

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



CORSO DI LAUREA IN INFORMATICA

**KARYA reports: realizzazione grafica di
un'applicazione Android per la
segnalazione di incendi boschivi**

Relatore:

Ch.ma Prof. Giuliana Vitiello

Candidata:

Francesca Pia Perillo

0512105563

ANNO ACCADEMICO 2019/2020

*Ai miei nonni,
ovunque voi siate.*

Indice

Elenco delle figure	4
1 Introduzione	7
2 La progettazione di KARYA reports	10
2.1 Delineazione dei profili utente	10
2.2 Delineazione dei task con la stesura degli scenari e dei casi d'uso	15
2.2.1 Segnalazione senza previa registrazione	15
2.2.2 Segnalazione previa registrazione	16
2.2.3 Segnalazione mediante il Bot di Telegram	17
2.3 Mock-up iniziali	18
2.3.1 Introduzione alle tecniche per la valutazione dell'usabilità	20
2.3.2 Tecnica del Mago di Oz	21
2.3.3 Valutazione del prototipo: cognitive walkthrough	23
2.3.4 Conclusioni scaturite dalla prima valutazione dell'usabilità	26
2.4 Evoluzione dei Mock-up iniziali	27
2.4.1 Tecnica del Mago di Oz	30
2.4.2 Valutazione del prototipo: cognitive walkthrough	31
2.4.3 Conclusioni scaturite dalla seconda valutazione dell'usabilità	35
3 Verso la realizzazione finale di KARYA reports	37
3.1 La scelta dei colori	37
3.2 La scelta del logo	39
3.3 Descrizione dei pattern utilizzati	40
3.4 Struttura del progetto in Android Studio	42
3.5 Feedback di KARYA reports	55
4 Conclusioni	69

Bibliografia	71
---------------------	-----------

Elenco delle figure

2.1	Schermate di accesso, di registrazione e di modifica dei dati	19
2.2	Schermate per la segnalazione di un incendio	19
2.3	Raffigurazione del cognitive walkthrough su task 7.2.1	24
2.4	Raffigurazione del cognitive walkthrough su task 7.2.2	25
2.5	Schermate di accesso e di registrazione a KARYA reports (rivisitazione dei primi prototipi)	28
2.6	Schermate di modifica dei dati (rivisitazione dei primi prototipi)	29
2.7	Schermate per la segnalazione di un incendio (rivisitazione dei primi prototipi)	29
2.8	Schermata di home page senza registrazione con modifiche aggiuntive	36
3.1	Palette della prima implementazione grafica di KARYA reports	37
3.2	Bozza della prima implementazione grafica di KARYA reports	38
3.3	Palette della seconda implementazione grafica di KARYA reports	39
3.4	Logo di KARYA reports	39
3.5	Pattern go back to a safe place	41
3.6	Pattern breadcrumbs	41
3.7	Pattern accordion	42
3.8	Struttura base di un progetto in Android Studio	43
3.9	Schermate di avvio e di prima apertura di KARYA reports . .	44
3.10	Esempio grafico della schermata del tutorial	45
3.11	Prime tre schermate del tutorial	48
3.12	Schermate di accesso e di registrazione a KARYA reports . . .	51
3.13	Home page con login effettuato	51
3.14	Schermate atte alla segnalazione di un incendio	52
3.15	Schermate per la modifica dei dati	55
3.16	A sinistra un Toast con il layout di default messo a disposizione dalla classe Toast in Android Studio. A destra un Toast personalizzato di KARYA reports.	56
3.17	Esempio grafico del funzionamento del metodo changeColor() .	59

3.18 Esempio di una finestra di dialogo di default	60
3.19 Finestre di dialogo utilizzate in KARYA reports	61
4.1 Mappa per la visualizzazione in Sala Operativa degli incendi segnalati dai singoli cittadini	70

Elenco dei codici

3.1	Esempio di configurazione dell'orientamento	43
3.2	Tutorial.java	46
3.3	home_page_no_login.xml	48
3.4	HomePageNoLogin.java	49
3.5	Metodo showPassword nella classe Utility.java	49
3.6	prop_rounded_button.xml	50
3.7	prop_segnala_incendio.xml	50
3.8	home_page_with_login.xml	51
3.9	user_profile.xml	53
3.10	UserProfile.java	54
3.11	custom_toast_error.xml	57
3.12	metodo newCustomToast() nella classe Utility.java	58
3.13	metodo changeColor() nella classe FirstRegistrationPage.java .	59
3.14	custom_popup_recovery_password.xml	61
3.15	custom_popup_confirm_operation.xml	63
3.16	HomePageNoLogin.java	64
3.17	TakePhoto.java	67

Capitolo 1

Introduzione

Negli ultimi anni il problema degli incendi boschivi [1] ha assunto dimensioni a dir poco drammatiche, tanto da destare un grido di preoccupato allarme all'intera popolazione mondiale. Il 21 novembre del 2000 la Camera dei Deputati ed il Senato della Repubblica hanno approvato la *Legge-quadro* in materia di incendi boschivi. Il *Capo 1* di tale legge batte sulla previsione, prevenzione e lotta attiva a questo problema. Focalizzando l'attenzione sul panorama italiano, nel decennio passato si sono perduti, per detta causa, più di 500 mila ettari di bosco. L'azione di rimboschimento¹e di ricostituzione boschiva non sono purtroppo riuscite a rimediare alle recenti devastazioni. Ogni anno, quasi a scadenze prestabilite, si ripete questo gravissimo problema, con ingentissimi danni sia economici che ecologici. Sono due le azioni che principalmente riescono a limitare tale danno: l'azione di *prevenzione* e quella di *segnalazione*. Ad oggi l'unica arma che il cittadino dispone per segnalare un incendio è comporre il numero verde. Questa modalità però, procura dati ridondanti e di poca precisione.

Nasce quindi la necessità di pensare ad un nuovo metodo di segnalazione che possa rappresentare un passo avanti sia per il singolo cittadino che per la Sala Operativa dei Vigili del Fuoco. Al fine di comprendere al meglio quale potrebbe essere una possibile soluzione a questo ingente problema, è stato di supporto disporre di un approccio partecipativo ottenuto mediante l'integrazione con un Vigile del Fuoco esperto. Il suo contributo ha permesso di evidenziare tutte le problematiche attualmente presenti per quanto riguarda la fase di segnalazione degli incendi. Grazie alla sua esperienza in questo campo, è stato appurato che migliorare il coinvolgimento delle comunità locali, attraverso un sistema software che possa essere utilizzato da tutti, sarebbe

¹Nelle scienze forestali il termine *rimboschimento* indica il processo con cui una zona da tempo priva di vegetazione o precedentemente non boscata viene ricoperta da alberi e arbusti adatti a quella zona, che di norma sono le specie autoctone.

una svolta cruciale per ridurre gli incendi e limitare notevolmente i loro danni. Inoltre tale sistema migliorerebbe il lavoro svolto dalla Sala Operativa, in quanto si contribuirebbe a delle segnalazioni più precise da parte del singolo cittadino. Alla base di questo, il lavoro di tesi concerne nella realizzazione di **KARYA reports**: un'applicazione che intende offrire un nuovo metodo per la segnalazione degli incendi. Con l'aiuto di KARYA reports tutti i cittadini che hanno compiuto la maggiore età, muniti di uno smartphone Android, potranno segnalare facilmente un incendio, mediante pochi click. KARYA reports chiede l'inserimento di poche informazioni atte alla segnalazione, quali: la posizione, l'inserimento facoltativo di una fotografia e una lista di ulteriori informazioni che possono essere d'aiuto per comprendere la gravità della situazione. Questi dati saranno un elemento di focale importanza in ricezione, in quanto offriranno alla Sala Operativa dei Vigili del Fuoco una completa panoramica dell'incendio. È bene notare che l'ottenimento della posizione mediante il GPS rende la segnalazione più precisa rispetto ai metodi attualmente in uso.

KARYA reports vede la realizzazione di due tesi separate. Nel corso di questo lavoro di tesi effettueremo un focus accurato su quelli che sono i passi che hanno portato alla progettazione grafica di KARYA reports; mentre in un secondo lavoro di tesi, intitolato "*KARYA reports: un'applicazione per la segnalazione di incendi boschivi*" stilato dalla studentessa Mariarosaria Esposito, verranno discusse le problematiche relative alla programmazione lato backend dell'applicazione proposta.

Nel 1989, lo scienziato *Don Norman* pubblicò "*The Psychology of Everyday Things*", ipotizzando che qualsiasi errore un utente commetta quando utilizza un'interfaccia è colpa del design, non dell'utente stesso. Una buona *User Interface* (in italiano, interfaccia utente) attira l'attenzione degli utenti e si traduce in una diminuzione del tempo di apprendimento delle funzionalità e di un aumento del tempo di interazione con l'applicazione. Quindi, il design di un qualsiasi applicativo è un fattore altamente determinante per la buona riuscita di un'applicazione. Ciò è dovuto al fatto che, quando viene scaricato un applicativo software per la prima volta su un dispositivo, l'attenzione si rivolge in primo luogo allo stile di progettazione, ai colori e alle animazioni. Queste sono le ragioni che rendono fondamentale il lavoro di progettazione delle interfacce. Una buona interfaccia, infatti, rende piacevole l'esperienza del singolo utente andando a migliorare quella che è la *User Experience*. Lo scopo della progettazione del design è quello di creare un'interfaccia usabile, che sia al contempo facile da comprendere, efficace, efficiente e sia capace di fornire all'utilizzatore un sentimento positivo. "*Good design is good business*" (in italiano: un buon design è un buon affare) è il motto alla base della storia di una delle più antiche aziende informatiche del mondo: la *IBM*.

- *International Business Machines Corporation.* Ciò significa che il design deve essere funzionale e piacevole, non deve rappresentare solo una questione di gusti personali. È indispensabile avere sempre bene chiaro che l'utente ha solo una necessità: svolgere un'attività ottenendo il risultato sperato nel minor tempo possibile e con facilità. Tutto ciò che non è strettamente legato a questo scopo, non è indispensabile. Tale concetto risulta maggiormente veritiero quando si muovono i primi passi verso la realizzazione di un nuovo prodotto. Un'applicazione avrà tanto più successo quanto più semplice sia la sua interfaccia, quando un'azione risulta talmente integrata con il contesto da sembrare naturale. L'utente deve aver l'impressione che la soluzione apportata sia l'unica possibile, la più semplice, e che non ci sia altra via per offrire quella funzionalità.

Per quanto concerne il seguente lavoro di tesi, verranno discusse tutte le motivazioni che hanno portato a fare determinate scelte di progettazione invece che altre.

Nello specifico, nel *Capitolo 2* verranno definiti i profili utente, gli scenari e i casi d'uso sui task ritenuti di maggiore interesse. Verranno spiegate le due tecniche che hanno permesso la valutazione dell'usabilità e descritti i mock-up iniziali, sui quali verrà effettuata tale valutazione in modo da cercare di migliorare quanto più possibile la fase di prototipazione.

Nel *Capitolo 3* ricadrà l'attenzione sulle motivazioni che hanno portato alla scelta dei colori e del logo. Verranno descritti i pattern utilizzati all'interno di KARYA reports. Si effettuerà anche una discussione sulla struttura del progetto nell'ambiente di sviluppo Android Studio, focalizzando l'attenzione anche sui feedback forniti all'utente dall'applicazione proposta.

Nel *Capitolo 4* verranno infine stilate le conclusioni. In queste ultime verrà mostrata un'anteprima di come le segnalazioni raccolte aiuteranno la Sala Operativa attraverso l'ausilio di un Mappa.

Capitolo 2

La progettazione di KARYA reports

In questo capitolo verranno descritti i *profili utente* e i *task*¹di maggiore interesse. Verrà inoltre effettuata la stesura degli *scenari* e dei relativi *casi d'uso*. I primi sono una narrazione di una possibile storia che evidenzia come il sistema proposto possa migliorare alcune tipologie di situazioni. I secondi sottolineano l'interazione fra l'utente e il sistema che si vuole progettare. Scrivere scenari e casi d'uso aiuta ad immaginare come si deve comportare un nuovo prodotto e ad individuarne correttamente i requisiti. In questo caso si vedranno protagonisti gli stessi attori della delineazione dei profili utente. L'utilizzo di tali metodologie rappresentano uno valido strumento di comunicazione tra i vari stakeholder² e risultano inoltre utili per collocare il prodotto nei suoi possibili contesti d'uso. Successivamente verranno introdotti i vari approcci utilizzati per la valutazione dell'usabilità, ovvero la tecnica del mago di Oz e quella del *Cognitive Walkthrough*. Verranno poi descritte le realizzazioni dei vari mock-up iniziali di progetto. Questi ultimi, una volta applicate le varie tecniche al fine di verificarne l'usabilità, subiranno dei cambiamenti atti migliorare la futura interazione degli utenti su di essi.

2.1 Delineazione dei profili utente

Un *profilo utente* è una fonte di dati relativa a tutte le informazioni dell'utente che possono essere impiegate per determinare il comportamento del sistema. La delineazione dei profili utente è un passaggio fondamentale in primo luogo

¹Nell'informatica il termine *task* indica il compito specifico da svolgere.

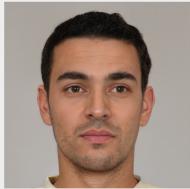
²Ciascuno dei soggetti direttamente o indirettamente coinvolti nel progetto.

per comprendere quelli che saranno i requisiti dell'applicazione, in secondo luogo per evitare di modellare il futuro prodotto sui bisogni dei progettisti. È perciò molto importante che i profili utente vedano protagoniste persone che, anche se fintizie, rappresentino un particolare tipo di utenza.

KARYA reports si occuperà nello specifico della segnalazione degli incendi da parte del cittadino. Per tanto sono stati delineati tre profili utente, composti da una fotografia del personaggio e una descrizione atta a definire un aspetto della vita di ognuno, che potrebbe essere migliorato mediante l'utilizzo di un applicativo software. La costruzione di tali profili nasce sulla base di ciò che si è compreso mediante le interviste effettuate al Vigile del Fuoco. Per mantenere il rispetto della privacy tutti i nomi utilizzati sono fintizi e tutte le foto dei personaggi sono state scaricate dal sito *Generated Photo* [4]. Questo sito utilizza una tecnologia secondo la quale, dando in input ad un algoritmo i volti di 69 modelli (volontari e d'accordo) in 29'000 foto scattate in studio, vengono generati i volti di 100'000 persone inesistenti ma al contempo realistiche.

A seguire verranno dunque riportati i *profili utente*, con le relative *checklist*. Queste ultime sono state realizzate con lo scopo di avere un'idea ancora più chiara dell'utente finale dell'applicazione, infatti, verranno prese in considerazione tutte le caratteristiche dell'utente tipo. Focalizzeremo l'attenzione sulle peculiarità di carattere psicologico e fisico, sulla conoscenza e l'esperienza e sulle caratteristiche di lavoro e task.

Profilo utente e Checklist: Antonio Cuozzolo



Antonio Cuozzolo è un giovane ragazzo di 28 anni della provincia di Salerno. Ha studiato fisica all'Università e sta cercando di coronare il suo sogno lavorativo: diventare un astrofisico. Ha tante passioni ed è sempre stato amante della natura, in tutte le sue sfaccettature. Da qualche anno ha deciso di iscriversi all'associazione Legambiente per tutelare la sua amata natura. Purtroppo nella sua zona sono molto diffusi gli incendi boschivi, per questo motivo Antonio vorrebbe avere un aiuto per la segnalazione di questi ultimi.

Caratteristiche psicologiche	
<i>Stile cognitivo</i>	sistematico e spaziale
<i>Attitudine</i>	positivo
<i>Motivazione</i>	alta
Conoscenza ed esperienza	
<i>Livello di lettura</i>	oltre il 12°
<i>Esperienza battitura</i>	alta
<i>Istruzione</i>	università
<i>Esperienza con il sistema</i>	novizio
<i>Esperienza sul task</i>	intermedio
<i>Esperienza di applicazioni</i>	nessun sistema simile
<i>Linguaggio</i>	italiano
<i>Uso di altri sistemi</i>	nessuno
<i>Conoscenza informatica</i>	medio alta
Caratteristica di lavoro e task	
<i>Frequenza d'uso</i>	alta
<i>Addestramento</i>	formale facoltativo
<i>Uso del sistema</i>	discrezionale
<i>Carattere di job</i>	disoccupato
<i>Turnover</i>	-
<i>Altri supporti</i>	telefono
<i>Importanza del task</i>	alta
<i>Complessità del task</i>	media
Caratteristiche fisiche	
<i>Distinzione dei colori</i>	si
<i>Predominanza</i>	destro
<i>Sesso</i>	uomo

Profilo utente e Checklist: Eugenio Pontale



Eugenio Pontale è un pensionato di 70 anni, della provincia di Salerno. Fin da piccolo ha sempre cercato di aiutare chiunque, voleva seguire le orme di suo padre e diventare un Vigile del Fuoco. Per questo ha deciso di provare vari concorsi e all'età di 24 anni è entrato a far parte del Corpo Nazionale dei Vigili del Fuoco. Eugenio comprende le difficoltà dei suoi ex colleghi, per questo vorrebbe migliorare la qualità delle segnalazioni degli incendi mediante l'utilizzo di uno strumento alternativo alla chiamata del numero verde.

Caratteristiche psicologiche	
<i>Stile cognitivo</i>	analitico e riflessivo
<i>Attitudine</i>	positivo
<i>Motivazione</i>	alta
Conoscenza ed esperienza	
<i>Livello di lettura</i>	oltre il 12°
<i>Esperienza battitura</i>	alta
<i>Istruzione</i>	università
<i>Esperienza con il sistema</i>	novizio
<i>Esperienza sul task</i>	intermedio
<i>Esperienza di applicazioni</i>	nessun sistema simile
<i>Linguaggio</i>	italiano
<i>Uso di altri sistemi</i>	nessuno
<i>Conoscenza informatica</i>	medio alta
Caratteristiche di lavoro e task	
<i>Frequenza d'uso</i>	alta
<i>Addestramento</i>	formale facoltativo
<i>Uso del sistema</i>	discrezionale
<i>Caratteregorie di job</i>	pensionato
<i>Turnover</i>	-
<i>Altri supporti</i>	telefono
<i>Importanza del task</i>	alta
<i>Complessità del task</i>	media
Caratteristiche fisiche	
<i>Distinzione dei colori</i>	si
<i>Predominanza</i>	ambidestro
<i>Sesso</i>	uomo

Profilo utente e Checklist: Giovanna Amodio



Giovanna Amodio è una ragazza di 21 anni, della provincia di Salerno. La sua più grande paura è il fuoco ed è molto attiva riguardo le notizie degli incendi boschivi. Non molto tempo fa le capitò di assistere ad un evento del genere. Chiamò immediatamente il numero verde dei Vigili del Fuoco per avviare una segnalazione. Ebbe però delle difficoltà nell'individuare la posizione dell'incendio. Giovanna vorrebbe uno applicativo che possa aiutare l'utente nella segnalazione dell'incendio, mettendo a disposizione strumenti di aiuto per determinare la posizione.

Caratteristiche psicologiche	
<i>Stile cognitivo</i>	impulsivo e verbale
<i>Attitudine</i>	positivo
<i>Motivazione</i>	alta
Conoscenza ed esperienza	
<i>Livello di lettura</i>	oltre il 12°
<i>Esperienza battitura</i>	media
<i>Istruzione</i>	università
<i>Esperienza con il sistema</i>	novizio
<i>Esperienza sul task</i>	intermedio
<i>Esperienza di applicazioni</i>	nessun sistema simile
<i>Linguaggio</i>	italiano/ inglese/ francese
<i>Uso di altri sistemi</i>	nessuno
<i>Conoscenza informatica</i>	medio alto
Caratteristica di lavoro e task	
<i>Frequenza d'uso</i>	alta
<i>Addestramento</i>	formale facoltativo
<i>Uso del sistema</i>	discrezionale
<i>Carattere di job</i>	disoccupato
<i>Turnover</i>	-
<i>Altri supporti</i>	telefono
<i>Importanza del task</i>	alta
<i>Complessità del task</i>	bassa
Caratteristiche fisiche	
<i>Distinzione dei colori</i>	no
<i>Predominanza</i>	destro
<i>Sesso</i>	donna

2.2 Delineazione dei task con la stesura degli scenari e dei casi d'uso

KARYA reports, come già accennato in precedenza, permette la segnalazione veloce di incendi boschivi, per tanto i task ritenuti di maggiore interesse sono tre:

- *Task 4.1*: segnalazione dell'incendio senza essersi precedentemente registrati all'applicazione;
- *Task 4.2*: segnalazione con una prima registrazione effettuata antecedentemente la segnalazione dell'incendio;
- *Task 4.3*: segnalazione dell'incendio mediante l'utilizzo di uno specifico Bot di Telegram predisposto alle segnalazioni di *KARYA reports*.

2.2.1 Segnalazione senza previa registrazione

Task 4.1: Segnalazione di un incendio con applicazione non ancora scaricata e senza registrazione

Scenario 4.1: Eugenio Pontale è un ex Vigile del Fuoco. In uno dei suoi soliti giri in auto, nota del fumo provenire da un piccolo bosco alla sua destra. Per segnalare l'incendio in corso si collega ad una rete Wi-Fi comunale, e si reca sul Play Store per procedere all'installazione dell'applicazione.

Caso d'uso 4.1: Riportato nella tabella che segue.

	Utente	Sistema
1	Apre per la prima volta l'applicazione;	
2		Chiede all'utente se si vuole procedere con il tutorial iniziale;
3	Preme sul pulsante per iniziare il tutorial;	
4		Mostra una serie di schermate atte far comprendere all'utente il funzionamento dell'applicazione, poi mostra la home page dell'utente non loggato; <i>Continua nella prossima pagina</i>

	Utente	Sistema
5	Preme sul pulsante per registrarsi all'applicazione;	
6		Mostra la schermata di registrazione;
7	Completa i campi e si registra all'applicazione;	
8		Mostra la schermata di home page dell'utente loggato;
9	Preme sul pulsante per segnalare un incendio	
10		Mostra le varie schermate atte a segnalare l'incendio;
11	Completa la segnalazione dell'incendio;	
12		Mostra una schermata di corretto invio dei dati;

2.2.2 Segnalazione previa registrazione

Task 4.2: Segnalazione di un incendio con applicazione già scaricata e registrazione avvenuta

Scenario 4.2: Antonio Cuozzolo ha da poco scaricato l'applicazione per segnalare gli incendi boschivi poiché abita in una zona in cui questi sono molto diffusi. In una delle sue passeggiate nota un incendio molto vicino a lui, così prende il cellulare e avvia l'applicazione.

Caso d'uso 4.2: Riportato nella tabella che segue.

	Utente	Sistema
1		Mostra la schermata di home page dell'utente non loggato;
2	Accede all'applicazione con le sue credenziali;	
3		Mostra la schermata di utente loggato <i>Continua nella prossima pagina</i>

	Utente	Sistema
4	Preme sul pulsante per segnalare un incendio;	
5		Mostra le varie schermate atte a segnalare l'incendio;
6	Completa la segnalazione dell'incendio;	
7		Mostra una schermata di corretto invio dei dati.

2.2.3 Segnalazione mediante il Bot di Telegram

Task 4.3: Segnalazione di un incendio mediante il Bot di Telegram

Scenario 4.3: Giovanna Amodio è una giovane ragazza che, passeggiando con il suo ragazzo, nota un incendio non molto lontano da loro. Lei ha già scaricato in precedenza l'applicazione e si trova molto bene ad utilizzare il bot di Telegram per le sue segnalazioni. Per iniziare la sua segnalazione, quindi, apre la chat con il bot di Telegram e inizia la segnalazione.

Caso d'uso 4.3: Riportato nella tabella che segue.

	Utente	Sistema (Telegram)
1		Mostra un messaggio iniziale contenente la parola chiave da utilizzare per iniziare la segnalazione;
2	Inserisce la parola chiave all'interno della casella di testo e preme su invia	
3		Mostra i vari messaggi contenenti le parole chiave atte a segnalare l'incendio;
4	Inserisce le parole chiave all'interno della casella di testo e preme su invia;	
5		Mostra un messaggio di avvenuta segnalazione.

2.3 Mock-up iniziali

A seguito dei primi incontri con il Vigile del Fuoco e della descrizione di scenari e casi d'uso, è stata effettuata una prima prototipazione del sistema. Sono stati realizzati i primi *mock-up*, ovvero modelli approssimati dell'interfaccia del sistema che si desidera sviluppare. Secondo lo standard ISO 13407, i benefici di questo approccio sono molteplici. Tra i più importanti vi è il coinvolgimento degli utenti, partendo dalle fasi iniziali di progetto. I prototipi realizzati sono di tipo *interattivo*, in modo da consentire una facile interazione con il sistema in corso di progettazione, per sperimentarne l'uso. Per la realizzazione dei prototipi è stato usato Adobe XD, un'applicazione completa per tutte le esigenze, dalla creazione di wireframe³, mockup⁴ e altri strumenti di prototipazione, alla distribuzione di risorse agli sviluppatori. Dovendo effettuare l'analisi dei prototipi in modalità telematica, l'utilizzo di Adobe XD ha eluso questa difficoltà, facilitando il lavoro della *valutazione dell'usabilità*. Al fine di una chiara analisi, questi ultimi verranno riportati di seguito a questo capitolo. Le varie schermate sono state divise in tre sotto categorie, in base alle varie funzionalità. Per quanto riguarda le *schermate di accesso e registrazione* (*Figure 2.1 (a), (b)*), entrambe sono state pensate in maniera molto semplice e schematica. La prima *(a)* si compone di due caselle di testo nelle quali inserire rispettivamente l'email e la password al fine di accedere alle funzionalità dell'applicazione. La seconda *(b)* si compone di cinque caselle di testo nelle quali inserire i propri dati personali per poi proseguire con l'accesso al sistema. Le *schermate per la segnalazione di un incendio* (*Figure 2.2*) si compongono di una schermata principale *(a)* che presenta tre pulsanti: un primo per la modifica dei dati utente, un secondo per l'avvio della segnalazione mediante un apposito Bot di Telegram e un terzo posizionato centralmente per avviare la segnalazione. Per un corretto invio dei dati, è buona norma permettere all'utente di inserire informazioni circa l'intensità delle fiamme *(b)*, la posizione dell'incendio *(c)* e l'invio di una fotografia esplicita del suddetto *(d)*. Relativamente all'invio della posizione dell'incendio, si è pensato di inserire un marker che di default verrà visualizzato sulla mappa per indicare la posizione attuale, in modo da permettere un orientamento su di essa quanto migliore possibile. Per quanto

³Bozza grafica per creare gli scheletri dei propri progetti. Rappresenta la base dell'idea.

⁴Wireframe con l'aggiunta di grafica, colori e solitamente contenuti indicativi più realistici.

concerne la *schermata di modifica dei dati* (Figure 2.1 (c)), la logica di questa realizzazione rimane la stessa delle schermate di accesso e registrazione. Si compone, pertanto, di varie caselle di testo atte a permettere la modifica dei dati personali.

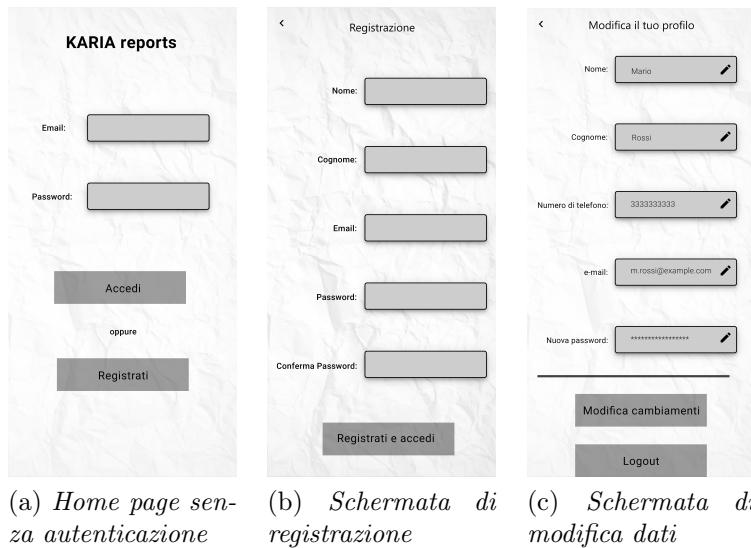


Figura 2.1: Schermate di accesso, di registrazione e di modifica dei dati

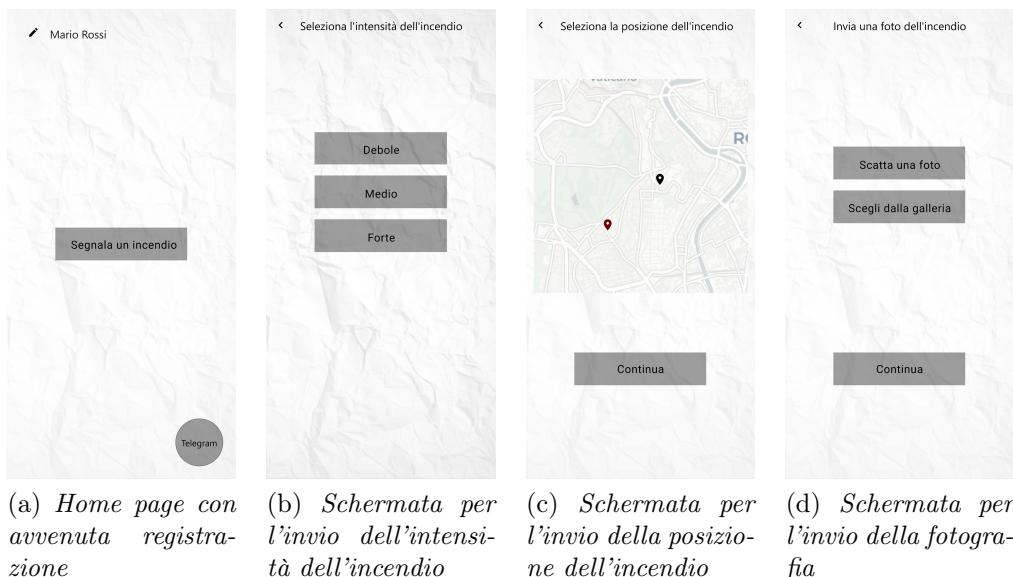


Figura 2.2: Schermate per la segnalazione di un incendio

2.3.1 Introduzione alle tecniche per la valutazione dell’usabilità

A seguito dei primi incontri con il Vigile del Fuoco e della descrizione di scenari e casi d’uso, sono stati strutturati i primi mock-up iniziali, descriventi l’interfaccia grafica del sistema. Per valutarne l’usabilità, è stata impiegata la tecnica del *Mago di Oz*. Quest’ultima prende nome dall’omonima favola e sfrutta come principio l’espeditivo utilizzato dal mago per manifestare i suoi trucchi agli occhi del pubblico. Così come nella storia il mago era nascosto nella macchina magica e l’azionava ad hoc per produrre l’effetto opportuno, si simula l’interazione dell’utente con il sistema grazie all’azione di un operatore che attua l’interazione in risposta all’input dell’utente. L’operatore osserva l’utente e le sue azioni; in base a queste adatta le risposte del sistema in tempo reale. L’utilizzo di questa metodologia, permette di valutare l’interazione ed il funzionamento di un prototipo nonostante il suo basso livello di definizione, prima di procedere con la realizzazione del sistema vero e proprio. I feedback che si otterranno saranno utili in futuro per migliorare i mock-up presentati ai partecipanti al test, al fine di ottenere un’interfaccia che possa migliorare l’esperienza del singolo utente.

Questa tecnica di valutazione verrà iterata due volte: una prima volta su quelli che sono i primi mock-up, realizzati conseguentemente alle fasi esplicative nei capitoli precedenti; una seconda volta sulle modifiche ai primi mock-up a seguito delle preferenze esposte dai partecipanti al test di usabilità.

Per avere una panoramica completa delle eventuali problematiche dei vari prototipi, ad ogni scelta dell’utente è stato chiesto se le sue aspettative fossero conformi all’output fornito dal sistema. Le loro risposte hanno aiutato a comprendere lo stato del *golfo di valutazione*. Questo permette di quantificare lo sforzo richiesto a una persona per interpretare lo stato fisico del sistema e confrontare le intenzioni con quello che è avvenuto.

È bene sottolineare che per effettuare la valutazione, è stato utilizzato il software di comunicazione Skype. Questa è stata una scelta necessaria in quanto, per esigenze dovute alle normative sul contenimento della diffusione del Covid-19, non è stato possibile effettuare incontri ravvicinati con i singoli partecipanti al test. Un altro importante motivo che ha influenzato la scelta di utilizzare Skype è legato alla facilità di installazione gratuita di quest’ultimo e alla possibilità di condivisione dello schermo, che ha permesso una facile interazione fra gli intervistati e i prototipi.

L’impiego della tecnica del Mago di Oz ha visto la partecipazione di cinque persone reali. Prima di dare inizio all’esperimento, è stato esposto loro il dominio del problema in una riunione generale, senza soffermarsi sull’aspetto grafico dell’applicazione. In questo modo ad ognuno sono state fornite

le medesime conoscenze di base. Per avere un'idea chiara di cosa stessero effettivamente pensando i partecipanti nel mentre del test, è stato adottato l'approccio *thinking aloud*⁵, in modo da permettere ai suddetti di sentirsi autorizzati di esprimere le loro opinioni in completa libertà. Terminata la prima breve presentazione del progetto KARYA reports, uno per volta ogni partecipante ha preso parte ad una apposita riunione personale, nella quale è stato proposto cadauno uno specifico compito (task) da portare a compimento.

Un'altra importante tecnica utilizzata per valutare l'usabilità del sistema proposto è il *Cognitive Walkthrough*. Questa non richiede la partecipazione di utenti, ma di uno o più esperti di valutazione. È una metodologia ispettiva per la valutazione dell'usabilità, che consiste nell'analizzare i passaggi richiesti per lo svolgimento di un compito (task), con lo scopo di individuare nell'interfaccia gli eventuali ostacoli che impediscono o rallentano il completamento del compito stesso. Si inizia definendo un compito che dovrà poi essere portato a termine mediante l'interazione con i prototipi. Le attività vengono suddivise in un semplice processo da seguire.

2.3.2 Tecnica del Mago di Oz

Nel seguente paragrafo, verrà analizzata la tecnica del mago di Oz attuata sui mock-up iniziali mostrati precedentemente.

Per motivi legati alla privacy non è stato possibile l'inserimento dei dati personali dei singoli partecipanti al test. Per questo motivo tutte le identità fornite di seguito a questo documento sono fittizie.

1° task proposto: registrarsi al sito e modificare il proprio nome una volta effettuato il login, successivamente eseguire il logout.

La prima persona ad essere intervistata è stata Marco Fernandi, un ragazzo di 21 anni iscritto all'Università degli Studi di Salerno. Egli, trovatosi di fronte la home page (*Figura 2.1 (a)*), ha esplorato la schermata e dopo qualche istante, ha espresso la volontà di premere sul pulsante "Registrati". Prima di proseguire con la navigabilità dei prototipi, gli è stato domandato quale risposta si aspettasse dal sistema e la sua descrizione è stata conforme a ciò che il prototipo seguente gli ha mostrato (*Figura 2.1 (b)*). L'interazione alla schermata di registrazione da parte di Marco è proseguita secondo

⁵Tecnica del “*pensiero ad alta voce*”. Nelle sessioni dei test di usabilità, l’utente viene incoraggiato ad agire e a verbalizzare tutti i pensieri che gli vengono in mente durante l'esecuzione dei compiti assegnatigli. Il thinking aloud consente di raccogliere misure qualitative relative alle strategie di soluzione del compito impartito.

le aspettative. L'unica anomalia riscontrata è stata la presenza sull'interfaccia di troppi oggetti grafici. Compilati i campi di registrazione, Marco ha espresso l'intenzione di premere sul pulsante "Registrati e accedi", per tanto è stato reindirizzato alla schermata di home page con avvenuta registrazione (*Figura 2.2 (a)*). Non è stato complesso per Marco individuare il pulsante che lo avrebbe poi portato alla schermata di modifica dei dati (*Figura 2.1 (c)*). Quest'ultima, essendo molto simile alla schermata di registrazione, non ha confuso l'intervistato. Dovendo seguire le indicazioni del task proposto, Marco ha poi espresso la decisione di premere sul pulsante "Logout" per terminare la sua esperienza.

2° task proposto: accedere al sito con delle credenziali già presenti nel database, successivamente iniziare la segnalazione di un incendio

Data la focale importanza della segnalazione degli incendi, questo task è stato proposto a due intervistati: Mariano Lamberti, lavoratore presso un'azienda informatica e Annarita Bonora, casalinga a tempo pieno. I due test sono avvenuti in momenti diversi, ma verranno analizzati di seguito parallelamente. Ad entrambi è stato detto di proseguire senza effettuare la registrazione, per tanto, trovandosi di fronte alla home page (*Figura 2.1 (a)*) e tenendo in considerazione la previa iscrizione a KARYA reports, entrambi hanno espresso la volontà di procedere premendo il pulsante "Accedi". Il prototipo che ha seguito la loro richiesta è risultato conforme alle aspettative di entrambi i partecipanti al test, che sono stati indirizzati alla schermata della home page con avvenuta registrazione (*Figura 2.2 (a)*). Mariano ha subito espresso il desiderio di premere sul pulsante di "Segnala un incendio", mentre Annarita è rimasta qualche secondo ad osservare la schermata per comprendere cosa fare, poi ha espresso la stessa richiesta di Mariano. Prima di proseguire con la visualizzazione del prototipo della schermata successiva (*Figura 2.2 (b)*), è stato chiesto loro di segnalare l'ultimo incendio visto o, in alternativa, di immaginarne uno. I due intervistati hanno preso due strade differenti: Annarita ha proseguito verso la segnalazione di un incendio di media intensità, Mariano ha deciso di segnalare un incendio di intensità debole. Entrambi sono stati reindirizzati alla schermata per l'invio della posizione dell'incendio (*Figura 2.2 (c)*). A questo punto sono state chieste le intenzioni di ognuno di fronte quest'ultima. Mariano ha esteriorizzato l'idea di comportarsi come farebbe con l'utilizzo di Google Maps sul proprio smartphone, successivamente di premere "Continua". Annarita è rimasta a riflettere per un minuto circa, poi ha pensato di proseguire con le stesse scelte esposte da Mariano. La schermata mostrata in successione, è stata quella dell'inserimento di una fotografia (*Figura 2.2 (d)*). Ancora una volta i due hanno preso scelte diverse:

Mariano ha deciso di premere su "Scegli dalla galleria", mentre Annarita ha optato per proseguire la sua esperienza premendo su "Scatta una foto". Non disponendo di prototipi atti a mostrare la conseguenza di tali scelte ad ognuno è stato spiegato oralmente come si sarebbe comportata KARYA reports. Entrambi hanno concluso l'esperimento spiegando la volontà di premere sul pulsante "Continua". Prima di terminare la sua esperienza, Mariano ha pensato che si troverebbe più in confidenza con l'applicazione se, una volta scelta o scattata la fotografia, questa comparisse a video prima di premere sul pulsante "Continua".

2.3.3 Valutazione del prototipo: cognitive walkthrough

Dopo aver stilato i tasks da analizzare, verranno riportati in una tabella le azioni effettuate dall'utente con le relative risposte del sistema. Successivamente, verrà riportato uno schema grafico numerato, per permettere una comprensione visiva dei vari passaggi messi in pratica per il compimento del task assegnato. I compiti proposti in questa fase sono due:

- *Task 1.1*: avendo già effettuato il login, modificare il nome ed effettuare il logout;
- *Task 1.2*: segnalare un incendio di grandi dimensioni, inserendo una foto scattata al momento della segnalazione.

Task 1.1: Avendo già effettuato il login, modificare il nome ed effettuare il logout.

<i>Azione 1</i>	L'utente preme sul pulsante che permette la modifica del profilo
<i>Risposta 1</i>	Il sistema mostra all'utente una schermata contenente i campi da modificare
<i>Azione 2</i>	L'utente effettua la modifica del nome e preme sul pulsante per effettuare i cambiamenti
<i>Risposta 2</i>	Il sistema mostra all'utente la stessa schermata con i cambiamenti effettuati
<i>Azione 3</i>	L'utente preme sul pulsante per effettuare il logout
<i>Risposta 3</i>	Il sistema mostra all'utente la schermata di accesso

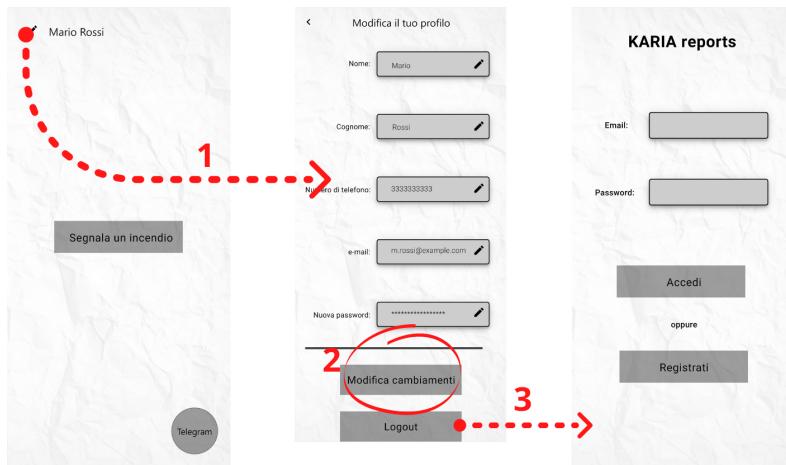


Figura 2.3: Raffigurazione del cognitive walkthrough su task 7.2.1

Task 1.2: Segnalare un incendio di grandi dimensioni, inserendo una foto scattata al momento della segnalazione.

Azione 1	L'utente preme sul pulsante che permette di segnalare un incendio
Risposta 1	Il sistema mostra all'utente una schermata contenente le varie intensità dell'incendio
Azione 2	L'utente sceglie di segnalare un incendio di dimensioni forti
Risposta 2	Il sistema mostra all'utente una schermata contenente una mappa
Azione 3	L'utente preme sulla mappa approssimativamente nel punto in cui visiona l'incendio
Risposta 3	Il sistema mostra all'utente un marker nel punto selezionato
Azione 4	L'utente preme sul pulsante che consente di continuare la segnalazione
Risposta 4	Il sistema mostra all'utente la schermata che permette l'inserimento della foto
Azione 5	L'utente seleziona la scelta che permette di scattare la foto
Risposta 5	Il sistema permette all'utente di scattare la foto
Azione 6	L'utente preme sul pulsante che consente di continuare la segnalazione
Risposta 6	Il sistema mostra all'utente la schermata iniziale

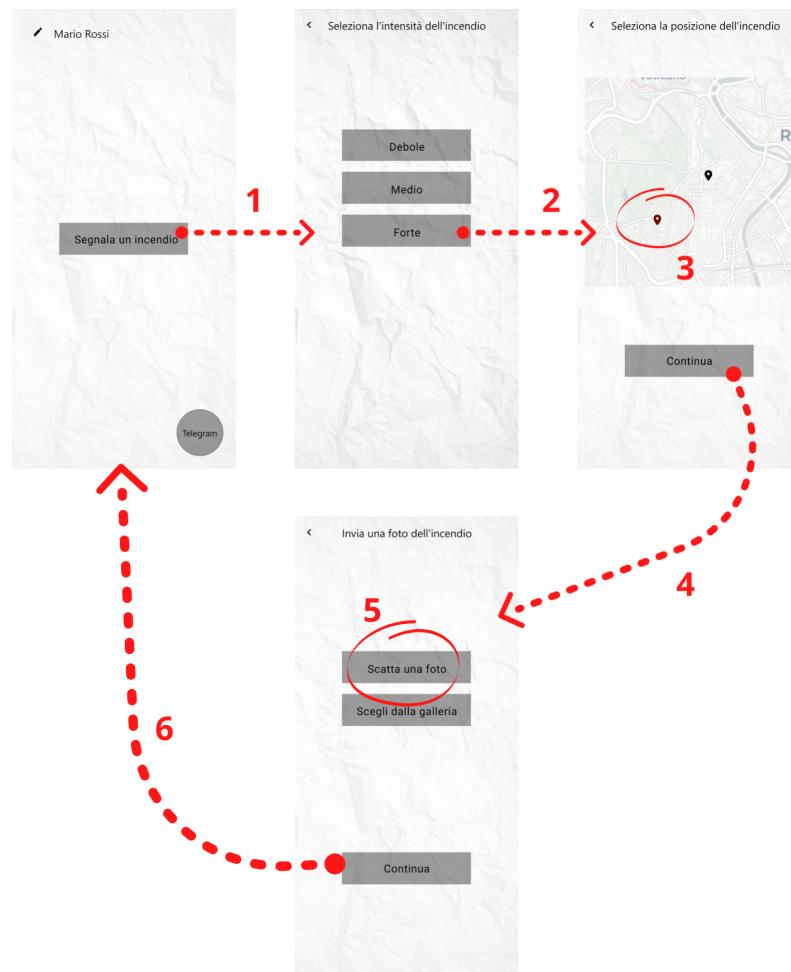


Figura 2.4: Raffigurazione del cognitive walkthrough su task 7.2.2

2.3.4 Conclusioni scaturite dalla prima valutazione dell’usabilità

Prima di tirare le conclusioni, stilando una lista di punti da migliorare all’interno dei primi prototipi, è stata effettuata una conferenza con il Vigile del Fuoco. Durante la riunione, particolare interesse è ricaduto sulla fase di registrazione (*Figura 2.1 (b)*) all’applicazione. In primo luogo, l’attenzione è ricaduta sugli attuali metodi di segnalazione, che avviene mediante la chiamata di un numero verde. Questa procedura determina la memorizzazione del numero di telefono di chi richiede soccorso. Per tanto nella fase di registrazione diventa necessario l’inserimento di una ulteriore casella di testo per la memorizzazione del suddetto. In secondo luogo, vi è la necessità di escogitare una procedura secondo la quale l’identità degli utilizzatori dell’applicazione sia verificata. Per tanto è stata proposta una registrazione a due fattori inviando una email contenente un codice di verifica alla stessa email inserita in fase di registrazione.

Effettuando una panoramica generale, questa prima iterazione sulla valutazione dell’usabilità ha evidenziato varie anomalie. A questo proposito, i successivi prototipi risulteranno migliorati secondo le seguenti specifiche:

- *dividere la registrazione in più schermate.* Questo in quanto il primo intervistato, durante lo svolgimento del primo task proposto, asserisce di aver riscontrato un disturbo nel vedere troppi oggetti grafici sull’interfaccia. A questo fattore, si unisce il dover inserire un ulteriore spazio per l’inserimento del numero di telefono all’interno della schermata di registrazione;
- *eliminazione della schermata della segnalazione dell’intensità dell’incendio.* Questo in quanto, durante lo svolgimento del secondo task proposto, Annarita e Mariano esprimono scelte diverse riguardo la potenza dell’incendio, ma nonostante ciò vengono reindirizzati alla stessa schermata. Alto motivo sostanziale che ha seguito la scelta è che l’intensità dell’incendio cambia radicalmente nel giro di pochi secondi, quindi risulta un dato ininfluente durante la segnalazione;
- *inserire uno spazio per la presa visione della foto durante la segnalazione, prima di inviarla.* Questo in quanto Mariano, prima di terminare il suo esperimento, ha espresso un parere che ha reso valida questa modifica. Durante l’ultima fase di segnalazione viene infatti richiesta una fotografia. Una volta inserita l’immagine però, non vi è alcuno spazio per visionarla prima di terminare la segnalazione;

- *inserimento di una ulteriore schermata per inviare informazioni addizionali riguardanti l'incendio.* Durante la segnalazione di un incendio è di particolare importanza inserire delle ulteriori informazioni utili per comprendere la gravità circa la posizione geografica dell'incendio;
- *aggiungere una ulteriore schermata che spieghi all'utente che la segnalazione effettuata ha avuto buon esito.* Il cognitive walkthrough ha evidenziato la problematica del possibile innalzamento del golfo di valutazione della schermata inerente all'invio della fotografia (*Figura 2.2 (d)*). Il pulsante denominato "Continua" infatti, rende poco evidente all'utente che la procedura di segnalazione stia per terminare.

2.4 Evoluzione dei Mock-up iniziali

L'analisi riportata nel precedentemente definisce un primo passo verso la reale progettazione grafica di KARYA reports. A seguito della prima valutazione dell'usabilità sono nati i nuovi prototipi. Di questi sono stati definiti, diversamente dai precedenti, colori e icone, per rendere quanto più vicina questa prototipazione allo sviluppo vero e proprio di KARYA reports. È stato utilizzato il colore blu. La decisione scaturisce da una ricerca, che dimostra come il blu sia un colore rilassante. Riporta al cielo e al mare e simboleggia la pace, la fiducia e la sicurezza. Anche in questo caso i prototipi realizzati sono di tipo *interattivo*, in modo da consentire una facile interazione con il sistema in corso di progettazione per sperimentarne l'uso. Per una chiara analisi, questi ultimi verranno riportati di seguito a questo capitolo. Le varie schermate sono state divise in tre sotto categorie, in base alle varie funzionalità. Per quanto riguarda le *schermate di accesso e registrazione* (*Figure 2.5*), entrambe sono state pensate in maniera molto semplice e schematica. La schermata di accesso (*Figura 2.5 (a)*) si compone di una breve presentazione iniziale dell'applicazione. Successivamente vi sono due caselle di testo nelle quali inserire rispettivamente l'email e la password al fine di accedere alle funzionalità dell'applicazione. In alternativa vi è la possibilità di registrarsi al sistema. La schermata di registrazione ha subito, rispetto ai precedenti mock-up proposti, una divisione in tre sotto schermate. Per rendere chiaro all'utilizzatore del sistema con quale delle tre sotto schermate stia interagendo, sono stati pensati dei selettori, composti da tre righe colorate in maniera differente. Per rendere chiaro questo concetto basti osservare le figure 2.5 (b), 2.5 (c) e 2.5 (d). La prima di queste ((2.5 (b)) si compone di tre caselle di testo nelle quali inserire una parte dei propri dati personali e un pulsante "Continua" che permette di proseguire con la registrazione. La seconda ((2.5

(c)), si compone di ulteriori caselle di testo per l'inserimento di maggiori dati. In fine, l'ultima parte della registrazione ((2.5 (d)) concerne l'inserimento di un codice di verifica inviato sull'indirizzo email specificato. Per quanto concerne la *schermata di modifica dei dati* (*Figura 2.6*), la logica di questa realizzazione rimane la stessa delle schermate di accesso e registrazione. Si compone, pertanto, di varie caselle di testo, atte a permettere la modifica dei dati personali. A differenza dei primi prototipi, è stato pensato ad un raggruppamento (*Figura 2.6 (b)*) fra i vari tipi di dati modificabili. Questo crea una divisione fra quelli che sono i dati personali dell'utente (*Figura 2.6 (c)*) e quelli che sono i dati di accesso all'applicazione (*Figura 2.6 (d)*). Le *schermate per la segnalazione di un incendio* (*Figura 2.7*) si compongono di tre semplici schermate atte ad guidare l'utente e un'ultima schermata che evidenzia la fine delle operazioni. Per un corretto invio dei dati, è buona norma permettere all'utente di inserire informazioni circa la posizione dell'incendio (*Figura 2.7 (a)*), l'invio di una fotografia esplicita del suddetto (*Figura 2.7 (b)*), l'inserimento di informazioni additive, atte a comprendere la gravità della situazione (*Figura 2.7 (c)*) e una schermata finale, con lo scopo di avvisare l'utente della conclusione della segnalazione (*Figura 2.7 (d)*).



Figura 2.5: Schermate di accesso e di registrazione a KARYA reports (riveditura dei primi prototipi)

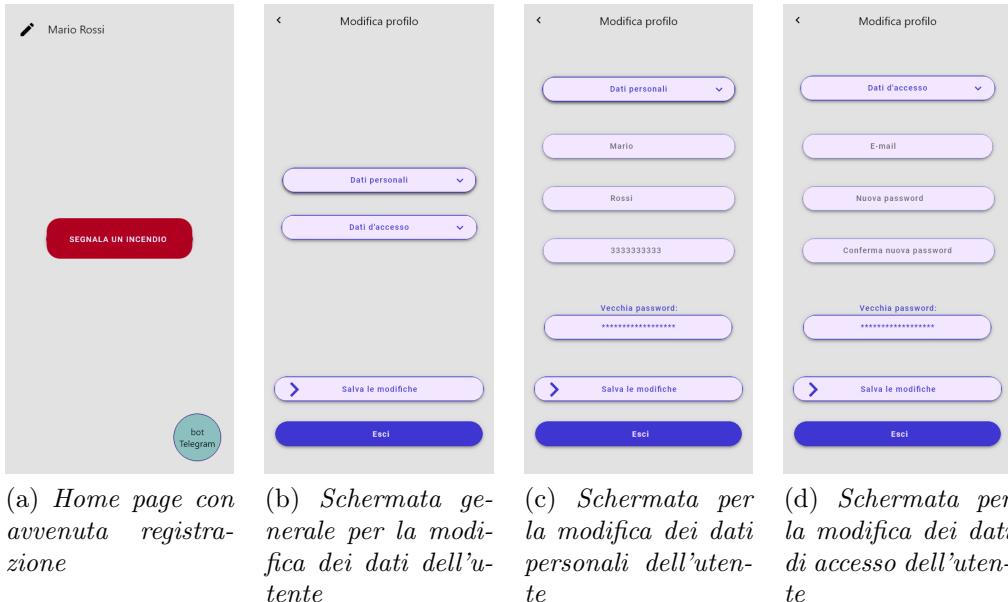


Figura 2.6: Schermate di modifica dei dati (rivisitazione dei primi prototipi)



Figura 2.7: Schermate per la segnalazione di un incendio (rivisitazione dei primi prototipi)

2.4.1 Tecnica del Mago di Oz

Nel seguente paragrafo, verrà analizzata la tecnica del mago di Oz attuata sui Mock-up mostrati precedentemente.

I task presentati in questa seconda iterazione della valutazione dell'usabilità sono rimasti gli stessi della prima, in quanto i prototipi hanno subito cambiamenti sostanziali per quanto riguarda le schermate di registrazione, presenti nello svolgimento del primo task, e le schermate di modifica dei dati e di segnalazione incendio, presenti nello svolgimento del secondo task.

Per motivi legati alla privacy non è stato possibile l'inserimento dei dati personali dei singoli partecipanti al test. Per questo motivo tutte le identità fornite di seguito a questo documento sono fittizie.

1° task proposto: registrarsi al sito e modificare il proprio nome una volta effettuato il login, successivamente eseguire il logout.

Lo svolgimento di questo compito vede protagonista Armando Bilillo, un operaio quarantenne in una fabbrica di cravatte. Egli, trovatosi di fronte la home page (*Figura 2.5 (a)*), ha espresso la volontà di premere sul pulsante "Registrati". Prima di proseguire con la navigabilità dei prototipi, gli è stato domandato quale risposta si aspettasse dal sistema e la sua descrizione è stata conforme a ciò che il prototipo seguente gli ha mostrato (*Figura 2.5 (b)*). Dopo aver compilato i campi relativi al nome, al cognome e alla email, ha espresso l'intenzione di premere su "Continua". La presenza dei divisorii nella parte superiore dello schermo, sotto la dicitura "Registrati", non ha confuso Armando, che si aspettava di dover inserire ancora altri dati prima di completare la registrazione. Dalla prima schermata di registrazione l'intervistato passa quindi alla seconda pagina (*Figura 2.5 (c)*), nella quale inserisce gli ulteriori dati richiesti. Una volta completate le form⁶, Armando afferma di voler cliccare sul pulsante "Continua". Tale azione lo reindirizza alla terza schermata di registrazione (*Figura 2.5 (d)*), nella quale finge di inserire un codice inviatogli precedentemente per email, in modo da completare l'operazione di registrazione. Prima di proseguire con il task, sono state poste domande all'intervisato circa la complessità di tali operazioni, le risposte di Armando sono state molto positive, in quanto le varie schermate gli risultano fluide e molto intuitive. Completata la prima parte del task, Armando viene reindirizzato alla schermata di home page con avvenuta registrazione (*Figura 2.7 (a)*). Non è stato complesso per Armando individuare il pulsante

⁶In informatica, l'interfaccia di un programma che consente a un utente di un sito web di inserire e inviare uno o più dati.

che lo avrebbe poi portato alla schermata di modifica dei dati (Figura 2.6 (a)). Il task propone all'intervistato di modificare il proprio nome, per tanto Armando decide di premere la sezione denominata "Dati personali", che rende visibili questi ultimi. Una volta modificati il candidato osserva la pagina proposta ed esprime la volontà di premere su "Salva le modifiche".

2° task proposto: accedere al sito con delle credenziali già presenti nel database, successivamente iniziare la segnalazione di un incendio

Il protagonista dello svolgimento di questo task è Cesare Glielmi, un ragazzo di 22 anni, iscritto all'Università degli Studi di Salerno. L'intervistato è stato avvisato di proseguire senza effettuare la registrazione, pertanto, trovatosi di fronte alla home page (*Figura 2.5 (a)*) e tenendo in considerazione la previa iscrizione a KARYA reports, egli ha espresso la volontà di procedere inserendo le proprie credenziali e successivamente premere sul pulsante "Accedi". Il prototipo seguente alla richiesta è risultato conforme alle aspettative del partecipante al test, che è stato reindirizzato alla schermata della home page con avvenuta registrazione (*Figura 2.7 (a)*). Cesare ha subito espresso il desiderio di premere sul pulsante di "Segnala un incendio". Prima di proseguire con la visualizzazione del prototipo della schermata successiva (*Figura 2.7 (b)*), è stato chiesto all'intervistato di segnalare l'ultimo incendio visto o, in alternativa, di immaginarne uno prima di proseguire con la segnalazione. Cesare ha deciso di segnalare un incendio nella città di Eboli, poi ha espresso la volontà di premere su "Continua" per proseguire la sua segnalazione. È stato quindi reindirizzato alla schermata che permette l'invio di una fotografia esplicativa del suddetto (*Figura 2.7 (c)*). Cesare ha deciso di premere su "Scegli dalla galleria". Non disponendo di prototipi atti a mostrare la conseguenza di tale scelta, è stato spiegato oralmente come si sarebbe comportata KARYA reports. Una volta compreso il funzionamento, l'intervistato ha esternato l'idea di premere su "Continua" e, trovatosi sulla pagina contenente le maggiori informazioni circa l'incendio da segnalare (*Figura 2.7 (d)*), Cesare ha selezionato le caselle "Zona industriale" e "Tralicci di alta tensione", prima di premere su "Continua" ed essere reindirizzato alla schermata di successo nell'invio della segnalazione (*Figura 2.7 (e)*). L'intervistato ha così concluso l'esperimento senza nessuna perplessità riguardo il task proposto.

2.4.2 Valutazione del prototipo: cognitive walkthrough

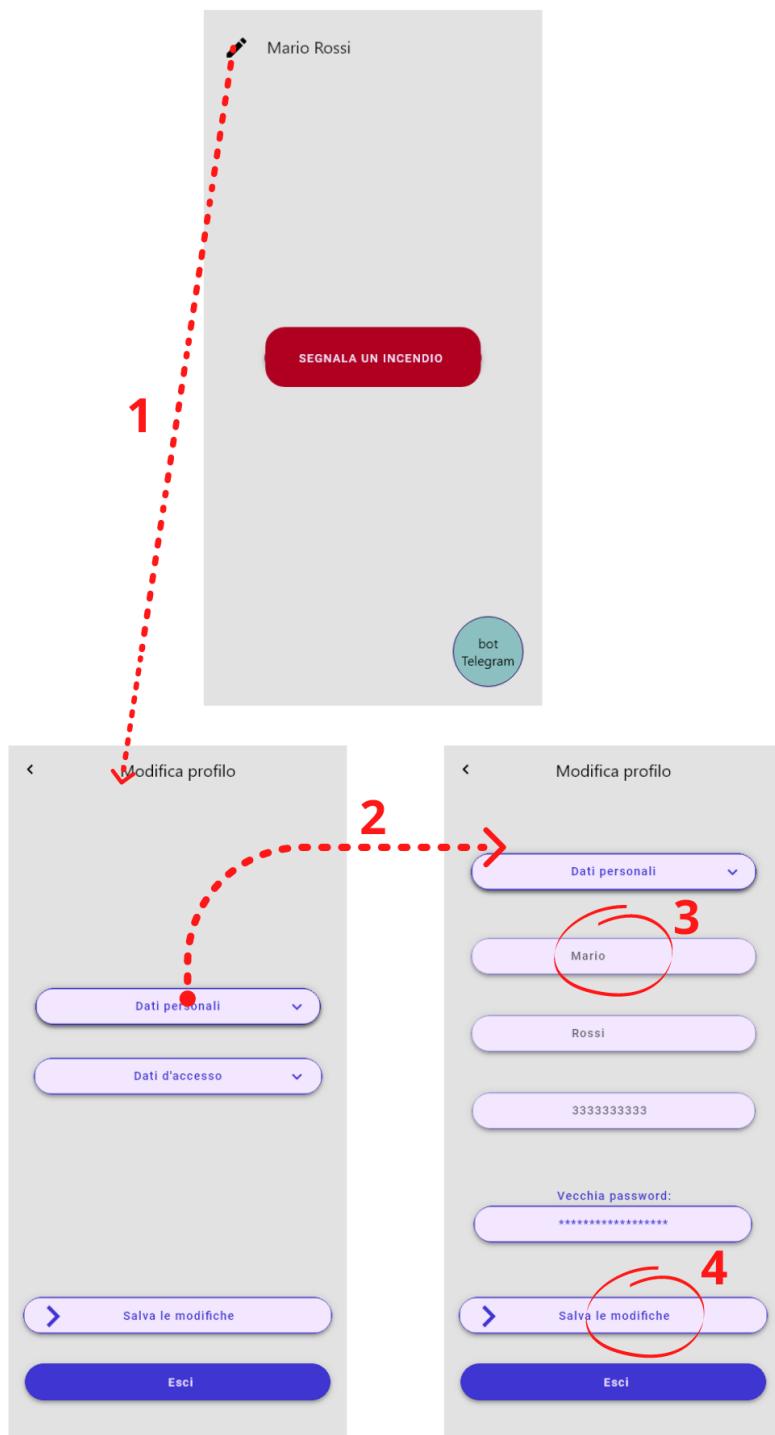
Dopo aver stilato i tasks da analizzare, verranno riportati in una tabella le azioni effettuate dall'utente con le relative risposte del sistema. Successivamente verrà riportato uno schema grafico numerato, per permettere una

comprendere visivamente i vari passaggi messi in pratica per il compimento del compito assegnato. I compiti proposti in questa fase sono due:

- *Task 9.2.1:* avendo già effettuato il login, modificare il nome ed effettuare il logout;
- *Task 9.2.2:* da utente loggato, segnalare un incendio nei pressi di un centro abitato e tralicci di alta tensione.

Task 9.2.1: avendo già effettuato il login, modificare il nome ed effettuare il logout.

<i>Azione 1</i>	L'utente preme sul pulsante per modificare il proprio profilo
<i>Risposta 1</i>	Il sistema mostra la schermata di modifica generale
<i>Azione 2</i>	L'utente preme sulla sezione relativa ai dati personali
<i>Risposta 2</i>	Il sistema mostra i campi personali da modificare
<i>Azione 3</i>	L'utente modifica il proprio nome
<i>Risposta 3</i>	Il sistema mostra la stessa schermata con il nome modificato
<i>Azione 4</i>	L'utente preme sul pulsante che consente di apportare i cambiamenti
<i>Risposta 4</i>	Il sistema effettua i cambiamenti



Task 9.2.2: da utente loggato, segnalare un incendio nei pressi di un centro abitato e tralicci di alta tensione.

<i>Azione 1</i>	L'utente preme sul pulsante per segnalare un incendio
<i>Risposta 1</i>	Il sistema mostra la schermata della segnalazione atta a inserire la posizione dell'incendio
<i>Azione 2</i>	L'utente preme su uno specifico punto della mappa per inserire la posizione
<i>Risposta 2</i>	Il sistema mostra un marker rosso sul punto toccato
<i>Azione 3</i>	L'utente preme sul pulsante che consente di continuare
<i>Risposta 3</i>	Il sistema mostra la schermata della segnalazione atta ad inserire una foto esplicativa dell'incendio
<i>Azione 4</i>	L'utente preme sul pulsante che gli consente di scattare la foto
<i>Risposta 4</i>	Il sistema permette all'utente di scattare la foto e la mostra sullo schermo
<i>Azione 5</i>	L'utente preme sul pulsante che consente di continuare
<i>Risposta 5</i>	Il sistema mostra la schermata atta a inserire maggiori informazioni sull'incendio
<i>Azione 6</i>	L'utente preme sulle informazioni delle quali è a conoscenza
<i>Risposta 6</i>	Il sistema evidenzia le informazioni ottenute dall'utente
<i>Azione 7</i>	L'utente preme sul pulsante che consente di continuare
<i>Risposta 7</i>	Il sistema mostra la schermata di completamento della segnalazione
<i>Azione 8</i>	L'utente preme sul pulsante che consente di tornare alla home page
<i>Risposta 8</i>	Il sistema mostra la home page



2.4.3 Conclusioni scaturite dalla seconda valutazione dell’usabilità

A seguito della seconda iterazione della valutazione dell’usabilità, non sono emerse sostanziali modifiche da effettuare ai secondi prototipi proposti in analisi. Tuttavia, dopo aver effettuato l’ultimo colloquio antecedente la reale progettazione mediante l’uso di Android Studio, sono venute alla luce ulteriori funzionalità, come il recupero della password e la memorizzazione delle credenziali dell’utente. In questo modo un utente che ha già effettuato la registrazione su uno specifico dispositivo X, non deve necessariamente inserire nuovamente le sue credenziali sullo stesso dispositivo X. Al seguito di ciò, l’unica schermata a subire cambiamenti è quella di accesso al sistema, che

verrà modificata in modo da contenere al di sopra del pulsante di Accesso a KARYA reports due componenti grafiche. La prima consente di evidenziare la volontà di memorizzare le credenziali, la seconda permette il recupero della password.



Figura 2.8: Schermata di home page senza registrazione con modifiche aggiuntive

Capitolo 3

Verso la realizzazione finale di KARYA reports

In questo capitolo verrà analizzato tutto ciò che concerne la realizzazione finale di KARYA reports mediante l'ambiente di sviluppo Android Studio. Partiremo con una chiara analisi della decisione dei colori, per poi passare alla scelta del logo e alla descrizione dei pattern tenuti in considerazione per la realizzazione dell'applicazione. In fine si passerà ad un'analisi della progettazione del layout di KARYA reports, focalizzando l'attenzione sulla struttura del progetto in Android Studio.

3.1 La scelta dei colori

La scelta dei colori [7] di KARYA reports ha subito un grande cambiamento in corso d'opera. Per quanto concerne la *prima realizzazione grafica*, si è pensato di costruire l'applicazione mediante una palette di colori pastello (Figura 3.1). Sono stati dunque utilizzati tre colori principali: il *purple navy* (*code*: `4D4D71`), utilizzato maggiormente come colore di sfondo e per pulsanti che determinano la conferma per varie azioni; i colori *cyclamen* (*code*: `E17897`) e *cool gray* (*code*: `818FB4`), pensati rispettivamente come colore secondario e terziario di sfondo.



Figura 3.1: Palette della prima implementazione grafica di KARYA reports

In questa prima fase è stato utilizzato uno sfondo molto ricco. Si è cercato per tanto di compensare questa sua sfarzosità inserendo componenti grafiche

sui toni del bianco, del blu e del rosso (Figura 3.2). Questa soluzione inizialmente sembrava creare un giusto equilibrio ma, navigando prolungatamente sulle interfacce di KARYA reports, si è notato che la complessità dello sfondo tendeva a rendere poco usabile l'applicazione. Questo perché le singole componenti dell'interfaccia, nonostante fossero state pensate per bilanciare lo sfondo appariscente, con un utilizzo prolungato tendevano a stressare la vista.

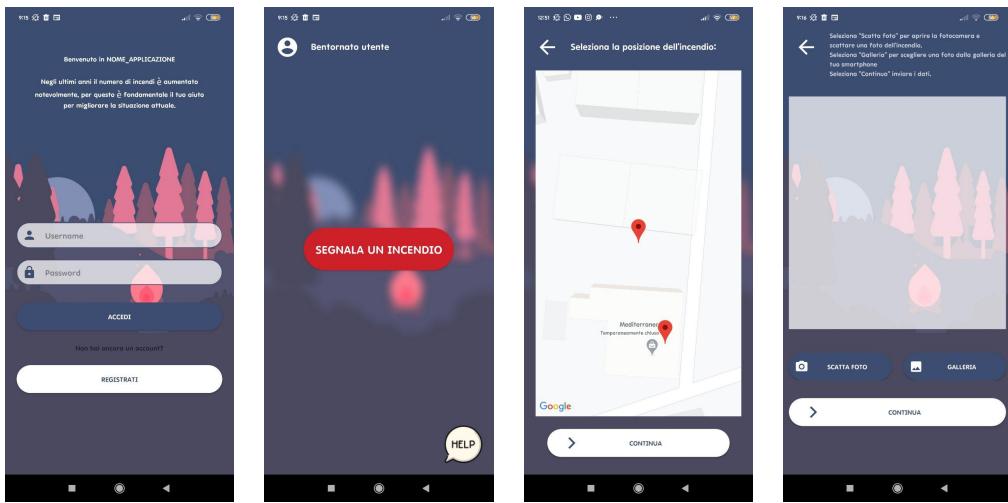


Figura 3.2: Bozza della prima implementazione grafica di KARYA reports

Per compensare questa difficoltà, si è ritenuto opportuno utilizzare una paletta di colori completamente alternativa (Figura 3.3). Si è pensato a colori che seguissero due linee di pensiero: fare da cornice al dominio del problema e sfruttare la loro comunicabilità. Sono stati utilizzati tre colori principali: il blu, il verde e il rosso.

Il *blu* rappresenta l'acqua, è considerato positivo per rilassare la mente e il corpo. Inoltre è capace di fornire serenità all'utente.

Il *verde* rappresenta la natura e procura protezione all'utente. Questo colore infatti è associato a operazioni che permettono di proseguire verso l'attuazione di un compito.

Il *rosso* simboleggia il fuoco, il pericolo. Questo infatti viene utilizzato per avviare la segnalazione di un incendio e per la conferma di operazioni irreversibili.



Figura 3.3: Palette della seconda implementazione grafica di KARYA reports

3.2 La scelta del logo

Il logo (Figura 3.4) ha visto la sua creazione grazie al sito *Canva.com*, che permette di creare e personalizzare grafiche in modo semplice e intuitivo. La realizzazione del logo nasce dal voler coniugare l'idea della natura al suo antagonista per eccellenza: il fuoco. Per questo motivo si è pensato ad una fiamma dai colori insoliti, che rispecchiano appieno la vegetazione. Il logo prende forma su una base circolare, questo conferisce una struttura lineare e ben definita che riesce ad evidenziare la semplicità del sistema stesso anche agli occhi del futuro utente. Per la realizzazione del logo è stata tenuta in considerazione la *psicologia della Gestalt*. Questa è una corrente psicologica che si sviluppò all'inizio del '900, in Germania. *Gestalt*, infatti, è una espressione tedesca che in italiano significa "forma". Gli psicologi della Gestalt hanno individuato una serie di principi della percezione visiva che sono fondamentali per un buon visual designer. Quando singoli elementi sono posizionati lungo un percorso circolare, prima vengono percepiti come un cerchio e solo in un secondo momento come una serie di elementi distinti. La tendenza a percepire le informazioni in questo modo è automatica e inconscia ed è presumibilmente attribuibile alla preferenza innata per la semplicità e l'organizzazione rispetto alla complessità e alla casualità. Quella descritta è definita *legge della chiusura* ed è uno dei principi della Gestalt. Tale legge vale in particolare quando gli elementi tendono a creare schemi semplici e riconoscibili, come nel caso della fiamma del logo, che crea un cerchio. La legge della chiusura consente di ridurre la complessità, utilizzando un numero inferiore di elementi o colori, e aumentare il grado di interesse del design.



Figura 3.4: Logo di KARYA reports

3.3 Descrizione dei pattern utilizzati

Definizione. Un *pattern* è una soluzione parzialmente specifica a un problema ricorrente all'interno di un contesto specifico. In altre parole si tratta di soluzioni progettuali riusabili che garantiscono a priori una buona usabilità.

Christopher Alexander [5], noto architetto austriaco, afferma che i *pattern* sono regole che mettono in relazione un problema comune ricorrente in un determinato contesto con la sua soluzione. Sono un potente strumento per adattare soluzioni di progettazione appropriate, sfruttando un patrimonio di conoscenze ed esperienze, per arrivare alla soluzione migliore possibile. Dalla metà degli anni '90, l'idea è stata adottata nel campo dell'ingegneria del software, in quanto i pattern iniziarono ad essere visti come un notevole corpus di conoscenze che catturano le migliori pratiche di progettazione del sistema.

I design pattern aiutano quindi a rafforzare quelli che sono i *principi dell'usabilità*. Questi sono un mezzo generale per comprendere il livello di usabilità di un sistema. Un sistema è *usabile* quando è user-friendly, in altre parole quando risulta facile da comprendere, efficace, efficiente (poco tempo, feedback rapido, spreco di risorse minimale) ed è capace di fornire all'utente un sentimento positivo. Queste sono però definizioni prettamente soggettive, ecco perchè ci si affida all'utilizzo dei design pattern. I principi e i fattori che incidono sull'usabilità sono fondamentalmente divisi in tre macro-principi: la *capacità di apprendimento* (*o learnability*) determina la facilità con cui nuovi utenti possono iniziare un'interazione e ottenere massime prestazioni; la *flessibilità*, ovvero la molteplicità di modi in cui l'utente il sistema scambiano informazioni e la *robustezza* che determina il livello di sostegno fornito all'utente nel determinare un comportamento di successo rispetto ai suoi obiettivi.

La progettazione di KARYA reports gode della presenza di vari design pattern [6]. Tra quelli di maggiori rilievo ne troviamo alcuni che verranno riportati di seguito.

- *Guided Tour*, ovvero un tour guidato che introduce le funzionalità del sistema all'utente finale. Vedremo l'utilizzo di questo pattern quando andremo ad analizzare la struttura del progetti in Android Studio. Guided Tour ha una durata breve e fornisce una prima interazione ad alto livello con l'applicazione.
- *Go back to a safe place*: fornisce una metodologia che consente di tornare a un checkpoint a scelta dell'utente. L'utilizzo di questo design pattern aiuta a rafforzare i principi legati alla *robustezza*. Infatti, mediante l'utilizzo del pattern go back to a safe place, il *principio della*

*recuperabilità*¹ è stato rafforzato. Ciò consente all’utente poco esperto di non aver timore nell’utilizzare l’applicazione, in quanto è possibile tornare indietro al fine di annullare operazioni sgradite. Tornare indietro risulta utile durante la segnalazione di un incendio, se ad esempio si sbaglia ad inviarne la posizione e si desidera tornare ad uno dei passi precedenti per aggiornarla prima di inviare i dati. La Figura 3.5 evidenzia il pattern poc’anzi espliato.

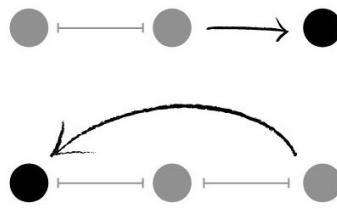


Figura 3.5: Pattern go back to a safe place

- *Breadcrumbs:* fornisce una traccia di collegamenti atti a far percepire all’utente la posizione nella quale si trova all’interno di un contesto. L’utilizzo di questo design pattern aumenta le principali forme di navigazione, senza però sostituirle. Un esempio generale dell’utilizzo del pattern breadcrumbs è illustrato nella figura sotto riportata. Questa si compone di tre sotto figure, rappresentanti ognuna tre frecce colorate in maniera differente. Immaginiamo di avere tre pagine differenti che rappresentano però la stessa cosa, vi è a necessità di differenziare le varie pagine. Per attuare tale differenziazione, si sfrutta il pattern breadcrumbs, secondo il quale vengono rappresentati dei separatori nella parte superiore dello schermo. Un esempio potrebbe essere proprio quello della registrazione a KARYA reports. Per effettuare la registrazione all’applicazione bisogna passare in tre pagine differenti e, per differenziare ogni pagina, sono stati introdotti dei selettori, che identificano le varie schermate.

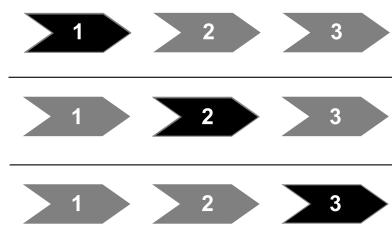


Figura 3.6: Pattern breadcrumbs

¹Il principio della recuperabilità batte sulla capacità dell’utente di intraprendere azioni correttive una volta rilevato un errore.

- *Accordion*: sono anche detti "menu a fisarmonica" e sono spesso utilizzati laddove lo spazio a disposizione non risulta essere abbastanza per contenere tutte le componenti grafiche dell'interfaccia. Facendo riferimento alla Figura 3.7, possiamo notare che questi menu si compongono di un pannello principale (*Main Category*) che si decomprime ad ogni clic in modo da rendere visibili varie componenti grafiche, ovvero le Sub Category, per poi ri-comprimersi quando si fa clic di nuovo sul pannello. Nello specifico questo pattern è stato utilizzato all'interno di KARYA reports al momento della modifica dei dati personali (Figure 3.15).

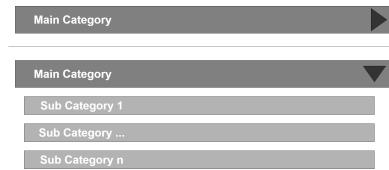


Figura 3.7: Pattern
accordion

3.4 Struttura del progetto in Android Studio

L'ambiente di sviluppo scelto per l'implementazione di KARYA reports è Android Studio, uno strumento leader per la creazione di applicazioni Android. Per la realizzazione delle componenti grafiche, delle interfacce e tutto ciò che comprende l'aspetto grafico dell'applicazione sono stati definiti dei file di layout. Questi sono dei file xml in cui sono organizzati staticamente gli elementi. Android Studio permette di erigere con facilità una netta separazione tra l'implementazione grafica e la logica dell'applicazione. Inoltre, grazie alla possibilità di introdurre un vasto numero di Virtual Device, ha reso possibile il poter testare la grafica su vari tipi di dispositivi elettronici. Un aspetto significativo di Android Studio consiste nell'anteprima di layout praticamente istantanea. Android Studio fornisce una propria struttura di default ben definita per i suoi progetti (Figura 3.8). Il progetto è costituito da tre parti principali: la cartella *java*, con il codice Java; la cartella *res*, con risorse per lo più realizzate in XML; un file di configurazione denominato *AndroidManifest.xml*. Per quanto concerne l'implementazione grafica di KARYA reports, ci concentreremo sulla cartella contenente le risorse grafiche, ovvero *res*. Questa a sua volta è divisa in: *drawable*, contenente tutte le icone e le immagini; *layout*, contenente i file xml per la creazione del design dell'applicazione; *values*, contenente i colori, le dimensioni e le stringhe atti alla realizzazione del progetto. KARYA reports è stata pensata sia per esse-

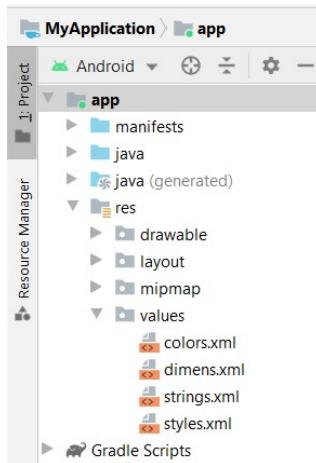


Figura 3.8:
Struttura base
di un progetto in
Android Studio

re utilizzata in portrait (verticale) che in landscape (orizzontale). Per questo motivo sono stati creati due file di layout per ogni schermata, in modo da impiegare al meglio lo spazio fisico del dispositivo in ogni posizione. Per fare ciò vengono sfruttate le caratteristiche del metodo *onCreate*. Questo metodo viene richiamato ogni volta l'*Activity*² viene creata. Si procede quindi comprendendo in primo luogo l'orientamento che il dispositivo ha nel momento specifico nel quale l'applicazione viene avviata, per poi indirizzare a due specifici layout differenti. Nell'*esempio di configurazione dell'orientamento* nel Codice 3.1 viene evidenziato il funzionamento di tale metodologia. Per comprendere la configurazione del dispositivo, viene utilizzata la classe *Configuration*, che permette di stabilire se il dispositivo è posto orizzontalmente o verticalmente. Nel primo caso verrà settata la view³ relativa al layout orizzontale (*LAYOUT_LANDSCAPE*), nel secondo caso verrà settata la view relativa al layout verticale (*LAYOUT_PORTRAIT*). Ogni qual volta il dispositivo viene ruotato, il metodo *onCreate* viene richiamato, per tanto la procedura di delineazione dell'orientamento viene reiterata. Questo assicura che se, ad esempio, un utente avvia KARYA reports con il cellulare posto orizzontalmente ma poi decide di cambiare la configurazione dello stesso, i due layout si daranno vicendevolmente "il cambio".

Codice 3.1: Esempio di configurazione dell'orientamento

```
@Override
protected void onCreate(Bundle savedInstanceState) {
```

²Un'activity in Android è essenzialmente una finestra che contiene l'interfaccia utente di un'applicazione ed il suo scopo è quello di permettere un'interazione con gli utenti.

³Qualunque elemento che appare in un'interfaccia utente e che svolge due funzionalità: mostra un aspetto grafico; gestisce eventi relativi all'interazione con l'utente.

```

super.onCreate(savedInstanceState);
int o = getResources().getConfiguration().orientation;
if(o==android.content.res.Configuration.ORIENTATION_LANDSCAPE){
    setContentView(R.layout.LAYOUT_LANDSCAPE);
} else {
    setContentView(R.layout.LAYOUT_PORTRAIT);
}
//some code here
}

```

Prima di accedere alle funzionalità di KARYA reports, è necessario attendere che l'applicazione sia pronta per l'utilizzo. Tale tempo di attesa è minimo, ma si è comunque pensato di inserire una schermata che si predispone di attendere quanto necessario prima di avviare l'applicazione. Questa consiste nella semplice visualizzazione del logo. Nel caso di una prima interazione con l'applicazione, verrà chiesto se si vuole iniziare con un tutorial, atto ad introdurre le funzionalità principali dell'applicazione (Figura 3.9).

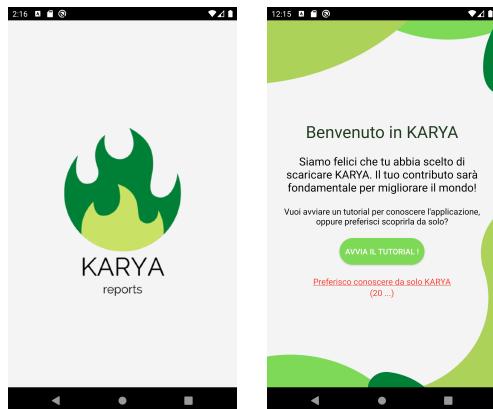


Figura 3.9: Schermate di avvio e di prima apertura di KARYA reports

A questo punto l'utente può decidere di avviare il tutorial, oppure di continuare senza quest'ultimo, iniziando a scoprire da solo l'applicazione. Se la scelta viene effettuata entro un lasso di tempo che supera i 20 secondi, KARYA reports si avvierà automaticamente senza mostrare alcun tutorial. Quest'ultimo è stato realizzato mediante una singola schermata di layout dinamica denominata *tutorial.xml*, contenuta della cartella *res/layout*. Per comprendere al meglio la grafica di tale interfaccia, nella figura 3.10 è stata riportata una bozza che rispecchia il posizionamento degli oggetti all'interno dello schermo. L'activity quindi si compone di un'*immagine* (1) rappresentante le varie schermate dell'applicazione, una *descrizione* (2) per queste ultime, un *indicatore di posizione* (3) e un *pulsante* (4) che permette di avanzare con il tutorial.

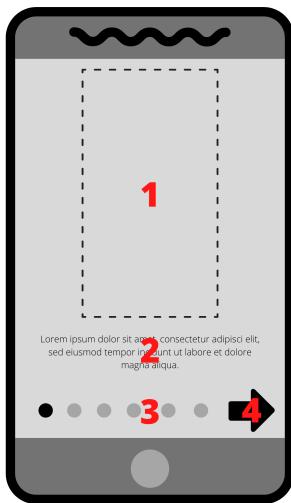


Figura 3.10:
Esempio grafico
della schermata
del tutorial

Concluse le schermate da mostrare, il tutorial cesserà automaticamente, presentando la schermata di accesso. Nel file *Utility.java* sono state definite le stringhe utilizzate per la descrizione delle varie schermate dell'applicazione, in modo da permetterne il cambiamento dinamico. La classe che si occupa di tale cambiamento e della modifica del colore dell'indicatore della posizione è *Tutorial.java* (Codice 3.2). Lo scopo è quello di rendere "intelligente" il bottone per proseguire con le schermate del tutorial e fare in modo che si accorga che qualcuno stia interagendo con esso. Per fare questo si devono introdurre i gestori di evento in Android, a tal proposito il metodo *onClickListener(...)* si occuperà di rendere sensibile al click il bottone. Questa procedura viene mostrata nel Codice 3.2, ma per comprenderne al meglio il funzionamento, bisogna innanzi tutto capire quali sono i riferimenti nel codice Java alle varie componenti del layout. Faremo riminiscenza alla Figura 3.10 per facilitare questo compito. L'immagine (1) è definita nella variabile denominata *layout_screen*; la descrizione (2) è definita nella variabile denominata *tv_description*; l'indicatore di posizione (3) si compone invece di nove immagini a forma di cerchio, denominate *image_view_N*, dove N è un numero progressivo compreso fra uno e nove; il pulsante (4) che permette di proseguire con il tutorial è definito invece nella variabile denominata *btn_next*. Quando il pulsante viene premuto, il suo riferimento *bnt_next* percepisce il click ed effettua i cambiamenti necessari per la corretta visualizzazione del tutorial. Ad esempio, se stiamo visualizzando la quarta schermata del tutorial e, dopo aver preso visione della descrizione decidiamo di andare avanti, vengono effettuati dei cambiamenti che permettono di visualizzare il prossimo passo del tutorial. Questi cambiamenti, nel caso specifico, comprendono:

- la modifica del riferimento *tv_description*, mediante la descrizione con-

tenuta della variabile stringa *HomePageWithLogin* all'interno della classe *Utility.java*;

- la modifica del colore dei pallini dell'indicatore della posizione, tramite l'utilizzo del metodo *changeColorPoint()* creato ad hoc. Questo prende in input due valori: il pallino che fa riferimento alla posizione precedente e quello che fa riferimento alla posizione attuale, modificando il colore del primo in *palette_two_transparent* (nome colore: Waikawa Grey Transparent, codice: #80084D6A) e del secondo in *palette_two* (nome colore: Waikawa Grey, codice: #576B95), in modo da far comprendere all'utente in quale posizione si trovi al momento;
- la modifica del *layout_screen*, mediante il settaggio di una nuova immagine, conforme alla descrizione inserita.

Codice 3.2: Tutorial.java

```
btn_next.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    @Override
    public void onClick(View v) {
        i++;
        switch (i){
            case 2:
                imageView_1 = findViewById(R.id.v1);
                imageView_2 = findViewById(R.id.v2);
                tv_description.setText(utility.FirstRegistrationPage);
                changeColorPoint(imageView_1,imageView_2);
                layout_screen.setBackgroundResource(R.drawable._2_bg);
                break;
            case 3:
                imageView_3 = findViewById(R.id.v3);
                changeColorPoint(imageView_2,imageView_3);
                tv_description.setText(utility.SecondRegistrationPage);
                layout_screen.setBackgroundResource(R.drawable._3_bg);
                break;
            case 4:
                imageView_4 = findViewById(R.id.v4);
                changeColorPoint(imageView_3,imageView_4);
                tv_description.setText(utility.ThirdRegistrationPage);
                layout_screen.setBackgroundResource(R.drawable._4_bg);
                break;
            case 5:
```

```
        imageView_5 = findViewById(R.id.v5);
        changeColorPoint(imageView_4,imageView_5);
        tv_description.setText(Utility.HomepageWithLogin);
        layout_screen.setBackgroundResource(R.drawable._5_bg);
        break;
    case 6:
        imageView_6 = findViewById(R.id.v6);
        changeColorPoint(imageView_5,imageView_6);
        tv_description.setText(Utility.UserProfile);
        layout_screen.setBackgroundResource(R.drawable._6_bg);
        break;
    case 7:
        imageView_7 = findViewById(R.id.v7);
        changeColorPoint(imageView_6,imageView_7);
        tv_description.setText(Utility.FireLocation);
        layout_screen.setBackgroundResource(R.drawable._7_bg);
        break;
    case 8:
        imageView_8 = findViewById(R.id.v8);
        changeColorPoint(imageView_7,imageView_8);
        tv_description.setText(Utility.TakePhoto);
        layout_screen.setBackgroundResource(R.drawable._8_bg);
        break;
    case 9:
        imageView_9 = findViewById(R.id.v9);
        changeColorPoint(imageView_8,imageView_9);
        tv_description.setText(Utility.MoreInformation);
        layout_screen.setBackgroundResource(R.drawable._9_bg);
        break;
    default:
        startApplication();
        break;
    }
}

});

private void changeColorPoint(ImageView iv1, ImageView iv2){
    iv1.setBackgroundTintList(ColorStateList.valueOf(getResources() .
        getColor(R.color.palette_two_transparent)));
    iv2.setBackgroundTintList(ColorStateList.valueOf(getResources() .
        getColor(R.color.palette_two)));
}
```

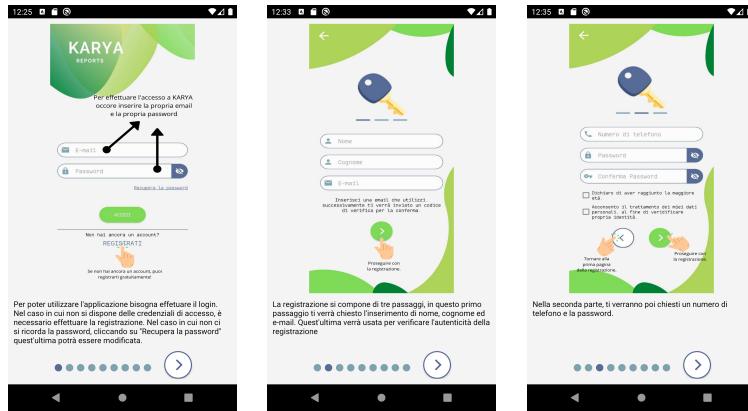


Figura 3.11: Prime tre schermate del tutorial

Concluso il tutorial viene mostrata la schermata di accesso al sistema, che si compone di uno sfondo che richiama i colori del logo. Centralmente è richiesto l'inserimento dell'*email* e della *password*, quest'ultima presenta la possibilità di essere nascosta, mediante l'utilizzo di un asterisco (*) ad ogni carattere, o visualizzata nella sua completezza. Per rendere possibile ciò è stato inserito all'interno del file *home_page_no_login.xml* (Codice 3.3) un apposito pulsante che permette di alternare le due visualizzazioni della password a discapito dell'utente. Di default la password viene mostrata a video come una successione di asterischi, ma premendo su pulsante a destra dell'apposita casella di testo, viene attivata dinamicamente la visualizzazione senza filtri della stessa. Data la presenza di più pulsanti atti alla duplice visualizzazione della password, questa modifica viene effettuata facendo riferimento ad un metodo creato nella classe *Utility.java* denominato *showPassword()* (Codice 3.5), nella classe *HomePageNoLogin.java* (Codice 3.4). In questo metodo vengono sfruttate le funzionalità di *getTransformationMethod()*, capace di comprendere come viene visualizzata la password a video.

Codice 3.3: *home_page_no_login.xml*

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="@dimen/dp10"
        android:password="true"/>
        <InputFilter</InputFilter>
    </EditText>
    <Switch
        android:id="@+id/show_password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
        <com.google.android.material.textfield.TextInputLayout</com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInput</com.google.android.material.textfield.TextInput>
                <com.google.android.material.textfield.PasswordToggle</com.google.android.material.textfield.PasswordToggle>
            </com.google.android.material.textfield.TextInput>
        </com.google.android.material.textfield.TextInputLayout>
    </Switch>
</RelativeLayout>
```

```
        android:layout_weight="2"
        android:autofillHints=""
        android:background="@drawable/prop_rounded_text"
        android:drawableStart="@drawable/ic_lock"
        android:drawableLeft="@drawable/ic_lock"
        android:drawablePadding="@dimen/dp15"
        android:hint="@string/password"
        android:inputType="textPassword"
        tools:ignore="TextFields" />
<ImageView
        android:id="@+id/showPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_password_visibility_off"
        android:background="@drawable/prop_show_password"
        android:padding="@dimen/dp10"
        android:layout_alignRight="@+id/password" />
</RelativeLayout>
```

Codice 3.4: HomePageNoLogin.java

```
iv_showPassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Utility.showPassword(et_password, iv_showPassword);
    }
});
```

Codice 3.5: Metodo showPassword nella classe Utility.java

```
public static void showPassword(EditText et, ImageView iv){
    if (et.getTransformationMethod().getClass().getSimpleName()
        .equals("PasswordTransformationMethod")) {
        et.setTransformationMethod(new
            SingleLineTransformationMethod());
        iv.setImageResource(R.drawable.ic_password_visibility);
    }
    else {
        et.setTransformationMethod(new PasswordTransformationMethod());
        iv.setImageResource(R.drawable.ic_password_visibility_off);
    }
    et.setSelection(et.getText().length());
}
```

Se non si dispone ancora di un account, è possibile registrarsi a KARYA reports premendo sull'apposito pulsante denominato "REGISTRATI". Il layout di queste Activity non si discosta molto da quello presentato nei prototipi. Le caselle di input del testo sono state personalizzate, non sono state perciò utilizzate quelle predefinite di Android, che presentano un layout composto da linee spezzate. Per attuare tale personalizzazione, è stato creato un file xml all'interno della cartella res/drawable, denominato *prop_rounded_button.xml* (Codice 3.6). La caratterizzazione fondamentale delle nuove caselle di testo è la loro forma tondeggiante, capace di rendere il design più armonico.

Codice 3.6: prop_rounded_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
<solid android:color="@color/white"/>
<corners android:radius="@dimen/dp150"/>
<stroke android:color="@color/palette_two"
        android:width="2dp"/>
<padding
    android:right="@dimen/dp10"
    android:left="@dimen/dp10"
    android:top="@dimen/dp10"
    android:bottom="@dimen/dp10"/>
</shape>
```

L'unico aspetto di design che si discosta dai prototipi è l'aggiunta della raffigurazione di una chiave al di sopra delle schermate di registrazione (Figura 3.12).

La funzionalità principale di KARYA reports è quella di segnalare un nuovo incendio, azione che può essere svolta dalla *home page con avvenuto login* (Figura 3.13). Si è pensato di evidenziare questa funzionalità dal punto di vista grafico realizzando un pulsante personalizzato per la segnalazione. A questo proposito è stato creato un file nella cartella res/drawable denominato *prop_segnala_incendio.xml* (Codice 3.7). L'obiettivo è quello di rappresentare una situazione di emergenza, un pericolo imminente da comunicare. L'utente deve comprendere l'importanza del bottone, per questo motivo presenta una forma tonda e un colore rosso lievemente acceso. Per rendere ancora più esplicita la sua funzione, a questo viene sovrapposta la scritta "Segnala un incendio". Il file *prop_segnala_incendio.xml* verrà poi richiamato nel file di layout *home_page_with_login.xml* (Codice 3.8), nel momento della creazione vera e propria del pulsante sull'interfaccia.

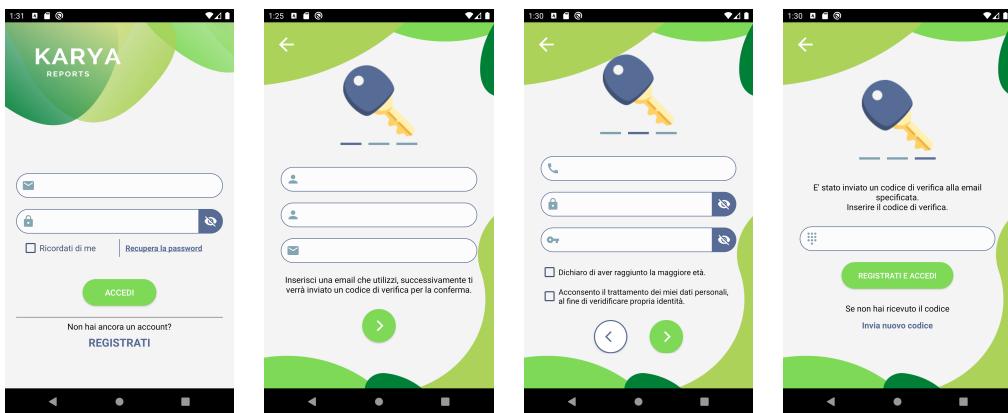


Figura 3.12: Schermate di accesso e di registrazione a KARYA reports



Figura 3.13:
Home page
con login
effettuato

Codice 3.7: prop_segnala_incendio.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
<solid android:color="@color/btn_red"/>
<corners android:radius="1000dp"/>
<size android:width="@dimen/dp200"
      android:height="@dimen/dp200"/>
</shape>
```

Codice 3.8: home_page_with_login.xml

```
<Button
    android:id="@+id/reporting"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_centerInParent="true"
```

```

    android:text="@string/segnala_incendio"
    android:textSize="@dimen/dp25"
    android:background="@drawable/prop_segnala_incendio"
    android:textColor="@color/white"
    android:backgroundTint="@color/redFire"
/>

```

L'obiettivo della segnalazione è quello di rendere il compito dell'utente quanto meno oneroso possibile, inviando comunque un quantitativo di dati opportuno per una corretta segnalazione dell'incendio. A tale scopo vi sono tre semplici passaggi da eseguire: l'invio della *posizione* approssimata dell'incendio, l'invio di una eventuale *fotografia* e l'invio di *informazioni aggiuntive* sulla zona interessata. Una volta ottenuti tutti i dati, il sistema informerà l'utente che la segnalazione è avvenuta con successo (Figura 3.14). Focalizzandosi sulla schermata inherente l'invio della foto (seconda immagine della Figura 3.14) si può notare l'utilizzo del *principio della similitudine*. Questo è uno dei principi della Gestalt, che afferma che elementi che condividono determinate caratteristiche visuali tendono ad essere percepiti come singolo gruppo. Nel caso specifico, all'interno della schermata che permette l'invio della fotografia, sono presenti tre pulsanti: "Scatta foto", "Scegli dalla galleria" e "Continua". I primi due sono rappresentati mediante la stessa tonalità di blu, mentre il terzo è di colore verde. L'occhio umano tende a distinguere due gruppi, il primo contenente i due pulsanti dello stesso colore, il secondo invece è rappresentato dal singolo pulsante che consente di continuare con la segnalazione.

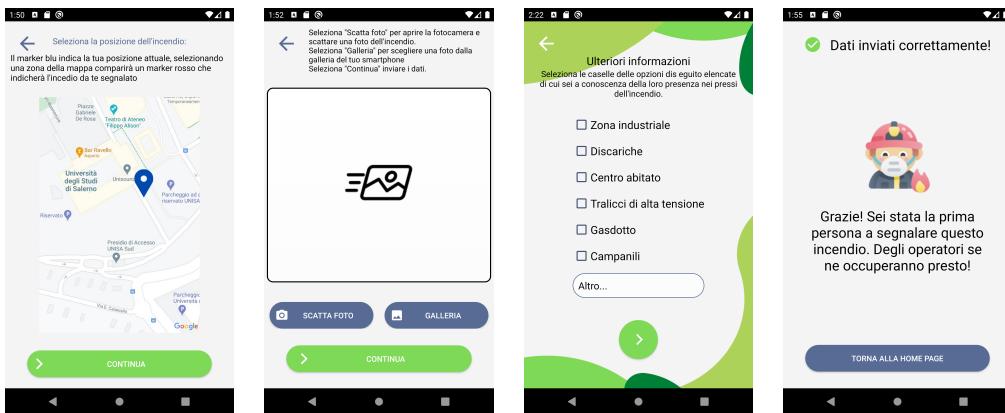


Figura 3.14: Schermate atte alla segnalazione di un incendio

Per quanto concerne la modifica dei dati personali e dei dati d'accesso, la sintassi grafica rispecchia appieno quelli che sono i prototipi mostrati nella

seconda fase di analisi (Figura 2.6). Una interfaccia grafica così strutturata è stata realizzata mediante il file *user_profile.xml* (Codice 3.9), contenente tutte le componenti grafiche, e la classe *UserProfile.java* (Codice 3.10), che si occupa di mostrare solo le componenti relative ai *dati personali* (nel caso l'utente abbia necessità di modificare il nome, il cognome o il numero di telefono) o ai *dati d'accesso* (nel caso l'utente abbia necessità di modificare l'email o la password).

Codice 3.9: user_profile.xml

```
[. . . some code here . . .]

<Button
    android:id="@+id/edit_personal_data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/dp20"
    android:layout_marginBottom="@dimen/dp10"
    android:background="@drawable/prop_rounded_button"
    android:drawableEnd="@drawable/ic_arrow"
    android:drawableRight="@drawable/ic_arrow"
    android:text="@string/button_personal_data"
    android:textColor="@color/palette_two"
    android:onClick="openPersonalData"/> // --> listener a un metodo
    per rendere visibili le form inerenti alla modifica dei dati
    personali sull'interfaccia

[. . . some code here . . .]

<Button
    android:id="@+id/edit_access_data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/dp10"
    android:layout_marginBottom="@dimen/dp10"
    android:background="@drawable/prop_rounded_button"
    android:drawableEnd="@drawable/ic_arrow"
    android:drawableRight="@drawable/ic_arrow"
    android:text="@string/button_access_data"
    android:textColor="@color/palette_two"
    android:onClick="openAccessData"/> // --> listener a un metodo
    per rendere visibili le form inerenti alla modifica dei dati
    di accesso sull'interfaccia
```

[. . . some code here . . .]

Codice 3.10: UserProfile.java

```
public void openPersonalData(View v){  
    if(openPD == false) {  
        openPD = true;  
        nome.setVisibility(View.VISIBLE);  
        cognome.setVisibility(View.VISIBLE);  
        numeroCellulare.setVisibility(View.VISIBLE);  
        confermaDati.setVisibility(View.VISIBLE);  
        iv_showPasswordConfirm.setVisibility(View.VISIBLE);  
        email.setVisibility(View.GONE);  
        password.setVisibility(View.GONE);  
        iv_showPassword.setVisibility(View.GONE);  
        confermaPassword.setVisibility(View.GONE);  
        iv_showConfirmPassword.setVisibility(View.GONE);  
    } else {  
        openPD = false;  
        nome.setVisibility(View.GONE);  
        cognome.setVisibility(View.GONE);  
        numeroCellulare.setVisibility(View.GONE);  
        if(openAD == false) {  
            confermaDati.setVisibility(View.GONE);  
            iv_showPasswordConfirm.setVisibility(View.GONE);  
        }  
    }  
}  
  
public void openAccessData(View v) {  
    if (openAD == false) {  
        openAD = true;  
        email.setVisibility(View.VISIBLE);  
        password.setVisibility(View.VISIBLE);  
        iv_showPassword.setVisibility(View.VISIBLE);  
        confermaPassword.setVisibility(View.VISIBLE);  
        iv_showConfirmPassword.setVisibility(View.VISIBLE);  
        confermaDati.setVisibility(View.VISIBLE);  
        iv_showPasswordConfirm.setVisibility(View.VISIBLE);  
        nome.setVisibility(View.GONE);  
        cognome.setVisibility(View.GONE);  
        numeroCellulare.setVisibility(View.GONE);  
    }  
}
```

```
    } else {
        openAD = false;
        email.setVisibility(View.GONE);
        password.setVisibility(View.GONE);
        iv_showPassword.setVisibility(View.GONE);
        confermaPassword.setVisibility(View.GONE);
        iv_showConfirmPassword.setVisibility(View.GONE);
        if (openPD == false) {
            confermaDati.setVisibility(View.GONE);
            iv_showPasswordConfirm.setVisibility(View.GONE);
        }
    }
}
```

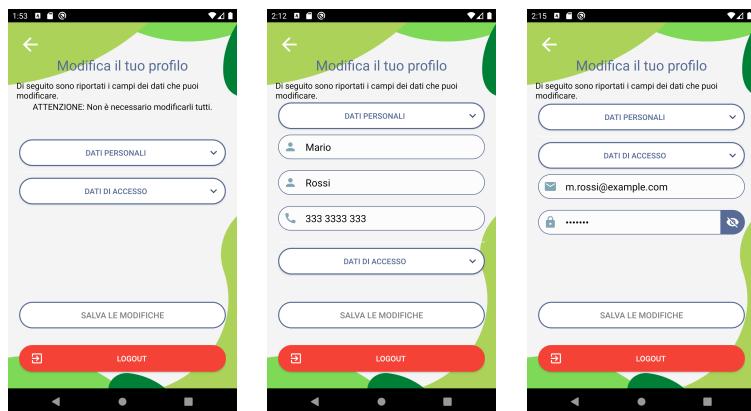


Figura 3.15: Schermate per la modifica dei dati

3.5 Feedback di KARYA reports

KARYA reports vanta la presenza di numerosi *pop-up*⁴ di navigazione e di avviso, per comunicare lo stato dell'applicazione all'utente. Per effettuare tali avvisi, sono stati utilizzati i *Toasts*. In Android un Toast è un semplice messaggio di notifica visualizzato in un pop-up. I toasts vengono utilizzati per dare un *feedback all'utente* in seguito ad una operazione compiuta. L'utilizzatore del sistema deve essere solo notificato dal messaggio, senza la possibilità di interagirvi. Un toast viene visualizzato solo per un determinato

⁴I *pop-up* (in italiano: finestre a comparsa) sono degli elementi dell’interfaccia grafica, quali finestre o riquadri, che compaiono automaticamente durante l’uso di un’applicazione in determinate situazioni.

intervallo di tempo, dopo il quale scomparirà. La classe statica *Toast* mette a disposizione due metodi: *makeText()*, utilizzato per costruire il messaggio da notificare, e *show()*, utilizzato per rendere visibile il messaggio. Di default, un toast viene posizionato nella parte bassa dello schermo e presenta un semplice sfondo bianco (o grigio scuro, dipendentemente dal tema scelto sul proprio dispositivo), con una scritta che contrasta il colore di background (Figura 3.16, immagine alla sinistra).

L'utilizzo dei toast garantisce che un qualsiasi utilizzatore del sistema sia sempre a conoscenza del motivo per il quale questo, in determinate circostanze, si comporti in maniera anomala. Ad esempio, se al momento della registrazione al sistema non ci si accorge che la password e la sua conferma non combaciano, sarà la stessa KARYA reports a segnalare tale anomalia mediante l'utilizzo di un pop-up ed evidenziando i campi in errore per un tempo limitato (Figura 3.16, immagine alla destra).

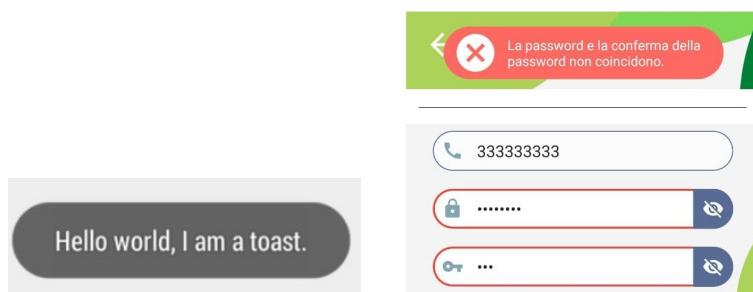


Figura 3.16: A sinistra un Toast con il layout di default messo a disposizione dalla classe *Toast* in Android Studio. A destra un Toast personalizzato di KARYA reports.

Come si può notare dalla figura precedentemente mostrata (Figura 3.16, immagine alla destra), KARYA reports sfrutta la possibilità di poter personalizzare i toast di default messi a disposizione dalla classe *Toast*. La modifica non concerne solo la grafica, ma anche la posizione, in quanto per ragioni di visibilità si è deciso di mostrare i messaggi nella parte superiore dello schermo, invece che su quella inferiore. Per la personalizzazione è stato creato un unico file di layout denominato *custom_toast_error.xml* (Codice 3.11), nel quale viene definito staticamente il layout del singolo toast. Per non dover creare un file di layout assestante per ogni errore da far presente all'utente, è stato definito il metodo *newCustomToast()* nella classe *Utility.java* (Codice 10.12). Per permettere la visualizzazione di un nuovo toast personalizzato bisognerà introdurre le seguenti linee di codice:

1. *Utility.newCustomToast()*

2. `getLayoutInflater().inflate(R.layout.custom_toast_error, null),`
3. `new Toast(getApplicationContext()),`
4. `"MESSAGGIO_DA_VISUALIZZARE");`

Nella prima riga viene invocato il metodo `newCustomToast()`, presente nella classe `Utility`, che prende in input tre parametri: la `View` che si vuole utilizzare (riga 2.), ovvero il file xml `custom_toast_error`, un nuovo `toast` (riga 3.) e una `stringa` (riga 4.) rappresentativa del messaggio da visualizzare.

Codice 3.11: custom_toast_error.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="@dimen/dp20">
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="@drawable/prop_toast"
        android:layout_marginRight="@dimen/dp5"
        android:layout_marginLeft="@dimen/dp5"
        android:layout_gravity="center_horizontal"
        android:backgroundTint="@color/red">
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="@dimen/dp10"
            android:background="@drawable/ic_error"
            android:backgroundTint="@color/white"/>
        <TextView
            android:id="@+id/customTextView"
            android:padding="@dimen/dp10"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="15sp"
            android:layout_gravity="center"
            android:text="Il messaggio di errore viene evidenziato qui!"
            android:textColor="@color/white"/>
    </LinearLayout>
</LinearLayout>
```

```
</LinearLayout>  
</LinearLayout>
```

Codice 3.12: metodo newCustomToast() nella classe Utility.java

```
public static void newCustomToast (View v, Toast toast, String s){  
    TextView tv = (TextView) v.findViewById(R.id.customTextView);  
    tv.setText(s);  
    toast.setView(v);  
    toast.setDuration(Toast.LENGTH_LONG);  
    toast.setGravity(Gravity.TOP, 0,0);  
    toast.show();  
}
```

Per segnalare errori di inserimento dei dati, oltre alla visualizzazione di toast personalizzati, KARYA reports è predisposta per la messa in evidenza dei campi in errore (Figura 10.12). Per fare ciò vengono sfruttate le funzionalità della classe Timer. Per comprenderne al meglio il funzionamento viene mostrato un esempio del suo utilizzo all'interno del metodo *changeColor()* nella classe *FirstRegistrationPage.java* (Codice 10.13). In primo luogo viene dichiarata una variabile *PERIOD*, che determina il tempo per cui i bordi delle caselle di testo assumono il colore rosso. La colorazione delle caselle di testo avviene parallelamente alla visualizzazione di un messaggio di errore tramite un toast. Siccome la durata di quest'ultimo è pari a 4 secondi, anche la colorazione dei bordi delle caselle di testo assumeranno tale colore per lo stesso lasso di tempo. Vengono poi dichiarati e creati due oggetti di tipo Timer: *redBorderColor* e *blueBorderColor*. Il primo si occuperà di colorare di rosso le cornici delle caselle di testo, il secondo si occuperà di colorare le stesse della loro tinta originale una volta conclusi i 4 secondi. Per avviare il processo di colorazione rosso, viene richiamato sull'oggetto Timer *redBorderColor* il metodo *schedule* che prende in input l'azione da svolgere, il ritardo e la durata. Nel caso specifico l'azione da svolgere è la colorazione in rosso dei campi che risultano vuoti o non validi, il ritardo è pari a 0, e la durata è pari alla variabile *PERIOD*, ovvero 4 secondi, tempo dopo il quale avviene il processo di colorazione blu. Viene richiamato sull'oggetto Timer *blueBorderColor* il metodo *schedule* che prende in input l'azione da svolgere e il ritardo. Nel caso specifico l'azione da svolgere prevede la cancellazione del primo oggetto Timer mediante la dicitura *redBorderColor.cancel()* e la successiva colorazione in blu di tutti i campi. Il ritardo è pari a 4 secondi, tempo prima del quale viene eseguito l'oggetto Timer *redBorderColor*. Per comprendere al meglio ciò che è stato appena spiegato è stato riportato il

funzionamento del metodo *changeColor()* nella Figura 3.17. Questa evidenzia, mediante un semplice schema, i tempi e le azioni compiute dagli oggetti *redBorderColor* e *blueBorderColor*.



Figura 3.17: Esempio grafico del funzionamento del metodo *changeColor()*

Codice 3.13: metodo *changeColor()* nella classe *FirstRegistrationPage.java*

```
//1000 = 1 secondo -> 4000 = 4 secondi
private static final int PERIOD =4000;
private Timer redBorderColor;
private Timer blueBorderColor;

private void changeColor(){
    redBorderColor = new Timer();
    blueBorderColor = new Timer();

    redBorderColor.schedule(new TimerTask() {
        @Override
        public void run() {
            //campo nome vuoto
            if(et_nome.getText().toString().trim().length() == 0) {
                Utility.setRedBorderColor(et_nome);
            }
            //campo cognome vuoto
            if(et_cognome.getText().toString().trim().length() == 0) {
                Utility.setRedBorderColor(et_cognome);
            }
            //campo email vuoto
            if(et_email.getText().toString().trim().length() == 0) {
                Utility.setRedBorderColor(et_email);
            }
        }
    }, PERIOD);
    blueBorderColor.schedule(new TimerTask() {
        @Override
        public void run() {
            Utility.setBlueBorderColor(et_nome);
            Utility.setBlueBorderColor(et_cognome);
            Utility.setBlueBorderColor(et_email);
        }
    }, PERIOD);
}
```

```

if(Utility.isValidName(et_nome.getText().toString()))
    Utility.setRedBorderColor(et_nome);

if(Utility.isValidName(et_cognome.getText().toString()))
    Utility.setRedBorderColor(et_cognome);
}

},0,PERIOD);

blueBorderColor.schedule(new TimerTask() {
    @Override
    public void run() {
        redBorderColor.cancel();
        Utility.setBlueBorderColor(et_nome);
        Utility.setBlueBorderColor(et_cognome);
        Utility.setBlueBorderColor(et_email);
        blueBorderColor.cancel();
    }
},PERIOD);
}

```

Vi sono occasioni nelle quali è opportuno che l’utente interagisca con gli avvisi di KARYA reports, per tanto il toast non risulta sufficiente a soddisfare questa necessità (abbiamo visto che vengono utilizzati solo per mostrare dei messaggi, senza che l’utente possa mai interagire con essi). Per superare tali mancanze sono state utilizzate le *finestre di dialogo*, ovvero piccole finestre che non riempiono lo schermo e vengono normalmente utilizzate per eventi modali che richiedono di eseguire un’azione prima di poter procedere. Di default, una finestra di dialogo viene posizionata centralmente allo schermo e presenta un semplice sfondo bianco (o nero, dipendentemente da tema scelto sul proprio dispositivo), un titolo e una descrizione che contrastano il colore di background e due scelte che sono una l’opposto dell’altra. L’azione positiva è posta alla destra, mentre quella negativa alla sinistra (Figura 3.18).

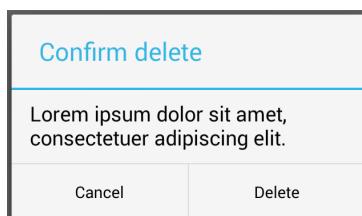


Figura 3.18: Esempio di una finestra di dialogo di default

All’interno di KARYA reports vengono utilizzate due finestre di dialogo, la prima inerente al recupero della password, la seconda alla fase di invio della fotografia durante segnalazione dell’incendio (Figure 3.19).

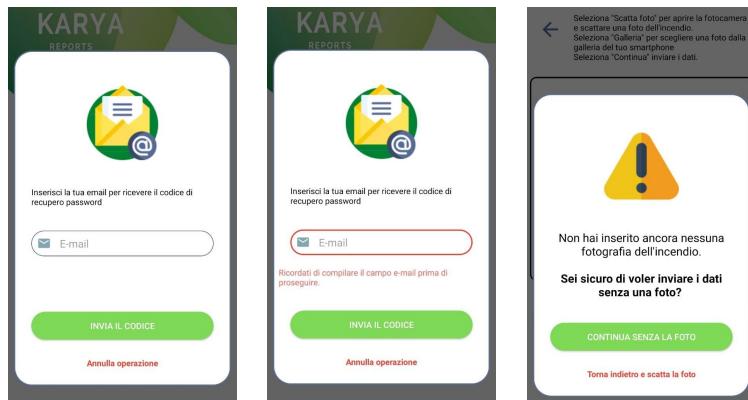


Figura 3.19: Finestre di dialogo utilizzate in KARYA reports

KARYA reports sfrutta la possibilità di poter personalizzare le finestre di dialogo fornite di default nella classe *AlertDialog.java*. Per fare ciò sono stati creati due file: *custom_popup_recovery_password.xml* (Codice 3.14), che definisce il layout della finestra di dialogo per il recupero della password, e *custom_popup_confirm_operation.xml* (Codice 3.15), che definisce il layout della finestra di dialogo per confermare il proseguimento della segnalazione senza l'inserimento della fotografia. Per ognuno di questi file è stato creato un apposito file per la definizione dello stesso in landscape (orizzontale).

In questo caso non è stato possibile creare un singolo file di layout riutilizzabile, poiché l'interazione dell'utente con ogni singola finestra di dialogo è diversa. Nel caso del recupero della password, all'interno dell'alert dialogue vi è una casella di testo dentro la quale inserire l'email. Qualora venga premuto il pulsante atto ad inviare l'email per il recupero password prima di aver inserito l'email, l'utente viene notificato con un messaggio di errore al di sotto della casella di testo (si veda la seconda immagine nelle Figure 3.19). Per permettere la visualizzazione di una nuova finestra di dialogo, bisogna dichiarare: un oggetto di tipo *AlertDialog.Buider*, un oggetto di tipo *AlertDialog* e tutte le componenti grafiche presenti all'interno della finestra di dialogo che si desiderano manipolare. Le classi java che mostrano i due alert dialogue sono *HomePageNoLogin.java* e *TakePhoto.java*. Queste sono riportate rispettivamente alle voci Codice 3.16 e Codice 3.17, escludendo le righe di codice che non appartengono alla visualizzazione delle finestre di dialogo.

Codice 3.14: *custom_popup_recovery_password.xml*

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:background="@drawable/prop_popup_recovery_password"
    android:orientation="vertical">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/ic_send_code"
        android:layout_gravity="center"
        android:layout_marginTop="@dimen/dp50"
        android:layout_marginBottom="@dimen/dp25"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/dp20"
        android:text="@string/insert_email"
        android:textColor="@color/black"/>
    <EditText
        android:id="@+id/popup_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/dp20"
        android:autofillHints=""
        android:background="@drawable/prop_rounded_text"
        android:drawableStart="@drawable/ic_email"
        android:drawableLeft="@drawable/ic_email"
        android:drawablePadding="@dimen/dp15"
        android:hint="@string/email"
        android:inputType="text" />
    <TextView
        android:id="@+id/rec_insert_email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rem_ins_email"
        android:layout_gravity="center"
        android:textColor="@color/red"
        android:visibility="invisible"/>
    <TextView
        android:id="@+id/rec_email_inexistent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/email_inexistent"
        android:layout_gravity="center"
        android:textColor="@color/red"
```

```
        android:visibility="invisible"/>
<Button
    android:id="@+id/b_send_code"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_margin="@dimen/dp20"
    android:background="@drawable/prop_rounded_button_green"
    android:text="@string/send_code"
    android:textColor="@color/white"
    android:textSize="@dimen/sp15"
    android:onClick="pressLog"/>
<TextView
    android:id="@+id/b_cancel_operation"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/dp10"
    android:text="@string/cancel_operation"
    android:layout_gravity="center"
    android:textStyle="bold"
    android:layout_marginBottom="@dimen/dp20"
    android:onClick="pressReg"
    android:textColor="@color/btn_red"/>
</LinearLayout>
```

Codice 3.15: custom_popup_confirm_operation.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/prop_popup_recovery_password"
    android:orientation="vertical">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/ic_miss_data_big"
        android:layout_gravity="center"
        android:layout_marginTop="@dimen/dp50"
        android:layout_marginBottom="@dimen/dp25"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_margin="@dimen/dp20"
        android:textAlignment="center"
        android:textSize="@dimen/dp20"
        android:text="@string/missing_photo"
        android:textColor="@color/black"
        android:gravity="center_horizontal"/>
<Button
        android:id="@+id/b_continue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="@dimen/dp20"
        android:background="@drawable/prop_rounded_button_green"
        android:text="@string/send_anyway"
        android:textColor="@color/white"
        android:textSize="@dimen/sp15"
        android:onClick="pressSendPhoto"/>
<TextView
        android:id="@+id/b_come_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/dp10"
        android:text="@string/come_back"
        android:layout_gravity="center"
        android:textStyle="bold"
        android:layout_marginBottom="@dimen/dp20"
        android:onClick="pressNulla"
        android:textColor="@color/btn_red"/>
</LinearLayout>
```

Codice 3.16: HomePageNoLogin.java

```
//per mostrare la finestra di dialogo
private AlertDialog.Builder dialogBuilder;
private AlertDialog dialog;
//per interagire con gli elementi grafici della finestra di dialogo
private EditText popup_email;
private Button btn_sendRecoveryCode;
private TextView btn_cancel_operation;
private TextView rec_insert_email;
private TextView rec_email_inexistent;

public void pressPasswordRecovery(View v){
```

```
dialogBuilder = new AlertDialog.Builder(this);
int orientation = getResources().getConfiguration().orientation;
if (orientation ==
    android.content.res.Configuration.ORIENTATION_LANDSCAPE) {
final View recoveryPasswordPopupView = getLayoutInflater().
    inflate(R.layout.custom_popup_recovery_password_landscape,
        null);

popup_email = (EditText) recoveryPasswordPopupView.
    findViewById(R.id.popup_email);
rec_insert_email = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.rec_insert_email);
rec_email_inexistent = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.rec_email_inexistent);
btn_sendRecoveryCode = (Button) recoveryPasswordPopupView.
    findViewById(R.id.b_send_code);
btn_cancel_operation = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.b_cancel_operation);

dialogBuilder.setView(recoveryPasswordPopupView);
dialog = dialogBuilder.create();
dialog.getWindow().setBackgroundDrawableResource(
    android.R.color.transparent);
dialog.show();
}
else{
    final View recoveryPasswordPopupView = getLayoutInflater().
        inflate(R.layout.custom_popup_recovery_password, null);

popup_email = (EditText) recoveryPasswordPopupView.
    findViewById(R.id.popup_email);
rec_insert_email = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.rec_insert_email);
rec_email_inexistent = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.rec_email_inexistent);
btn_sendRecoveryCode = (Button) recoveryPasswordPopupView.
    findViewById(R.id.b_send_code);
btn_cancel_operation = (TextView) recoveryPasswordPopupView.
    findViewById(R.id.b_cancel_operation);

dialogBuilder.setView(recoveryPasswordPopupView);
dialog = dialogBuilder.create();
dialog.getWindow().setBackgroundDrawableResource(
```

```
        android.R.color.transparent);
    dialog.show();
}

if(et_email.getText().toString().trim().length() != 0){
    popup_email.setText(et_email.getText().toString());
}

//l'utente preme sul pulsante per inviare il codice di recupero
//password
btn_sendRecoveryCode.setOnClickListener(new
    View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(popup_email.getText().toString().trim().length() == 0) {
            popup_email_ok = false;
            Utility.setVisibilityOn(rec_insert_email);
            changeColor();

        } else if(popup_email.getText().toString().trim().length()
        != 0){

            [ . . . ]

            HomepageNoLogin.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    //metodo che si occupa di inviare l'email e
                    //cambiare la password
                    sendEmailANDChangingPassword();
                    dialog.cancel();
                    et_email.setText(popup_email.getText().toString());
                }
            });
        }
    }
}

[ . . . ]

//l'utente preme sul pulsante per chiudere la finestra di dialogo
btn_cancel_operation.setOnClickListener(new
    View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        dialog.cancel();
    }
});
}
```

Codice 3.17: TakePhoto.java

```
//per mostrare la finestra di dialogo
private AlertDialog.Builder dialogBuilder;
private AlertDialog dialog;
//per interagire con gli elementi grafici della finestra di dialogo
private Button btn_send_anyway;
private TextView btn_come_back;

public void pressContinue(View v){
    if(string_photo != null){

        [ . . . ]

    } else { //se l'utente non ha inviato alcuna foto
        dialogBuilder = new AlertDialog.Builder(this);
        final View dialogShow = getLayoutInflater().
            inflate(R.layout.custom_popup_confirm_operation, null);
        btn_come_back = dialogShow.findViewById(R.id.b_come_back);
        btn_send_anyway = dialogShow.findViewById(R.id.b_continue);
        dialogBuilder.setView(dialogShow);
        dialog = dialogBuilder.create();
        dialog.getWindow().setBackgroundDrawableResource(
            android.R.color.transparent);
        dialog.show();

        //l'utente preme sul pulsante per inviare la segnalazione
        //senza password
        btn_send_anyway.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),
                    MoreInformation.class);
                startActivity(intent);
            }
        });
    };
}

//l'utente decide di ritornare sui suoi passi e torna indietro
```

```
    per scattare la foto
btn_come_back.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dialog.cancel();
    }
});
```

Capitolo 4

Conclusioni

L'obiettivo cardine del lavoro tesi è stato fin dall'inizio quello di trovare una metodologia per la segnalazione degli incendi boschivi, che coniugasse semplicità e velocità. A tal fine è stata realizzata *KARYA reports*, un'applicazione che fornisce un aiuto al suddetto scopo. Il target d'utenza è molto vasto, in quanto la segnalazione di un incendio può avvenire da parte di tutti i cittadini che hanno raggiunto la maggiore età. L'applicazione nasce da due fasi, *la prima* è stata presa in considerazione nella stesura di questa tesi. Nel particolare sono stati effettuati focus riguardanti il *l'interfaccia grafica* dell'applicazione. La *seconda fase* è stata invece presa in considerazione in un secondo lavoro di tesi, intitolato "*KARYA reports: sviluppo di una applicazione Android per la segnalazione di incendi boschivi*", stilato dalla studentessa Mariarosaria Esposito.

Il lavoro svolto nella prima fase di progettazione evidenziato in questa tesi, ha cercato la soluzione ad un quesito in particolare, ovvero "*quale potrebbe essere un possibile design grafico capace di coniugare semplicità e sicurezza, al fine di dare un sentimento positivo per chi la utilizza?*". Per rispondere a questa domanda si è cercato di comprendere in primo luogo le funzionalità che il prodotto deve implementare. Alan Cooper, noto informatico e designer americano, ha infatti affermato che l'obiettivo è quello di "*definire cosa farà il prodotto, prima di progettare come lo farà*" e che "*se vogliamo che gli utenti apprezzino i nostri lavori dobbiamo comportarci come se fossimo noi gli utilizzatori del sistema*". Di fatto, lo scopo principale è sempre stato quello di soddisfare gli utenti, partendo dalla propria della grafica, che rappresenta lo strumento primario con il quale l'utente interagisce con l'applicativo software. Lo studio è partito da zero, e ha determinato tutte le caratteristiche grafiche dell'applicazione, dalla scelta dello stile e della posizione delle varie componenti grafiche, alla preferenza di determinati colori anziché altri. Grazie quindi allo studio e al successivo utilizzo della *scienza cognitiva* è nato

quello che un design creato ad hoc per KARYA reports.

KARYA reports rappresenta solo un primo mattone, infatti l'applicazione al momento approfondisce solo quelle che sono le funzionalità che consentono al singolo cittadino di poter segnalare un incendio. È necessario però che le informazioni raccolte grazie all'aiuto dei singoli civili, vengano fornite alla Sala Operativa dei Vigili del Fuoco, che si occuperà di manipolare tali dati. Un'ipotetica visualizzazione dei dati inviati alla Sala Operativa è mostrato nella mappa in figura 4.1, sulla quale vi sono dei marker a forma di fiamma che evidenziano la posizione degli incendi segnalati.

Concludendo, sono ancora molti gli sforzi che si devono compiere affinché l'applicazione diventi uno strumento fondamentale per i Vigili del Fuoco, nonostante ciò, questa rappresenta un passo importante per innovare e rendere ancora più efficiente il loro prezioso lavoro.

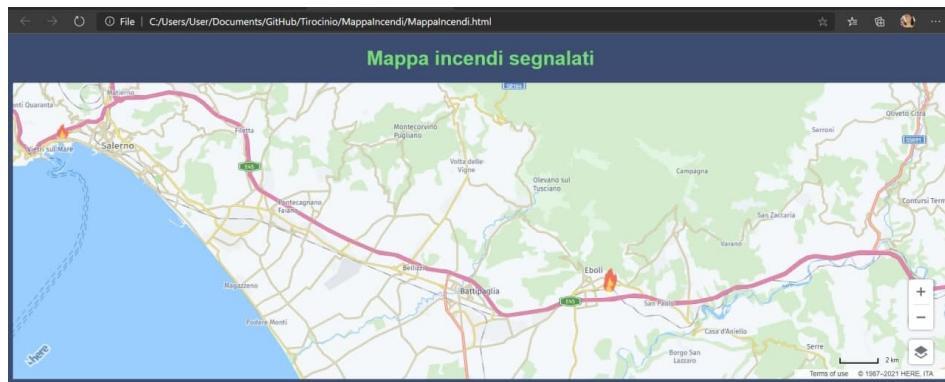


Figura 4.1: Mappa per la visualizzazione in Sala Operativa degli incendi segnalati dai singoli cittadini

Bibliografia

- [1] Comune di Vitulano. *Gli incendi boschivi: lo studio e le soluzioni.*
<https://www.comune.vitulano.bn.it/avvvpc/pages/studio.html>
- [2] Parlamento Italiano *Legge-quadro in materia di incendi boschivi.*
<https://www.camera.it/parlam/leggi/003531.html>
- [3] Wikipedia, l'enciclopedia libera. *Donald Norman*
https://it.wikipedia.org/wiki/Donald_Norman
- [4] Generated Photos | Unique, worry-free model photos *Generated Photos*
<https://generated.photos/>
- [5] Wikipedia, l'enciclopedia libera *Christopher Alexander*
https://it.wikipedia.org/wiki/Christopher_Alexander
- [6] User Interface Design patterns *Patterns*
<http://ui-patterns.com/>
- [7] Colors.co *Creazione della palette dei colori*
<https://colors.co>

Ringraziamenti

Sembra quasi surreale aver raggiunto questo traguardo. Mentirei nel dire di avercela fatta solo grazie a me stessa. Se avessi fatto affidamento solo alla mia persona, avrei mollato alla prima difficoltà. Sono arrivata a questo capitolo della mia vita grazie a tutte le persone che mi hanno sempre sostenuta, che hanno creduto in me fino alla fine. Ognuno ha avuto un ruolo fondamentale all'interno del mio percorso di studi, persino il collaboratore scolastico che con allegria mi salutava con un bel "Ci vediamo domani!". Non ho mai dato peso a quella frase, fino a quando il domani all'interno del campus è col tempo diventato un ricordo.

Ringrazio la mia relatrice, la prof.ssa Giuliana Vitiello, per avermi seguita in questo percorso e il Vigile del Fuoco Domenico Giordano, per la sua immensa disponibilità e per aver reso concreto questo bellissimo progetto.

Ringrazio con tutta me stessa la mia famiglia, perché ha saputo indirizzarmi verso la strada giusta, dandomi tutti gli strumenti a loro disposizione per poter diventare la donna che voglio essere.

Ringrazio Gisella, che mi ha aiutata a capire come approcciarmi allo studio e come riuscire a dare sempre il massimo di me stessa.

Ringrazio la mia vicina di casa Mimma, che mi ha salvata portandomi i documentati durante uno degli esami online.

Ringrazio di cuore Alfredo, Biagio, Emanuele, Eugenio, Giusy e Nicola per aver condiviso con me forse l'esame più lungo della storia del mio corso di studi. Si sono rivelate delle persone fantastiche, capaci di strapparti un sorriso anche nei momenti in cui sembravano regnare malinconia e ansia.

Ringrazio Marco, Sara e Silvio, per aver reso i miei studi in quarantena meno solitari.

Ringrazio Angela, Anna Chiara, Daniele, Francesco B., Francesco F., Giovanni, Giuseppe, Ilaria, Michela e Pio per avermi quasi obbligata a concedermi delle pause costruttive dallo studio. Anche quelle giornate passate a ridere e a scherzare hanno contribuito alla fine di questo percorso.

Sono e sarò perennemente grata alle due persone che hanno reso magico il mio percorso di studi. Incontrare Ciro e Chicca (non riesco a chiamarti Mariarosaria, scusami) è stato un crescendo di emozioni positive. Eravamo

semplici compagni universitari, uniti dalla voglia di studiare e dalla soddisfazione nel superare gli esami. Il tempo ci ha portati a diventare complici di vita. Spero di aver dato loro anche solo un briciole della forza che loro hanno avuto la capacità di donarmi.

Un ringraziamento speciale va a Cesare, che pur non essendone consapevole, mi ha dato la forza di reagire alle situazioni, di crescere sempre di più come persona. Mi ha spronata a dare il massimo di me stessa in ogni cosa, riuscendo a farmi capire che alla fine anche "una come me" può farcela sempre. Sono fiera di avere lui nella mia vita, e spero di poter contare sulla sua presenza non solo in questo percorso, ma in tanti altri che seguiranno.

Francesca, febbraio 2021