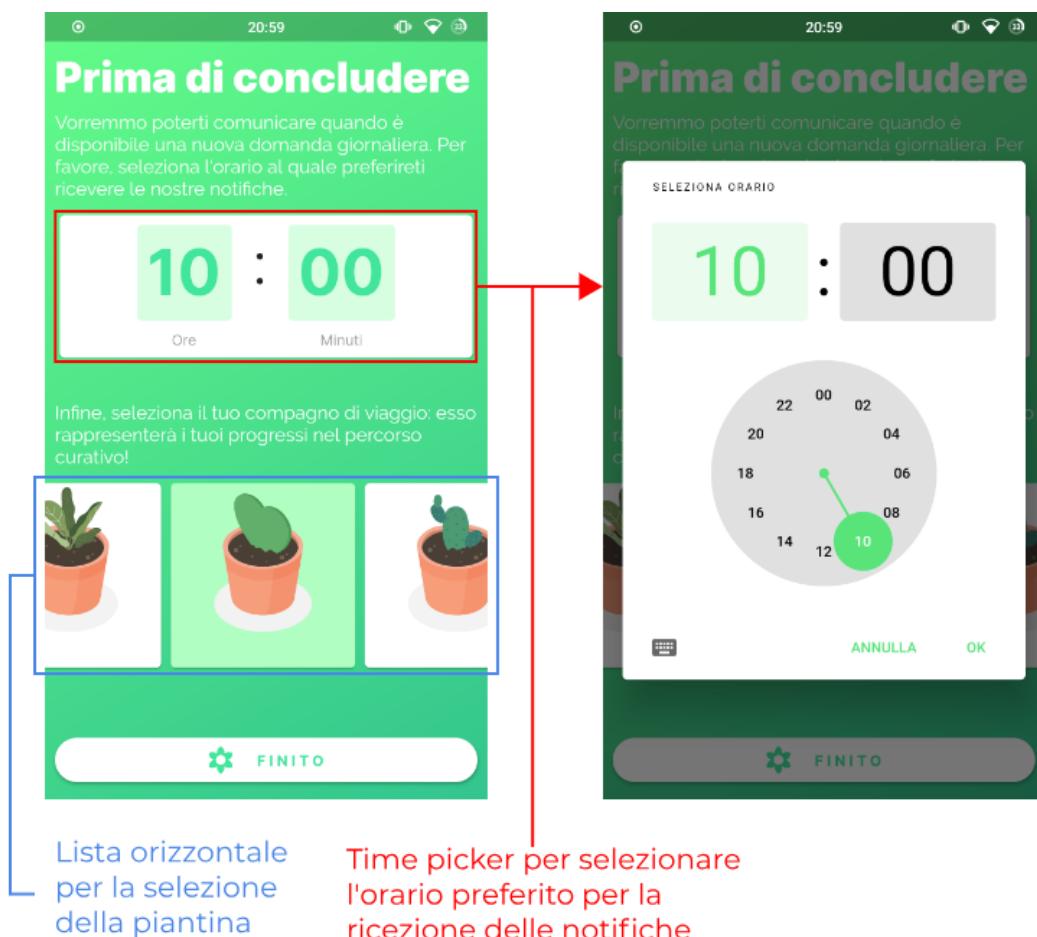


Figura 4.6: Schermata impostazioni iniziali

4.8 Homepage

La homepage, la schermata che l'utente vedrà appena aperta l'applicazione una volta conclusa la fase di screening, è composta, come è osservabile in Figura 4.7, da un calendario che mostrerà all'utente le scadenze dei vari esercizi e daily screening e dalla piantina che mostrerà l'andamento del suo percorso. In particolare, il calendario evidenzierà gli esercizi in scadenza nel giorno corrente e quelli completati.



Figura 4.7: Schermata Homepage

4.9 Esercizi

In questa schermata l'utente può visualizzare gli esercizi disponibili e rivedere quelli già completati, in modo che l'utente abbia sempre accesso alle risorse multimediali messe a disposizione dall'applicazione. Come mostrato in Figura 4.8, la schermata contiene uno switch che permette all'utente di vedere gli esercizi disponibili o quelli già completati. Accedendo ad un esercizio, all'utente verrà presentata una schermata di sezioni, ognuna delle quali contenente delle domande, come quella vista nella sezione 4.4.

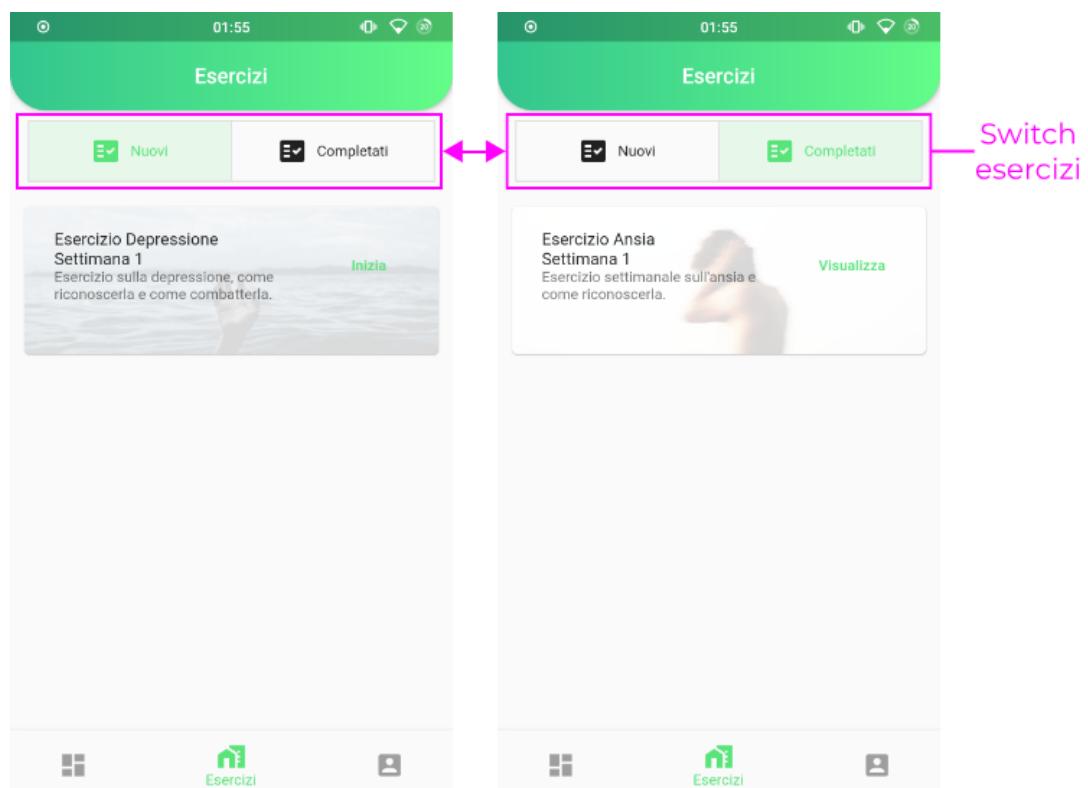


Figura 4.8: Schermata Esercizi

4.10 Profilo

Infine, nella schermata profilo l'utente potrà modificare le proprie impostazioni, ovvero, come visibile in Figura 4.9, cambiare l'orario delle notifiche e la propria piantina. Notiamo che i widget utilizzati per la modifica di queste impostazioni sono i medesimi utilizzati nell'impostazione iniziale mostrata nella sezione 4.7.

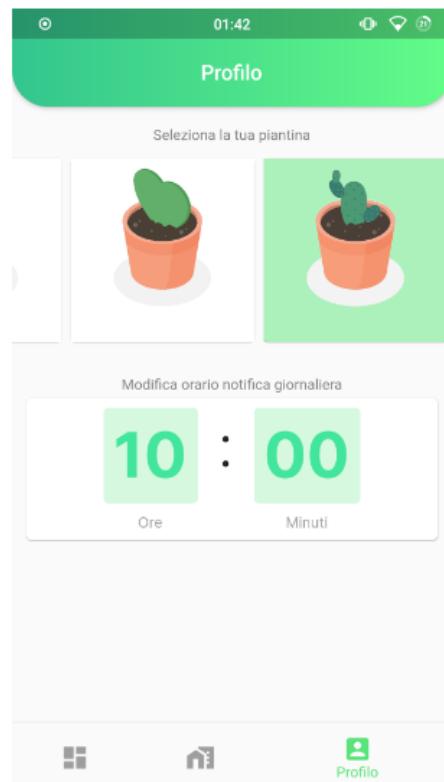


Figura 4.9: Schermata Profilo

CAPITOLO

5

CONCLUSIONI E SVILUPPI FUTURI

L'obiettivo prefissato per questo progetto di stage era lo sviluppo di una versione *alpha* di quello che sarebbe stato *MindBlooming*, una versione che avrebbe stabilito se il progetto fosse stato fattibile, di quali dati avrebbe avuto bisogno, di che struttura ci sarebbe dovuta stare dietro. È innegabile che l'applicazione sia ancora acerba, ma aver potuto stabilire questi punti iniziali permetterà un'evoluzione di essa molto più facilitata sapendo già quali dati creare e/o usare.

5.1 Idee future

Al momento tutto ciò che concerne gli sviluppi futuri è il perfezionamento ed il testing di quanto fatto fino ad ora, in modo da avere una base stabile per poi concentrarsi sull'implementazione di nuove feature come la piantina citata nella sezione 4.7 del Capitolo 4. Tra i vari perfezionamenti c'è sicuramente anche un miglioramento di alcune parti dell'UI come ad esempio la Homepage che al momento potrebbe sembrare un po' spoglia e l'aggiunta di immagini significative che accompagnino le patologie in modo da dare un suggerimento immediato di cosa tratta ognuna di esse.

5.1.1 Backend proprio

Un possibile sviluppo futuro, nel caso l'applicazione venisse finanziata e supportata, è sicuramente lo sviluppo di un backend proprio ideato per l'applicazione stessa che fornirebbe una maggiore flessibilità rispetto a Qualtrics e permetterebbe operazioni come l'aggiunta di nuovi moduli (*per esempio abuso di sostanze, disturbi del comportamento alimentare, disturbi ossessivi compulsivi e così via*), nuovi screening, nuovi esercizi il tutto senza dover ricompilare l'applicazione poiché sarebbe possibile definire delle strutture dati adeguate a questo dominio e spostare alcune computazioni come quella dell'algoritmo di screening direttamente sul server; questo permetterebbe all'applicazione di offrire sempre nuove funzionalità. Inoltre, per quanto riguarda sondaggi e domande la struttura potrebbe rimanere pressoché invariata, mantenendo tutto il lavoro svolto su Qualtrics.

5.1.2 E-Coaches

Un'altra funzionalità aggiuntiva che può essere molto utile agli utenti di questa applicazione è la possibilità di avere un contatto diretto con le tutor che hanno creato i vari moduli o comunque con qualcuno che possa fornire loro consigli utili ed eventualmente indirizzarli verso un trattamento professionale se necessario. Il metodo ideale è quello di utilizzare una chat in tempo reale o comunque un sistema di comunicazione stile e-mail ma che mantenga l'anonimato degli utenti.

5.1.3 Integrazione in altri ambienti

Un altro possibile sviluppo per l'applicazione è l'integrazione di essa in diversi ambienti informatici: questo permetterebbe, per esempio, di poter offrire l'applicazione a diversi atenei e/o aziende che la rendano poi disponibile ai propri studenti/dipendenti, cosa che permetterebbe, oltre a migliorare le condizioni dei diretti interessati, di offrire a queste istituzioni la possibilità di avere un feedback diretto di come i loro clienti vivono il loro ambiente, potendo così migliorarlo.

Appendice

APPENDICE

A

NOTE PER GLI SVILUPPATORI

In questa appendice saranno forniti alcuni dettagli implementativi che possono essere utili per gli sviluppatori per continuare o migliorare il lavoro fatto fino ad ora. In particolare verranno elencati l'ambiente di sviluppo e i vari pacchetti utilizzati durante lo sviluppo.

A.1 Ambiente di sviluppo

Come ambiente di sviluppo è stato utilizzato **Visual Studio Code**¹ accompagnato dalle seguenti estensioni:

- **Flutter**: aggiunge il supporto al editing, refactoring, running e hot reloading di applicazioni flutter in Visual Studio Code.
- **Dart**: aggiunge il supporto al linguaggio Dart, utilizzato da Flutter, e alla formattazione automatica del codice secondo le linee guida di Google.
- **Bracket Pair Colorizer 2**: estensione che colora diversamente le varie coppie di parentesi in un sorgente; molto utile per sviluppare in Flutter in quanto creando diversi widget le parentesi diventano tante molto facilmente.

¹code.visualstudio.com

A.2 Pacchetti utilizzati

Nella Tabella A.1 sono elencati i vari pacchetti utilizzati dall'applicazione e la funzione che ognuno di essi svolge. Sono riportate anche le versioni dei vari pacchetti, nel caso qualcuno di essi subisca una *breaking change*. Per l'installazione basta aggiungere il pacchetto al file `pubspec.yaml` e lanciare il comando `flutter pub get`. Tutti i pacchetti sono disponibili presso `pub.dev/packages`.

Tabella A.1: Pacchetti utilizzati

NOME	FUNZIONE	URL
Provider 5.0.0	Permette di utilizzare i provider citati nella sottosezione 2.5.1 del Capitolo 2.	/provider
Shared Preferences 2.0.6	Utilizzato per poter scrivere dati sulla memoria di massa del dispositivo.	/shared_preferences
Flutter Native Splash 1.2.0	Permette di creare uno splash screen nativo che viene mostrato mentre viene caricato il framework di Flutter.	/flutter_native_splash
Drag And Drop Lists 0.3.2	Permette di creare delle liste con supporto al trascinamento e riordinamento degli elementi anche tra liste diverse.	/drag_and_drop_lists
Better Player 0.0.72	Player video adattabile che implementa diversi controlli sullo stream video.	/better_player
Youtube Player iframe 2.1.0	Permette di riprodurre video da YouTube tramite iframe senza bisogno di chiave API.	/youtube_player_iframe

AudioPlayers 0.19.1	Plugin di basso livello che permette di riprodurre file audio, i controlli sono stati creati da zero assegnando le funzioni esposte.	/audioplayers
http 0.13.3	Permette di creare richieste http.	/http
html 0.15.0	Parser html, utile per estrarre il parametro src dai media del Question Text.	/html
Simple Html CSS 3.0.1	Traduce le proprietà CSS relative al testo in widget RichText , permettendo di replicare in app i testi editati in un rich text editor.	/simple_html_css
Table Calendar 3.0.1	Calendario provvisto di eventi. Permette di renderizzare i vari widget personalmente con vari builder esposti.	/table_calendar
intl 0.17.0	Internazionalizzazione e localizzazione, usato principalmente per il calendario.	/intl
Intro Views Flutter 3.2.0	Pacchetto usato per creazione rapida di schermate di introduzione.	/intro_views_flutter
Google Fonts 2.1.0	Permette di utilizzare i font gratuiti di Google Fonts in app.	/google_fonts

BIBLIOGRAFIA

- [1] Dott.ssa Martina Spelta. *Depressione e università: gli interventi mirati dei Servizi Clinici della SFU Milano.* (Visitato Maggio 2021). URL: www.milano-sfu.it/depressione-e-universita-gli-interventi-mirati-dei-servizi-clinici-della-sfu-milano/ (cit. a p. 1).
- [2] C Barr Taylor e Kristine H Luce. «Computer and internet-based psychotherapy interventions». In: *Current directions in psychological science* (). (Visitato Giugno 2021). URL: mental.jmir.org/2018/3/e10200?utm_source=TrendMD&utm_medium=cpc&utm_campaign=JMIR_TrendMD_1 (cit. a p. 3).
- [3] Ash Turner. *How Many Smartphones Are In The World?* (Visitato Maggio 2021). URL: www.bankmycell.com/blog/how-many-phones-are-in-the-world (cit. alle pp. 3, 4).
- [4] Wikipedia. *mHealth*. (Visitato Maggio 2021). URL: en.wikipedia.org/wiki/mHealth#Definitions (cit. a p. 4).
- [5] Wikipedia. *Cognitive behavioral therapy*. (Visitato Maggio 2021). URL: en.wikipedia.org/wiki/Cognitive_behavioral_therapy (cit. a p. 5).
- [6] Codecademy. *MVC: Model View Controller*. (Visitato Giugno 2021). URL: <https://www.codecademy.com/articles/mvc> (cit. a p. 15).
- [7] Interview Cake. *Hash Table Data Structure*. (Visitato Luglio 2021). URL: www.interviewcake.com/concept/java/hash-map (cit. a p. 28).
- [8] Wikipedia. *Beck Depression Inventory*. (Visitato Luglio 2021). URL: https://en.wikipedia.org/wiki/Beck_Depression_Inventory#BDI-II (cit. a p. 60).

- [9] Wikipedia. *State-Trait Anxiety Inventory*. (Visitato Luglio 2021). URL: https://en.wikipedia.org/wiki/State-Trait_Anxiety_Inventory#Scoring (cit. a p. 61).
- [10] Wikipedia. *Pittsburgh Sleep Quality Index*. (Visitato Luglio 2021). URL: https://en.wikipedia.org/wiki/Pittsburgh_Sleep_Quality_Index#Scoring_and_interpretation (cit. a p. 62).
- [11] Laura Montanari. *Brief Pain Inventory*. (Visitato Luglio 2021). URL: <https://www.fisioscience.it/blog/brief-pain-inventory-bpi/> (cit. a p. 62).
- [12] Fiorilli C., Galimberti V., De Stasio S., Di Chiacchio C. e Albanese O. *L'utilizzazione dello School Burnout Inventory (SBI) con studenti italiani di scuola superiore di primo e secondo grado*. (Visitato Luglio 2021). URL: https://www.researchgate.net/publication/267099535_L'_utilizzazione_dello_School_Burnout_Inventory_SBI_con_studenti_italiani_di_scuola_superiore_di_primo_e_secondo_grado (cit. a p. 63).
- [13] Wikipedia. *Columbia Suicide Severity Rating Scale*. (Visitato Luglio 2021). URL: https://en.wikipedia.org/wiki/Columbia_Suicide_Severity_Rating_Scale (cit. a p. 64).
- [14] Lausi G., Quaglieri A., Burrai J., Mari E. e Giannini A. *Development of the DERS-20 among the Italian population: a study for a short form of the Difficulties in Emotion Regulation Scale*. (Visitato Luglio 2021). URL: https://www.researchgate.net/publication/343988837_Development_of_the_DERS-20_among_the_Italian_population_a_study_for_a_short_form_of_the_Difficulties_in_Emotion_Regulation_Scale (cit. a p. 64).
- [15] Wikipedia. *Universally unique identifier*. (Visitato Luglio 2021). URL: [https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_2_\(date-time_and_MAC_address,_DCE_security_version\)](https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_2_(date-time_and_MAC_address,_DCE_security_version)) (cit. a p. 66).