UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA
E MATEMATICA APPLICATA

Corso di Laurea in Ingegneria Informatica

Elaborato di Laurea

Monitor4U": progettazione e sviluppo di un'app iOS a supporto della sicurezza domestica "

Relatore:

Ch. Prof. Luca Greco

Candidato:

Beniamino Squitieri

Mat. 0612704798

ANNO ACCADEMICO 2021/2022

Sommario

Introduzione

L'esigenza di tenere sotto controllo la propria abitazione quando non si è a casa costituisce uno dei principali problemi per quanto riguarda la sicurezza delle proprie dimore. Questo forte stato di ansia e preoccupazione causato sia dalla possibile intromissione di intrusi, sia da eventuali fughe di gas che possono causare danni notevoli sia alle case che alle persone, non consente di godere dei momenti importanti della vita quotidiana, come una semplice passeggiata con la famiglia, o di concentrarsi a pieno nel lavoro o in qualsiasi altro contesto dove sia richiesta la massima attenzione, impedendo di poter rendere al meglio.

Dalle analisi condotte negli ultimi anni, le denunce di furti in abitazione hanno superato quota 182mila. In altre parole, quasi 500 al giorno, più di venti all'ora, uno ogni tre minuti circa. Quella dei furti in appartamento è una piaga così diffusa che viene studiata e monitorata sistematicamente. L'Osservatorio sulla Criminalità italiana de Il Sole 24 ore realizza annualmente un'indagine che rileva, incrociando i dati delle denunce sporte per i vari reati, la geografia dei crimini commessi sul territorio, provincia per provincia. La peggiore è Ravenna **c**on 410 denunce ogni 100.000 abitanti, seguono Bologna con 360, e Modena con 328. La struttura urbanistica bolognese facilita molto le effrazioni, poiché le stradine del centro storico ospitano edifici antichi, vulnerabili tanto attraverso le porte che le finestre. Secondo l'ISTAT negli ultimi 10 anni i furti in casa denunciati sono più che raddoppiati: l'ordine di grandezza è di 689 al giorno, cioè 29 all'ora, uno ogni due minuti e sono concentrati principalmente nelle città del Centronord. Peggiora sensibilmente la tendenza delle denunce anche in città apparentemente tranquille, come: Livorno, Rieti, Crotone, Matera e Potenza, Campobasso, Firenze, Pistoia.
Il problema dei furti, quindi, risulta aggravarsi sempre di più con il passare del tempo, sebbene molti italiani (e non solo) tendino a prestare attenzione a quei dettagli come chiudere le finestre, la porta di casa o in casi estremi usare anche una doppia serratura.
Per quanto riguarda le fughe di gas, oltre alle cause più note, come il lasciare acceso qualche attrezzo elettrodomestico (fornelli da cucina nella maggior parte dei casi), vi possono essere anche altre cause non imputabili direttamente agli abitanti dell'abitazione. Numerosi studi statistici hanno evidenziato un aumento soprattutto nelle grandi città come Roma, dove la procura ha addirittura aperto un fascicolo. Si indaga per "pericolo contro l'incolumità pubblica", per via delle fughe di gas causate dalle troppe condutture spaccate durante i lavori stradali. Sebbene la fuga non si avvenuta direttamente nelle dimore delle persone può comunque rappresentare un problema in quanto porta ad inquinare l'aria circostante.

Tutte queste problematiche descritte rappresentano quanto l'elaborato tratterà in termini di progettazione ed implementazione. Il focus è definito in relazione a Monitor4U, un'applicazione in grado di permettere all'utilizzatore il costante monitoraggio dello stato di sicurezza della propria abitazione, mirando al monitoraggio della qualità dell'aria,

avvertendo nel caso in cui il livello dell'aria non si attesti intorno a valori ottimali e all'eventuale presenza di intrusi nei pressi dell'abitazione in quanto è collegata ad un sistema di allarme che suona nell'eventualità in cui un individuo non autorizzato si trovi nei pressi della propria dimora senza alcun permesso. L'applicazione permette anche il monitoraggio di più case (nell'eventualità in cui la persona ne sia in possesso) tramite una mappa in esso presente e collegata anche al servizio di Mappe offerto da Apple. Per ragioni di privacy è stato introdotto un servizio di autenticazione, sia tramite e-mail sia tramite Facebook, in maniera tale da evitare che informazioni personali delle proprie abitazioni come locazione o dispositivi per il monitoraggio attivi sia reso noto ad estranei. Infine, nell'interfaccia home dopo aver effettuato il login vi è una sezione grazie alla quale è possibile accedere ad una schermata dove vi è una sintesi dei dispositivi presenti nelle proprie abitazioni ed inoltre è possibile anche aggiungerne una qualora un individuo si trasferisca in una nuova località.

Quanto concerne la stesura dell'elaborato, quest'ultimo sarà strutturato in quattro capitoli. Nel capitolo 1 si discuterà delle tecnologie utilizzate, sia hardware che software, per la realizzazione dell'obiettivo proposto: nello specifico delle componenti software, l'utilizzo di piattaforme quali Visual Studio Code per l'implementazione delle funzionalità lato "controllori collegati alle abitazioni" attraverso il linguaggio Python, XCode attraverso il linguaggio Swift e i framework MapKit,FacebookLogin,Firebase, MQTT, SwiftUI per l'implementazione delle funzionalità lato applicazione, la piattaforma Figma per quanto riguarda l'UI e UX; per quanto riguarda le componenti hardware, tutte le componenti relative allo "Sistema di Controllo e Cura" quali l'MCU (Micro Controller Unit) "Zerynth ZM1-Eva" e l'insieme dei sensori utilizzati per l'implementazione di quanto messo a disposizione dal dispositivo.

Successivamente, attraverso il capitolo 2, si discuterà la fase progettuale di Monitor4U: nello specifico sarà discusso l'approccio tramite Challenge Based Learning attraverso cui è stata formalizzata l'idea e la strutturazione del progetto, tutta la parte di progettazione della User Interface e della User Experience attraverso il wireframing di tutte le viste appartenenti all'applicazione, l'integrazione dei vari framework per le funzionalità del progetto e la gestione degli avvisi rivolti all'utente nel caso di intrusi.

Nel capitolo 3 il focus sarà spostato sulla fase di implementazione di quanto discusso nella fase progettuale, a supporto della quale verrà fornita una descrizione delle funzionalità del progetto, l'implementazione delle viste principali dell'applicazione con le funzionalità a corredo, la gestione dei charts utilizzati per rappresentare in sintesi i risultati dei controllori, ed infine le funzionalità implementate per la User Interface & User Experience, il tutto accompagnato da una serie di esempi dimostrativi dell'utilizzo delle funzionalità descritte.

Infine, nel capitolo 4 viene presentata una sintesi del lavoro sviluppato, tenendo conto di quanto affermato nei capitoli precedenti e discutendo di tutti gli spunti di miglioramento ed evoluzione rispetto alla progettazione ed implementazione condotta, in particolar modo facendo riferimento alle migliorie rivolte su aspetti ben specifici del progetto sviluppato quali il servizio di autenticazione riservato all'utente e alla connessione con il sistema di controllo (SafeAndCareSystem).

1. Panoramica delle tecnologie utilizzate

In evidenza di seguito saranno elencate tutte le componenti, sia hardware che software, che hanno dato vita al progetto di tesi, a partire dalla fase di progettazione sino a giungere alla fase di realizzazione. Si definiranno dunque i linguaggi di programmazione utilizzati, le piattaforme utilizzate per lo sviluppo sia lato applicazione che lato SafeAndCareSystem (sistema di controllo), la definizione di quest'ultimo, la scelta dei sensori che lo riguardano e la gestione dell'interfacciamento tra l'applicazione e il sistema di controllo.

1.1. Piattaforme utilizzate

Per lo sviluppo del software e della comunicazione tra l'applicazione e il sistema di controllo si è fatto utilizzo di cinque piattaforme:

- Visual Studio Code
- XCode
- GitHub
- Figma
- UIzard

Mentre lato della connettività si è fatto utilizzo di:

- Firebase
- Meta for developers

1.1.1. Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e MacOS. Include il supporto per debugging, un controllo per GIT integrato, snippet e refactoring del codice, syntax highlighting e tra le tante altre feature anche la possibilità di utilizzare framework esterni mediante i quali è possibile programmare software sull'ambiente nativo, come nel caso della casa produttrice dell'ESP-32 utilizzata nella realizzazione del sistema di controllo (SafeAndCareSystem). Visual Studio Code è un editor di codice sorgente che può essere utilizzato con vari linguaggi di programmazione, tra cui la famiglia di linguaggi C, HTML, altri linguaggi web, tra cui PHP, Java, Ruby e molti altri tra cui Python, utilizzato per lo sviluppo e il caricamento del software verso l'MCU associata al sistema di controllo. Incorpora un insieme di funzioni che variano a seconda del linguaggio che si sta utilizzando.

1.1.2. XCode

XCode è un ambiente di sviluppo integrato, completamente sviluppato e mantenuto da Apple contenente una suite di strumenti utili allo sviluppo di software per i sistemi macOS, iOS, iPadOS, watchOS e tvOS. Estende e rimpiazza il precedente tool di sviluppo della

Apple, *Project Builder*, che era stato ereditato dalla NeXT e lavora in congiunzione con Interface Builder, un tool grafico per la realizzazione delle interfacce grafiche. Una delle caratteristiche tecnologicamente più avanzate di XCode è che supporta la distribuzione in rete del lavoro di compilazione. Oltretutto supporta anche la compilazione incrementale, ovvero XCode è in grado di compilare il codice mentre viene scritto, in modo da ridurre i tempi di compilazione. Dalla versione 3.1, XCode è anche lo strumento per sviluppare applicazioni native per IPhone e IPod touch. Dalla versione 3.2 è invece possibile anche sviluppare applicazione per IPad. Con il rilascio della versione 6 è stato aggiunto il nuovo linguaggio di programmazione Swift ed il Live Rendering, che permette di visualizzare oggetti in tempo reale mentre vengono sviluppati, così come verrebbero visualizzati a runtime. Anche le modifiche fatte alle proprie *custom view* vengono mostrate in tempo reale e si possono definire comportamenti diversi per diverse misure del display che sono mostrati sempre in tempo reale con proprietà definite da Interface Builder e anche con dati fittizi per pre-popolare le UI in modo da poter avere esempi reali di come apparirà l'interfaccia grafica. Un altro vantaggio di XCode è che fornisce il controllo della versione tramite GIT e subversion. È facile eseguire branch e unire perfettamente le operazioni per i team distribuiti. È facile confrontare due versioni di file, vedere i registri di commit e chi ha apportato modifiche al codice utilizzando l'editor delle versioni. Grazie a ciò è stato reso possibile lo sviluppo in team delle diverse funzionalità delle quali l'applicazione è composta. XCode 6 include anche un altro strumento, denominato Playground. Esso è un'area di test dove poter iniziare ad apprendere il linguaggio velocemente, scrivendo codice con risultati mostrati in tempo reale. XCode supporta C, C++, Objective-C, Objective C++, Java, AppleScript, Python, Ruby, Rex e Swift, con una varietà di modelli di programmazione inclusi e non limitati a Cocoa, Carbon e Java. Tool di terze parti permettono il supporto a GNU Pascal, Free Pascal, Ada, C#, Perl e D.

1.1.3. GitHub

GitHub è un servizio di hosting per progetti software, di proprietà della società *GitHub Inc*. Il nome deriva dal fatto che "*GitHub*" è una implementazione dello strumento di controllo di versione distribuito GIT, ovvero una tipologia di controllo di versione che permette di tenere traccia delle modifiche e delle versioni apportate al codice sorgente del software. Con questo sistema gli sviluppatori possono collaborare individualmente e parallelamente non connessi su di un proprio ramo (branch) di sviluppo, registrare le proprie modifiche (commit) ed in seguito condividerle con altri o unirle (merge) a quelle di altri, senza bisogno del supporto di un server centralizzato. Grazie a GitHub e allo strumento di controllo di versione distribuito Git su cui esso si fonda, è stata resa possibile la collaborazione parallela che ha portato alla realizzazione del progetto, tramite i servizi offerti come il sistema di issue tracking, pull request e commenti che permette di migliorare il codice del repository, ovvero un ambiente di un sistema informativo in cui vengono gestiti i metadati, attraverso tabelle relazionali.

1.1.4. Figma

Figma è un tool per sviluppare grafica per i più disparati fini. Locandine, bigliettini da visita, siti web, app Android, app iOS o software desktop. Si propone come alternativa al più conosciuto Sketch o al rivale Adobe Experience Design e non ha assolutamente nulla da invidiare loro.

Vediamo nel dettaglio le sue caratteristiche, offre la possibilità di:

- Integrare gli elementi grafici **nativi** di Android e iOS nei propri progetti;
- Customizzare dimensioni dell'ambiente di lavoro o scegliere tra quelle adatte a piattaforme specifiche;
- Esportare il codice in CSS, XML o Swift;
- Realizzare animazioni per ottenere un vero prototipo;
- Modificare immagini vettoriali.

Completezza ed attenzione alle esigenze dell'autore sono le sue parole chiave. L'interfaccia utente è assolutamente gradevole e semplice da capire per facilitare l'avvicinamento delle mani più inesperte e curiose.

1.1.5 Firebase

Firebase è un potente servizio on line che permette di salvare e sincronizzare i dati elaborati da applicazioni web e mobile. Si tratta di un database NoSQL dalle grandissime risorse, ad alta disponibilità ed integrabile in tempi rapidissimi in altri progetti software, semplicemente sottoscrivendo un account al servizio. Firebase rientra in una categoria di servizi on-line in rapidissima diffusione e noti come backend, la cui necessità è divenuta sempre più evidente negli ultimi anni. Le crescenti capacità di dispositivi e sistemi operativi mobili, insieme alle connessioni in rete sempre più diffuse e veloci hanno offerto nuovi strumenti alla fantasia degli sviluppatori. In questo contesto le nuove applicazioni tengono ad essere sempre meno "chiuse" nei dispositivi, e maggiormente capaci di interagire con servizi remoti creando reti "social" di utenti.
Per consentire il funzionamento di tutto ciò, è necessaria una controparte server in grado di rendere disponibili, tramite API, servizi quali autenticazione, storage dei dati, push notification, comunicazione tra utenti e molto altro: i backend, appunto.
La creazione di un backend per conto proprio può portare gli sviluppatori ad incontrare delle difficoltà di vario genere. Innanzitutto, di carattere tecnico: i programmatori potrebbero non avere le competenze e l'esperienza necessarie in ambito server per realizzare un backend di qualità, sicuro ed efficiente. Inoltre, potrebbero esservi difficoltà organizzative, per esempio nel conciliare tempistiche e risorse umane coinvolte nello sviluppo delle componenti server e client.
Esistono, infine, problematiche di carattere infrastrutturale. Viviamo nell'epoca del Cloud, delle risorse e delle funzionalità offerte "as a service", in cui senza avere a disposizione proprie macchine fisiche o virtuali si possono trovare servizi in rete che offrono elaborazione e storage in maniera affidabile, flessibile e con costi contenuti.

In casi in cui si richiede un backend per la propria app mobile o web ma si vogliono evitare difficoltà come quelle appena citate, si può fare affidamento ad un servizio online, offerto in cloud e già pronto all'interazione con i nostri software: Firebase è uno di questi.
Le caratteristiche peculiari di Firebase sono:

- capacità di sincronizzazione dei dati oltre che di storage: Firebase è in grado di aggiornare i dati istantaneamente, sia se integrato in app web che mobile. Non appena l'app recupera la connettività sincronizza i dati mostrati con le ultime modifiche apportate;
- disponibilità di librerie client per integrare Firebase in ogni app. Android, Javascript e framework con esso realizzati (Node.js, Angular.js, Ember.js, Backbone.js e molti altri), Java e sistemi Apple: per tutte le più comuni tecnologie web e mobile esistono librerie già pronte per essere importate nei propri progetti;
- API REST: rendono disponibili le funzionalità di Firebase per ogni tecnologia per cui non esistano librerie apposite o in caso di operazioni non contemplate in esse;
- Sicurezza: i dati immagazzinati in Firebase sono replicati e sottoposti a backup continuamente. La comunicazione con i client avviene sempre in modalità crittografata tramite SSL con certificati a 2048-bit;
- costi differenziati in base all'uso e alle capacità richiesti.

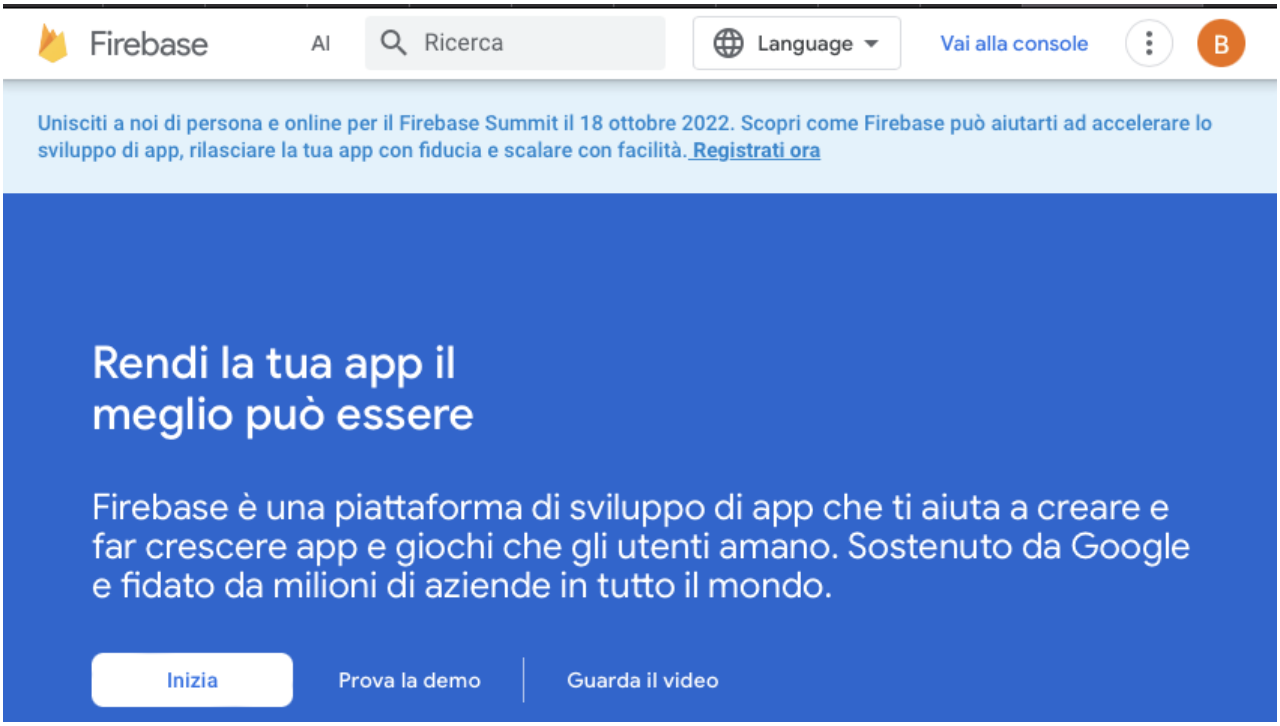

Figura 1 Firebase Dashboard

Firebase offre una documentazione chiara e completa, accessibile tramite l'apposita sezione ufficiale. Vengono mostrate subito le tecnologie più comunemente usate con Firebase, e per ognuna di esse è possibile consultare documentazioni di tipo differente:

- Quick start: guida rapida per iniziare ad usare in pochi minuti le funzionalità;
- Guida: realizzata passo passo per accompagnare l'utente in un approfondimento imperniato sui capisaldi di Firebase;
- API Reference con la documentazione approfondita.

A corredo, inoltre, ci sono esempi di codice e librerie da scaricare liberamente.
In ogni caso, risulta necessario per prima cosa ottenere un account di accesso al servizio
All'account creato sarà associato un indirizzo, nel formato <username>. firebaseio.com.
Per quanto riguarda lo storage dei dati in Firebase è importante sottolineare come non vi siano tabelle né record, nessuna traccia dell'approccio relazionale. Tutti i dati inseriti in Firebase vanno a costituire un albero JSON. I dati al suo interno possono essere strutturati secondo questo formato: si possono inserire mappe, liste, stringhe, tipi numerici e booleani.
Infine, per i più novizi Firebase mette anche a disposizione una console vera e propria grazie alla quale è possibile comunicare con il database senza la necessità di scrivere codice.

1.1.6 Meta for developers

La piattaforma Meta(Facebook) è l'insieme di servizi, strumenti e prodotti forniti dal servizio di social networking Facebook per gli sviluppatori di terze parti per creare le proprie applicazioni e servizi che accedono ai dati in Facebook.
L'attuale piattaforma Facebook è stata lanciata nel 2010. La piattaforma offre una serie di interfacce e strumenti di programmazione che consentono agli sviluppatori di integrarsi con il "grafico sociale" aperto delle relazioni personali e altre cose come canzoni, luoghi e pagine Facebook. Le applicazioni su facebook.com, siti web esterni e dispositivi sono tutte autorizzate ad accedere al grafico.
Offre numerosi servizi, tra i quali quello di Facebook Connect (Facebook Login)

1.1.6.1. Facebook Connect

Facebook Connect, chiamato anche Accedi con Facebook, come OpenID, è un insieme di API di autenticazione di Facebook che gli sviluppatori possono utilizzare per aiutare i loro utenti a connettersi e condividere con gli amici di Facebook di tali utenti (da e fuori Facebook) e aumentare il coinvolgimento per il loro sito web o applicazione. Quando vengono utilizzati, i membri di Facebook possono accedere a siti web, applicazioni, dispositivi mobili e sistemi di gioco di terze parti con la loro identità Facebook e, mentre sono connessi, possono connettersi con gli amici tramite questi media e pubblicare informazioni e aggiornamenti sul loro profilo Facebook.
L'accesso con Facebook non può essere utilizzato dagli utenti in posizioni che non possono accedere a Facebook, anche se il sito di terze parti è altrimenti accessibile da quella posizione.

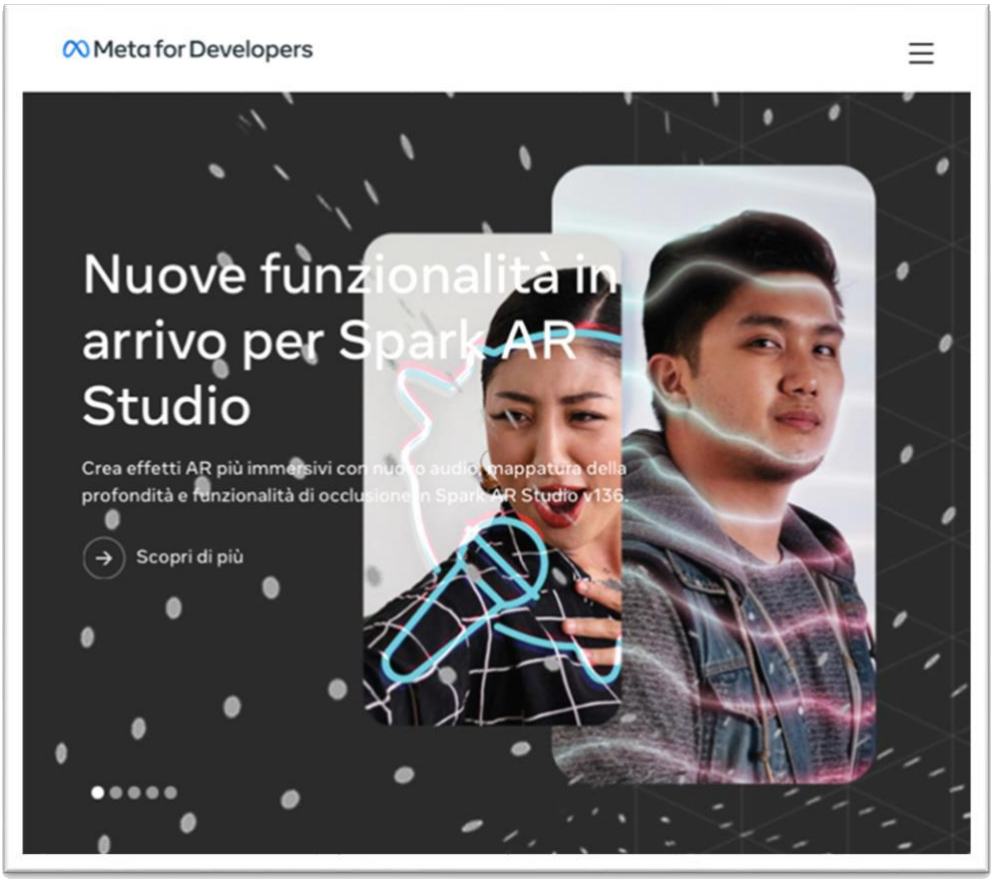

Figura 2 Meta for developers' dashboard

1.1.7 UIZard

UIzard è uno strumento di prototipazione rapida utilizzato per trasformare automaticamente i tuoi wireframe in prototipi, creare guide di stile personalizzate, esportare come file Sketch, scaricare il codice frontend e iterare il più velocemente possibile. Scompone il problema e il tempo per trasformare il wireframe a bassa fedeltà in un wireframe ad alta fedeltà.
Uizard utilizza algoritmi di visione artificiale e di apprendimento automatico per trasformare le immagini wireframe catturate da uno schizzo disegnato a mano in mockup ad alta fedeltà. Ha un sistema di guida di stile integrato che è tre per impostazione predefinita (mup semplice, accogliente e avventuroso) per personalizzare i componenti nell'interfaccia utente e anche un motore prototipo per costruire un flusso utente interattivo.
Uizard offre la possibilità di trasformare i tuoi wireframe su carta o su lavagne in file Sketch che possono essere importati nei tuoi strumenti di progettazione preferiti.
Uizard converte i tuoi wireframe in codice che ti consente di visualizzare le tue idee senza lo stress di riscrivere il codice semplicemente configurando le preferenze di codice, scegliere una piattaforma di destinazione HTML e CSS, React o Android e generare il codice in pochi secondi.
Per utilizzare Uizard basta andare sul sito "https://app.uizard.io " accedere con il proprio account e creare un nuovo prototipo grazie alle linee guida fornite dal sito stesso. Il sito

risulta molto utile e versatile in quanto consente anche la creazione di mockup e wireframes partendo da bozze fatte a mano e scannerizzate tramite la fotocamera del proprio telefonino.

1.2. Linguaggi di programmazione

I linguaggi di programmazione utilizzati per lo sviluppo del progetto di tesi sono:

- Swift, per la realizzazione dell'applicazione per iOS;
- Python, a supporto della realizzazione e l'interfacciamento del software verso l'MCU associata al sistema di controllo.

1.2.1. Python

Python è un linguaggio di programmazione ad alto livello, rilasciato pubblicamente per la prima volta nel 1991 dal suo creatore Guido van Rossum, programmatore olandese attualmente operativo in Dropbox. Deriva il suo nome dalla commedia *Monty Python's Flying Circus* dei celebri Monty Python, in onda sulla BBC nel corso degli anni 70. Attualmente, lo sviluppo di Python (grazie e soprattutto all'enorme e dinamica comunità internazionale di sviluppatori) viene gestito dall'organizzazione no-profit Python Software Foundation.
Python supporta diversi paradigmi di programmazione, come quello object-oriented (con supporto all'ereditarietà multipla), quello imperativo e quello funzionale, ed offre una tipizzazione dinamica forte. È fornito di una libreria built-in estremamente ricca, che unitamente alla gestione automatica della memoria e a robusti costrutti per la gestione delle eccezioni fa di Python uno dei linguaggi più ricchi e comodi da usare.
Comodo, ma anche semplice da usare e imparare. Python, nelle intenzioni di Guido van Rossum, è nato per essere un linguaggio immediatamente intuibile. La sua sintassi è pulita e snella così come i suoi costrutti, decisamente chiari e non ambigui. I blocchi logici vengono costruiti semplicemente allineando le righe allo stesso modo, incrementando la leggibilità e l'uniformità del codice anche se vi lavorano diversi autori.
Python è un linguaggio pseudocompilato: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo. In Python, non esiste una fase di compilazione separata (come avviene in C, per esempio) che generi un file eseguibile partendo dal sorgente.
L'esser pseudo-interpretato rende Python un linguaggio portabile. Una volta scritto un sorgente, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme attualmente utilizzate, siano esse di casa Apple (Mac) che PC (Microsoft Windows e GNU/Linux). Semplicemente, basta la presenza della versione corretta dell'interprete.
Infine, Python è free software: non solo il download dell'interprete per la propria piattaforma, così come l'uso di Python nelle proprie applicazioni, è completamente gratuito; ma oltre a questo Python può essere liberamente modificato e così ridistribuito, secondo le regole di una licenza pienamente open-source.
Queste caratteristiche hanno fatto di Python il protagonista di un enorme diffusione in tutto il mondo, e anche in Italia, negli ultimi anni. Questo perché garantisce lo sviluppo rapido (e divertente) di applicazioni di qualsiasi complessità in tutti i contesti: dal desktop al web, passando dallo sviluppo di videogiochi e dallo scripting di sistema.

Dopotutto, se è vero che Python rappresenta una delle tecnologie principali del core business di colossi come Google (YouTube è pesantemente basato su Python) e ILM, allora probabilmente sarà ottimo anche per le tue necessità.

1.2.2. Swift

Swift è un linguaggio di programmazione orientato agli oggetti per sistemi macOS, iPadOS, iOS, watchOS, tvOS e Linux, presentato dalla Apple durante la WWDC 2014. Swift è concepito per coesistere con il linguaggio Objective-C, tipico degli sviluppi per i sistemi operativi Apple, semplificando la struttura del codice. Rispetto ad Objective-C, il codice risulta più conciso e leggibile. È stato introdotto il concetto di type inference grazie al quale non è necessario specificare esplicitamente il tipo di variabili e costanti se questo può essere determinato dal contesto. Non è più necessario avere un header file ed un implementation file separato per ogni classe, ma la definizione di queste avviene in un unico file. Possiamo dunque affermare che Swift è un successore sia dei linguaggi C che Objective-C. Include primitive di basso livello come tipi, controllo del flusso e operatori. Fornisce anche funzionalità orientate agli oggetti come classi, protocolli e generici, dando agli sviluppatori Cocoa e Cocoa Touch le prestazioni e la potenza che richiedono.

La prima versione di Swift è stata rilasciata il 9 settembre 2014, mentre l'ultima versione risale al 14 marzo 2022.

1.2.2.1. SwiftUI

SwiftUI è un framework di sviluppo realizzato da Apple e scritto interamente in Swift. Presentato nel WWDC 2019, SwiftUI è stato pensato per costruire interfacce utente tramite un unico set di tool e API che coinvolge tutto l'ecosistema Apple, ricordando che precedentemente si faceva uso di frameworks distinti: AppKit per macOS, UIKit per iOS e WatchKit per watchOS. A differenza di UIKit, SwiftUI per la realizzazione delle interfacce grafiche sfrutta un modello di tipo "dichiarativo": non si creano degli oggetti, ma si va a dichiarare quale deve essere la struttura grafica dell'interfaccia, si costruiscono componenti grafici complessi a partire da componenti grafici base e componenti diversi sono allineati tra loro tramite l'ausilio di altri specifici componenti grafici, che si prendono cura di definire i vincoli, senza che sia lo sviluppatore a farlo. La logica di business comunica alla schermata non solo quali informazioni deve mostrare, ma anche come deve cambiare la struttura grafica stessa, senza dover andare ad agire sui singoli componenti (o oggetti, nel caso di UIKit). Infine, i componenti grafici prima definiti inviano sempre degli eventi alla business logic, ma la gestione di questi ultimi è molto più semplice rispetto ad UIKit. SwiftUI è stato progettato dunque per costruire interfacce grafiche complesse molto più velocemente e con un rischio minore di introdurre bug oltre che implementare feature quali Dark Mode, font dinamici e animazioni custom, con il vantaggio di riutilizzare le stesse componenti grafiche su più fronti. Grazie al supporto di tutte le piattaforme Apple, il risultato del suo utilizzo è che una sola applicazione scritta con SwiftUI ne sostituirà diverse, con logico risparmio di tempi

e costi di sviluppo. Questo, insieme alla facilità di apprendimento e di utilizzo del framework mediante il quale è possibile definire interfacce grafiche con diversi livelli di complessità, rappresenta il motivo principale per il quale la realizzazione e progettazione del software relativo all'app iOS sviluppata è stata condotta tramite l'utilizzo di questo framework, garantendo comunque tramite l'ausilio di appositi bridge la sua interoperabilità con UIKit, utilizzato durante lo sviluppo per la gestione di altri framework nativi del framework appena citato.

1.3. Componenti Hardware

Oltre ad un sistema software, la realizzazione del progetto di tesi è accompagnata anche da un sistema hardware, il sistema di controllo (SafeAndCareSystem), mediante il quale è stata resa possibile l'automatizzazione di tutte le azioni che accompagnano nel monitoraggio dell'abitazione domestica. Alla base della sua realizzazione vi è la presenza del board ZM1-EVA della casa produttrice Zerynth, attraverso la quale è stato reso possibile lo sviluppo del software a corredo delle funzionalità di cui il sistema è dotato, permettendo di interconnettere fra di essi e con il mondo esterno un insieme di sensori, analogici e digitali, scelti in base alle loro funzionalità e soprattutto alla loro efficienza.

1.3.1. ZM1 Evaluation Board

ZM1-EVA consente la programmazione e il debug di applicazioni IoT utilizzando il sistema operativo Zerynth, attraverso l'SDK ufficiale di Zerynth, utilizzabile nativamente in Visual Studio Code, tramite il linguaggio Python. Tutti i pin e l'elettronica che la definiscono sono fondamentali per il funzionamento dell'MCU. Il modulo principale è un ESP-32. I pin che caratterizzano la board sono General Purpose Input Output, non assegnati ad una particolare funzione e dunque utilizzabili per scopi diversi. Essi sono utilizzabili sia come Digital Input che come Digital Output, oltre che ad avere a disposizione una serie di funzioni addizionali riservate ad alcuni pin come l'Analog/Digital Converter, attraverso il quale è possibile convertire un voltaggio analogico in un valore numerico.

1.3.2. Sensori utilizzati

A supporto del sistema software, l'interfacciamento del sistema di controllo verso l'esterno è stato realizzato tramite l'ausilio di diversi sensori, ognuno dei quali assegnati ad un ruolo specifico. Essendo il sistema di controllo proposto come un sistema in grado di rilevare la presenza di intrusi e valutare la qualità dell'aria la parte dei sensori utilizzati per il raggiungimento di tali obiettivi comprende:

- RFID (Radio Frequency Identification): una tecnologia di comunicazione wireless. Un sistema RFID completo è generalmente costituito dal transponder e dal lettore. Il lettore è un dispositivo per leggere (o scrivere) informazioni sui tag. I prodotti derivati dalla tecnologia RFID possono essere suddivisi in tre categorie: prodotti RFID

passivi, attivi e semi-attivi. I passivi sono i primi, i più maturi e i prodotti più utilizzati sul mercato, tra gli altri.

- Un buzzer è un componente audio. Ci sono sia tipi attivi che passivi. I buzzer attivi hanno un oscillatore all'interno, questi suoneranno finché l'alimentazione è fornita. Sono davvero facili da usare perché producono solo una frequenza sonora specifica.
- Il modulo Gas Sensor (MQ2) è utile per il rilevamento di perdite di gas (in casa e nell'industria). Può rilevare gas combustibili e fumo. La tensione di uscita dal sensore Gas aumenta proporzionalmente con la concentrazione di gas. La sensibilità può essere regolata ruotando il potenziometro.
- LED RGB; non sono altro che tre tipi di LED montati sulla striscia: rosso, verde e blu (questo è ciò che sta per RGB). Mescolando colori con intensità diverse e poi sovrapponendo un colore all'altro, si può avere una luce caratterizzata da diverse sfumature che illuminano l'ambiente in modo elegante; ad esempio, il viola chiaro si ottiene mescolando la luce blu con la luce rossa
- Un motore DC il quale ha due uscite: positiva e negativa. La corrente che scorre nel motore carica gli avvolgimenti posti alle estremità di un rotore centrale con segno opposto. Il campo magnetico generato è costantemente spinto da magneti, posti ai lati del rotore. In questo modo il motore rimane sempre in movimento. Se colleghiamo due positivi o due negativi sugli ingressi del motore, si ferma. Se, invece, colleghiamo un positivo e un negativo, girerà in una direzione o nell'altra.
- Uno stepper motor: un dispositivo di controllo ad anello aperto, che converte un segnale di impulso elettronico in spostamento angolare o spostamento lineare. In una condizione di non sovraccarico, la velocità del motore e la posizione degli arresti dipendono solo dalla frequenza del segnale degli impulsi e dal numero di impulsi e non sono influenzati dalle variazioni di carico come con un motore DC.
- LCD1602, o display a cristalli liquidi di tipo 1602 caratteri, è una sorta di modulo a matrice di punti per mostrare lettere, numeri e caratteri e così via. È composto da posizioni di matrice di punti 5x7 o 5x11; ogni posizione può visualizzare un carattere. C'è un punto tra due caratteri e uno spazio tra le righe, separando così caratteri e righe. Il modello 1602 significa che visualizza 2 righe di 16 caratteri.
- Un sensore ad ultrasuoni il quale ha un campo di misura che va da due centimetri a quattro metri e la precisione raggiunge 1 cm. Il modulo include il trasmettitore ad ultrasuoni, il ricevitore e il circuito di controllo. Questi sensori a ultrasuoni non misurano direttamente la distanza, ma forniscono il tempo necessario affinché un segnale acustico raggiunga un ostacolo e ritorni nuovamente al sensore.
- Un Vibration Switch, dispositivo che riconosce l'ampiezza della vibrazione a cui è esposto e fornisce una sorta di risposta quando questa ampiezza supera un valore di soglia predeterminato. La risposta dell'interruttore è in genere una chiusura o apertura di un contatto elettrico.

2. Progettazione della soluzione

In questo capitolo si tratterà la progettazione, l'implementazione e l'integrazione di tutti i componenti che formano il progetto di tesi designato. In particolar modo, verranno messi in

evidenza tutti gli aspetti salienti che riguardano la progettazione software, a partire dall'idea sulla quale il progetto si fonda sino ad arrivare alla sua concretizzazione. Il processo di sviluppo dell'idea che ha portato alla sua realizzazione è stato condotto mediante la tecnica del Challenge Based Learning, un approccio didattico multidisciplinare coinvolgente ideato da Apple che ha come obiettivo principale sfruttare la tecnologia d'uso quotidiano per risolvere problemi del mondo reale.

2.1. Challenge Based Learning

Il Challenge-Based Learning (CBL) è un approccio pedagogico attraverso il quale gli studenti e le studentesse sono coinvolti attivamente per identificare, analizzare e disegnare una soluzione che risolva una challenge, ovvero una sfida su problematiche attuali e temi reali.
Il Challenge-Based Learning può essere una modalità di apprendimento utile poiché presenta tre fasi principali**:** engagement**,** ovvero l'impegno che studenti e studentesse si assumono nell'affrontare una sfida, nel definire il problema da risolvere e nel porsi le giuste domande; investigate**,** cioè la fase di indagine attraverso cui si trovano e si analizzano le informazioni rilevanti; act**,** ovvero la fase di progettazione, implementazione e valutazione della soluzione.
Il Challenge Based Learning non va confuso con il Project Oriented Learning o con il Problem Based Learning. A differenza degli ultimi due, che si basano sull'analisi di casi studio o sulla risoluzione di un problema fittizio, il CBL è imperniato nella risoluzione di un problema attuale fornito da un "challenge provider", cioè un'azienda o un'istituzione motivata a trovare una soluzione rispetto a una tematica reale.
Nel CBL, infatti, si possono distinguere tre momenti di valutazione: iniziale, che consente di misurare il livello delle proprie competenze, prima dell'inizio dell'esperienza; formativa**,** che consente di monitorare il progresso degli studenti durante l'apprendimento, in modo da applicare eventuali aggiustamenti; sommativa che è quella applicata sul risultato ottenuto alla fine del percorso.

2.1.1. Engage

Durante la fase di coinvolgimento, si passa da una grande idea astratta (big idea) a una sfida concreta e fattibile utilizzando il processo di domande essenziali (essential question). L' obiettivo è quello di connettersi personalmente con il contenuto dell'idea attraverso l'identificazione e lo sviluppo di una sfida avvincente (challenges).

2.1.1.1. Big Ideas

Una Big Idea è un tema o un concetto ampio che può essere esplorato in più modi. Esempi di Big Ideas includono Comunità, Relazioni, Sostenibilità, Creatività e soprattutto Salute, la big idea associata alla nostra progettazione, dalla quale tutto è stato progettato e sviluppato seguendo con attenzione le fasi designate in precedenza.

2.1.1.2. Essential Questioning

Per la progettazione, a partire dalla big idea è possibile generare un'ampia varietà di domande essenziali che riflettono gli interessi personali e i bisogni della comunità. Alla fine del processo di Essential Questioning c'è l'identificazione di una domanda essenziale che ha un significato personale e permette di proseguire con le fasi successive dell'approccio Challenge Based Learning.

2.1.1.3. Challenges

La sfida trasforma la domanda essenziale in una chiamata all'azione per imparare ancor di più sull'argomento trattato. La fase di Engage si conclude con l'identificazione e la dichiarazione di una sfida convincente e fattibile.

2.1.2. Investigate

La fase di Investigate costruisce dalla sfida definita una serie di esperienze di apprendimento contestualizzate e ricerche rigorose, basate sul contenuto e il concetto della sfida, per creare una base per soluzioni fattibili e sostenibili.

2.1.2.1. Guiding Questions

La fase di investigazione inizia con la generazione di domande relative alla sfida: esse includono tutto ciò che deve essere appreso per sviluppare una soluzione informata alla sfida. Le domande sono categorizzate e classificate in ordine di priorità.

2.1.2.2. Guiding Activities/Resources

Per quanto concerne questa fase, può essere utilizzata qualsiasi risorsa o attività che aiuti a rispondere alle Guiding Questions e a sviluppare una soluzione innovativa, perspicace e realistica. Esempi di risorse guida includono: contenuti e corsi online, banche dati, libri di testo e social network. Esempi di attività guida includono: simulazioni, esperimenti, progetti, serie di problemi, ricerche e giochi.

2.1.2.3. Synthesis

Una volta che tutte le domande guida sono state affrontate e i risultati delle attività guida registrati, si analizzano i dati accumulati e si identificano i temi. La fase di investigazione si conclude con la dimostrazione dei risultati ottenuti e le conclusioni sviluppate che porranno le basi per la soluzione.

2.1.3. Act

Nella fase dell'Act le soluzioni ottenute durante la fase di Investigate sono sviluppate e implementate con un pubblico autentico e i risultati sono valutati.

2.1.3.1. Solutions Concepts

I Solutions Concepts possono comprendere piani per una campagna di informazione, progetti di miglioramento, sviluppo di prodotti o altre attività.

2.1.3.2. Solution Development

Dopo che il Solution Concept è stato approvato, si sviluppano prototipi, sperimentazioni e test. Questo ciclo iterativo di progettazione porta, in alcuni casi, a sollevare nuove domande guida che richiedono ulteriori ricerche, riportando il focus nella fase di investigazione.

2.1.3.3. Implementation and Evaluation

Dopo aver sviluppato le soluzioni, si passa alla fase di implementazione, di misurazione dei risultati, con una successiva analisi su cosa ha funzionato e cosa no, determinando l'impatto di quanto appena affermato sulla sfida.

2.2. Monitor4U e l'approccio Challenge Based Learning

Seguendo l'approccio della Challenge Based Learning, la soluzione progettata e realizzata tramite Monitor4U è stata sviluppata partendo da un concetto, anche detto Big Idea, fondamentale: la sicurezza domestica. La motivazione alla base della scelta di questo concetto così ampio risiede in una serie di situazioni e problematiche a noi vicine, riguardanti in particolar modo familiari e conoscenti i quali devono far fronti a svariati problemi di questo tipo durante il loro normale svolgimento della vita quotidiana. A partire da quanto affermato è stata generata un'ampia varietà di essential questions che riflettono interessi e idee personali delle persone a noi vicine per quanto concerne il tema trattato, finalizzando come domanda essenziale "*Quanto ti senti al sicuro riguardo la tua abitazione?*".

È stata scelta come domanda essenziale in quanto va dritta al focus della ricerca e pone subito dinanzi all'utente la questione della problematica della sicurezza domestica quando egli non si trova nella propria dimora. Per quanto concerne la fase di investigazione, al fine di poter condurre una ricerca rigorosa per creare una base per soluzioni fattibili e sostenibili, sono state generate una serie di domande relative al tema trattato, includendo tutto quanto necessario per sviluppare una soluzione informata alla sfida. Le Guiding Questions sviluppate sono state le seguenti:

- Il totale controllo dello "stato" della propria dimora contribuisce a trascorrere meglio la propria giornata?
- Com'è possibile utilizzare l'informatica per andare in contro alle esigenze delle persone?
- È possibile tenere traccia delle variazioni dello stato di qualità dell'aria, facilitando anche eventuali monitoraggi utili non solo alle persone ma anche ad eventuali enti pubblici?
- In che modo è possibile evitare la dispersione di informazioni pubbliche relative alla propria abitazione?

Al fine di poter trovare un insieme di risposte per guidare e sviluppare una soluzione innovativa, perspicace e realistica, sono state condotte una serie di attività guida comprendenti ricerche in rete, approfondimento delle attuali soluzioni informatiche in ambito “safety e care” e la realizzazione di un questionario mediante Google Form attinente al tema. Di seguito, si riportano i risultati ottenuti a seguito della ricezione di 95 feedback:

1. *Quando esci di casa ti capita mai di avere il dubbio di aver lasciato la porta o le finestre aperte?*

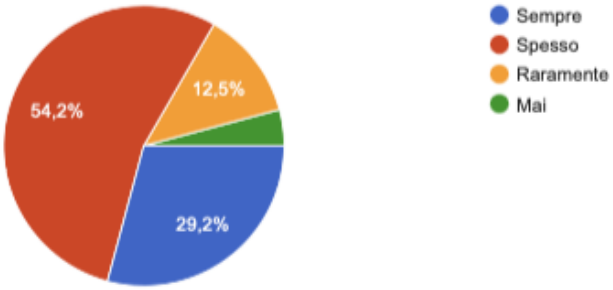

Figura 3 Risposte alla domanda 1

2. *Ti capita mai quando non sei a casa di pensare di esserti dimenticato qualche apparecchio elettrodomestico acceso?*

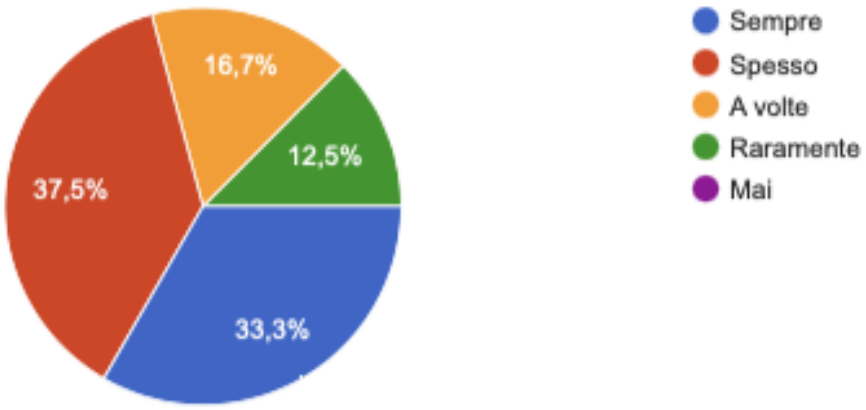

Figura 4 Risposte alla domanda 2

3. *Sei preoccupato per la possibile presenza di intrusi quando non sei a casa?*

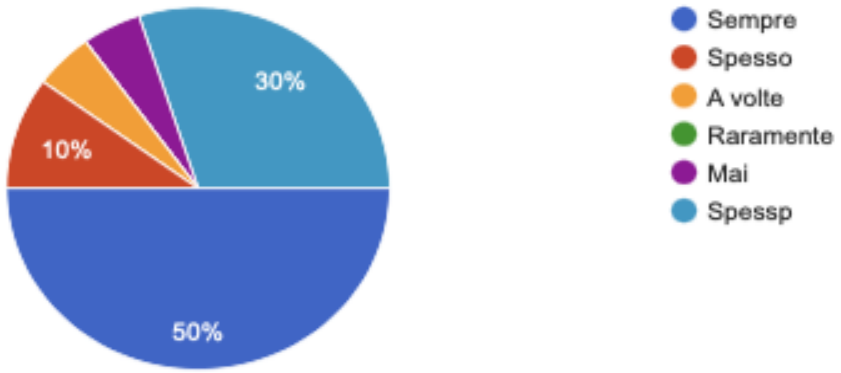

Figura 5 Risposte alla domanda 3

4. *Hai dispositivi per controllare lo stato di sicurezza della tua casa? Se sì quali?*

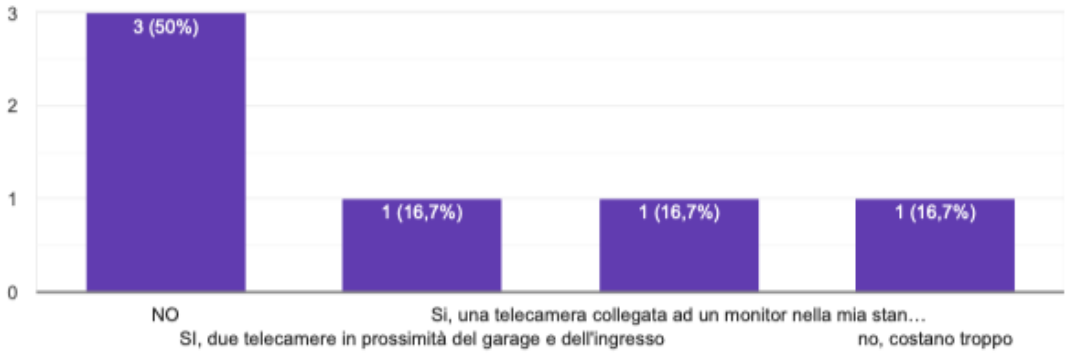

Figura 6 Risposte alla domanda 4

5. *Utilizzeresti un'applicazione che ti fornisca la possibilità di gestire in maniera rapida ed efficace i dispositivi che usi per controllare la sicurezza della tua casa?*

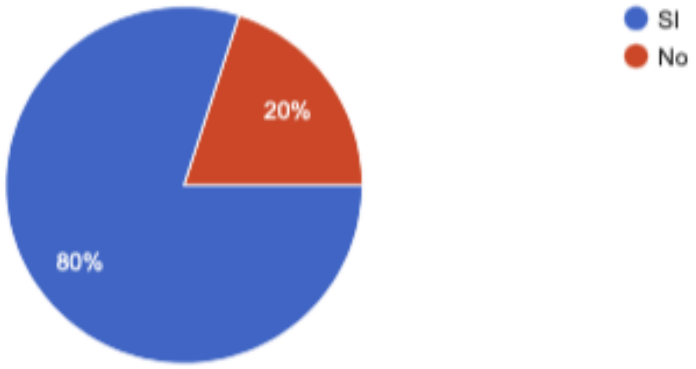

Figura 7 Risposte alla domanda 5

6. *Utilizzeresti un'applicazione che sia in grado si segnalarti l'evetuale presenza di intrusi anche quando non sei a casa?*

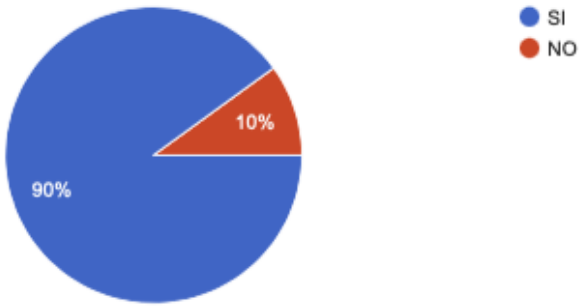

Figura 8 Risposte alla domanda 6

7. *Sarebbe utile un'app per monitoreare lo stato della tua casa a distanza? In caso affermativo cosa vorresti che offrisse?*

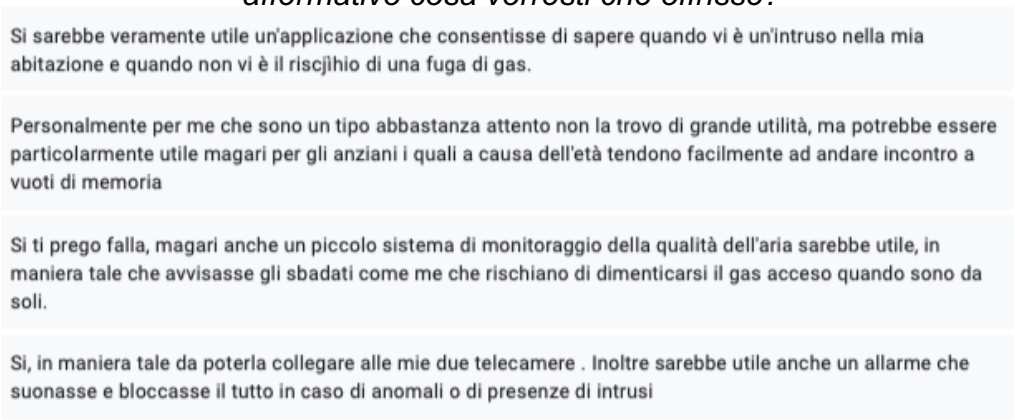

Figura 9 Risposte alla domanda 7

Una volta conseguiti i risultati appena elencati, la fase successiva della progettazione si è spostata verso l'attuazione e lo sviluppo di concetti di soluzioni, anche conosciuta come Act. In particolar modo, e soprattutto a supporto delle risposte ricevute, si è scelto di poter centrare a pieno l'obiettivo posto dall'essential question attraverso lo sviluppo di un prodotto software, e più nello specifico di un'applicazione mobile, Monitor4U. Monitor4U si promette di creare un "sistema di monitoraggio e gestione della casa digitale" con l'obiettivo di aiutare a gestire i propri dispositivi di sicurezza domestica, fornendo anche un servizio di localizzazione della/e propria/e abitazione/i. L'applicazione in totale linea con l'idea di conservare la privacy dell'utente all'avvio è caratterizzata da una pagina di Login, nella quale l'utente può sia iscriversi, se è al primo accesso, sia accedere con la propria email o con le credenziali Facebook. Nel caso in cui l'utente decidesse di registrarsi con le proprie credenziali di Facebook, verrà mostrato un segnale di pop-up con un link alla pagina iniziale di Facebook nel quale l'utente potrà inserire i propri dati e dopo aver dato il consenso al trattamento dei dati può accedere all'applicazione. In aggiunta l'applicazione è caratterizzata anche da un servizio di autenticazione a due livelli, in quanto consente all'utente una volta effettuato l'accesso di migliorare la sicurezza del proprio account integrando le proprie credenziali, con quelle di Facebook nel caso in cui egli abbia effettuato l'accesso con Gmail e con quelle di quest'ultima nel caso in cui l'accesso sia stato effettuato usufruendo dell'altro servizio. L'applicazione si propone per poter permettere all'utente l'inserimento di una o più case con la relativa posizione geografica garantendo l'accesso a queste informazioni attraverso una sezione appositamente dedicata. Infine, nella sezione dedicata alla mappa vi è la possibilità per l'utente di poter visualizzare la localizzazione della propria abitazione (anche nel caso in cui ve ne siano di più) e cliccare su di essa per vedere

lo stato dei dispositivi connessi dopo essersi collegato alla casa, tramite l'apposito bottone per visualizzarne lo stato.

2.3. User Interface, User Experience

La trattazione successiva, per quanto riguarda lo sviluppo dell'applicativo iOS, è incentrata sulla progettazione della User Interface, la gestione della User Experience per il soddisfacimento delle linee guida imposte da Apple, l'integrazione con i vari framework utilizzati e l'interfacciamento con il protocollo MQTT.

2.3.1. Progettazione della User Interface e gestione della User Experience

Come ben sappiamo, le persone dipendono dal loro dispositivo mobile per molti aspetti, a partire dalla visualizzazione dei media sino a giungere allo svolgimento di attività per tenere traccia dei dati personali in un qualsiasi momento e in un qualsiasi luogo. La User Interface è il fondamentale tramite nell'interazione tra l'utilizzatore e l'applicazione d'interesse; non è altro quindi che l'interfaccia visuale con la quale avviene quest'ultima. La User Interface, per quanto possa essere indipendente, in realtà è solo una - anche se fondamentale – branca della User Experience. È, dunque, completamente basata sulle esigenze dell'utente, e saranno proprio queste a determinare tutte le scelte di design. In generale, e soprattutto per i target di Monitor4U, le principali necessità da garantire durante la fase di interazione sono:

- Accedere alla propria sezione di Login senza difficoltà
- Accedere alle info relative alla propria abitazione in maniera rapida
- Gradire l'esperienza generale, anche da un punto di vista prettamente estetico.

È importante sottolineare che quest'ultimo punto, non è come spesso succede, l'unico fattore da considerare; d'altra parte, però, dire che un'interfaccia non deve essere curata nell'aspetto per offrire un'esperienza ottimale è come dire che non serve una fase di progettazione per lo sviluppo di un prodotto, che sia hardware o software. Infatti, l'utilizzatore che non riesce a completare il proprio task nel minor tempo possibile prova frustrazione a prescindere da quanto sia curata esteticamente l'interfaccia, ma allo stesso tempo può decidere di abbandonare l'applicazione nel momento in cui viene presentata un'interfaccia esteticamente aberrante. Nello specifico, per quanto concerne lo sviluppo di applicazioni per il sistema operativo iOS, è necessario comprendere sin dall'inizio le caratteristiche e i modelli fondamentali del dispositivo che contraddistinguono l'esperienza. L'utilizzo di queste caratteristiche e modelli per formare le decisioni di progettazione risulta fondamentale, al fine di poter garantire da parte dell'utilizzatore finale apprezzamento. Durante la fase di progettazione della User Interface e più in generale della User Experience, inizialmente gli aspetti fondamentali da considerare sono i seguenti: display, ergonomia, ingressi, interazione con l'app e funzionalità di sistema

2.3.1.1. Display

Tener conto di quelle che sono tutte le caratteristiche tecniche dei dispositivi d'interesse, a partire dalle dimensioni fino ad arrivare alla risoluzione è fondamentale. Per quanto possano sembrare caratteristiche banali, è necessario essere a conoscenza di quest'ultime dal momento in cui definiscono quella che sarà la disposizione degli elementi che caratterizzeranno ogni singola schermata di cui l'applicazione fa uso.

2.3.1.2. Ergonomia

Le persone tengono generalmente il loro dispositivo in una o entrambe le mani mentre interagiscono con esso, passando da orientamento verticale a orientamento orizzontale a seconda delle necessità. Oltretutto, durante la fase di interazione con il dispositivo, la distanza di visione tende a non essere troppo superiore. Dunque, l'insieme di questi aspetti andrà in particolar modo a definire tutte quelle che sono le caratteristiche degli elementi visivi dell'applicazione, a partire dalla grandezza fino a giungere alla loro disposizione a seconda della modalità d'uso dell'applicazione stessa.

2.3.1.3. Ingressi

I gesti multitouch, le tastiere sullo schermo ed altri elementi d'ingresso consentono all'utilizzatore di eseguire azioni e svolgere compiti specifici. È da queste necessità che dunque, per l'applicazione d'interesse, vengono definiti tutti gli elementi d'ingresso che possano garantire l'interazione dell'utilizzatore con tutti gli elementi presenti sullo schermo, ognuno dei quali con un significato predefinito.

2.3.1.4. Interazione con l'app

Molto importante durante la fase di progettazione è tener conto, per quanto concerne il target di utenti verso cui l'applicazione è rivolta, il tempo medio trascorso in relazione alle funzionalità messe a disposizione, fattore molto importante il quale andrà a condizionare quella che sarà poi l'esperienza d'uso. Al fine del miglioramento di quest'ultime, è necessario durante la fase di progettazione prevedere che l'accessibilità alle funzionalità messe a disposizione e il loro utilizzo sia quanto più semplice e al tempo stesso veloce.

2.3.1.5. Funzionalità di sistema

Le funzionalità messe a disposizione da iOS sono diverse, al fine di aiutare gli utilizzatori a interagire con il sistema e l'applicazione in modo familiare e coerente. La scelta di quest'ultime deve essere dunque condizionata solo da quelli che sono gli scopi per la quale l'applicazione è pensata.

In linea generale, nella fase di progettazione della User Interface, tenendo conto della User Experience, di Monitor4U, le fondamenta sulla quale quest'ultima è stata costruita sono state redatte tenendo conto dei seguenti criteri:

- • garantire all'utilizzatore la possibilità di concentrarsi su attività e contenuti primari limitando il numero di controlli sullo schermo e rendendo rilevabili dettagli e azioni secondarie con un'interazione minima;
- adattarsi perfettamente ai cambiamenti di aspetto, e più nella fattispecie nell'utilizzo anche della modalità scura, consentendo alle persone di scegliere le configurazioni che funzionano meglio per loro;
- abilitare interazioni che supportino il modo in cui le persone di solito tengono il loro dispositivo: difatti tende ad essere, ad esempio, più facile e comodo per le persone raggiungere un controllo quando si trova nella zona centrale o inferiore del display; quindi, è particolarmente importante lasciare che le persone scorrano per tornare indietro o avviare azioni in una riga di elenco;
- con il permesso delle persone, integrare le informazioni disponibili con le funzionalità in modo da migliorarne l'esperienza, e più nello specifico di Monitor4U.

Di seguito viene riportata la prototipazione delle viste descritte.

2.3.1.6. Wireframing di Monitor4U

Dopo aver redatto le caratteristiche di Monitor4U, sia dal punto di vista funzionale che dal punto di vista dei requisiti a livello della User Interface, la fase di progettazione di quest'ultima continua con la prototipazione di base dell'interfaccia, noto anche come wireframing. Un wireframe è come lo scheletro di un'interfaccia, il minimo necessario per capire come un progetto sarà definito a livello funzionale. In particolar modo, essendo Monitor4U pensata per gestire la privacy dell'utente, la gestione dei dispositivi del sistema di controllo e la visualizzazione delle case disponibili associate al singolo utente loggato, possiamo suddividere l'applicazione in 3 viste principali, alle quali è possibile accedere mediante un tab menu. Di seguito viene riportata la prototipazione delle viste descritte.

Dal momento che la privacy costituisce un importante focus dell'applicazione la prima vista che si pone dinanzi all'utente appena effettua l'accesso all'applicazione è proprio la vista per l'autenticazione, per il corretto trattamento dei propri dati.

La vista "AuthenticationView"

L'utente non appena effettua l'accesso all'applicazione si trova dinanzi a questa vista nella quale può sia effettuare l'accesso se già registrato sia registrarsi nel caso di primo utilizzo dell'applicazione. Sono possibili due tipi di autenticazione, la prima tramite E-mail la seconda tramite Facebook, grazie agli appositi pulsanti posizionati al centro della vista. Nel caso in cui l'utente non sia registrato è possibile cliccare sul bottone rappresentato da una freccia per avere accesso ad una finestra(sheet) che si aprirà nella quale è possibile inserire le proprie credenziali, nel totale rispetto del trattamento dei dati. Questa vista è bloccante in quanto se non si effettua l'accesso l'utente è impossibilitato ad accedere alle altre funzionalità dell'applicazione. Dal momento che Monitor4U si pone come uno dei tanti obiettivi anche quella di essere UserFriendly; quindi, di fornire un'interfaccia UI e UX che sia molto semplice e comoda per l'utente, notiamo la presenza dei bottoni per accedere con le credenziali "E-mail" o "Facebook" predisposte al centro dello schermo in maniera tale da

non essere distanti dal pollice dell'utente, potendo quindi essere cliccati con relativa facilità. Discorso analogo per il bottone che porta alla vista, accessibile nel caso in cui l'utente non abbia già un proprio account

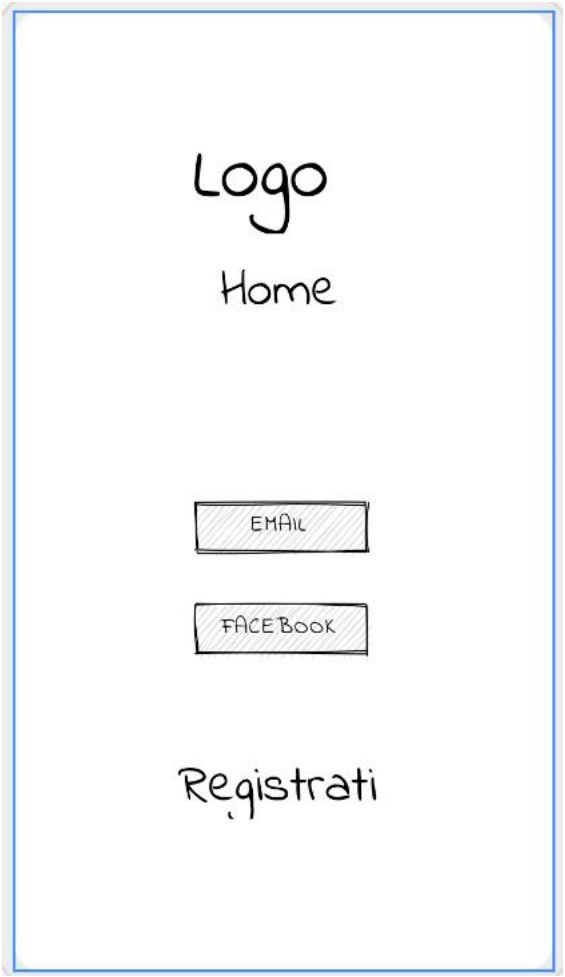

Figura 10 Wireframe vista "AuthenticationView"

La vista "LoginEmailView"

La vista "LoginEmailView" appare all'utente qualora quest'ultimo abbia già effettuato l'accesso in passato e prema il pulsante per accedere con le proprie credenziali e-mail. La vista si compone di una descrizione iniziale, volta a dare il bentornato all'utente e successivamente di due texfield tramite i quali l'utilizzatore può inserire le proprie credenziali. Infine, dopo l'inserimento delle credenziali tramite il pulsante di login può accedere alle altre schermate dell'applicazione precedentemente bloccate, oppure può chiudere la vista sia scorrendo verso il basso che premendo il pulsante in alto a destra,

Figura 11 Wireframe vista "LoginEmailView"

La vista “RegisterEmailView”

La vista “RegisterEmailView” appare all’utente qualora quest’ultimo non abbia mai effettuato l’accesso in passato e prema il pulsante per la registrazione. La vista si compone di una descrizione iniziale, e successivamente di due texfield tramite i quali l’utilizzatore può inserire le proprie credenziali. Infine, dopo l’inserimento delle credenziali tramite il pulsante “Accedi” può accedere alle altre schermate dell’applicazione precedentemente bloccate, oppure può chiudere la vista sia scorrendo verso il basso che premendo il pulsante in alto a destra.

Figura 12 Wireframe vista "RegisterEmailView"

La vista “homeView”

La seguente vista nonché vista principale dell’applicazione è pensata per poter garantire all’utilizzatore l’accesso a tutti i dispositivi collegati con il sistema di controllo della propria abitazione. La vista permetterà all’ utente di avere sempre a disposizione la vista dei dispositivi disponibili e attivi in quel momento e cliccando su ognuno di essi vederne il relativo stato. Inoltre, è presente anche un bottone tramite il quale è possibile aggiungere altre case nell’ipotesi in cui l’utente si sia trasferito oppure abbia acquistato una nuova abitazione. In alto la vista presente un pulsante di “logout” nel caso in cui l’utente voglia disconnettersi e accedere con un altro account.

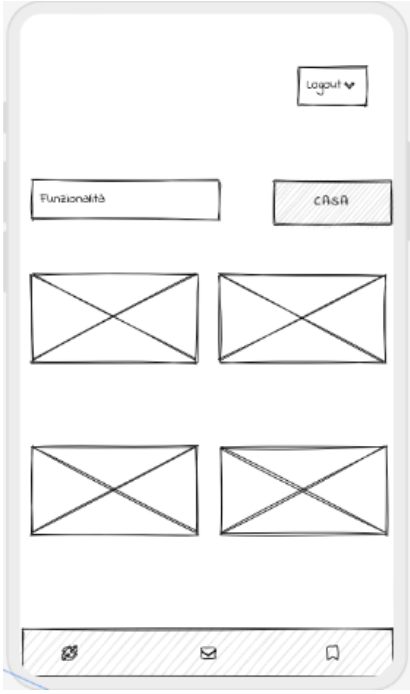

Figura 13 Wireframe vista "homeView"

La vista “DetailedHome”

La vista “DetailedHome” si apre allorquando l’utente ha cliccato su uno dei dispositivi del quale vuole saperne di più in relazione allo stato di funzionamento. In questa vista egli può visualizzare una descrizione dei dispositivi in sé, la quale fornisce dei dettagli su come devono essere interpretati eventuali valori/risultati e la lista di abitazioni dove quel determinato dispositivo è attivo.

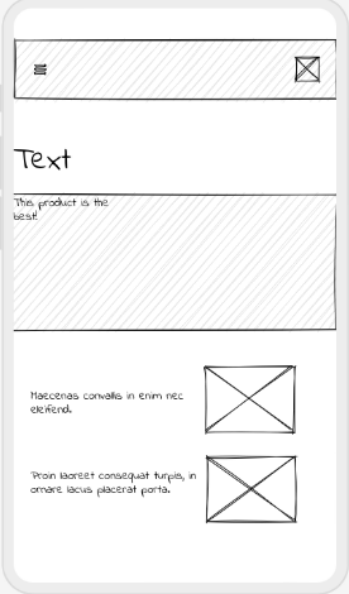

Figura 14 Wireframe vista "DetailedHome"

La vista “MoreHouseView”

La vista “AddHouse” viene mostrata allorquando l’utente nella schermata Home clicca sul bottone “Aggiungi Casa”. Viene mostrato un Form nel quale l’utente deve aggiungere le info relative alla nuova abitazione come: il nome, la latitudine, la longitudine e la via, le quali verranno usati dalla vista “Map” per mostrare la posizione sulla Mappa della relativa abitazione.

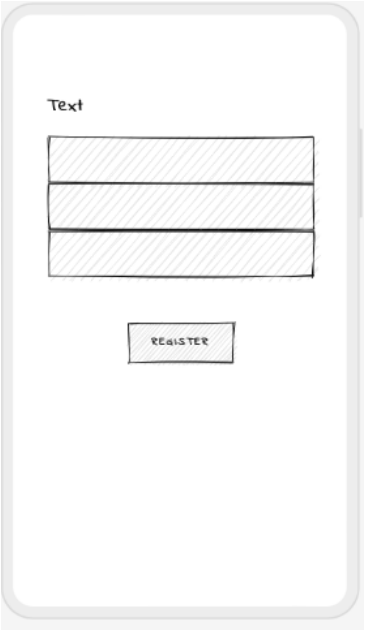

Figura 15 Wireframe vista "MoreHouseView"

Ritornando alla vista “DetailedHome” è possibile notare come la descrizione del tasto vari in base a quale bottone viene premuto e nel caso particolare in cui il dispositivo del quale si vuole conoscere lo stato è quello relativo alla presenza di intrusi, vi è un testo dinamico, il quale cambia se vi è o meno la presenza di persone estranei nei pressi dell’abitazione.

La vista “Profile”

Questa sezione è accessibile all’utente in due modi: cliccando sulla tabMenu oppure sull’icone del profilo presente nella schermata Home (in quel caso la tabMenu non è visibile. Vengono mostrati all’utente tramite un form due bottoni tramite i quali è possibile integrare le proprie credenziali con ulteriori informazioni. Nel caso in cui l’utente abbia effettuato l’accesso con la propria e-mail allora sarà disponibile il tasto “Collegati con Facebook” mentre nel caso in cui l’abbia effettuato l’accesso con le proprie credenziali facebook sarà visibile il tasto “Collega Email” il quale premuto darà accesso ad un form dove è possibile inserire le credenziali relative al proprio account E-mail.

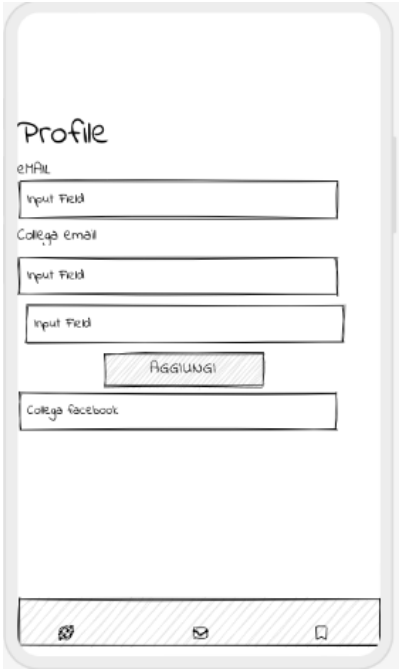

Figura 16 Wireframe vista "ProfileView"

La vista “LocationsView”

Attraverso la tabMenu è possibile accedere anche alla mappa. In questa vista l’utente si trova dinanzi ad una “lista dinamica” dove vengono mostrate tutte le case da egli possedute oltre alla relativa posizione in termini di coordinate geografiche. In basso è presente un bottone la cui funzione è quella di scorrere la lista delle abitazioni precedentemente descritta. Inoltre, quando l’utente si trova lungo un elemento della lista diverso dalla sua posizione attuale nei pressi del bottone compare un ulteriore tasto “Open in Maps” il quale offre un collegamento diretto con il servizio di Google Maps direttamente fornito da Apple e che nella fattispecie consente di calcolare il percorso più conveniente dalla posizione attuale alla destinazione (la propria casa nello specifico caso preso in esame).

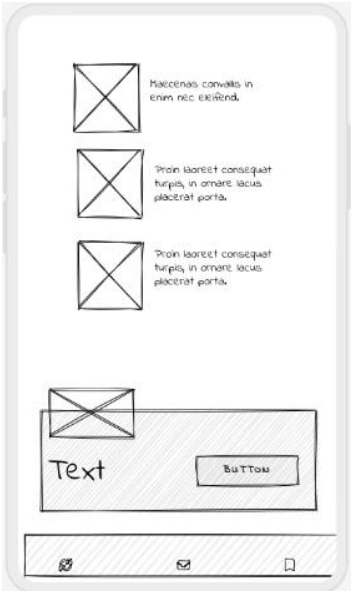

Figura 17 Wireframe vista "LocationsView"

La vista “HomeView”

A conclusione del wireframing di Monitor4U, l’ultima vista da analizzare è relativa all’interfacciamento con il sistema di controllo. Una volta che l’utente ha cliccato su una delle case disponibili egli si trova dinanzi alla vista “HomeView”, la quale ha lo scopo di presentare la propria dimora con tutti i dispositivi connessi. Al primo accesso però, questi non sono ancora visibili in quanto l’applicazione non è ancora connessa con la casa, motivo per cui l’utente dovrà usufruire del bottone “Connetti alla casa” il quale consentirà la visualizzazione dei dispositivi connessi al sistema di controllo SafeAndCareSystem. Nel caso in cui la connessione con l’abitazione non andasse a buon fine l’utente vedrebbe comparire una schermata caratterizzata da un testo, rimarcando l’assenza di un’adeguata connessione ad internet.

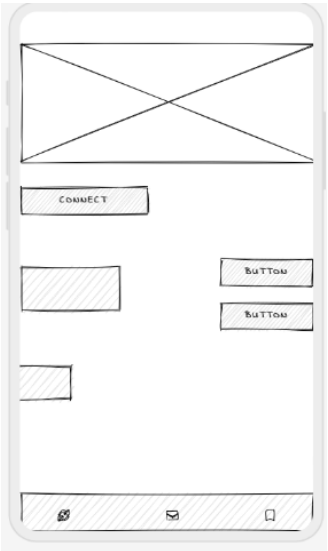

Figura 18 Wireframe vista "HomeView"

Una volta connessosi con l'abitazione l'utilizzatore potrà selezionare il dispositivo del quale vuole monitorarne lo stato e vi si aprirà una schermata aggiuntiva, con informazioni utili su come gestire la connessione. Dal momento che le funzionalità che Monitor4U si prefigge di offrire sono riguardano prevalentemente il controllo della qualità dell'aria e la presenza di intrusi, passeremo in esame il modo in cui l'utente può accedere a queste determinate funzionalità.
Nel caso in cui l'utente voglia visualizzare il livello del gas nella sua abitazione, tramite l'utilizzo dell'apposito bottone potrà visualizzare l'andamento del livello del Gas tramite l'apposito grafico e l'inserimento i pulsanti di "Grafico a linea" e "Azzera grafico" i quali consentono di avere una visuale lineare del grafico stesso e di azzerarne i valori. In particolare, qualora l'utente voglia essere a conoscenza del valore attuale del livello di gas in questo momento all'interno della sua abitazione grazie a questa funzionalità potrà vedere i valori del gas che compariranno al di sotto dello slider, il quale svolge una funzione particolare in quanto consente all'utente stesso di impostare il livello del gas oltre il quale il sistema deve segnalare lo stato critico.
Nel caso in cui l'utente voglia visualizzare lo stato del sistema di "Care" (cura della casa) allora la vista mostrerà oltre agli elementi di quella base anche un ulteriore Testo accompagnato da un bottone grazie al quale l'utente può abilitare o disabilitare lo stato del care.
Qualora l'utente voglia visualizzare lo stato del sistema di "Safety" (sicurezza della casa) allora la vista mostrerà oltre agli elementi di quella base anche un ulteriore Testo accompagnato da un bottone grazie al quale l'utente può abilitare o disabilitare lo stato del Safety. In aggiunta sullo schermo verrà visualizzato se attivo, all'interno della Gauge il valore della ventola azionata nel sistema di aerazione.

Gerarchizzazione delle viste

Conclusa la parte di prototipazione, all'interno della quale è stata gestita anche la parte di User Experience per cui viene ad essere stabilito come ogni istanza si collega e si relaziona alla successiva, scegliendo accuratamente le posizioni funzionali di ogni singolo elemento in relazione a quelle che sono note come Human Interface Guidelines, una raccolta di linee guida di interfaccia e principi di funzionamento legati al comportamento umano nell'utilizzo di un device, applicate con cura su dispositivi e sistemi operativi di casa Apple, che possiamo dare un quadro più generale della fase di progettazione della User Interface insieme alla User Experience, permettendo di poter avere ad un livello più alto la descrizione e l'interfacciamento di ogni singola vista dell'applicazione. Di seguito viene riportato uno schema delle viste gerarchico, descrivendo tutti i concetti poc'anzi affermati. Con le frecce si indicano le viste contenute in altre viste, mentre con il segmento le viste alle quali si accede passando per altre viste.

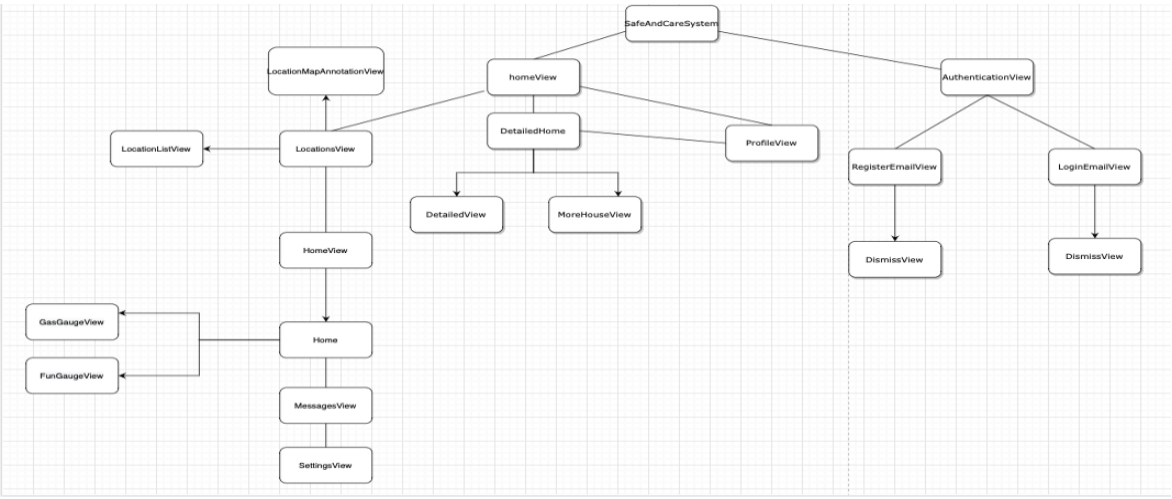

Figura 19 Gerarchia Views

2.3.2. Il framework Chart

Swift Charts è un framework SwiftUI potente e conciso per trasformare i dati in visualizzazioni informative. Con Swift Charts, è possibile creare grafici efficaci e personalizzabili con un codice minimo. Questo framework fornisce segni, scale, assi e leggende come elementi costitutivi che è possibile combinare per sviluppare una vasta gamma di grafici basati sui dati. Ci sono molti modi in cui puoi usare Swift Charts per comunicare modelli o tendenze nei tuoi dati. È possibile creare una varietà di grafici, tra cui grafici a linee, grafici a barre e grafici a dispersione come mostrato sopra. Quando crei un grafico utilizzando questo framework, genera automaticamente scale e assi che si adattano ai tuoi dati.

Swift Charts supporta le funzionalità di localizzazione e accessibilità. Puoi anche ignorare il comportamento predefinito per personalizzare i tuoi grafici utilizzando i modificatori del grafico. Ad esempio, è possibile creare un'esperienza dinamica aggiungendo animazioni ai tuoi grafici.

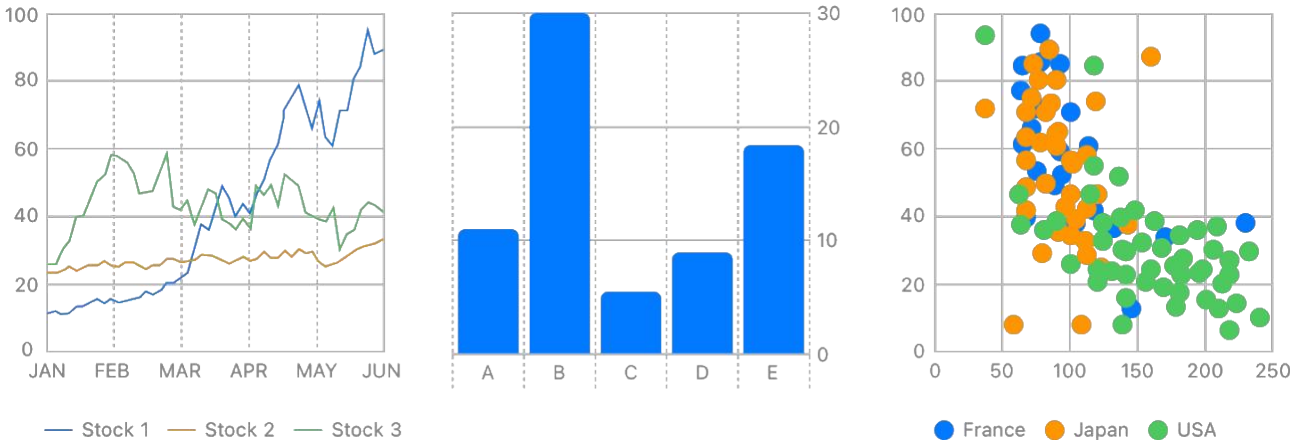

Figura 20 Charts

Come già accennato in precedenza, le funzionalità messe a disposizione da Charts sono totalmente in accordo agli obiettivi che a livello di funzionalità Montior4U garantisce di ottenere in termini di interfacciamento con il sistema di controllo, ed è per questo motivo che

fra altri framework messi a disposizione per la gestione della salute, quest'ultimo risulta tra tutti quello più aderente.

2.3.3. Il framework Mapkit

La vista Map è stata introdotta nel mondo di SwiftUI con il rilascio di iOS 14. Map è una vista che visualizza un'interfaccia mappa incorporata. La vista Map visualizza un'area e puoi utilizzarla in modo nativo nella visualizzazione SwiftUI. Ha molte opzioni di configurazione che consentono di visualizzare o tracciare una posizione e persino di interagire con la mappa nella vista. La mappa SwiftUI è ottima per visualizzare le mappe e visualizzare alcune delle funzionalità di base della mappa, ma non corrisponde alla versione UIKit di MKMapView. Ma il porting sulla mappa UIKit è facile. Il framework si presta bene a quelle che sono le funzionalità che Monitor4U si pone di offrire, come la visualizzazione della propria casa su una mappa grazie ad un MapAnnotationView realizzabile grazie al framework come vedremo nella parte di implementazione, la possibilità di tracciare un percorso il più veloce possibile nel caso in cui vi sia l'urgenza di tornare subito alla propria abitazione(eventuali fughe di gas oppure presenza di intrusi) grazie al collegamento fornito con il servizio di Mappe di Apple.

2.3.4. MQTT

MQTT è il protocollo di messaggistica più comunemente usato per l'Internet of Things (IoT). MQTT sta per MQ Telemetry Transport. Il protocollo è un insieme di regole che definiscono come i dispositivi IoT possono pubblicare e iscriversi ai dati su Internet. L'MQTT viene utilizzato per la messaggistica e lo scambio di dati tra dispositivi IoT e IoT industriale (IIoT), come dispositivi integrati, sensori, PLC industriali, ecc. Il protocollo è guidato da eventi e collega i dispositivi utilizzando il modello publish /subscribe (Pub/Sub). Il mittente (Publisher) e il destinatario (Subscriber) comunicano tramite argomenti e vengono disaccoppiati l'uno dall'altro. La connessione tra loro è gestita dal broker MQTT. Il broker MQTT filtra tutti i messaggi in arrivo e li distribuisce correttamente agli abbonati. Il modello di pubblicazione/iscrizione MQTT (noto anche come pub/sub) fornisce un'alternativa all'architettura client-server tradizionale. Nel modello client-server, un client comunica direttamente con un endpoint. Il modello pub/sub disaccoppia il client che invia un messaggio (publisher) dal client o dai client che ricevono i messaggi (gli abbonati). Gli editori e gli abbonati non si contattano mai direttamente. In effetti, non sono nemmeno consapevoli che l'altro esiste. La connessione tra loro è gestita da un terzo componente (il broker). Il compito del broker è filtrare tutti i messaggi in arrivo e distribuirli correttamente agli abbonati. Quindi, approfondiamoci un po' in profondità in alcuni degli aspetti generali del pub/sub (parleremo delle specifiche di MQTT tra un minuto).

L'aspetto più importante del pub/sub è il disaccoppiamento del publisher del messaggio dal destinatario (sottoscrittore). Questo disaccoppiamento ha diverse dimensioni:

- Space decoupling**:** il publisher e il subscriber non hanno bisogno di conoscersi (ad esempio, nessun scambio di indirizzo IP e porta).
- Time decoupling: il publisher e il subscriber non devono essere eseguiti contemporaneamente.
- Synchronization decoupling**:** le operazioni su entrambi i componenti non devono essere interrotte durante la pubblicazione o la ricezione.

In sintesi, il modello MQTT pub/sub rimuove la comunicazione diretta tra il publisher del messaggio e il destinatario/sottoscrittore. L'attività di filtraggio del broker consente di controllare quale cliente/sottoscrittore riceve quale messaggio. Il disaccoppiamento ha tre dimensioni: spazio, tempo e sincronizzazione.

2.3.4.1 Scalabilità

MQT Pub/Sub scala meglio dell'approccio client-server tradizionale. Questo perché le operazioni sul broker possono essere altamente parallelizzate e i messaggi possono essere elaborati in modo basato su eventi. La memorizzazione nella cache dei messaggi e il routing intelligente dei messaggi sono spesso fattori decisivi per migliorare la scalabilità. Tuttavia, scalare fino a milioni di connessioni è una sfida. Un livello così elevato di connessioni può essere ottenuto con nodi di intermediazione cluster per distribuire il carico su più singoli server utilizzando bilanciatori di carico.

2.3.4.2 Filtraggio dei messaggi

È chiaro che il broker MQTT svolge un ruolo fondamentale nel processo pub/sub. Ma come fa il broker a filtrare tutti i messaggi in modo che ogni subscriber riceva solo messaggi di interesse? Come vedrai, il broker ha diverse opzioni di filtraggio:

OPZIONE 1: SUBJECT-BASED FILTERING

Questo filtro si basa sul subject o sul topic che fa parte di ciascun messaggio. Il cliente ricevente sottoscrive il broker per argomenti di interesse. Da quel momento in poi, il broker si assicura che il cliente ricevente riceva tutti i messaggi pubblicati sugli argomenti sottoscritti. In generale, gli argomenti sono stringhe con una struttura gerarchica che consente il filtraggio basato su un numero limitato di espressioni.

OPZIONE 2: CONTENT-BASED FILTERING

Nel content-based-filtering, il broker filtra il messaggio in base a uno specifico linguaggio di filtro del contenuto. I clienti riceventi si iscrivono alle query di filtro dei messaggi a cui sono interessati. Uno svantaggio significativo di questo metodo è che il contenuto del messaggio deve essere conosciuto in anticipo e non può essere crittografato o facilmente modificato.

OPZIONE 3: TYPE-BASED FILTERING

Quando si usano linguaggi orientati agli oggetti, il filtraggio in base al tipo/classe di un messaggio (evento) è una pratica comune. Ad esempio, un subscriber può ascoltare tutti i messaggi, che sono di tipo Exception o qualsiasi sottotipo.

Naturalmente, pubblicare/iscriversi non è la risposta per ogni caso d'uso. Ci sono alcune cose che bisogna considerare prima di usare questo modello. Il disaccoppiamento di publisher e subscriber, che è la chiave nel pub/sub, presenta alcune sfide a sé stante. Ad esempio, è necessario essere consapevoli di come sono strutturati i dati pubblicati in anticipo. Per il filtraggio basato sul soggetto, sia il publisher che il subscriber devono sapere quali argomenti utilizzare. Un'altra cosa da tenere a mente è la consegna dei messaggi. Il publisher non può presumere che qualcuno stia ascoltando i messaggi inviati. In alcuni casi, è possibile che nessun subscriber legga un particolare messaggio.

MQTT incarna tutti gli aspetti del pub/sub che abbiamo menzionato

- MQTT disaccoppia il publisher e il subscriber spazialmente. Per pubblicare o ricevere messaggi, gli editori e gli abbonati devono solo conoscere il nome host/IP e la porta del broker.
- MQTT si disaccoppia per tempo. Anche se la maggior parte dei casi d'uso MQTT consegna messaggi in tempo quasi reale, se lo si desidera, il broker può memorizzare i messaggi per i clienti che non sono online. (Per memorizzare i messaggi devono essere soddisfatte due condizioni: il client si è connesso con una sessione persistente e si è iscritto a un argomento con una qualità del servizio superiore a 0).
- MQTT funziona in modo asincrono. Poiché la maggior parte delle librerie client funziona in modo asincrono e si basano su callback o un modello simile, le attività non vengono bloccate durante l'attesa di un messaggio o la pubblicazione di un messaggio. In alcuni casi d'uso, la sincronizzazione è desiderabile e possibile. Per attendere un determinato messaggio, alcune librerie hanno API sincrone. Ma il flusso è di solito asincrono.

Un'altra cosa che va menzionata è che MQTT è particolarmente facile da usare sul lato client. La maggior parte dei sistemi pub/sub ha la logica sul lato broker, ma MQTT è davvero l'essenza di pub/sub quando si utilizza una libreria client e questo lo rende un protocollo leggero per dispositivi piccoli e vincolati. MQTT utilizza il filtraggio dei messaggi basato sull'oggetto. Ogni messaggio contiene un topic (subject) che il broker può utilizzare per determinare se un client abbonato riceve o meno il messaggio. Per gestire le sfide di un sistema pub/sub, MQTT ha tre livelli di qualità del servizio (QoS). È possibile specificare facilmente che un messaggio venga recapitato correttamente dal client al broker o dal broker a un client. Tuttavia, c'è la possibilità che nessuno si iscriva all'argomento particolare. Se questo è un problema, il broker deve sapere come gestire la situazione.

2.3.4.3 MQTT Client e Broker

Sia i publisher che i subscriber sono clienti MQTT. Queste etichette si riferiscono al fatto che il client stia attualmente pubblicando messaggi o sia iscritto a ricevere messaggi (la funzionalità di pubblicazione e sottoscrizione può anche essere implementata nello stesso client MQTT). Un client MQTT è qualsiasi dispositivo (da un microcontrollore a un server a tutti gli effetti) che esegue una libreria MQTT e si connette a un brokerMQTT su una rete. Ad esempio, il client MQTT può essere un dispositivo molto piccolo e limitato di risorse che si connette su una rete wireless e dispone di una libreria minima. Il client MQTT può anche essere un computer tipico che esegue un client MQTT grafico per scopi di test. Fondamentalmente, qualsiasi dispositivo che parla MQTT su uno stack TCP/IP può essere chiamato client MQTT. L'implementazione client del protocollo MQTT è molto semplice e semplificata. La facilità di implementazione è uno dei motivi per cui MQTT è ideale per i dispositivi di piccole dimensioni.

La controparte del client MQTT è il broker MQTT. Il broker è al centro di qualsiasi protocollo di pubblicazione/iscrizione. A seconda dell'implementazione, un broker può gestire fino a milioni di client MQTT connessi contemporaneamente.

Il broker è responsabile della ricezione di tutti i messaggi, del filtraggio dei messaggi, della determinazione di chi è iscritto a ciascun messaggio e dell'invio del messaggio a questi clienti iscritti**.** Il broker conserva anche i dati di sessione di tutti i clienti che hanno sessioni persistenti, inclusi abbonamenti e messaggi persi. Un'altra responsabilità del broker è l'autenticazione e l'autorizzazione dei clienti. Di solito, il broker è estensibile, il che facilita l'autenticazione personalizzata, l'autorizzazione e l'integrazione nei sistemi di back-end. L'integrazione è particolarmente importante perché il broker è spesso il componente che è direttamente esposto su Internet, gestisce molti clienti e ha bisogno di passare messaggi ai sistemi di analisi ed elaborazione a valle. Dal momento che il broker è l'hub centrale attraverso il quale ogni messaggio deve passare. Pertanto**,** è importante che il tuo broker sia altamente scalabile, integrabile nei sistemi di back-end, facile da monitorare e (ovviamente) resistente ai guasti**.**

2.3.4.4 Connessione MQTT

Il protocollo MQTT si basa su TCP/IP. Sia il client che il broker devono disporre di uno stack TCP/IP.

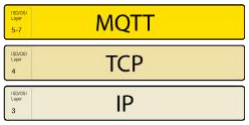

Figura 21 Struttura MQTT

La connessione MQTT è sempre tra un client e il broker. I clienti non si connettono mai direttamente tra loro. Per avviare una connessione, **il client invia un messaggio CONNECT al broker. Il broker risponde con un messaggio CONNACK** e un codice di

stato. Una volta stabilita la connessione, il broker la mantiene aperta fino a quando il client invia un comando di disconnessione o la connessione si interrompe.

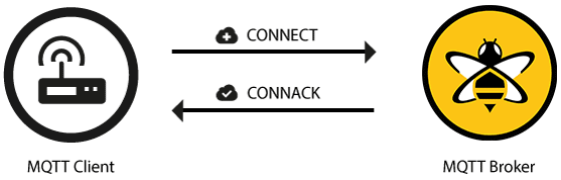

Figura 22 Connessione Client Broker

Per avviare una connessione, il client invia un messaggio di comando al broker. Se questo messaggio CONNECT è malformato (secondo la specifica MQTT) o passa troppo tempo tra l'apertura di un socket di rete e l'invio del messaggio di connessione, il broker chiude la connessione. Questo comportamento scoraggia i clienti malintenzionati che possono rallentare il broker.

ID client

Un clientID identifica ogni client MQTT che si connette a un broker MQTT. Il broker utilizza il ClientID per identificare il client e lo stato corrente del client. Pertanto, questo ID deve essere univoco per client e broker. In MQTT 3.1.1 è possibile inviare un ClientId vuoto, se non è necessario uno stato da tenere dal broker. Il ClientId vuoto si traduce in una connessione senza alcuno stato. In questo caso, il flag della sessione pulita deve essere impostato su true o il broker rifiuterà la connessione.

Sessione pulita

Il flag clean session indica al broker se il client desidera stabilire o meno una sessione persistente. In una sessione persistente (CleanSession = false), il broker memorizza tutte le sottoscrizioni per il client e tutti i messaggi mancati per il client che si è abbonato con un QoS (Quality of Service) di livello 1 o 2. Se la sessione non è persistente (CleanSession = true), il broker non memorizza nulla per il client ed elimina tutte le informazioni di qualsiasi sessione persistente precedente.

Will Message

Il will message fa parte della funzione Last Will and Testament (LWT) di MQTT.Questo messaggio avvisa gli altri client quando un client si disconnette in modo indegno. Quando un client si connette, può fornire al broker un'ultima volontà sotto forma di messaggio MQTT e argomento all'interno del messaggio CONNECT. Se il client si disconnette in modo sgradevole, il broker invia il messaggio LWT per conto del client.

KeepAlive

Il keep alive è un intervallo di tempo in secondi che il client specifica e comunica al broker quando la connessione è stata stabilita. Questo intervallo definisce il periodo di tempo più lungo che il broker e il cliente possono sopportare senza inviare un messaggio. Il client si impegna a inviare normali messaggi di richiesta PING al broker. Il broker risponde con una risposta PING. Questo metodo consente a entrambe le parti di determinare se l'altra è ancora disponibile.

Risposta del broker con un messaggio CONNACK:

Quando un broker riceve un messaggio CONNECT, è obbligato a rispondere con un messaggio CONNACK.

Il messaggio CONNACK contiene due voci di dati:

- Un session present flag
- Un codice di ritorno della connessione

Session present Flag

Il session present flag indica al client se il broker dispone già di una sessione persistente disponibile dalle precedenti interazioni con il client. Quando un client si connette con Clean Session impostato su true, il flag session present è sempre false perché non è disponibile alcuna sessione. Se un client si connette con Clean Session impostato su false, ci sono due possibilità: se le informazioni sulla sessione sono disponibili per l'ID client e il broker ha memorizzato le informazioni sulla sessione, il flag presente della sessione è true. In caso contrario, se il broker non dispone di alcuna informazione di sessione per il clientId, il flag presente della sessione è falso. are se devono iscriversi agli argomenti o se gli argomenti sono ancora archiviati in una sessione persistente.

Pubblicare:

Un client MQTT può pubblicare messaggi non appena si connette a un broker. MQTT un topic-based filtering dei messaggi sul broker. Ogni messaggio deve contenere un topic che il broker può utilizzare per inoltrare il messaggio ai clienti interessati. In genere, ogni messaggio ha un payload che contiene i dati da trasmettere in formato byte**.** MQTT è indipendente dai dati. Il caso d'uso del client determina la struttura del payload. Il client di invio (publisher) decide se inviare dati binari, dati di testo o anche XML o JSON a tutti gli effetti.

Iscrizione:

Pubblicare un messaggio non ha senso se nessuno lo riceve mai. In altre parole, se non ci sono clienti per iscriversi ai topic dei messaggi. Per ricevere messaggi su topic di interesse, il cliente invia un messaggio di Subscribe al broker MQTT. Questo messaggio di subscribe

è molto semplice, contiene un identificatore di pacchetto univoco e un elenco di abbonamenti.

Suback:

Per confermare ogni abbonamento, il broker invia un messaggio di conferma SUBACK al cliente. Questo messaggio contiene l'identificatore del pacchetto del messaggio di iscriviti originale (per identificare chiaramente il messaggio) e un elenco di codici di ritorno.

Unsubscribe:

La controparte del messaggio subscribe è il messaggio Unsubscribe. Questo messaggio elimina gli abbonamenti esistenti di un client sul broker. Il messaggio UNSUBSCRIBE è simile al messaggio SUBSCRIBE e ha un identificatore di pacchetto e un elenco di argomenti.

Unsuback:

Per confermare l'annullamento dell'iscrizione, il broker invia un messaggio di conferma Unsuback al cliente. Questo messaggio contiene solo l'identificatore del pacchetto del messaggio UNSUBSCRIBE originale (per identificare chiaramente il messaggio). Dopo aver ricevuto il UNSUBACK dal broker, il client può presumere che gli abbonamenti nel messaggio UNSUBSCRIBE siano stati eliminati.

3. Implementazione e descrizione delle funzionalità

Nel capitolo successivo si tratterà l'implementazione e l'integrazione di tutti i componenti che formano il sistema desiderato, mettendo in evidenza tutti gli aspetti salienti che riguardano la progettazione software, a partire dal linguaggio utilizzato fino ad arrivare alle integrazioni fra elementi appartenenti a framework distinti. Infine, verranno mostrati i risultati con le immagini raffiguranti il sitema prodotto

Nello specifico, le funzionalità che verranno ad essere descritte nel capitolo successivo, che concorrono nella realizzazione di Monitor4U sono le seguenti: l'integrazione del framework Charts in SwiftUI, framework adottato per l'interfacciamento con il sistema di controllo con Monitor4U comprendente la visualizzazione di tutti i diagrammi che rendono possibile visualizzare lo stato dei vari componenti; successivamente sarà discussa tutta la parte implementativa riguardante la sezione di Login, delegata alla gestione dei dati dell'utente ,grazie all'integrazione dei framework Firebase e Facebook integrati da github e di cui parleremo approfonditamente, e garantendo anche la possibilità di lasciare recensioni e valutazioni riguardante l'app sulla pagina Facebook; inoltre verrà approfondito l'utilizzo del framework MapKit il quale consente all'utente di localizzare la propria abitazione sulla mappa e integrare quest'ultima con il servizio di Mappe fornito da Apple, calcolando il percorso più breve dalla propria posizione attuale all'abitazione di destinazione nel caso in cui ce ne fosse bisogno; verrà discusso l'utilizzo del framework MQTT, combinato con

MapKit ,il quale permette all'utente di poter venire a conoscenza di tutte le info necessarie riguardo una delle prorprie dimore scelte sulla Mappa; in conclusione una gestione completa della parte di implementazione delle interfacce grafiche discusse durante la fase di progettazione, in concorrenza alla User Experience, per ogni singola vista dell'applicazione garantendo integrità sia dal punto di vista della progettazione che tra le viste differenti definite all'interno di Monitor4U.

3.1. L'integrazione di Firebase in SwifUI

Come descritto precedentemente, il framework Firebase è utile nei casi in cui si abbia la necessità di gestire azioni quali l'autenticazione ad un determinato servizio, come nel nostro caso.

Per usufruire del servizio di Firebase dapprima è stato necessario registrarsi al sito ufficiale Firebase.google.com, ed importare nel progetto Xcode, grazie all' SPM, la libreria di Firebase (nello specifico i framework FirebaseAuth, FirebaseAnalytic, FirebaseAnalyticsSwiftBeta) disponibile su GitHub. Una volta registratosi il passo successivo consisteva nella creazione del progetto della piattaforma Firebase. Durante questa fase dopo aver scelto il nome del progetto, il sito mette a disposizione il servizio di "Google Analytics", una soluzione di analisi gratuita e illimitata che abilita il targeting, il reporting e molto altro su Firebase Crashlytics, Cloud Messaging, messaggistica in-app, Remote Config, A/B Testing e Cloud Functions. In particolar modo questo servizio risulta molto utile qualora lo sviluppatore voglia tener conto di tutti gli accessi effettuati sulla propria applicazione attraverso un opportuno andamento grafico disponibile nella "home" di Firebase.

Per una corretta e più semplice gestione del nostro codice la gestione della connessione con Firebase avverrà seguendo una struttura a quattro livelli: al primo livello abbiamo un datasource il cui compito è quello di collegarsi con l'SDK di Firebase e fornire i servizi di base, i quali saranno passati ad un Repository collegato al ViewModel il quale per ultimo si collegherà alla vista che man mano richiederà i servizi dell'SDK di Firebase.

3.1.1.1 *AuthenticationFirebaseDatasouce* per la connessione con il servizio di Firebase

Questa classe risulta fondamentale in quanto costituisce il cardine di tutta la connessione con il database di Firebase, nella quale grazie all' import di FirebaseAuth è possibile definire numerose funzioni utili per l'interfacciamento con il database che saranno poi richiamate da altre classi sovrastanti con l'obiettivo di gestire queste funzionalità tramite le apposite funzioni create. Affinché l'utente possa interagire con il database Monitor4U si pone come obiettivo la possibilità di creazione e posizionamento all'interno del database di Firebase. Tutto ciò è possibile grazie alla funzione "createNewUser", comprendente come parametri due Stringhe, l'e-mail e la password, ed un completionBlock, parametro che ha il compito di

dire agli altri livelli della gerarchia qualora la creazione dell'utente sia andata a buon fine o meno. Un'altra funzionalità che il database deve fornire è il riconoscimento dell'utente che sta utilizzando l'applicazione, questa funzionalità viene svolta tramite la funzione "getCurrentCredential" la quale restituisce un set contenente le credenziali dell'utente corrente che sta utilizzando l'applicazione e che viene utilizzato dalla funzione "linkEmailAndPassword" la quale lo autentica nel database.

Ma come fare a capire se l'utente è gia presente nel database o meno? Questa funzionalità viene fornita dalla funzione di "login", la quale analizza i parametri e-mail e password presi in input insieme al "completionBlock analizzato precedentemente nella funzione "createNewUser", e verifica la presenza di queste credenziali invocando la funzione "Auth.auth(). signIn" la quale restituisce un errore se l'utente non è presente e inzializza la variabile e-mail.

Per quanto riguarda la connessione con il servizio di Facebook questa avviene tramite la funzione di "loginWithFacebook" la quale prende come parametri un "completionBlock" e usufruisce delle funzionalità della libreria FacebookLogin usufruibile grazie al framework importato da GitHub seguendo la falsariga della funzione di "login" per l'e-mail.

Per una corretta gestione di queste funzioni all'interno del codice dell'applicazione e per una più corretta integrazione del framework Facebook, del quale si parlerà in seguito, eventuali migliorie di queste funzioni sono state effettuate basandosi su altre due classi: AuthenticationRepository e AuthenticationViewModel, le quali unite alla precedente creano una gerarchia di classi a tre livelli grazie alla quale la gestione della connessione con il database viene resa più semplice.

3.1.1.2. AuthenticationRepository per la gestione dei dati dell'AuthenticationFirebaseDatasource

All'interno dell'AuthenticationRepository che costituisce il secondo livello della gerarchia di connessione con il database, viene creata un'istanza dell'AuthenticationFirebaseDatasource per usufruire dei dati precedentemente creati. Dopo aver inizializzato la variabile viene creata la funzione "createNewUser" la quale richiama l'omonima funzione del livello precedente e inizializza l'istanza dell'AuthenticationFirebaseDatasource precedentemente creata.

3.1.1.3. AuthenticationViewModel: interfaccia prima delle viste con il database

La classe "authenticationViewModel costituisce il fulcro della gerarchia di classi precedentemente descritta, in quanto essendo alla cima della gerarchia di classi precedentemente descritta, è quella che verrà utilizzata in tutte le altre viste della nostra applicazione e che consentirà l'interfacciamento con Firebase. Come nella classe precedente anche qui viene creata un'istanza dell'AuthenticationRepository con lo scopo di gestirne i dati forniti e inizializzata tramite l'opportuno inizializzatore.

Successivamente viene creato l'omonimo metodo "createNewUser" il quale inizializzerà i dati del variabile prima dichiarata e in aggiunta darà un risultato, memorizzato nella variabile "result" dichiarata "weak self" per evitare memory leak all'interno della nostra applicazione, a seconda dell'esito della connessione, grazie alle variabili "@Published" user e error, rispettivamente di tipo User e String entrambe optional, utilizzate per fornire maggiori informazioni all'utente riguardo l'errore. Procedimento analogo viene svolto grazie alle funzioni ": login, getCurrentProvider e linlkEmailPassword". Vengono inoltre aggiunta la funzione "isEmailPasswordLinked" per controllare che il database contenga effettivamente degli utenti connessi in un dato momento.

3.1.1.4. Utilizzo viewModel nelle View

A questo punto abbiamo collegato il nostro viewModel al Repository, il quale è collegato al FirebaseDatasource ma come è possibile avviare la configurazione della nostra applicazione con il database all'avvio.

Questa funzionalità è fornita dalla classe "AppDelegate", richiamata nella struttura principale usata all'avvio dell'applicazione e che quindi avvia la configurazione con il database grazie alla creazione delle variabili @UIApplicationDelegateAdaptor (AppDelegate. Self) var delegate e @StateObject var authenticationViewModel = AuthenticationViewModel ()

All'interno del body del main è possibile così gestire la view che deve essere presentata all'utente. In particolare, se l'utente risulta loggato verrà resa visibile "homeView", viceversa sarà resa visibile "AuthenticationView".

```
class AppDelegate: NSObject, UIApplicationDelegate {
  func application(_ application: UIApplication,
                   didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
      ApplicationDelegate.shared.application(application, didFinishLaunchingWithOptions: launchOptions)
      FirebaseApp.configure()
      return true
  }
}

@main
struct SafeAndCareSystemApp: App {
    @StateObject private var viewModel: LocationViewModel = LocationViewModel()
    @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate
    @StateObject var authenticationViewModel = AuthenticationViewModel()
//    @StateObject var remoteConfiguration = RemoteConfiguration()
//    @EnvironmentObject private var mqttManager: MQTTManager
    var body: some Scene {
        WindowGroup {
            if let _ = authenticationViewModel.user {
                homeView(authenticationViewModel: authenticationViewModel)
                    .environmentObject(viewModel)
            } else {
                AuthenticationView(authenticationViewModel: authenticationViewModel)
            }
        }
    }
}
```

Figura 23 Codice vista di avvio

Una volta creata la nostra istanza per collegarsi al database nella nostra applicazione andiamo su Firebase, nella quale è possibile notare che grazie alla corretta configurazione della nostra applicazione con il database, il servizio di Google Analytics ha già iniziato a tracciare tutti i membri che hanno effettuato l'accesso all'applicazione.

Figura 24 Firebase dashboard in azione

Successivamente andando nella sezione Authentication e successivamente in "sign-in method" è possibile selezionare i nostri metodi di accesso usufruibili da coloro che usano l'applicazione, nel nostro caso registrazione per e-mail e tramite credenziali Facebook,

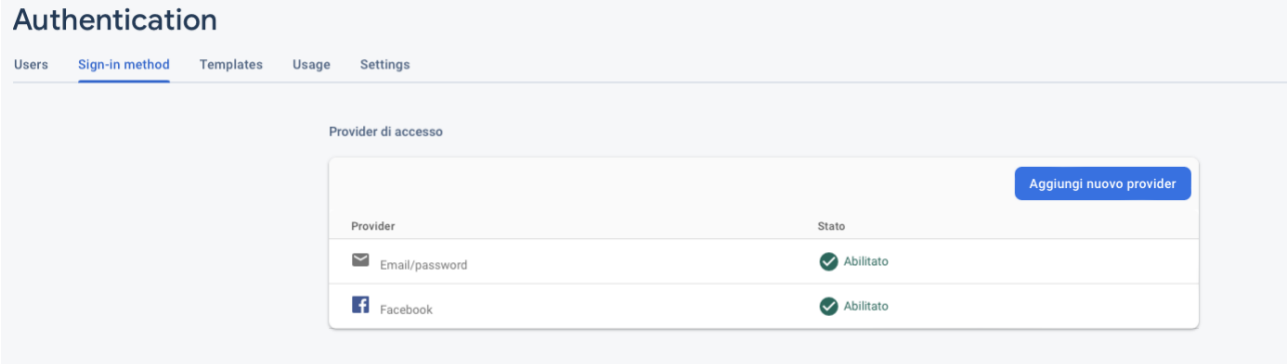

Figura 25 Firebase Dashboard Sign-in

Una volta configurata l'applicazione e il database si passa alla gestione delle varie funzionalità all'interno delle viste dove si dichiara in ognuna di esse una variabile di tipo "ObservedObject" che fa riferimento ad un'istanza dell'AuthenticationViewModel per gestire tutte le funzioni di autenticazione

3.1.2.1. Meta for developer: un tool per la connessione ai server di Facebook

Come è stato detto nella fase di autenticazione, uno dei metodi tramite i quali l'utente può accedere alla propria applicazione è tramite le proprie credenziali Facebook. Questa funzionalità è resa disponibili grazie al sito meta for developers. Ma come collegare la nostra applicazione all'API di Meta?

Il primo passo da seguire è, una volta creato un progetto con la procedura guidata del sito, inserire nella sezione di “Avvio Rapido” del sito l’ID Bundle dell’applicazione visionabile nelle impostazioni del progetto nell’IDE Xcode, successivamente una volta inserito l’ID Bundle, si configura il file “info. plist” situato all’interno del nostro progetto aprendolo come codice sorgente. Dopo aver aperto il file bisogna modificarlo inserendo il seguente codice:

```
<key>CFBundleURLTypes</key>
<array>
<dict>
<key>CFBundleURLSchemes</key>
<array>
<string>fbAPP-ID</string>
</array>
</dict>
</array>
<key>FacebookAppID</key>
<string>APP-ID</string>
<key>FacebookClientToken</key>
<string>CLIENT-TOKEN</string>
<key>FacebookDisplayName</key>
<string>APP-NAME</string>
```

inserendo al posto dell’APP-ID l’id dell’applicazione Facebook fornito dall’API, al posto di APP-NAME il nome della nostra applicazione (Monitor4U nel nostro caso) e al posto di “FacebookClientToken” il client token visibile nelle impostazioni della nostra applicazione all’interno dell’API fornita dal sito.

Conclusasi la configurazione del file “info. plist “per concludere la configurazione dell’ambiente è necessario importare o tramite CocoaPods o tramite SPM il framework Facebook disponibile sulla piattaforma di GitHub. Per poter usufruire dei dati mandati e ricevuti all’API di Facebook, per ragioni di semplicità del codice si è deciso di usufruire dell’approccio dell’”AuthenticationViewModel” precedentemente descritto.

Si è istanziata all’interno dell’AuthenticationFirebaseDatasource un’istanza della classe “FacebookAuthentication” presente nella libreria FacebookLogin importata grazie al framework scaricato dalla piattaforma di GitHub. Le funzionalità che Monitor4U si prefige di fornire come la connessione con il servizio di Facebook e la seguente possibilità di lasciare delle recensioni sulla pagina avvengono grazie alle funzioni: “loginWithFacebook” e “linkFacebook” le quali consentono la manipolazioni dei dati proveniente dall’API di Facebook, con l’unica differenza rispetto alle omonime funzioni delle credenziali riguardanti il servizio Email, che non devono gestire nessun email o password in quanto quest’ultime verranno passate dalla funzione “linkFacebook” che apre un link dove l’utente può inserire le proprie credenziali.

3.2. L’implementazione della vista “AuthenticationView”

Dopo aver introdotto gli elementi fondamentali che hanno contribuito alla configurazione di quella che sarà l’implementazione effettiva delle funzionalità proposte, possiamo spostare il focus sulla vista “AuthenticationView”. Come descritto in fase di progettazione, quest’ultima avrà come funzionalità principali da garantire le seguenti: consentire l’accesso all’applicazione tramite le credenziali sia e-mail che Facebook, in caso di primo accesso consentire all’utente di registrarsi trasferendo le informazioni al database di Firebase. Questi controlli avvengono grazie alla variabile di tipo “@ObservedObject “che fa riferimento ad un’istanza dell’AuthenticationViewModel, la quale viene passata come parametro alle viste “RegisterEmailView” per gestire la registrazione dell’utente e “LoginEmailView” per gestire l’inserimento delle credenziali di un utente già registrato. Queste viste vengono azionate dalla mutazione della variabile di tipo “@State” che fa riferimento ad un’istanza dell’enumerazione “AuthenticationSheetView” per gestire le varie viste che si devono aprire. La connessione con l’API di Facebook viene gestita grazie all’istanza dell’authenticationViewModel la quale all’azionarsi del bottone di Facebook richiama la funzione “loginFacebook” collegata all’AuthenticationFirebaseDatasource.

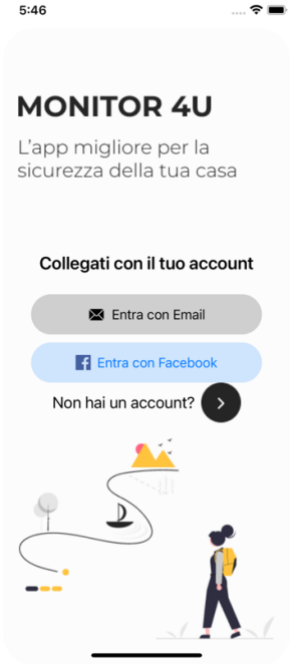

Figura 26 Schermata "AuthenticationView"

3.2.1. Vista “RegisterEmailView”: connessione con il database, vista “LoginEmailView”: accesso al database

Parte fondamentale della vista risulta la connessione al database, possibile grazie alla vista “RegisterEmailView”. Questa si presenta sotto forma di form nel quale l’utente può inserire l’e-mail e la password e infine inviarli tramite l’apposito pulsante di “Accedi”. La connessione al database avviene grazie alla variabile di tipo “@ObservedObject” che usa l’istanza dell’AuthenticationViewModel. Questa variabile è responsabile anche di eventuali controlli

quali; la presenza dell'e-mail nel database all'atto di registrazione, mandando un messaggio all'utente nel quale riferisce che l'e-mail è già in uso, un controllo riguardante la correttezza dell'e-mail stessa, nella fattispecie che deve soddisfare i caratteri tipici che un'e-mail deve avere e infine un controllo riguardante la lunghezza della password che deve essere almeno di sei caratteri. La vista "LoginEmailView" contribuisce all'interfacciamento con il database insieme alla vista precedente in quanto all'inserimento dei dati dell'utente viene verificata la presenza di quest'ultimi nel database e se presenti viene dato l'accesso alle altre funzioni dell'applicazioni altrimenti viene restituito un messaggio di errore, riguardante la mancata presenza di credenziali all'interno del database.

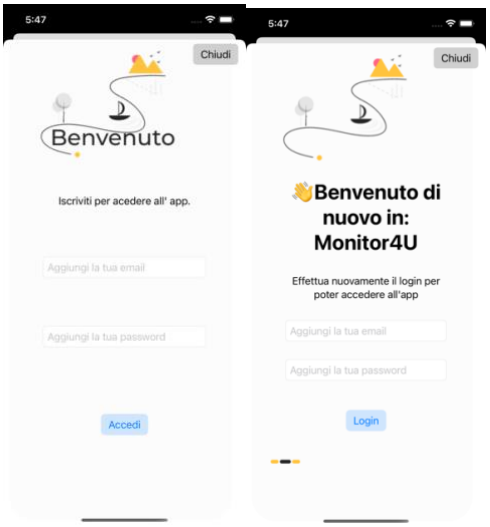

Figura 27 Schermate " RegisterEmailView " e "LoginEmailView"

3.2.2 ProfileView: altra gestione dati Firebase

La vista profileView si occupa di fornire un'implementazione del servizio di autenticazione fornito nella schermata di ingresso e precedentemente descritto. L'obiettivo è consentire all'utente che si è inscritto con le credenziali Facebook la possibilità di integrare le proprie credenziali E-mail qualora ne sia in possesso e viceversa nel caso in cui si sia autenticato con le credenziali della propria E-mail.Questa funzionaltià viene forntia sembre con una variabile di tipo "ObservedObject" che fa riferimento ad un'istanza dell'authenticationViewModel grazie alla quale è possibile richiamare le funzioni "linkEmailAndPassword" e "linkFacebook" per effettuare le funzionalità che Monitor4U si prefige di fornire all'utente. Qualora la connessione non andasse a buon fine verrebbe stampato un messaggio di errore controllato grazione alla gunzione "isAccountLinked" sempre dell'authenticationViewModel la quale controlla all'interno del database se le credenziali richieste dall'utente siano state inserite correttamente.

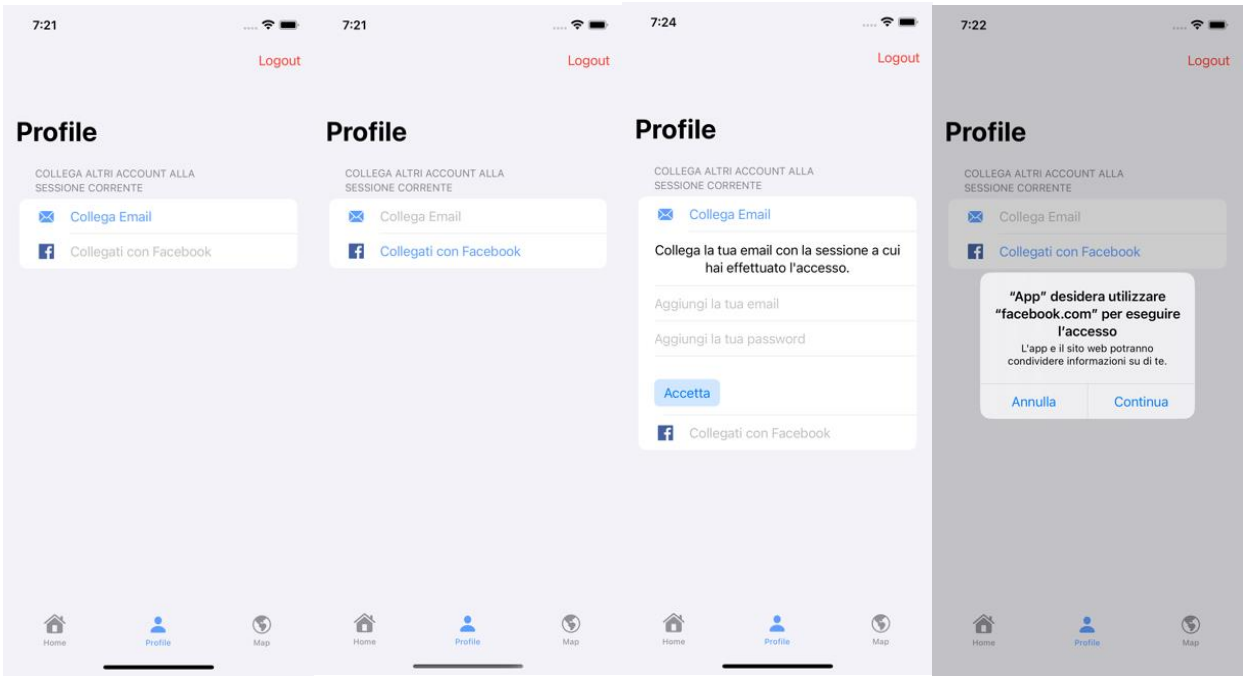

Figura 28 Schermata " ProfileView"

3.3. L'implementazione della vista "LocationsView"

La descrizione delle funzionalità implementate si sposta a questo punto sulla seconda vista sviluppata, denominata in Monitor4U come "LocationsView". Il nome assegnato alla vista è strettamente correlato alle informazioni che tende a rappresentare, in quanto sarà in quest'ultima gestita tutta la parte di registrazione e tracciamento delle abitazioni dell'utente, nella fattispecie , nome dell'abitazione, coordinate geografiche ed un collegamento al servizio di Mappe utile nel caso in cui ce ne sia bisogno di impostare il percorso più breve possibile, dalla propria posizione attuale calcolata grazie al servizio di geolocalizzazione e la casa di destinazione, per potervisi recare nel minor tempo possibile. Oltre alla visualizzazione delle varie abitazioni disponibili la LocationsView offre anche la possibilità di risalire ad ulteriori informazioni riguardanti la propria abitazione selezionata sulla mappa, aprendo su richiesta specifica dell'utente (cliccando sulla mappa), una vista denominata "Home" grazie alla quale è possibile connettersi all'abitazione per visualizzare lo stato dei dispositivi connessi. La vista si presenta composta da una sezione principale nella quale è rappresentata la mappa, sulla quale sono presenti tutte le informazioni di interesse. Oltre a questa sezione principale affinché l'utente non abbia difficoltà a ricercare tutte le varie abitazioni in suo possesso, sono state introdotte due ulteriori sezioni con lo scopo proprio di aiutarlo nello svolgimento di quest'attività, attenendosi alle Human Interface Guidelines. La prima sezione è composta dalla "LocationListView", una vista rappresentante tutte le località ubicate sulla mappa, con la possibilità di accedere subito ad ognuna di esse semplicemente cliccandoci sopra. La seconda sezione introdotta, riguarda la vista "LocationPreviewView"

la quale consente all'utente di scorrere ogni elemento della lista sovrastante cliccando sul bottone "Prossimo", ad ogni scorrimento della lista vi è, tramite animazione, un cambiamento del testo vicino al bottone , il quale andrà ad indicare il nome corrispondente all'elemento attuale della lista, mentre la figura sovrastante un omino nel caso in cui l'elemento rappresenti la posizione attuale, nel caso in cui l'elemento corrisponda ad una casa avremo una figura raffigurante il proprietario di quest'ultima.

3.3.1. LocationViewModel: gestione della Mappa

La gestione delle variabili associate alla Mappa avviene attraverso la classe LocationViewModel, nella vengono dichiarate le variabili "Locations" contenenti tutte le abitazioni visibili nella LocationsListView, e altre variabili di tipo "Published" utili per gestire la variazione della vista principale in base ai parametri cliccati dall'utente. Poiché l'applicazione deve adattare la vista in base alle richieste dell'utente, sono state introdotte le funzioni "nextButtonPressed" e "showNextLocation" le quali gestiscono le sotto viste precedentemente descritte modificando le viste con le informazioni che man mano vengono richieste.

3.3.2. La visualizzazione delle abitazioni sulla Mappa

Dal momento che Monitor4U si pone come obiettivo la visualizzazione dell'abitazione dell'utente all'interno della mappa integrata grazie al framework Mapkit, si è utilizzata la vista MapAnntoationView, la quale rappresenta ogni marcatore della mappa che può essere aggiunto di volta in volta, identificando l'abitazione dell'utente. Ovviamente dato che una mappa ha dimensioni molto grandi e quindi sarebbe risultato molto complicato per l'utente ricercare al suo interno la propria casa (seppur visualizzata dalla MapAnnotationView) sono state implementate le due sotto viste, considerando LocationsView come vista principale, LocationsListView e LocationPreviewView. Nella prima la visualizzazione delle località dell'utente avviene grazie alla variabile di tipo "EnvironmentObject" che fa riferimento ad un'istanza dell'LocationViewModel, consentendo alla vista di poter accedere all'elenco di abitazioni disponibili dell'utente. Nella seconda viene sempre dichiarata una variabile di tipo "EnvironmenteObject" che fa riferimento ad un'istanza del LocationViewModel grazie alla quale è possibile accedere a "Nome" e "immagine" associate ad ogni dimora e raffigurarle ogni qual volta su richiesta dell'utente il riferimento da una di esse si sposta ad un'altra. La connessione con il servizio di Mappe avviene grazie ad una variabile denominata "apriInMappeButton" la quale restituisce una View ed è caratterizzata da un bottone, il quale aziona l'apertura del programma Mappe passando come parametri della variabile "url" le coordinate geografiche della posizione in termine di altitudine e longitudine.

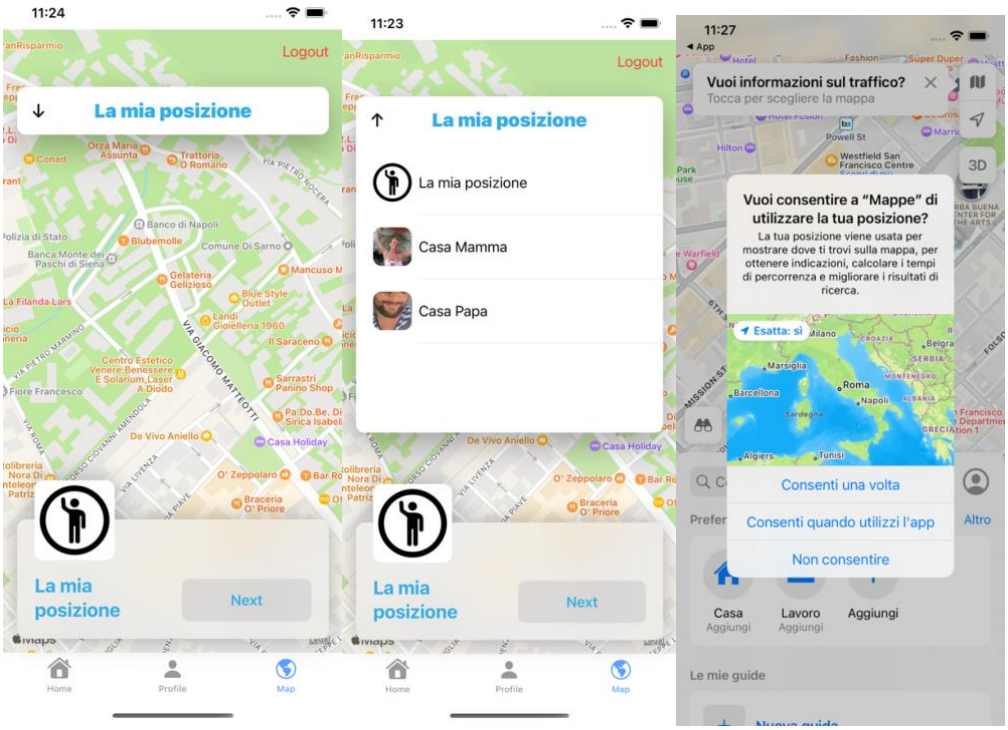

Figura 29 Schermata "LocationView"

3.4. L'implementazione della connessione con MQTT

Come detto in fase di progettazione, Monitor4U assiste l'utilizzatore nella connessione con la propria abitazione con lo scopo di vedere lo stato dei propri dispositivi e attivare/disattivare i sistemi di Care e Safety (cura e sicurezza della casa). Per poter implementare ciò, è stato necessario l'utilizzo del framework *MQTT* implementato grazie all'omonima libreria scaricata grazie al SPM dalla piattaforma di GitHub. L'approccio con il quale si è deciso di utilizzare la libreria è simile a quello utilizzato per implementare Firebase all'interno dell'applicazione, oppure il framework Mapkit in quanto i corrispettivi AuthenticationViewModel e LocationViewModel sono stati sostituiti dall'MQTTmanager, una classe responsabile della connessione con il broker "test.mosquitto.org" e delle funzioni utilizzate per gestire i vari topic che vengono utilizzati dall'utente e dai vari dispositivi messi a disposizione dal sistema di controllo.

3.4.1. MQTTManager: connessione tramite MQTT

Come anticipato, la connessione con MQTT avviene grazie alla classe MQTTManager. Un parametro importante di questa classe risulta essere la variabile di tipo "Published" *currentAppState* la quale fornirà ogni qual volta invocata lo stato corrente della connessione con il broker "Test.mosquitto.org". La classe "MQTTAppState" è un'ulteriore classe definita allo scopo di estendere le funzionalità della libreria precedentemente importa e per motivi di leggibilità di codice, in quanto si è preferito adottare questa scelta di creare una classe a parte perché altrimenti il codice sarebbe potuto risultare di difficile interpretazione. In quest'ultima classe vengono rappresentate le variabili "isConnected" e "isSubscribed" ti tipo

Bool, le quali hanno il compito di informare qualora la connessione sia attiva o meno e qualora l'applicazione si sia sottoscritta ad un particolare topic e la variabile "description" la quale invece ha lo scopo di informare sullo stato della connessione, connessa, disconnessa, sottoscritta/non sottoscritta (ad un topic). Tornando alla classe MQTTManager, al suo interno vengono definite le funzioni per inizializzare la connessione("inizializeMQTT"), connettersi o disconnettersi e gestire i vari topic utilizzati dai dispositivi della nostra abitazione.

3.5. L'implementazione della vista "HomeView"

Come definito in fase di progettazione, Monitor4U consente all'utilizzatore di controllare lo stato dei propri dispositivi e non solo tramite l'utilizzo di alcuni grafici, i quali aiutano l'utente ad avere una visione più chiara e globale di tutto il sistema. Per poter implementare ciò, è stato necessario l'utilizzo del framework *Charts*. In particolar modo, la visione di questi diagrammi avviene in parti diverse e con modalità altrettanto diverse all'interno della vista Home. Questo perché al primo accesso la vista si presenta con una figura raffigurante l'abitazione inserita e un bottone utile per connettersi ad essa. Una volta connessi alla casa all'utente appare una griglia con tutti i dispositivi connessi e dei quali può ricevere informazioni. A seconda di quale dispositivo viene azionato tramite l'apposito pulsante presente all'interno della griglia, appariranno dei grafici con dei bottoni utili per la connessione e la visualizzazione dello stato corrente del dispositivo.

Nel caso in cui l'utente voglia avere informazioni sullo stato del livello del Gas, cliccando sul bottone vedrà comparire una finestra contente un grafico con i livelli di gas durante la giornata precedente. Di default il grafico apparirà contenente le informazioni sotto forma di rettangoli ma tramite l'apposito pulsante "Grafico a linea" presente vicino ad esso sarà possibile visualizzare le informazioni seguendo un andamento lineare, a seconda di quali sono i gusti dell'utente. Al di sotto di quest'ultimo pulsante si trova un ulteriore pulsante con lo scopo di azzerarne i valori. Infine, riguardo la sezione del Gas vi è un ulteriore slider utile all'utente per visualizzare la soglia del livello del Gas oltre il quale la situazione è" critica" e per visualizzarne il livello attuale.

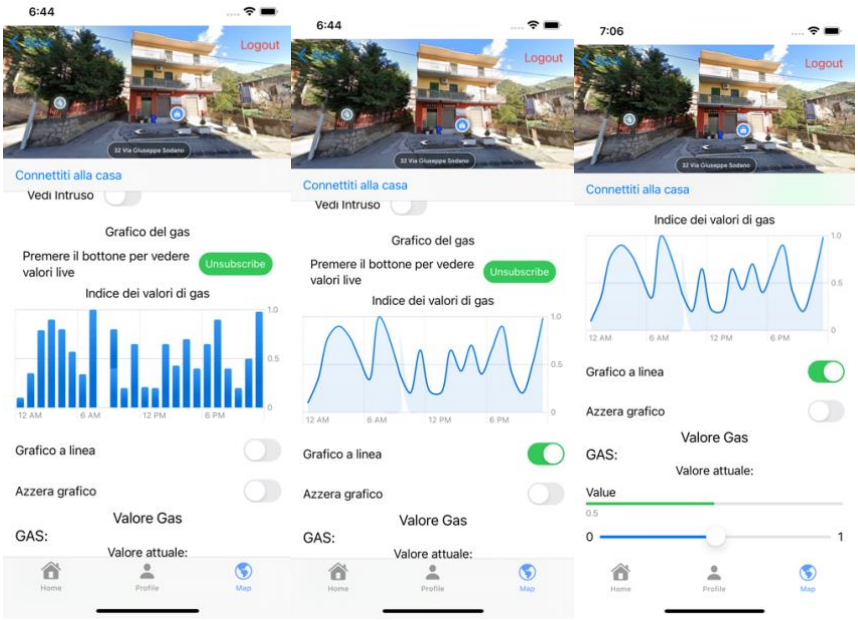

Figura 30 Schermata "HomeView"

Nel caso in cui l'utente voglia avere informazioni sullo stato del sistema di Care oppure del sistema di Safety, cliccando sul bottone vedrà comparire una finestra contente un bottone da premere per accedere al sistema e un pulsante per attivare e disattivare il sistema , sia di Care che di Safety.Ragionamento analogo nel caso del sistema di Aerazione, ricordiamo che questo viene azionato nel caso in cui la temperatura del Gas superi il livello critico e i valori mandati tramite il protocollo MQTT sono solo superiori alla velocità minima di 50, in quanto velocità minori non contribuiscono ad una corretta pulizia dell'aria all'interno dell'abitazione. Quando il sistema di azione la velocità della ventola viene resa visibile all'interno della Gauge. Infine, per quanto riguarda l'eventuale presenza di intrusi questa è resa visibile grazie all'apposito pulsante "Allarme" grazie al quale compare un testo che informa se il sistema ha rilevato o meno qualcuno di sospetto.

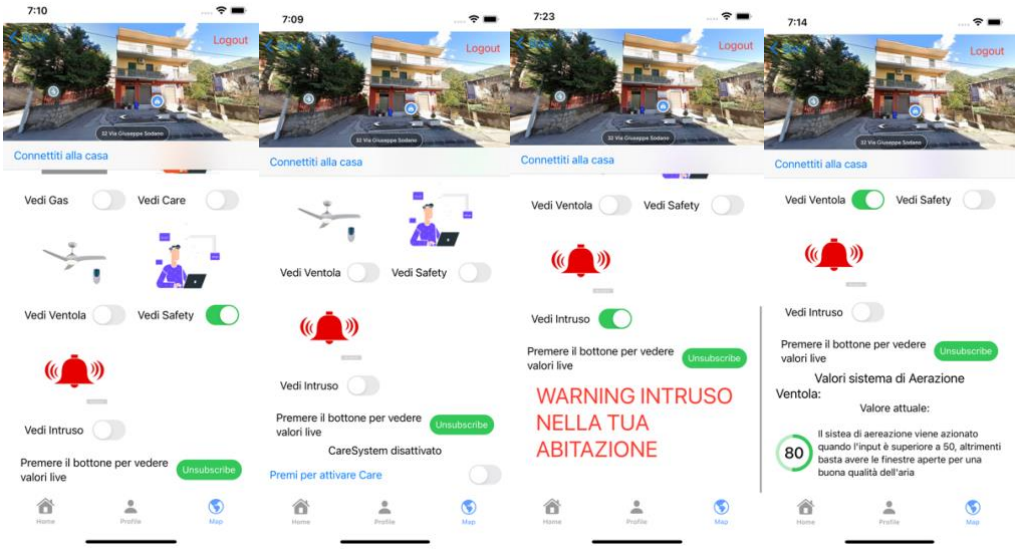

Figura 31 Schermata "HomeView" gestione sistema di controllo"

3.6. Gestione della User Interface & User Experience

La parte finale dell'implementazione e descrizione delle funzionalità è rivolta alla gestione della User Interface & User Experience. Nello specifico, Monitor4U è stata progettata attenendosi completamente alle Human Interface Guidelines. A partite da quest'ultime, per quanto riguarda la navigazione tra le viste si è deciso di implementarla attraverso la struttura *TabView*, una vista che passa tra più viste figlie utilizzando elementi dell'interfaccia utente, consentendo una rapida, efficace e semplice navigazione da parte dell'utilizzatore. Per ogni vista figlia, per quanto riguarda l'accesso alle viste rappresentative i form di compilazione si è deciso di implementare quest'ultime attraverso delle *Sheet View*, utilizzate per presentare nuove viste rispetto a quelle esistenti consentendo comunque all'utilizzatore di trascinare verso il basso per ignorare la nuova vista. Invece, la gestione degli elementi informativi contenenti le informazioni salienti dell'utilizzatore quali lo stato dei dispositivi dell'abitazione è stata implementata mediante l'utilizzo di *NavigationView*, le quali permettono di creare delle viste basate sulla navigazione in cui l'utente può attraversare a sua volta una raccolta di viste di destinazione selezionando un *NavigationLink* fornito. Di seguito verranno mostrate una serie di esempi di utilizzo di Monitor4U nelle situazioni descritte, in particolar modo facendo riferimento alla "homeView" visibile all'utente non appena accede all'applicazione con le proprie credenziali. La vista si compone di una prima descrizione volta a dare il bentornato all'utente e affianco ad essa vi è un'icona implementata con un "navigationLink" con un collegamento alla "ProfileView" precedentemente descritta. Successivamente vi è una griglia contenente un riepilogo delle funzionalità del sistema di controllo e implementate anche esse con un navigationLink rimandante ad una vista nella quale è presente una descrizione del dispositivo selezionato dall'utente e una lista di abitazioni dove esso è applicato.

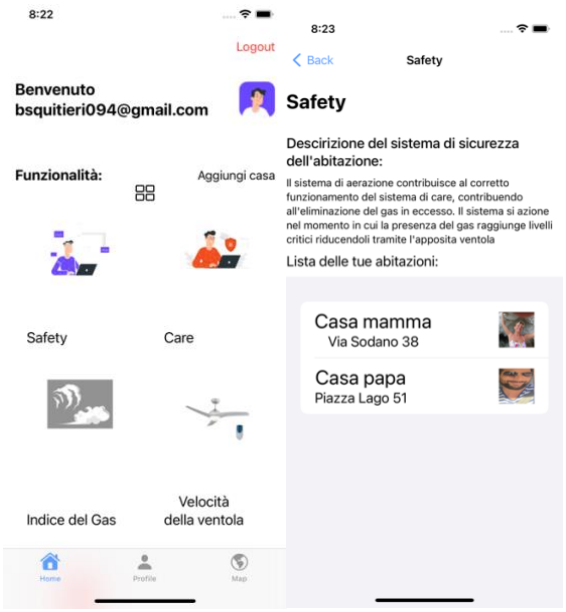

Figura 32 Schermate per uso NavigationLink

Al fine di poter garantire una corretta User Experience all'utilizzatore anche in assenza di informazioni registrate all'interno dell'applicazione durante il processo di connessione con l'abitazione è stata definita una parte informativa contenente rispettivamente un'immagine e un testo entrambi rappresentativi dell'assenza di connessione. Nel seguito vengono riportati un esempio di utilizzo di Monitor4U nella situazione appena descritta.

Figura 33 Schermata "no-connection"

4. Conclusioni e prospettive future

A conclusione di quanto affermato nei capitoli precedenti, andiamo a discutere da un livello più alto delle conclusioni tratte dalle fasi precedenti della progettazione, a partire dalle tecnologie utilizzate, passando per la fase di progettazione per poi concludere con la fase di implementazione e descrizione delle funzionalità, attraverso cui si definiscono in ottica futura tutte le migliorie ed estensioni che possono essere implementate in un secondo momento rispetto a quanto sviluppato e descritto all'interno dell'elaborato. La progettazione ed implementazione di Monitor4U è stata definita seguendo le fasi successive: si è partiti da una fase di progettazione attraverso la quale sono stati innanzitutto inquadrati gli obiettivi da finalizzare attraverso l'approccio del Challenge Based Learning ed i riscontri ottenuti, per passare successivamente al fulcro della fase condotta per la quale sono state scelte le piattaforme da utilizzare, i framework e successivamente la finalizzazione dell'idea tramite l'implementazione di un applicazione, dalla quale sono stati definiti tutti i prototipi associati per gestire la fase successiva di implementazione, la gestione dell'integrazione dei framework scelti rispetto alle piattaforme utilizzate e la definizione di tutte le funzionalità da implementare. Tutto questo è stato concretizzato attraverso Monitor4U , proposta per poter garantire all'utente un ottimo servizio di autenticazione, tramite credenziali E-mail e tramite credenziali Facebook e l'integrazione con un ottimo servizio di storage dei dati offerto dal

database Firebase, l'accesso ad una sezione tramite la quale è possibile monitorare lo stato del SafeAndCareSystem(sistema di cura e qualità dell'aria), un sistema fisico composto da dispositivi comunicanti grazie al protocollo MQTT, il protocollo per definizione dell'IOT. In particolar modo è resa disponibile all'utente una visione in tempo reali dei valori assunti dai dispositivi che compongono il sistema di controllo e la possibilità di controllarli tramite appositi comandi messi a disposizione di Monitor4U. Tutta la progettazione condotta è discussa successivamente attraverso l'implementazione e descrizione di tutte le funzionalità elencate, le quali sono state ottenute partendo da una fase di integrazione di viste e controller nativi SwiftUI, per passare successivamente all'implementazione in sé delle viste curate, con tutte le funzionalità a corredo per passare poi alla cura della User Interface & User Experience.

Dall'implementazione finale il risultato ottenuto si propone, nello scenario in cui è stato inquadrato, a nuove implementazioni e migliorie, soprattutto in riferimento alla parte di Monitor4U delegata alla registrazione e la visualizzazione dello stato dei dispositivi. Seppur sia stata garantita la connessione tramite il protocollo MQTT con i dispositivi componenti il sistema di controllo SafeAndCareSystem, l'applicazione potrebbe comunque fornire funzionalità aggiuntive come un miglior storage dei dati riguardo i dispositivi stessi, con lo scopo di fornire all'utente maggiori info riguardo i valori assunti non soltanto durante la giornata precedente ma anche durante periodi temporali più ampi. Per quanto riguarda la fase di autenticazione si potrebbero implementare ulteriori metodi di accesso, come uno basato su Instagram o Twitter sulla falsa riga della modalità vista per Facebook, fornendo così un livello di sicurezza ancora più alto. Infine, per quanto riguarda la fase di aggiunta delle abitazioni potrebbe essere associata un'ulteriore sezione per personalizzare anche i dispositivi costituenti il sistema di controllo stesso, con l'obiettivo di fornire quindi un servizio di personalizzazione migliore e non limitativo ai dispositivi maggiormente usati in tutti i sistemi di controllo, come si è evinto dal nostro sondaggio durante le fasi iniziali del CBL. Per quanto riguarda la fase della gestione dei dispositivi si potrebbe utilizzare il servizio di immagazzinamento dei dati grazie a firebase, integrandolo con quello di CoreData utilizzabile tramite Swift, con lo scopo di fornire un maggior bagaglio di informazioni all'utente. Tutte queste migliorie saranno applicate all'applicazione, con lo scopo di fornire un servizio più completo all'utilizzatore.

Bibliografia e Sitografia

1. Wikipedia, «Visual Studio Code,» [Online]. Available: https://it.wikipedia.org/wiki/Visual_Studio_Code.
2. Wikipedia, «XCode,» [Online]. Available: https://it.wikipedia.org/wiki/Xcode.
3. Wikipedia, «GitHub,» [Online]. Available: https://it.wikipedia.org/wiki/GitHub.
4. Wikipedia, «Swift,» [Online]. Available: https://it.wikipedia.org/wiki/Swift_(linguaggio_di_programmazione).
5. C. B. Learning, «Challenge Based Learning,» [Online]. Available: https://www.challengebasedlearning.org/it/framework/.
6. https://developer.apple.com/design/human-interface-guidelines/platforms/designing-for-ios/
7. https://kiteblue.it/blog/ux-ui-principali-differenze
8. https://developer.apple.com/documentation/charts/
9. https://www.figma.com/
10. https://uizard.io/
11. https://firebase.google.com/?gclid=Cj0KCQjwjvaYBhDlARIsAO8PkE3NHq-PEhmunl58XGdcIZprI21fbGfY8aDovysQ7-cJS6f1WTky0R4aAp2iEALw_wcB&gclsrc=aw.ds
12. https://developers.facebook.com/?locale=it_IT
13. https://www.hackingwithswift.com/quick-start
14. https://www.hivemq.com/mqtt-essentials/?utm_source=adwords&utm_campaign=&utm_term=mqtt&utm_medium=ppc&hsa_tgt=kwd-296477891479&hsa_cam=17295502954&hsa_src=g&hsa_net=adwords&hsa_kw=mqtt&hsa_ad=594094363642&hsa_grp=138486496202&hsa_ver=3&hsa_acc=3585854406&
15. https://www.hackingwithswift.com/swift/5.3/spm-improvements
16. https://www.hackingwithswift.com/books/ios-swiftui/integrating-mapkit-with-swiftui

Indice delle figure

Ringraziamenti

Ed eccoci giunti alla fine di questo percorso, il più importante della mia vita. Grazie ad esso sono cresciuto e maturato professionalmente, ho capito cosa significhi davvero impegnarsi e soprattutto come ogni obiettivo, con la costanza e la dedizione, sia raggiungibile. Questo percorso di studi è stato fondamentale per capire quale fosse davvero la mia inclinazione, dalla quale sono nate tutte le mie passioni, ed anche se rappresenta un traguardo raggiunto, in realtà è un punto di partenza per la mia carriera, un percorso che vuole essere ancor di più coltivato e arricchito di conoscenze, per poter dare in futuro un contributo notevole. Mi è doveroso ringraziare tutte le persone che hanno contribuito al raggiungimento di questo primo traguardo, con le quali mi auguro di poterne condividere altrettanti.

Vorrei ringraziare tutti coloro che, durante la stesura di questa tesi e durante i miei tre anni di attività accademica, mi hanno supportato, facendomi crescere sotto ogni punto vista. Un ringraziamento speciale e sincero va anzitutto al professore nonché mio relatore, Luca Greco, per la sua disponibilità nel seguirmi nel mio percorso di tirocinio e di stesura della tesi, nonché per la sua fiducia, permettendomi di sviluppare conoscenze e competenze che avrò modo di completare e approfondire al meglio nei seguenti anni accademici.

Grazie ai miei genitori che, con tutti gli alti e bassi che abbiamo avuto in famiglia, mi hanno sempre supportato per il raggiungimento di tutti gli obiettivi posti, ed è anche grazie a loro se ora sono qui a scrivere questo.
Ringrazio i miei colleghi del “Gruppo di fuoco”, che hanno reso questi tre anni universitari semplici e divertenti, colmi di soddisfazioni e di un interesse crescente nello studio e nello sviluppo dei nostri progetti.

Ed infine, ringrazio tutte le persone che in questi tre anni mi hanno accompagnato, in un modo o nell’altro, in questo percorso, formandomi sia dal punto di vista accademico che professionale.