



Università degli Studi di Camerino

SCUOLA DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica (Classe L-31)

**Analisi e sviluppo di un sistema software
per la domotica d'ufficio: il frontend di
"Proximity System"**

Laureando

Marco D'Argenio

Matricola 090315

Relatore
Prof. Francesco De Angelis

Correlatore
Dott. Francesco Strazzullo

Indice

1	Introduzione	11
1.1	E-xtrategy	11
1.2	Confronto e scelta del metodo di lavoro	12
2	Business Model Canvas	15
2.1	Business Model	15
2.2	Business Model Canvas	15
2.2.1	Esperienza lavorativa	16
2.2.2	Clienti	18
2.2.3	Valore offerto	18
2.2.4	Canali	20
2.2.5	Relazione con i clienti	20
2.2.6	Attività	21
2.2.7	Risorse chiave	22
2.2.8	Partner	23
2.2.9	Ricavi	24
2.2.10	Costi	24
3	Frontend	27
3.1	Scelta Tecnologica	27
3.1.1	Javascript	27
3.1.2	Scelta del framework: AngularJS	27
3.2	AngularJS	28
3.3	MVC	28
3.4	Funzionalità principali di Angular	28
3.4.1	Uso dei moduli	28
3.4.2	Direttive	29
3.4.3	Data Binding bidirezionale	31
3.4.4	Controller	31
3.4.5	Filtri	33
3.4.6	Servizi	33
3.4.7	Dependency injection	34
3.4.8	Tipologie delle applicazioni mobili	34
3.4.9	Ionic	35

4 Material Design	37
4.0.1 Scheumorfismo	37
4.0.2 Flat Design	38
4.0.3 Material come sintesi	39
4.1 Caratteristiche Material Design	40
4.1.1 Comportamento nello "spazio"	40
4.1.2 Uso delle Ombre	41
4.1.3 Animazioni	41
4.1.4 Colori, Immagini e Font	42
4.2 Componenti principali	43
4.2.1 Bottom Navigation	43
4.2.2 Button	44
4.2.3 Card	44
4.2.4 Dialog	45
4.2.5 List	46
4.2.6 Bottom Sheets	47
5 Angular Material	49
5.1 Accessibilitá del prodotto	49
5.2 Flex	49
5.3 Gestione dei Temi	51
5.4 Componenti Principali	52
5.4.1 Sidenav	52
5.4.2 Toolbar	53
5.4.3 Card	54
5.4.4 Raised Button	55
5.4.5 Flat Button	55
5.4.6 Float Button	56
5.4.7 Button md-fab-speed-dial	56
6 Screenshot Ionic App	59
7 Conclusioni	61
A Installazione Proximity System	63
A.1 Backend e WebApp	63
A.2 App	64

Elenco dei codici

3.1	metodo module()	29
3.2	servizio e controller in modulo	29
3.3	unicam-beacon-server/webserver/angular/index.html	29
3.4	unicam-beacon-server/webserver/angular/templates/home.html	30
3.5	unicam-beacon-server/webserver/angular/templates/actions.html	30
3.6	Data binding	31
3.7	Controller	32
3.8	webserver/angular/js/controllers/login.js	32
3.9	Filtri GPIO e IO	33
3.10	webserver/angular/js/services/devices.js	34
5.1	webserver/angular/templates/home.html	49
5.2	/webserver/angular/js/routes.js	51
5.3	webserver/angular/index.html	52
5.4	webserver/angular/templates/gpio.html	53
5.5	webserver/angular/templates/gpio.html	54
5.6	esempio Raised Button	55
5.7	esempio Flat Button	56
5.8	CSS Float Button	56
5.9	codice FAB	56

Elenco delle figure

1.1	Trello del nostro progetto	13
2.1	BMC	17
2.2	Clienti	18
2.3	Valore offerto	19
2.4	Canali	20
2.5	Relazioni con i clienti	21
2.6	Attività	21
2.7	Risorse chiave	22
2.8	Partner	23
2.9	Ricavi	24
2.10	Costi	25
3.1	Use Case della nostra Applicazione	27
3.2	Data binding	31
4.1	Calcolatrice nello Scheumorfismo	38
4.2	Calcolatrice nel Flat Design	38
4.3	Calcolatrice nel Material Design	39
4.4	Ambiente del Material Design	40
4.5	Uso delle ombre	41
4.6	Effetto Ripple	42
4.7	Esempio di distribuzione dei colori Primary (viola) e Accent (verde) . .	42
4.8	Esempio Navigation Bar	43
4.9	I Button	44
4.10	Esempio di Card	44
4.11	Esempio di Dialog	45
4.12	Esempio di List	46
4.13	Esempio di Bottom Sheet	47
5.1	Home Responsive	50
5.2	Gestione Temi	52
5.3	Sidenav Proximity System	52
5.4	Card GPIO Proximity System	54
5.5	Raised Button in una Card	55

5.6	Flat Button di logout in una Card	55
5.7	Float Button	56
5.8	md-fab-speed-dial	56
6.1	Connessione iniziale con il server di Proximity System.	59
6.2	Alcune delle pagine principali dell'app.	60

Elenco delle tabelle

2.1 Leggenda BMC	17
----------------------------	----

1. Introduzione

Questo lavoro rappresenta la terza ed ultima parte di un lavoro composto da tre step:

1. stage in azienda;
2. progetto;
3. tesi finale;

Nelle attività di stage sono stato affiancato dal tutor Francesco Strazzullo, che per me è stato una guida essenziale e un amico con cui condividere aspirazioni e la passione per l’informatica. In azienda ci è stato chiesto di realizzare un’applicazione per la domotica. La domotica è una scienza interdisciplinare che si occupa dell’applicazione dell’informatica e dell’elettronica alla gestione dell’abitazione. Attraverso l’uso di piccoli trasmettitori bluetooth (beacon) si riesce a gestire dispositivi ed impianti usati non solo in casa, ma anche in luoghi come uffici. Al prodotto da offrire sul mercato abbiamo dato il nome di ”Proximity System”. Il prodotto che dovevamo progettare deve, per l’appunto, permettere la geolocalizzazione interna nei locali ed il controllo dei dispositivi che possiamo trovare in un’abitazione. Per la realizzazione di Proximity System abbiamo pensato di iniziare con il backend, eseguito su di un Raspberry PI, un’applicazione mobile in Ionic per la gestione dei dispositivi e infine una web app dove avviene l’amministrazione del sistema.

Il codice del progetto è disponibile nei due repository:

1. [www.github.com/e-xtrategy/unicam-beacon-server](https://github.com/e-xtrategy/unicam-beacon-server)
2. [www.github.com/e-xtrategy/unicam-ionic-beacon-app](https://github.com/e-xtrategy/unicam-ionic-beacon-app)

1.1 E-xtrategy

Lo stage è stato svolto presso l’azienda E-xtrategy di Monsano, in provincia di Ancona. Innovativa internet company, costituita nel 1995 da tre soci fondatori, che iniziarono a lavorare all’interno di un’agenzia di comunicazione, dedicandosi all’approfondimento delle nuove tecnologie e di Internet, mondo allora ancora in larga parte inesplorato.

Nel marzo 2001 questo sodalizio fortuito tra i tre amici si cristallizza in una scelta, quella di dare vita alla start-up E-xtrategy. Il 2009 è la chiave di volta per E-xtrategy: viene ridisegnato il proprio approccio al lavoro attraverso il metodo agile¹ e il metodo Lean. L’azienda aiuta altre aziende e startup a raggiungere i propri obiettivi business utilizzando internet, la tecnologia e un approccio lean; rientrano nella sua linea di prodotti:

¹ Agile Software Development, ASD

1. mentoring;
2. digital strategy;
3. user experience;
4. web development;
5. mobile development;

E-xtrategy é l'azienda a cui vanno i miei ringraziamenti per la sua ospitalitá e per avermi dato la possibilità, oltre allo sviluppo del progetto, di vivere per tre mesi la realtá del mondo lavorativo, a me ancora sconosciuta, insieme al carissimo amico e collega di universitá Saverio Tosi e all'aiuto indispensabile e prezioso del mio relatore Prof.re Francesco De Angelis.

1.2 Confronto e scelta del metodo di lavoro

Visto il mercato a cui si rivolge la nostra azienda (il web), un mercato molto dinamico, il lavoro di analisi e progettazione non é stato quello tradizionale, ma innovativo. Abbiamo lavorato in piccoli team interfunzionali attraverso una progettazione partecipata con il coinvolgimento diretto anche del cliente. Prima di poter iniziare il nostro progetto abbiamo dovuto scegliere il programma di progettazione lavoro. In azienda si lavorava con un mix tra Scrum e Kanban².

Lo Scrum é un framework agile per la gestione del ciclo di sviluppo del software. Esso ci ha permesso di creare una serie di iterazioni, cioè atti ripetitivi di un processo che ci permettevano di raggiungere l'obiettivo a determinati intervalli regolari chiamati sprint, dando la possibilità di consegnare al cliente con cadenze costanti. Questo framework é molto utile in quanto permette un'attività di feedback e di stime in tempo reale. Ogni sprint é formato dalla seguente struttura:

1. Pianificazione: un incontro iniziale in cui definire cosa fare con lo sprint corrente.
2. Scrum giornaliero: un mini incontro giornaliero della durata di una decina di minuti per sincronizzare il team.
3. Demo: un incontro per illustrare il lavoro svolto dal team
4. Retrospettiva: una revisione di ciò che é stato fatto e cosa fare per farlo meglio la volta successiva.

Il Kanban, invece, é un metodo operativo per far circolare le informazioni in modo sistematizzato all'interno dell'azienda ed eventualmente tra azienda e fornitori eliminando la necessità di sistemi complessi di programmazione.

Il Kanban si configura come un cartellino quadrato o una board virtuale che contiene le informazioni necessarie. Di conseguenza esso rappresenta il motore dell'attività dell'azienda gestendo in modo automatico la quotidianità degli ordini di lavoro, consentendo al team, nel caso nostro, di risolvere le criticità e di sviluppare i miglioramenti del sistema. Le board virtuali diventano molto importanti per lo sviluppo del software, in quanto il lavoro svolto dal team puó essere visualizzato in qualsiasi momento, come

²termine giapponese che vuol dire insegnare e inventato da Taichi Ohno

anche qualsiasi imprevisto puó essere subito visualizzato per essere risolto. Inoltre, il flusso delle informazioni risulta essere standardizzato.

Kanban puó essere sintetizzato in cinque fasi. Esse sono:

1. Visualizzare il flusso di lavoro
2. Limitare il Work-in.Progress
3. Misurare e gestire il flusso
4. Rendere le politiche di processo esplicite
5. Utilizzare i modelli per riconoscere le opportunitá di miglioramento.

Abbiamo avuto modo di utilizzare il Kanban in modo non del tutto completo, in quanto il nostro team essendo abbastanza piccolo non necessitava di tutte le procedure, per cui alla fine é stato piú utilizzato Trello, in quanto piú utile per le nostra finalitá.

Trello é uno strumento di project management molto semplice e molto versatile. Esso si basa sul sistema Kanban, utilizzato negli anni '80 per gestire il processo produttivo. Kanban vuol dire cartello, che é praticamente il mattone di base di Trello.

Gli elementi chiave di Trello sono 3:

1. La Board: la lavagna o tabellone che rappresenta il layout dove impostare il progetto;
2. La List: a seconda del board rappresentano: i macro elementi di un progetto (es. attivitá "da fare" oppure "In corso" o "Eseguito"), dei raccoglitori di idee o possono riguardare le figure che fanno parte di un team con le loro liste dei loro rispettivi compiti nel board;
3. la Card: é il modulo singolo del board e rappresenta l'idea o il compito da svolgere.

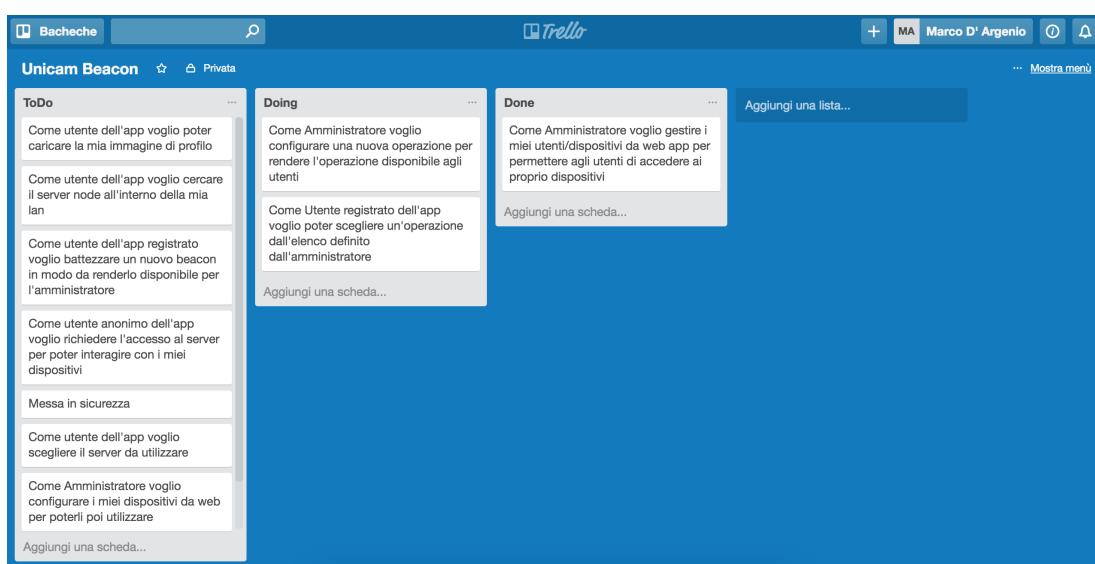


Figura 1.1: Trello del nostro progetto

Utilizzando il "To do, Doing e Done" secondo la figura si crea un flusso di lavoro, che dà subito una visione del progetto in corso. I board possono essere condivisi o meno,

quindi si possono anche rendere pubblici e visibili a tutti quelli del team di lavoro con la possibilità di collaborare in maniera immediata con altre persone. Le card, che contengono i vari task, sono dentro alla lista e si possono muovere facendo drag and drop da una lista all'altra, con l'avanzare del progetto.

Questo software usa il "card back" composto da quattro parti:

1. Card description: cartella con le informazioni utili per uno specifico task da usare come un promemoria per ricordare ad esempio al designer gli input di briefing del progetto;
2. Comments and Activity: questa sezione fa da timeline delle attività e dei commenti sul task. Si può indicare una figura del team del board con una @ prima del suo nome per avere una conversazione con lui. Nei commenti si possono allegare file o link ad altre card;
3. Add: sulla destra c' è la sezione "Add" con cui si può: aggiungere nuovi membri, dare un task a uno o più membri, fare checklist, indicare termini di scadenza, ci si può connettere con Dropbox, Google Drive, Box e OneDrive;
4. Actions: per poter spostare, copiare, archiviare una card oppure sottoscriverla. Le notifiche arriveranno ad ogni modifica fatta.

L'uso di Trello si è rilevato per noi molto utile, in quanto ci ha aiutato ad avere una visione completa del nostro progetto in ogni momento.

2. Business Model Canvas

2.1 Business Model

I cambiamenti sociali, culturali, scientifici e tecnologici influenzano l'attività aziendale costringendola ad affrontare situazioni di incertezza e di rischio sempre nuove ed imponendole di offrire situazioni rapide e adeguate a consumatori diversi ed in continua evoluzione. Spesso in mercati molto dinamici come quello dei prodotti tecnologici diventa di fondamentale importanza cercare di essere innovativi anticipando le aspettative dei clienti. Oggi l'innovazione è diventata una componente indispensabile del successo aziendale costruito sull'idea che "si produce ciò che si può vendere".

L'impegno e l'attenzione verso i consumatori e la capacità di instaurare relazioni continuative e differenziate con i clienti ripagano l'impresa in termini di: fidelizzazione della clientela, visibilità e successo dei prodotti, aumento dei clienti abituali, giro d'affari più soddisfacente, calo dei reclami. Per fare tutto ciò diventa indispensabile attuare politiche di marketing che mirino a regolare i rapporti tra impresa e mercato, ad identificare bisogni e desideri della clientela, a trovare soluzioni in grado di soddisfarli.

L'insieme delle soluzioni organizzative e strategiche che l'azienda intende adottare possono essere rappresentate in un Business Model, che tradotto in italiano vuol dire Modello Economico. Esso serve per descrivere come funziona un'azienda dal punto di vista economico, come, cioè, essa riesce a creare, distribuire e catturare valore per i propri clienti. Creare valore vuol dire risolvere un problema, soddisfare un bisogno, svolgere un'attività. Se si è prodotto oppure no un buon Business Model dipende dalla sua capacità di creare valore per i clienti dell'azienda e ottenerlo, così, un vantaggio competitivo. Esso diventa il primo documento da redigere da parte dell'azienda ed uno dei metodi per la sua redazione è il Business Model Canvas.

2.2 Business Model Canvas

Il Business Model Canvas è un metodo per redigere un Business Model. La comodità di questo metodo è che il tutto sta su un foglio di carta, dove tutti insieme costruiscono il modo in cui l'azienda organizza se stessa e la sua offerta per creare il massimo valore per i suoi clienti.

Il Business Model Canvas è stato inizialmente proposto da Alexander Osterwalder nel suo primo lavoro Business Model Ontology (2004) e successivamente sviluppato dallo stesso Osterwalder, da Yves Pigneur e da Alan Smith insieme ad una community di 470 esperti in 45 paesi del mondo: questo ha portato alla pubblicazione del libro Business Model Generation (in Italia "Creare modelli di business"). Oggi un best seller mondiale tradotto in 30 lingue, per cui il Business Model Canvas è diventato uno standard internazionale. Grazie all'intuizione di Alexander Osterwalder questo modello ha rivo-

luzionato il modo di rappresentare un business model. Infatti, con facilitá, tutti hanno la possibilità di comprendere elementi complessi che riguardano il funzionamento di un'intera azienda, in quanto offre un grande vantaggio comunicativo. Infatti il lavoro di Osterwalder propone un unico modello realizzato a partire dalle similitudini di una vasta gamma di altri frameworks che si sono succeduti nel tempo, creando una sintesi funzionale decisamente esaustiva.

Il Business Model Canvas, descritto nel best seller, é uno strumento strategico che utilizza il linguaggio visuale per creare e sviluppare modelli di business innovativi. Consente di rappresentare visivamente il modo in cui un'azienda crea, distribuisce e cattura valore. Esso diventa una specie di cartellone, una stampa di grandi dimensione, che mostra con facilitá le strategie adottate, che possono essere cambiate in qualsiasi momento a seconda delle esigenze. Basta modificare, aggiungere, togliere o sostituire un semplice post-it in qualsiasi momento. In questo modo lo strumento favorisce la comprensione, la discussione e l'analisi del business, ma allo stesso tempo favorisce anche la creatività e la condivisione. Sono, inoltre disponibili applicazioni web-based che replicano il Business Model Canvas online.

Il Business Model Canvas é rappresentato da blocchi di nove elementi costitutivi di un'azienda. I nove elementi sono:

1. Customer Segments (Clienti): i segmenti di clientela ai quali l'azienda si rivolge;
2. Value Proposition (Valore offerto): la proposta di valore contenente i prodotti/ servizi che l'azienda vuole offrire;
3. Channels (Canali): i canali di distribuzione e contatto con i clienti;
4. Customer Relationships (Relazioni con i clienti): il tipo di relazioni che si instaurano con i clienti;
5. Revenue Streams (Ricavi): il flusso di ricavi generato dalla vendita di prodotti/- servizi;
6. Key Resources (Risorse chiave): le risorse necessarie per cui l'azienda funzioni;
7. Key Activities (Attività chiave): le attivitá necessarie che servono per rendere funzionante il modello di business aziendale;
8. Key Partners (Partner): i partner chiave con cui l'impresa puó stringere alleanze;
9. Cost Structure (Costi): la struttura dei costi che l'azienda dovrá sostenere.

Per capire come creare un Business Model Canvas é importante soffermarci sui singoli 9 blocchi che lo costituiscono per poter poi costruire il modello riguardante il nostro prodotto da introdurre sul mercato.

2.2.1 Esperienza lavorativa

Io e il mio collega Saverio Tosi abbiamo lavorato con l'aiuto e i consigli dell'azienda E-xtrategy alla creazione del Business Model Canvas, costruito per la creazione e la commercializzazione del nostro prodotto "Proximity System". Esso non é altro che un'applicazione di domotica. Nei prossimi paragrafi analizzeremo punto per punto i vari step del Business Model Canvas e come esso stato costruito rispetto alla nostra applicazione.

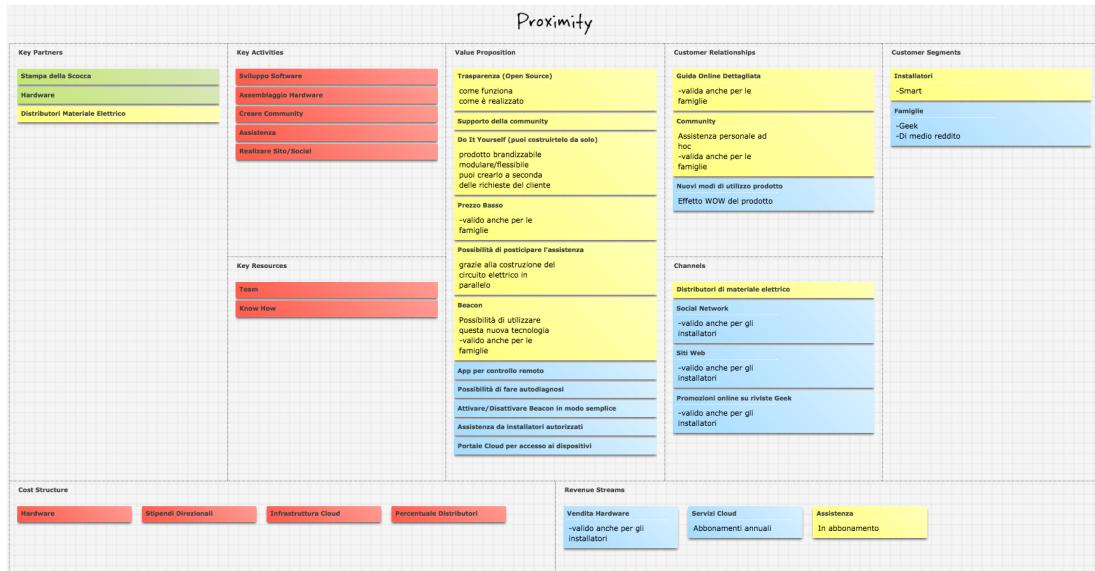


Figura 2.1: BMC

Colori	Attributi
Giallo	Installatori
Azzurro	Famiglie
Rosso	Team
Verde	Parner

Tabella 2.1: Leggenda BMC

2.2.2 Clienti



Figura 2.2: Clienti

I clienti rappresentano la domanda, i potenziali consumatori del nostro prodotto. Essi formano il mercato a cui rivolgersi. Nessuna impresa, purtroppo, può soddisfare con la sua offerta di beni e servizi tutti i consumatori, diversi fra loro per classe sociale, collocazione geografica, reddito, gusti, bisogni, desideri, cultura. Quindi, ogni impresa deve individuare il settore di mercato sul quale intende concentrare l'offerta: tecnicamente, deve attuare la segmentazione del mercato. La segmentazione del mercato è un genere di classificazione che consente all'impresa di suddividere il mercato in gruppi di persone (segmenti), aventi caratteristiche proprie ed omogenee. Definito il segmento di mercato, l'impresa deve decidere su quale mercato obiettivo deve concentrare le strategie. Il mercato obiettivo (o mercato target) è costituito da un gruppo di persone o di organizzazioni verso le quali l'impresa indirizza i propri obiettivi e le proprie strategie di marketing. Con il termine target, quindi, ci riferiamo ad un sottogruppo nell'ambito di un segmento ed è a questo sottogruppo che un'impresa si rivolge per trasformare un consumatore generico in un cliente, possibilmente in un cliente fedele. Ciò è possibile solo se si propongono prodotti e/o servizi che interpretino al meglio i bisogni e le aspettative dei consumatori. Noi abbiamo pensato di dividere i clienti in due grandi Target:

1. Le famiglie di medio reddito e con una passione per la tecnologia (Geek);
2. Gli Installatori "Smart".

Abbiamo scelto di puntare sul target di medio reddito in quanto il nostro prodotto ha un prezzo di vendita molto più basso rispetto a ciò che i competitor offrono. Il nostro punto di forza, infatti, è il costo di realizzazione del prodotto, in quanto non è formato da componenti che richiedono costi elevati.

2.2.3 Valore offerto

Quando si introduce un prodotto si deve fare in modo che venga identificato dai consumatori come unico per i suoi tratti distintivi. I prodotti per imporsi sul mercato devono



Figura 2.3: Valore offerto

avere una "personalità" che li contraddistingua dagli altri, che li faccia emergere e riconoscere dal pubblico di riferimento. Ciò implica che con la nostra offerta produttiva si debba costruire una forte identità verso l'esterno: è grazie al prodotto che il cliente riconosce i simboli che caratterizzano l'impresa. Importante è il posizionamento del prodotto, che è l'insieme delle decisioni e delle politiche imprenditoriali che hanno come scopo la creazione o il mantenimento di un'idea di quel determinato prodotto nella mente dei consumatori. Pertanto il posizionamento può diventare un vero e proprio patrimonio aziendale che incide sulle decisioni e orienta le scelte della clientela in tutte le fasi che precedono, che accompagnano e che seguono l'acquisto: quando il consumatore pensa di acquistare, quando acquista, nel momento in cui utilizza il bene o il servizio e quando valuta, eventualmente, di sostituirlo. In base a queste considerazioni abbiamo creato i nostri punti di forza del nostro prodotto che possono essere vantaggiosi per i nostri target di clientela:

1. Trasparenza e "Do it yourself": l'installatore ha il vantaggio di poter visionare l'intero codice del prodotto su Github. Inoltre l'installatore può creare il proprio prodotto personalizzato a seconda dell'esigenza del cliente;
2. Supporto della community e Assistenza: tutti e due i nostri target hanno una tipologia di assistenza diversa. Da una parte abbiamo le famiglie che possono contare su personale qualificato, dall'altra, invece, si punta alla creazione di una community con la quale l'installatore possa trovare riscontro dei problemi che ha avuto, per trovarne una soluzione. Inoltre è possibile da parte delle famiglie effettuare autodiagnosi. L'installatore, d'altra parte, può posticipare l'assistenza, in quanto Proximity System in caso di guasto, essendo costruito in parallelo, non influisce sul quadro elettrico;
3. Tecnologie utilizzate: il punto di forza del nostro prodotto è sicuramente l'utilizzo dei Beacon, una piccola antenna trasmettitore bluetooth, che riesce a rendere automatizzate le varie tipologie di funzioni. Inoltre, abbiamo pensato di introdurre in futuro un portale Cloud per permettere di effettuare operazioni a distanza.

2.2.4 Canali



Figura 2.4: Canali

I canali riguardano le modalità di trasferimento dei prodotti dal produttore al consumatore e la possibilità di rendere i prodotti disponibili al momento giusto e nella quantità giusta. Dal canale distributivo dipendono quindi la presenza dei prodotti sul mercato e l'accessibilità ai prodotti da parte della clientela.

Un canale di distribuzione può essere:

1. Breve: il prodotto va dal produttore al consumatore senza nessun intermediario, esso è detto anche "vendita diretta";
2. Medio: la merce va dal produttore al cliente tramite un intermediario detto dettagliante. Tale canale è detto "vendita indiretta";
3. Lungo: la merce va dal produttore al cliente tramite il grossista ed il dettagliante, attraverso, cioè, due o più intermediari.

Il canale scelto per la vendita del nostro prodotto è di tipo medio. Infatti, essendo i nostri clienti amanti della tecnologia, abbiamo previsto di realizzare una campagna marketing che prevedesse soprattutto la vendita tramite vendite online, accompagnate, se possibile anche da promozioni.

2.2.5 Relazione con i clienti

Al giorno d'oggi è necessario comunicare determinate informazioni riguardanti il prodotto e la sua organizzazione. È, infatti, importante suscitare curiosità e fiducia nei consumatori, i quali senza alcuna informazione, non acquisterebbero prodotti sconosciuti o di dubbia provenienza. Inoltre, le relazioni, per essere efficaci con i clienti devono raggiungere direttamente il target di riferimento ed implicano anche costi inferiori rispetto a quelli della pubblicità. Per questo possono rivelarsi uno strumento di comunicazione particolarmente efficace e un valido investimento con il fine di stabilire con il proprio interlocutore proficue e durature relazioni. Abbiamo pensato di suscitare la curiosità del cliente attraverso tre tipologie di relazioni:

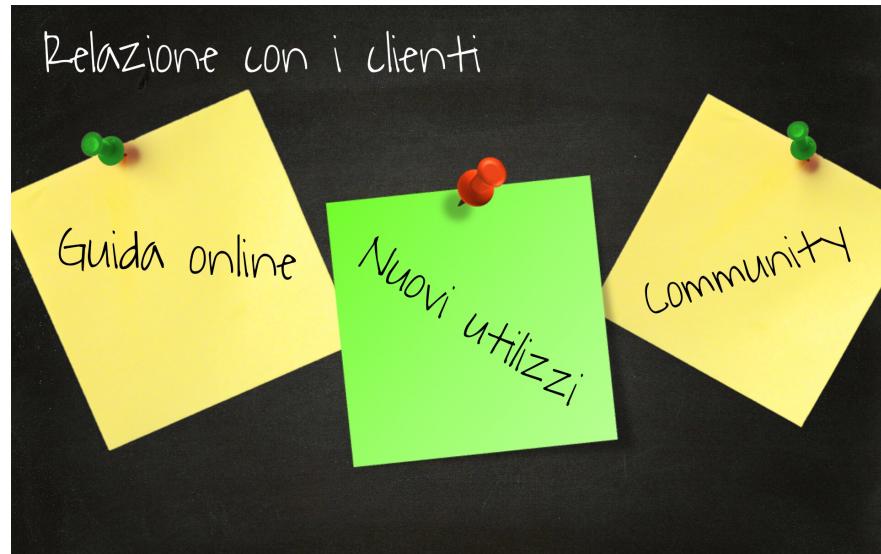


Figura 2.5: Relazioni con i clienti

1. Guida online: all'interno del nostro futuro sito pensavamo di introdurre una guida dettagliata che possa essere esaustiva per tutti i target di clientela da noi selezionati;
2. Community: un forum dove la gente possa discutere dei problemi o di soluzioni innovative per le impostazioni del prodotto;
3. Nuovi metodi di utilizzo: creare un settore apposito all'interno del sito dove i clienti possono vedere le impostazioni del prodotto piú divertenti e inusuali, in modo tale da creare un effetto "WOW".

2.2.6 Attività

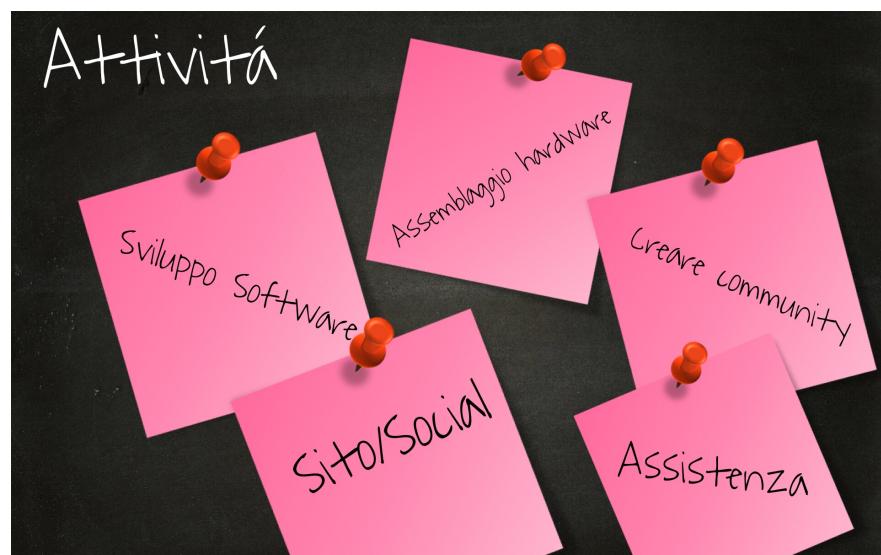


Figura 2.6: Attività

Diventa sempre piú importante pianificare tutte le attività da porre in essere, per una maggiore efficacia ed efficienza. Infatti, con un’adeguata pianificazione di tutto ciò che deve essere eseguito, dalla creazione, produzione alla pubblicizzazione e commercializzazione del prodotto sarà possibile tenere sotto controllo i costi, evitando una dispersione di energie. Per la creazione di Proximity System dovremo prima di tutto sviluppare il software e assemblare l’hardware. Inoltre, dato che la vendita avviene per lo piú tramite siti web, dovremmo realizzare il sito, una pagina Facebook e creare una community. Infatti Internet deve diventare parte integrante della politica generale di marketing, non essere una semplice opzione. L’attivazione di un sito internet affianca alle analisi di mercato offline ed alle strategie di promozione/ vendita tradizionali una relazione con il pubblico del web. La progettazione di un sito web é un’operazione piuttosto complessa che si articola nelle seguenti fasi:

1. raccolta e definizione delle informazioni e dei contenuti da pubblicare;
2. classificazione ed organizzazione delle informazioni all’interno del sito;
3. definizione del progetto grafico;
4. scelta ed implementazione delle tecnologie.

Per l’assistenza in un primo momento sarà svolta dai membri del team per poi essere ampliata, qualora fosse necessario, a persone esterne.

2.2.7 Risorse chiave

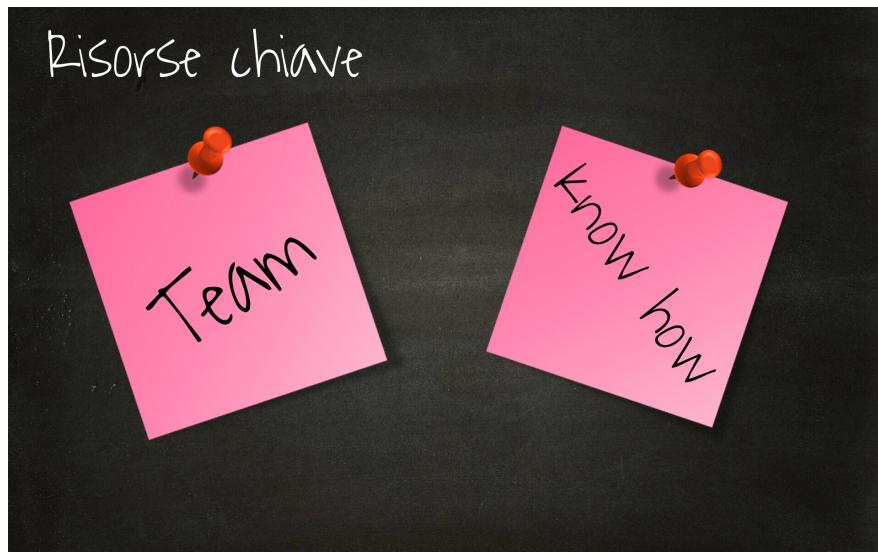


Figura 2.7: Risorse chiave

Diventa indispensabile reperire le risorse necessarie per poter produrre e commercializzare il prodotto e i relativi servizi ad esso connesso. Tali risorse consistono in risorse materiali (materie prime, beni strumentali, immobilizzazioni ecc.), risorse fisiche ed intellettuali (personale dipendente, rapporti di collaborazione ecc.), risorse finanziarie (finanziamenti effettuati tramite capitale proprio o tramite operatori finanziari esterni all’azienda, quali banche). Diventa importante per ogni azienda cercare di reperire determinate risorse a costi competitivi e ciò lo si può ottenere attraverso accordi con i

fornitori, attraverso la scelta di determinate forme di finanziamento rispetto ad altre e forme contrattuali che possono apportare maggiori vantaggi.

Le risorse chiave di cui necessitá "Proximity System" sono le risorse fisiche ed intellettuali. La risorsa fisica é il team, composto sempre da me e il mio collega Saverio. Le risorse intellettuali, invece, si basano sulle nostre conoscenze di informatica. In questa prima fase sar importante informarsi su eventuali finanziamenti europei per queste tipologie di attivit o agevolazioni previste per l'imprenditoria giovanile.

2.2.8 Partner



Figura 2.8: Partner

Importante é cercare dei partner con cui poter lavorare, in modo tale da creare rapporti duraturi e alleanze con fornitori o utenti diversi, cos da poter ridurre i costi per essere pi competitivi sul mercato. Con questa fase vengono definiti rapporti di collaborazione tra attori diversi (fornitori, clienti, enti, ecc) sotto forma di accordo di investimento. Con tali accordi si cerca di ottenere benefici nell'approccio al proprio mercato di riferimento utilizzando iniziative di marketing di tipo diverso; pu capitare che una iniziativa di marketing si concentri maggiormente sulla negoziazione, un'altra si differenzi per il tipo di comunicazione, un'altra ancora per la sua regolamentazione giuridica. Si possono inoltre distinguere partnership verticali e partnership orizzontali. Nel primo caso si tratta di accordi mirati all'integrazione delle imprese costituenti i vari anelli della catena del valore: dal produttore al distributore al consumatore e viceversa. Nel caso delle partnership di tipo orizzontale si tratta di accordi che si attuano tra attori dello stesso livello della catena del valore. Pu trattarsi quindi di accordi tra imprese di settori diversi, ma tra le quali si possono stabilire sinergie basate sulle competenze e sui segmenti di clientela raggiunti. Allo stesso tempo pu trattarsi di accordi tra imprese dello stesso settore che decidono di unire la forza commerciale ed affrontare il mercato con una strategia comune, almeno per alcuni comparti.

In questa prima fase di progettazione, abbiamo previsto tre nostri partner:

1. Fornitore della scocca: ovvero quel produttore che realizzi l'involucro che conterr il nostro prodotto;

2. Hardware: i fornitori che ci forniscono le componenti necessarie, affinché Proximity System sia realizzato;
3. Distributori di componenti elettriche: utili agli installatori per trovare il nostro prodotto;

2.2.9 Ricavi



Figura 2.9: Ricavi

Per la clientela il prezzo del prodotto (generatore di ricavi per l'imprenditore) é uno dei fattori guida nelle scelte di acquisto ed é un elemento di confronto fra i prodotti delle diverse aziende. É, dunque, una componente importantissima ed ha un ruolo preminente e determinante per il condizionamento che é in grado di esercitare. Per fissare il prezzo d'impresa lo si deve adattare al proprio prodotto a seconda che sia di massa, di lusso, business ecc.. Nella fissazione del prezzo si deve tener conto dei costi di produzione e commercializzazione del prodotto, della pressione della concorrenza, della quota di mercato che l'impresa vuole raggiungere. É necessario, inoltre, individuare gli obiettivi di prezzo, cioé stabilire che cosa l'impresa intende ottenere con le politiche di prezzo. Gli obiettivi possono essere: sopravvivenza, profitto, rendimento del capitale investito, quota di mercato, qualitá del prodotto. Importante sará anche fissare differenti prezzi. Si tratta di creare una gamma di offerte intorno al prezzo base, in modo tale da invogliare continuamente il cliente all'acquisto. Si fissano i differenziali di prezzo configurando sconti , riduzioni, gratuitá in rapporto alla tipologia della clientela ed alle modalitá di pagamento: in contante, rateizzato ecc. Avendo scelto, comunque, un mercato di penetrazione, definiremo un prezzo piú basso rispetto alla concorrenza per ampliare la nostra quota di mercato. Ció é reso possibile anche dai bassi costi di produzione. Altri ricavi deriveranno dagli abbonamenti; infatti sia il servizio cloud, sia l'assistenza saranno date in abbonamenti annuali.

2.2.10 Costi

Cercare di controllare i costi diventa una strategia importante per ogni azienda, dato che per essere competitivi bisogna vendere il prodotto ad un prezzo concorrenziale. I



Figura 2.10: Costi

costi influenzano drasticamente il prezzo di un prodotto, il quale condiziona le scelte dei clienti.

Distinguiamo tre gruppi fondamentali di costi: costi fissi, costi variabili e semi variabili.

1. I costi fissi non variano al variare della quantità prodotta. Un esempio di costo fisso potrebbe essere il fitto di un locale che l'imprenditore dovrebbe pagare qualora il locale dove viene svolta l'attività non è di sua proprietà.
2. I costi variabili sono tutti quei costi che variano al variare della produzione. I costi relativi alla materia prima sono costi variabili. Infatti essi aumentano all'aumentare della produzione.
3. I costi semi variabili sono formati da una componente di costi fissi e una di costi variabili. I costi riguardanti il consumo di energia elettrica o utenze telefoniche sono composte da una quota di costi fissi ed una quota di costi variabile che varia in base al consumo.

Per il nostro prodotto abbiamo previsto i seguenti costi:

1. Hardware: tutte le componenti elettroniche sono fornite dall'esterno, per cui avranno un costo variabile;
2. Stipendi direzionali: rappresenta il compenso che l'imprenditore dovrebbe percepire per la sua attività lavorativa nell'impresa;
3. Infrastruttura cloud: costo dato per il mantenimento delle funzioni del cloud;
4. Percentuale distributori: costo sostenuto dall'azienda per pagare coloro che distribuiranno il nostro prodotto sul mercato.

3. Frontend

3.1 Scelta Tecnologica

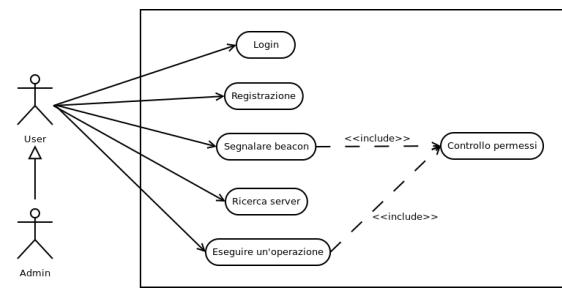


Figura 3.1: Use Case della nostra Applicazione

3.1.1 Javascript

Abbiamo scelto di utilizzare per prima cosa il linguaggio di programmazione javascript per questi motivi:

1. La sua Diffusione: naturalmente javascript é uno dei linguaggi piú popolari e diffusi (se non il piú diffuso), ciò fa sì che dietro di esso ci sia un continuo interesse da parte di tutta la community mondiale di informatica;
2. È Rapido: esso infatti é uno dei linguaggi piú veloci e performanti, sia nella sua versione server (nodeJS), sia nella sua versione client (un esempio può essere jQuery).
3. Adattabile e Multipiattaforma: javascript é infatti perfettamente integrato con HTML5, inoltre é semplice la portabilitá di una web app in javascript su tutti i dispositivi mobili.
4. Full Stack Development: la possibilitá di utilizzare lo stesso linguaggio di programmazione per ogni parte applicativa che puó andare dal client e dal server fino al database. Questo é il vero punto di forza del Javascript.

3.1.2 Scelta del framework: AngularJS

Possiamo considerare AngularJS come un framework all-or-nothing, ovvero non si puó usare solo una parte di Angular, come per esempio il data Binding, senza usare tutto il resto che questo framework concede, come le dependance injecton o le direttive. Le scelte che ci hanno fatto propendere per esso rispetto ad altri possono essere riassunti nei seguenti punti:

1. azienda E-xtrategy: AngularJS è e continua a essere uno dei punti di forza per i programmatore che lavorano all'interno dell'azienda E-xtrategy. Essi si sono rivelati dei punti fermi ai quali abbiamo potuto chiedere spiegazioni e dubbi nati durante la creazione della nostra Applicazione.
2. Popolaritá e Mercato: ad oggi é sicuramente uno dei framework piú richiesti dal mercato, anche grazie alla sua enorme community.
3. Soluzione piú completa: Angular, a discapito di una complessitá maggiore di apprendimento e uso, laddove altri framework (come per esempio React) si presentano già accessibili e focalizzati, ha la possibilità di ricorrere a numerose librerie note per le parti che non sono strettamente legate all'interfaccia utente.

3.2 AngularJS

AngularJS é un framework appartenente al linguaggio di programmazione Javascript sviluppato da Google. Esso ha lo scopo di facilitare lo sviluppo di applicazioni Web client side. Un framework si puó definire come una struttura che ha il compito di facilitare lo sviluppo, sia in termini di velocitá sia in termini di ordine e mantenibilitá del codice. Utilizzando JavaScript puro, esso é perfetto per realizzare applicazioni web mono pagine, ovvero che non richiedono la necessitá di ricaricare tutta la pagina, ma le risorse di quest'ultima vengono ricaricate dinamicamente a seconda della richiesta.

3.3 MVC

Il framework si ispira al pattern MVC, ovvero un paradigma architettonale utilizzato da molteplici linguaggi di programmazione, con lo scopo sempre di facilitare la struttura del codice. Analizziamo brevemente e singolarmente i termini del MVC che analizzeremo in maniera piú approfondita piú tardi:

1. Model: rappresenta la struttura dati che sta dietro a una parte dell'applicazione. Rappresenta lo stato dell'applicazione.
2. View: é la componente grafica che viene implementata all'interno dei file HTML. Qui troviamo l'interazione con l'utente.
3. Controller: ha lo scopo di controllare le interazioni dell'utente e gestire l'aggiornamento dei dati. In sintesi dovrá contenere l'intelligenza della nostra web-app.

3.4 Funzionalitá principali di Angular

A differenza di altri framework famosi, Angular oltre ad avere funzionalitá utili agli sviluppatori, ha anche funzioni che permettono di agevolare i designer. Qui di seguito elenchiamo le principali e le piú interessanti.

3.4.1 Uso dei moduli

In un'applicazione che ha un certo livello di complessitá, deve esserci un meccanismo che permette di raggruppare funzionalitá a seconda dei criteri dettati dall'architettura dell'applicazione stessa. I moduli favoriscono la separazione dei compiti definendo

un’interfaccia pubblica e limitando la visibilità del funzionamento interno. Essi tuttavia non si trovano a livello nativo all’interno delle versioni Javascript attualmente diffuse. Angular copperisce a questo problema con un proprio sistema che consente un’ampia flessibilità nell’organizzazione del codice. Abbiamo visto come AngularJS preveda diversi componenti ciascuno con un suo specifico ruolo: controller, servizi, filtri, ecc. In Angular un modulo è un contenitore di componenti, indipendente dalla loro natura e dalla loro collocazione fisica. Per creare un modulo basta utilizzare il metodo module(), come possiamo vedere nel seguente codice:

```
1 angular.module("exampleModule", ["module1", "module2"]);
```

Codice 3.1: metodo module()

Come abbiamo già detto, un modulo è un insieme di componenti, vediamo qui di seguito come aggiungere un servizio o un controller all’interno del modulo:

```
1 var exampleModule = angular.module("exampleModule");
2 exampleModule.controller("myController", function() . . .);
3 exampleModule.factory("myService", function() . . .);
```

Codice 3.2: servizio e controller in modulo

3.4.2 Direttive

Le direttive sono una funzionalità unica, predominante e disponibile solo in AngularJS. Esse hanno la capacità di nascondere le complessità della struttura DOM, creando componenti HTML che possono essere sia personalizzati a seconda delle proprie preferenze, sia riusabili. Infatti una direttiva permette di espandere la sintassi che troviamo all’interno del codice HTML, facendone creare nuove e successivamente incorporarle nelle funzionalità della pagina. Tutte le direttive hanno il prefisso ”ng-” e funzionano come elementi standalone. Qui di seguito vengono riportate le direttive più usate:

1. ng-app: probabilmente la direttiva più importante. Dichiara un elemento root dell’applicazione e viene posizionato all’interno dei tag <body> o <html>. Per dichiarare una pagina come applicazione angular è sufficiente scrivere <html ng-app>.
2. ng-controller: permette di definire una classe controller Javascript per risolvere le espressioni che troviamo all’interno dell’HTML ng-repeat: questa direttiva è utile per ciclare delle variabili in una collection e le istanzia in un template per gli item.

```
1 <body class="background" layout="column" md-theme="{{theme.color}}"
2   md-theme-watch ng-controller="NavigationController">
3   <section layout="row" flex class="height">
4     <md-sidenav ng-if="user.username !== ''"
5       class="height md-whiteframe-z2 md-sidenav-left"
6       md-component-id="left" md-is-locked-open="$mdMedia('gt-sm')">
7       <md-toolbar class="md-whiteframe-z2 md-hue-2" md-scroll-shrink>
8         <center><p class="white">Proximity <strong>System</strong></p></center>
9       </md-toolbar>
10      <md-content class="height">
11        <md-toolbar class="image" layout="column" layout-align="start center">
12          <span flex></span>
13          <center>
14            <p class="white">{{user.firstname}}</p>
15          </center>
16        </md-toolbar>
17        <md-toolbar class="height md-hue-2">
```

```

18     <md-list class="contact_list">
19         <md-list-item ng-repeat="nav1_in_nav">
20             <md-button class="contactButton_no-hover_larghezza"
21                 href="{{nav1.link}}" ng-class="md-raised"
22                 ng-click="toggleSidenav('left')">
23                 <md-icon class="position_one" md-svg-icon="{{nav1.icon}}>
24             </md-icon>
25             <p class="position_white_contact_list">{{nav1.name}}</p>
26             </md-list-item>
27         </md-list>
28     </md-toolbar>
29     </md-content>
30 </md-sidenav>
31 <md-content class="background" layout="column" flex ng-view>
32 </md-content>
33 </section>
34 </body>
35

```

Codice 3.3: unicam-beacon-server/webserver/angular/index.html

3. ng-class: è utilizzata per caricare in maniera dinamica gli attributi delle varie classi.
4. ng-hide e ng-show: direttive che permettono di nascondere o mostrare un determinato elemento

```

1 <md-fab-speed-dial hide-xs hide-sm hide-gt-md ng-hide="demo.isHidden"
2   md-direction="down" md-open="demo.isOpen"
3   class="md-scale_md-fab-bottom-right_move-up1"
4   ng-class="{'md-hover-full': demo.hover}" ng-mouseenter="demo.isOpen=true"
5   ng-mouseleave="demo.isOpen=false">
6     <md-fab-trigger>
7       <md-button aria-label="menu" class="md-fab_md-accent">
8         <md-icon style="fill:_white" md-svg-icon="img/palette.svg"></md-icon>
9       </md-button>
10    </md-fab-trigger>
11    <md-fab-actions>
12      <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme1')">
13        <md-icon style="fill:_blue" md-svg-icon="img/paint.svg"></md-icon>
14      </md-button>
15      <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme')">
16        <md-icon style="fill:_red" md-svg-icon="img/paint.svg"></md-icon>
17      </md-button>
18      <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme2')">
19        <md-icon style="fill:_green" md-svg-icon="img/paint.svg"></md-icon>
20      </md-button>
21    </md-fab-actions>
22 </md-fab-speed-dial>

```

Codice 3.4: unicam-beacon-server/webserver/angular/templates/home.html

5. ng-if: a seconda del valore true o false, è usata per inserire o eliminare un elemento all'interno del DOM

```

1 <md-input-container data-ng-if="device.io=='output'>
2   <label>Permessi</label>
3   <md-select data-ng-model="device.permission"
4             data-ng-change="saveDevice(device)">
5     <md-option data-ng-repeat="n in [] | range:11">{{n}}</md-option>
6   </md-select>
7 </md-input-container>
8 <md-input-container data-ng-if="device.io=='output'>
9   <label>iBeacon</label>
10  <md-select data-ng-model="device._Beacon"
11            data-ng-change="saveDevice(device)">
12    <md-option data-ng-value="ibeacon._id">

```

```

13      data-ng-repeat="ibeacon_in_devices/BeaconFilter">{{ibeacon.name}}
14      </md-option>
15  </md-select>
16  </md-input-container>
17  <md-input-container data-ng-if="device.io=='input'>
18      <label>Comanda</label>
19      <md-select data-ng-model="device._Output"
20          data-ng-change="saveDevice(device)">
21          <md-option data-ng-value="output._id"
22              data-ng-repeat="output_in_devices/DeviceLampadeFilter">{{output.name}}
23          </md-option>
24      </md-select>
25  </md-input-container>
```

Codice 3.5: unicam-beacon-server/webserver/angular/templates/actions.html

3.4.3 Data Binding bidirezionale

Direttive come ng-model e ng-bind costituiscono la seconda funzionalità più importante di AngularJS. Esse infatti permettono di risparmiare una buona parte di codice inutile che si dovrebbe scrivere sul lato server per la gestione dei template. Il data binding, ovvero "l'associazione dati", è appunto il processo di connessione tra gli oggetti e gli elementi del DOM. Di solito il data binding avviene spesso in una sola direzione; infatti se consideriamo il sistema classico di template, ogni alterazione che l'utente compie all'interno della view non sarà riportata all'interno del model. Ciò succede perché il sistema unisce il template e il model nella view. Per ovviare a questo problema, ovvero per far rimanere in sincronizzazione questi elementi, lo sviluppatore dovrebbe manipolare manualmente gli attributi e gli elementi nel DOM. Il template AngularJS funziona in maniera migliore e dinamica. Infatti fa risparmiare allo sviluppatore l'onere di inserire dati nella view manualmente, sincronizzando in maniera automatica i dati tra il model e il view e viceversa. Esempio di Data Binding:

```

1 <body>
2 il prodotto di
3 <input type="number" ng-model="number1">
4 e
5 <input type="number" ng-model="number2">
6 e'
7 {{number1*number2}}
8 </body>
```

Codice 3.6: Data binding

il prodotto di e è

Figura 3.2: Data binding

3.4.4 Controller

Uno dei componenti principali di Angular è sicuramente il controller. All'interno del pattern MVC, come abbiamo visto in precedenza, il controller assolve il ruolo di creare un collegamento tra il model, ovvero la fonte di dati, e la view, cioè l'interfaccia grafica. Ciò accade esattamente anche all'interno di Angular. Per creare un'applicazione che integri un controller quindi sarà necessario:

1. valorizzare l'attributo ng-app

2. aggiungere alla root della nostra applicazione la direttiva ng-controller

```

1 <div ng-app="exampleApp" ng-controller="myCtrl">
2 Nome: <input type="text" ng-model="nome"><br/>
3 Cognome: <input type="text" ng-model="cognome"><br/>
4 <br/>
5 Il tuo nome: {{ nome + " " + cognome}}
6 </div>
7 <script>
8 var app = angular.module('myApp', []);
9 app.controller('myCtrl', function($scope) {
10   $scope.nome = "Giuseppe";
11   $scope.cognome = "Rossi";
12 });
13 </script>
```

Codice 3.7: Controller

Nella prima riga all'interno dello script definiamo una nuova applicazione metodo module() visto in precedenza. Successivamente abbiamo definito il nostro controller che funge da estensione per la nostra applicazione. Abbiamo usato il metodo controller() sulla variabile "app" per poter creare il nostro controller. Dentro, infatti abbiamo passato lo scope, ovvero il "contesto", della nostra applicazione. Infine abbiamo dichiarato il valore iniziale delle nostre variabili, in questo caso Giuseppe Rossi. L'utilità del controller, come possiamo vedere, è quello di permetterci di gestire i dati associati allo scope mediante proprietà. Uno degli errori più comuni delle persone che si affacciano ad Angular per la prima volta è il sovraccaricare i controller con funzionalità che non gli appartengono. Infatti il Controller ha come suo scopo principale quello di impostare lo stato iniziale del modello dei dati e la definizione di funzionalità per la view. Gli errori più comuni che un Controller non deve mai fare sono:

1. formattare l'input (compito che spetta ai form);
2. condividere dati con altri controller (compito dei Service);
3. manipolare il DOM (compito della Directive);

```

1 angular.module('beaconApp.controllers.login', [])
2
3 .controller('LoginCtrl', function($scope, $location, $http, Theme, Login, $mdDialog) {
4   $scope.user = Login.user;
5   $scope.theme = Theme;
6
7   $scope.login = function(username, password) {
8     Login.login(username, password).then(function(res) {
9       $scope.autenticato = true;
10      $scope.theme.color = $scope.user.theme;
11    },
12    function(res) {
13      alert = $mdDialog.alert()
14        .title('Attenzione')
15        .content('Username o Password errato/i!')
16        .ok('Chiudi');
17      $mdDialog
18        .show( alert )
19        .finally(function() {
20          alert = undefined;
21        });
22    });
23  };
24  $scope.errore = function($scope) {
25    $scope.project = {
26      rate: 500
27    };
28  };
29});
```

```

27     };
28   };
29   $scope.logout = function() {
30     Login.logout();
31   }
32 })

```

Codice 3.8: webserver/angular/js/controllers/login.js

3.4.5 Filtri

Un altro componente abbastanza importante all'interno di AngularJS è sicuramente il filter. Il filter ha lo scopo di formattare o applicare un'elaborazione al risultato di un'espressione. L'applicazione del filter è abbastanza semplice, infatti è basato sull'uso dell'operatore pipe (-). Esistono anche filtri "preimpostati", i più utilizzati sono:

1. orderBy: ordina gli elementi di un array
2. lowercase/uppercase: trasformano rispettivamente la stringa in caratteri minuscoli e maiuscoli
3. currency: formatta un numero come valuta
4. date: serve a formattare il numero come data
5. limitTo: estrae i primi N elementi di un insieme, di una stringa o un array.
6. filter: estrae gli elementi di un array che devono soddisfare un determinato criterio

```

1 angular.module('beaconApp.filters.gpio', [])
2 .filter('GPIOFilter', function() {
3   return function(input, type, id) {
4     var out = [];
5     for (var i = 0; i < input.length; i++) {
6       if(input[i].type === type && (input[i]._Device === null || input[i]._Device === id ))
7         out.push(input[i]);
8     }
9     return out;
10   };
11 });
12 angular.module('beaconApp.filters.io', [])
13 .filter('IOFilter', function() {
14   return function(input, type) {
15     var out = [];
16     for (var i = 0; i < input.length; i++) {
17       if(input[i].type === type)
18         out.push(input[i]);
19     }
20   };
21 });
22

```

Codice 3.9: Filtri GPIO e IO

3.4.6 Servizi

I servizi hanno come compito quello di offrire delle funzionalità che non dipendono dalla User Interface. Il loro scopo è quello di implementare le funzionalità che si occupano di elaborare o recuperare i dati da visualizzare sulle view tramite i controller, ovvero

la parte logica dell'applicazione. Un altro compito importante è la condivisione delle varie funzionalità accessibili dalle altre componenti. Un esempio per capire meglio a cosa serve il servizio è il metodo per calcolare il codice fiscale, che può essere utilizzato da più componenti della nostra applicazione; il modo migliore di gestire questa esigenza è la sua implementazione come servizio.

```
1 angular.module('beaconApp.services.devices', [])
2
3 .factory('Devices', function($http) {
4   return {
5     getAll: function() {
6       return $http.get('/api/v2.0/device');
7     },
8     getIos: function() {
9       return $http.get('/api/v2.0/device/ios');
10    },
11    add: function(device) {
12      return $http.post('/api/v2.0/device', device);
13    },
14    deleteDevice: function(device) {
15      return $http.delete('/api/v2.0/device/' + device._id);
16    },
17    getDevice: function(device) {
18      return $http.get('/api/v2.0/device/' + device._id);
19    },
20    editDevice: function(device) {
21      return $http.put('/api/v2.0/device/' + device._id, device);
22    },
23    setBeaconDevice: function(device, beacon){
24      return $http.post('/api/v2.0/device/ibeacon',
25        {id_device: device._id, id_beacon:beacon._id} )
26    },
27    getOutputDevice: function() {
28      return $http.get('/api/v2.0/device/output');
29    }
30  };
31 });
32});
```

Codice 3.10: webserver/angular/js/services/devices.js

3.4.7 Dependency injection

Angular è un framework che dispone di un sistema che si occupa sia di creare componenti, sia di caricare dipendenze e passarle ad altri componenti. Tutto ciò è importante, in quanto porta servizi che comunemente sono sul server al lato client. Ciò comporta una netta diminuzione del lavoro sul backend, con un conseguente aumento delle prestazioni. Infatti la dependency injection consente di combinare insieme componenti allo scopo di strutturare un'applicazione. Se all'interno di un componente Angular abbiamo bisogno delle funzionalità offerte da un altro componente non dobbiamo fare altro che dichiararne la dipendenza.

3.4.8 Tipologie delle applicazioni mobili

In ambito mobile esistono tre tipi di tipologie di applicazioni:

1. App native: sono applicazioni scritte e compilate per una specifica piattaforma, utilizzando i linguaggi di programmazione e librerie supportati dal sistema operativo mobile. Un esempio è il linguaggio di programmazione Java per Android.

Le applicazioni native hanno il piú grande vantaggio nello sfruttare a pieno le caratteristiche e le funzionalità del dispositivo integrate con l’interfaccia del sistema operativo. Tuttavia se si vuole programmare un’app multipiattaforma, bisognerà sviluppare applicazioni diverse con linguaggi diversi

2. Applicazioni web: sono pagine web che sono state ottimizzate per dispositivi mobili, utilizzando linguaggi come HTML5, Javascript e CSS3. A differenza delle app native, condividono lo stesso codice anche tra piattaforme diverse, tuttavia non potranno essere usate se il dispositivo è offline e non avranno a disposizione tutte le funzionalità del dispositivo.
3. App Ibride: sono applicazioni che cercano di sfruttare al meglio le due categorie. Infatti sono scritte con tecnologie Web ma vengono eseguite localmente all’interno di un’applicazione nativa. Esse come le applicazioni web, permettono di usare lo stesso codice per ogni piattaforma, ma il loro aspetto grafico, generalmente, non sarà integrato come quello della piattaforma.

3.4.9 Ionic

Per semplicitá abbiamo deciso dunque di creare come applicazione per Proximity System una app ibrida, per far ciò abbiamo utilizzato Ionic.

Ionic è un framework che mette insieme Apache Cordova e AngularJs per la creazione di app ibride. Inoltre consente di sfruttare al meglio le tecnologie Web per creare applicazioni simili a quelle native. In questo ambito è una delle soluzioni di miglior successo. Ciò che l’ha portata a questo successo è sicuramente il suo orientamento verso il design e la user interaction dell’app, prediligendo le funzionalità di frontend a quelle di backend. Il framework è stato realizzato dal team Drifty. Gli stessi creatori affermano che Ionic può essere considerato ”il Bootstrap per le applicazioni ibride”, ovvero essendo quest’ultimo un punto di riferimento per la creazioni di siti e applicazioni Web responsive, Ionic cerca di esserlo per quanto riguarda la creazioni delle applicazioni mobili ibride. Ionic, per funzionare al meglio, richiede l’utilizzo di AngularJS, di cui ne abbiamo parlato ampiamente nel capitolo precedente. Ciò fa sì che si abbia a disposizione tutta la potenza di AngularJS per creare interazioni fluide, animazioni e tutto ciò che occorre ad un’app per risultare accattivante. Phonegap e Cordova sono strumento di sviluppo open source, il primo creato dallo Sviluppatore Nitobi, per fare da ponte tra le applicazioni Web e i dispositivi mobili. Grazie a esse possiamo sviluppare applicazioni utilizzando HTML, Javascript e CSS, ma utilizzare anche le principale risorse del dispositivo, quali la fotocamera, il gps, il multitouch ecc.. Attraverso Ionic Framework si ha a disposizione tutti i vantaggi del framework Phonegap e la potenza di Angular. In poche parole si abbattono quasi tutti i costi ed i tempi per la progettazione di app native e in piu’ si hanno a disposizione le App piú utilizzate basate sull’HTML5 e sulle integrazioni delle API.

4. Material Design

Abbiamo deciso di utilizzare per la nostra app come comparto grafico il Material Design, perché ci è sembrata una scelta migliore per quanto riguarda una grafica che comprenda un ecosistema. Il Material Design è un design language sviluppato da Google e presentato il 25 giugno 2015, in occasione della conferenza Google I/O. A partire dal 2015, la maggior parte delle applicazioni mobili di Google per Android hanno applicato il nuovo linguaggio di design, tra cui Gmail, Youtube, Google Drive, Google Documenti ecc.. Il lancio del Material Design è avvenuto in concomitanza con la presentazione di Android Lollipop. Essa infatti, è la prima versione del sistema operativo made in Google che supporta questa nuova interfaccia grafica e può considerarsi uno stravolgiamento rispetto a kitkat. Il Material Design è una perfetta sintesi tra l'interfaccia flat di Windows e lo scheumorfismo dei primi iPhone. Esso sintetizza i principi classici del buon design con le innovazioni e le possibilità tecnologiche e della scienza. Google ha anche realizzato per Lollipop più di 5000 API, in modo tale da permettere agli sviluppatori di interagire in modo completo con tutti i prodotti Google.

4.0.1 Scheumorfismo

Abbiamo parlato del Material come sintesi tra lo scheumorfismo di Apple e il flat design di metro Style, ma andiamo a capire le maggiori differenze tra queste tre tipologie di interfacce grafiche. Il termine "scheumorfismo" deriva dalle parole greche (contenitore o attrezzo) e (forma), esso indica un ornamento grafico apposto su un oggetto allo scopo di richiamare le caratteristiche estetiche di un altro oggetto. Un esempio può essere una ceramica ornata con dei rivetti per far in modo che ricordi una pentola di metallo. In ambito digitale invece acquisisce un significato simile, ma leggermente diverso. Infatti lo scopo dell'interfaccia grafica è quella di simulare oggetti del mondo reale. Esso non si ferma solo agli effetti visivi, ma anche a quelli sonori (come simulare il suono della fotocamera quando si scatta una foto). Lo scheumorfismo ha più a che fare con la rappresentazione di oggetti del passato. Negli ultimi anni, grazie anche ai primi iPhone, questo language design si è unito molto a quello del realismo.

I punti di forza dello scheumorfismo sono:

1. Aiuta gli utenti a comprendere meglio lo scopo e l'utilizzo di un'app grazie alla sua grafica che rimanda a quella di un oggetto esistente;
2. si può definire un approccio familiare con l'utente da parte del designer.
3. è un processo creativo molto lento e macchinoso. Creare textures fotorealistiche non è per nulla semplice e richiede molto tempo;
4. è poco adatto al web e soprattutto ai codici di programmazione CSS e HTML5 usati per rendere i siti web responsive.

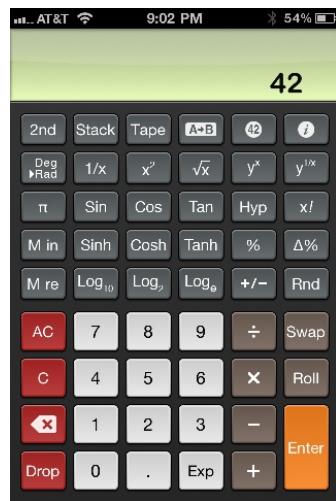


Figura 4.1: Calcolatrice nello Scheumorfismo

4.0.2 Flat Design

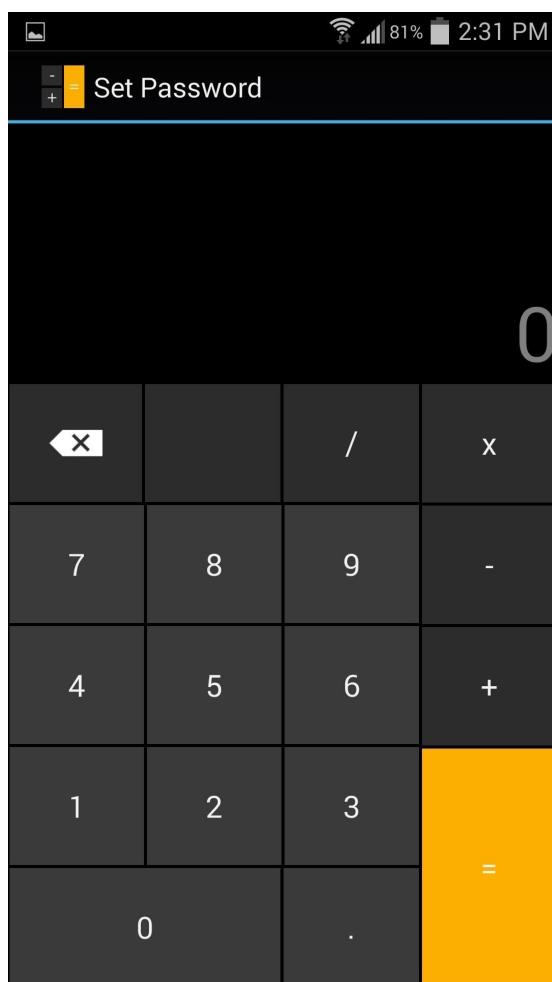


Figura 4.2: Calcolatrice nel Flat Design

Il flat design invece è basato sul minimalismo e la rappresentazione degli oggetti

tramite forme piatte e monocolore, a differenza del material design che nasce come ecosistema, il flat design nasce come stile.

I punti di forza del flat design sono:

1. È adattabile. E intendo estremamente adattabile. Una grafica flat può essere adattata senza problemi ad interfacce grafiche di ogni tipo;
2. È perfetto per il web e per i cellulari, sia perché è adattabile, sia perché la sua semplicità lo rende leggero per il caricamento, ad esempio, di un sito web;

I contro sono:

1. Estrema semplicità e il forte minimalismo possono far risultare questo stile troppo semplice, mancante di personalità o addirittura poco professionale;
2. A volte l'usabilità dell'interfaccia può risentirne, specialmente quando ci sono tante informazioni da dover trasmettere.

4.0.3 Material come sintesi

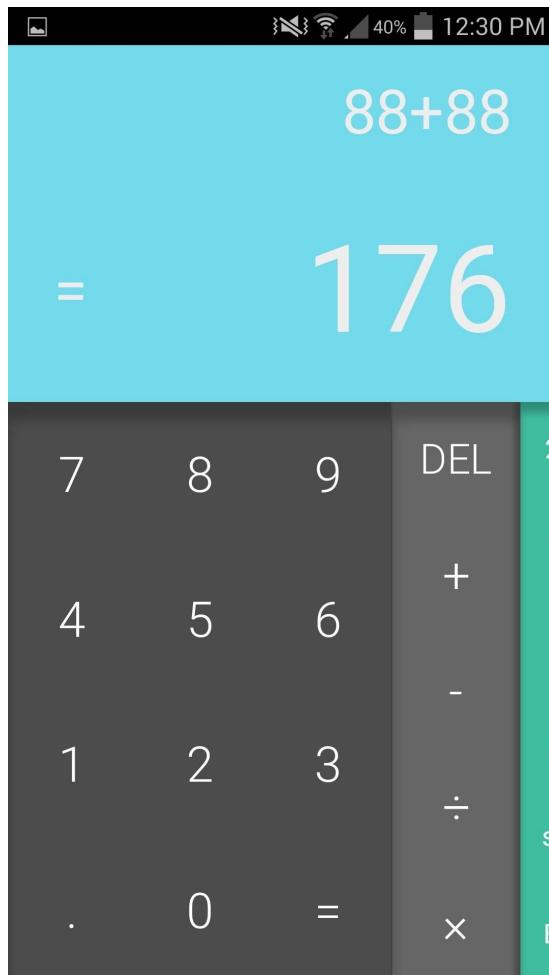


Figura 4.3: Calcolatrice nel Material Design

Il Material Design riesce a essere una sintesi tra loro due perché riesce a portare la piattezza dei componenti con il movimento tra le varie dimensioni.

Matias Duarte, ovvero il designer che ha gestito tutto questo processo creativo, avendo anche influito sulla prima release di android 3.0 "honeycomb", ha dichiarato: "proprio come la carta, il materiale digitale si può espandere o restringere riformando in maniera intelligente. I materiali hanno superfici fisiche bordi. Dettagli come le ombre e le cuciture forniscono il significato di quello che si tocca. Tutto ciò si traduce in un'interfaccia estremamente responsive, ovvero capace di adeguarsi alle varie situazioni, tutto ciò in maniera molto "smart"."

4.1 Caratteristiche Material Design

4.1.1 Comportamento nello "spazio"

Il Material Design è un ambiente tridimensionale che contiene luci, ombre e materia. I componenti e lo stesso ambiente all'interno del material design hanno dei canoni "rigidi" da dover rispettare.

I più importanti sono:

1. Ogni oggetto materiale all'interno dell'ambiente ha un asse x, y, z;
2. L'asse z è perpendicolarmente allineato al display;
3. Ogni elemento occupa un preciso spessore all'interno dell'ambiente
4. L'effetto 3D è dato dalla manipolazione dell'asse Z;

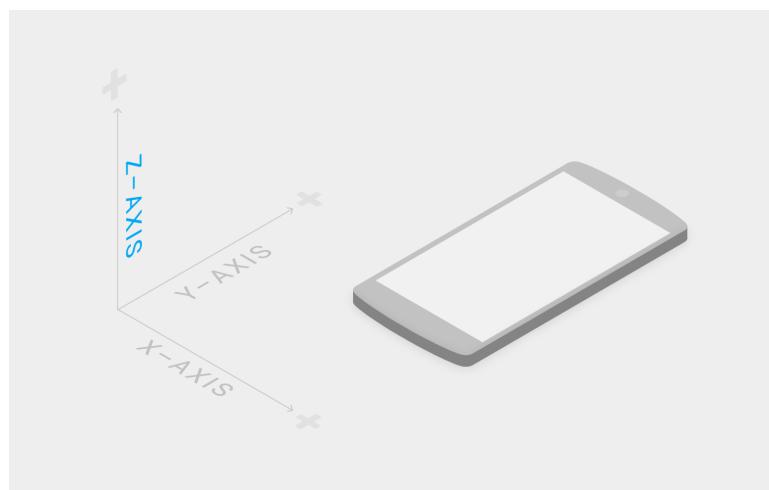


Figura 4.4: Ambiente del Material Design

4.1.2 Uso delle Ombre

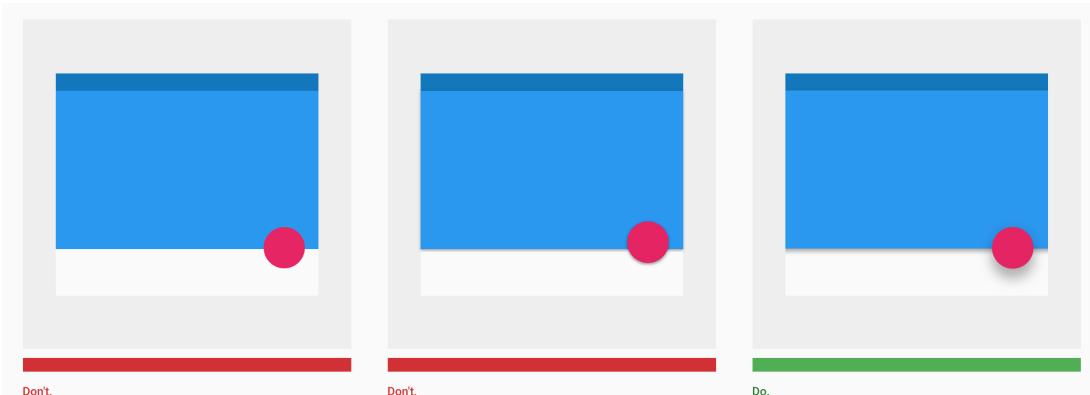


Figura 4.5: Uso delle ombre

Ciò che rende tutto così realistico è l'uso estremamente intelligente delle ombre, ogni elemento è disposto in modo assolutamente gerarchico. Analizzando l'immagine possiamo capire come funziona la struttura che crea questo effetto particolare:

1. prima immagine: notiamo che senza ombre non c' è niente che indichi che il bottone sta su una superficie superiore a quella del background; (non conforme al Material)
2. seconda immagine: le ombre nette fanno capire che il bottone e la toolbar sono separate dallo sfondo, ma rimangono comunque troppo vicine a esso, in quanto non c' è differenza tra loro due; (non conforme al Material)
3. terza immagine: l'ombra più lunga e morbida fa capire che il bottone è in una posizione più elevata rispetto all'appbar.

Tutto ciò è realizzabile grazie al quantum paper, ovvero la superficie tattile: ogni elemento è disposto in maniera gerarchica lungo l'asse z, grazie a questa dipendenza dalla loro elevazione che le ombre risultino naturali, esse comunque possono variare lungo l'asse x e y e hanno sempre spessore unitario (1DP). Il material design è solido, ovvero gli input non possono passare tra i componenti e i materiali. Molteplici elementi inoltre non possono occupare lo stesso spazio simultaneamente.

4.1.3 Animazioni

Google introduce il significato di meaningful nelle animazioni, ovvero le animazioni forzano il focus dell'utente sul contenuto e non sulla transizione, in modo tale che quest'ultimo possa già aspettarsi cosa ci sia nel contenuto successivo. Ogni interazione ora viene introdotta da transizioni grafiche che donano all'esperienza fluidità, armonia e un tocco di realismo che aiuta gli utenti a rapportarsi con gli oggetti virtuali come se fossero reali. Le animazioni possono esistere all'interno di tutti i componenti di un'applicazione, dalle icone alle transizioni o nelle azioni chiave.

Tutti gli elementi lavorano insieme per costruire un'esperienza di continuità e bellezza, rimanendo comunque funzionali. Gli oggetti fisici hanno una propria massa e solo una forza esterna può farli muovere, ciò avviene anche all'interno dell'applicazione, solo l'utente può far partire le animazioni. Inoltre esse non possono iniziare e fermarsi da

sole, tutto ciò per dare un senso di continuità all'intera interfaccia. Ogni animazione deve dare un Feedback visivo, questo effetto viene chiamato "Ripple" e attraverso ciò l'utente rompe lo stato di quiete dell'interfaccia.

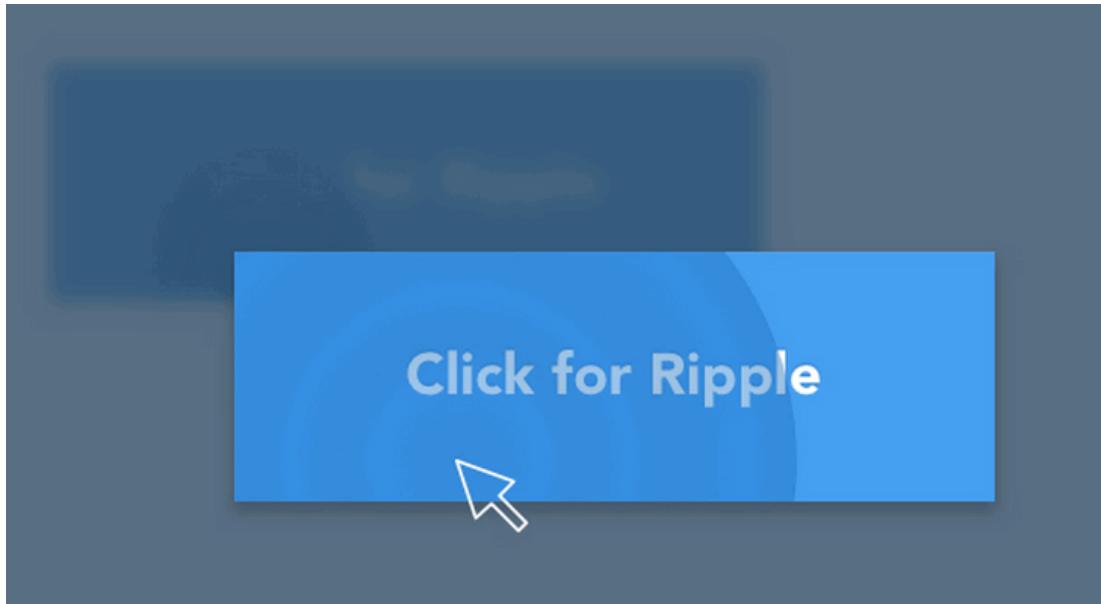


Figura 4.6: Effetto Ripple

4.1.4 Colori, Immagini e Font

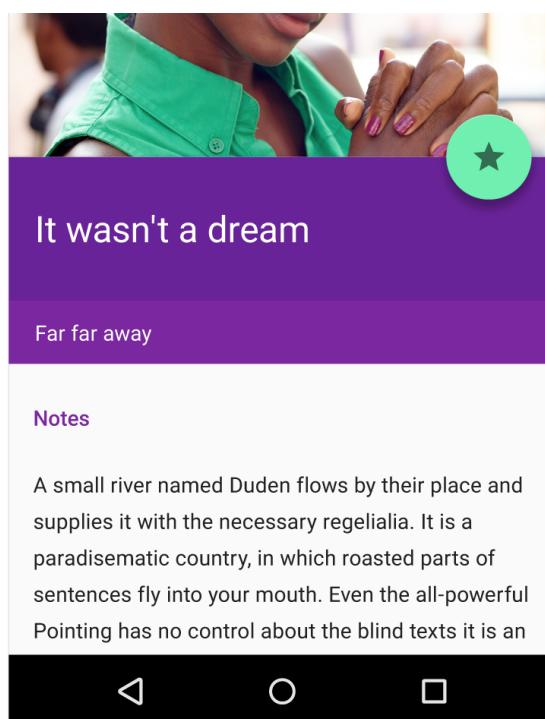


Figura 4.7: Esempio di distribuzione dei colori Primary (viola) e Accent (verde)

I colori nel material design si ispirano a tinte forti contrapposte a tinte piú tenue. Google per aiutare i designer, mette a disposizione già diciannove palette di colore predefinite, ognuna di esse poi ha 15 sfumature diverse di colore. Secondo i canoni del Material Design esistono due tipi di colore:

1. Primary Color: ovvero il colore principale che deve essere usato ampiamente in tutte le interfacce della nostra applicazione.
2. Accent Color: ossia il colore che viene utilizzato all'interno degli elementi interattivi (campi di testo, selezioni, barre di progresso, link e cosí via).

Per quanto riguarda le immagini invece devono essere centrate e appartenenti al contesto, inoltre devono trasmettere piú informazioni possibili, in maniera semplice e immediata. Il font scelto é il Roboto, usato già ai tempi di Ice Cream Sandwich, é leggermente piú ampio e rotondo rispetto al vecchio Nodo usato all'interno di Froyo, ciò comporta una maggiore leggibilitá ed eleganza.

4.2 Componenti principali

4.2.1 Bottom Navigation

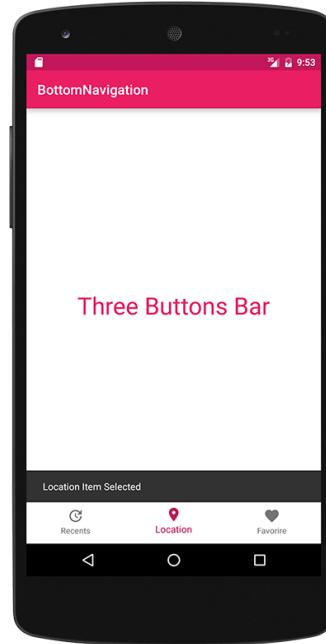


Figura 4.8: Esempio Navigation Bar

La Bottom Navigation é uno dei componenti principali del material design, nonché una delle ultime novità introdotte da Google. La Bottom Bar é principalmente utilizzata all'interno dei dispositivi mobili, infatti per la visualizzazione dell'applicazione all'interno del formato desktop si é soliti usare la sidenav.

La Navigation Bar non é altro che un insieme di pulsanti che, se cliccati, riporta l'applicazione nella pagina indicata. Per la scelta del colore, essa deve utilizzare il colore Primary. Inoltre, la bottom bar non puó contenere piú di cinque icone e non meno di tre icone.

4.2.2 Button

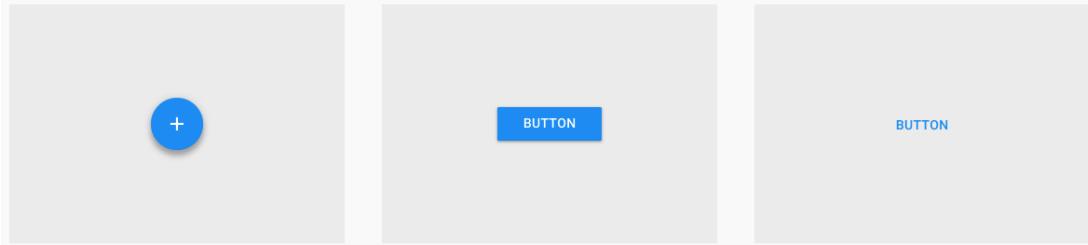


Figura 4.9: I Button

I button devono sempre comunicare l'azione che eseguiranno se cliccati. Per assicurare l'uso di questi pulsanti anche a persone con disabilità, la grandezza minima deve essere di 36dp fino ad arrivare ad un massimo di 48dp.

I button si possono dividere in tre grandi categorie:

1. Floating Action Button: il pulsante principale della nostra pagina, esso di solito esegue l'azione più importante all'interno della nostra applicazione. Lo scopo di questo pulsante, infatti, è quello di guidare l'utente verso un'azione chiara e precisa;
2. Raised Button: questo tipo di bottone aggiunge un livello di elevazione rispetto l'interfaccia, ciò li rende facilmente individuabili in un UI piena di elementi;
3. Flat Button: bottoni che sono a "livello" di interfaccia, solitamente usati per dialog, toolbar e cards;

4.2.3 Card

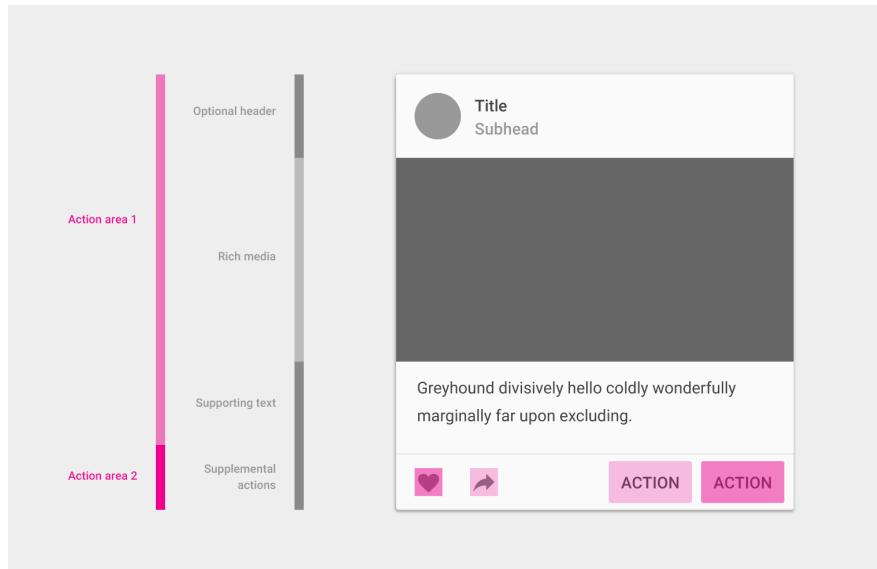


Figura 4.10: Esempio di Card

Le Card sono componenti del Material Design usati per mostrare contenuti composti da elementi differenti. Vengono usate soprattutto per mostrare elementi le cui dimensioni o azioni possono variare. Le Card possono supportare gestures come Swipe e

Pick-up-and-move.

Nello specifico le card si usano:

1. Come una collezione, contenenti più tipi di informazioni, come immagini, film e testi;
2. Supportano più contenuti di lunghezza variabile, come per esempio possono essere i commenti;
3. Possono racchiudere contenuti interattivi, come per esempio può essere un bottone +1;
4. Contengono oggetti che non richiedono una comparazione con contenuti al di fuori della Card;
5. Possono essere utilizzati per racchiudere elementi che si dovrebbero trovare in una list grid, ma non hanno spazio sufficiente per essere mostrati all'interno di essa.

4.2.4 Dialog

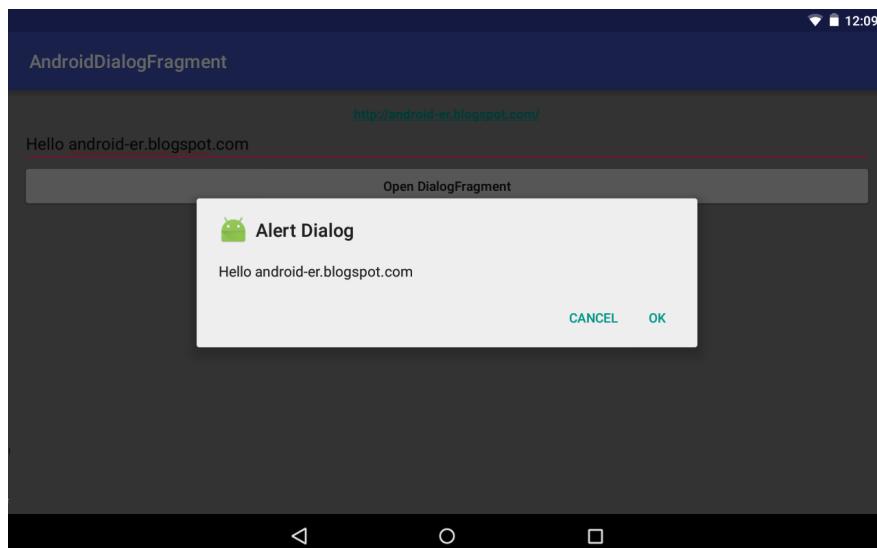


Figura 4.11: Esempio di Dialog

I Dialog contengono il testo e i controlli della UI. Essi centrano il focus dell'utente e hanno una funzione molto interattiva, per questo devono essere usati con parsimonia. I Dialog a schermo intero possono essere usati solo in ambito mobile, e hanno di solito funzioni più complesse, come quelle di richiedere un input di testo. Essi hanno la priorità su tutta l'interfaccia, quindi non devono mai essere oscurati da nessun altro elemento.

Alcuni tipi di Dialog possono essere:

1. Alert: un'interruzione urgente che informa l'utente di un errore o di una richiesta di riconoscimento;
2. Menu: può essere usato anche per una lista di oggetti, dove il dialog provvede a dare più dettagli o azioni;
3. Dialog di Conferma: chiedono esplicitamente la conferma ad un'azione.

4.2.5 List

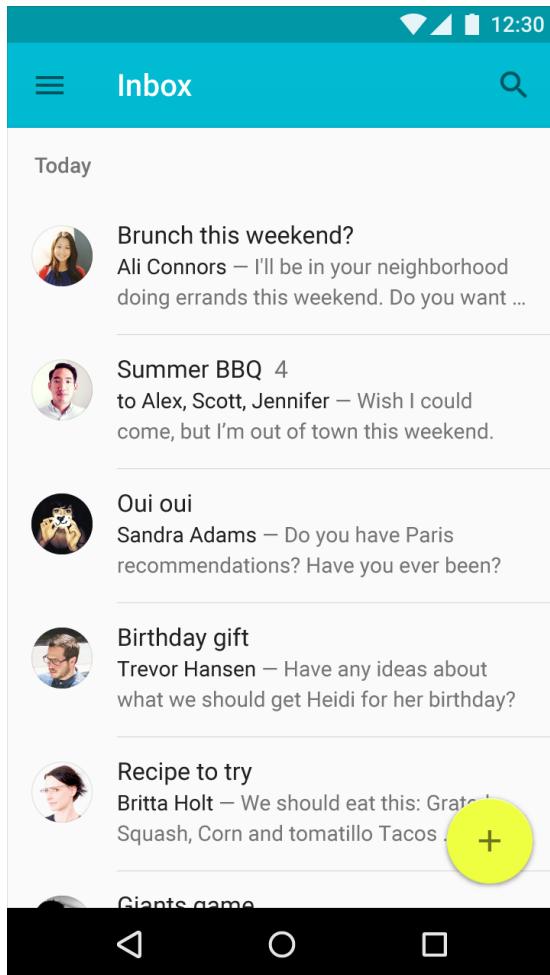


Figura 4.12: Esempio di List

Una list consiste in una singola colonna divisa in settori di uguale altezza chiamati Row che funzionano come contenitori di Tile, componenti che possono variare in altezza se non fossero contenuti in una lista. Le List, a differenza delle Card, si adattano benissimo a rappresentare tipi di informazioni che sono omogenei tra loro, come alcune immagini e testi. Le liste sono ottimizzate per la comprensione e la lettura. Esse permettono di fare uno scroll solo in verticale e tutti i contenuti all'interno di essa devono avere un ordine gerarchico, come può essere quello alfabetico. Le liste per quanto riguarda la struttura delle informazioni all'interno di esse, devono seguire una struttura gerarchica; infatti una list non può contenere più di tre linee di testo. Se si dovessero superare le tre linee di testo, a quel punto si userebbe la Card.

Specifiche tecniche:

1. La maggior parte dello spazio deve essere dedicata all'azione principale;
2. Il contenuto che distingue una Row da un'altra deve essere situato a sinistra;
3. Nelle Tile con più contenuti, il contenuto più importante va nella prima linea;
4. Le azioni secondarie devono essere situate a destra.

4.2.6 Bottom Sheets

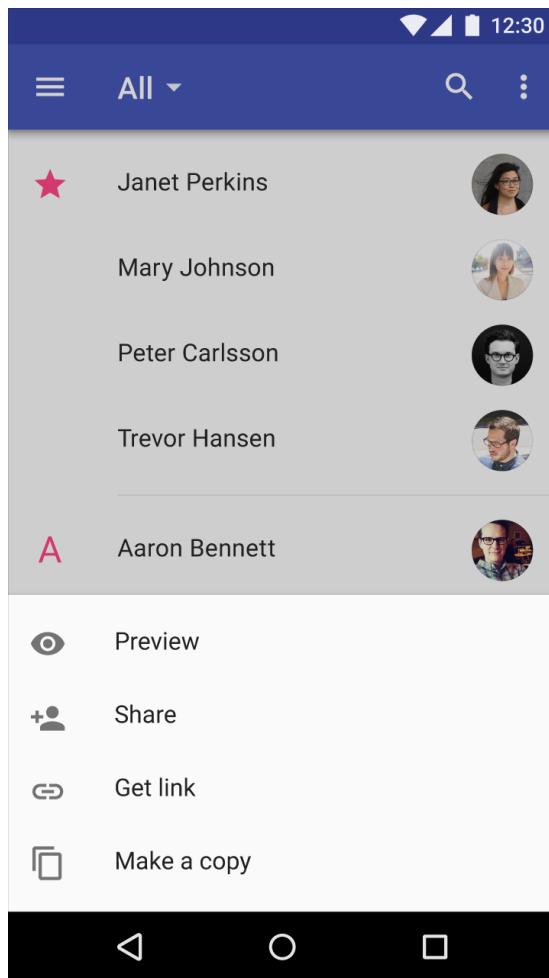


Figura 4.13: Esempio di Bottom Sheet

Il Bottom Sheets è un componente Material che si richiama facendo uno slide da sotto e rivela più contenuti. Esistono due tipi di Bottom Sheet:

1. Un bottom sheet che rappresenta un'alternativa al menu e che deve essere richiamato con le gestures, usato principalmente per ambienti mobili.
2. Un bottom sheet persistente che mostra i contenuti dell'app, usato soprattutto in ambito desktop.

All'interno delle applicazioni in material Design, esso rappresenta l'unico modo dell'app per comunicare con applicazioni esterne; infatti dentro di esso molte volte troviamo i pulsanti che servono per condividere i contenuti o le informazioni con altre applicazioni.

5. Angular Material

Per gli sviluppatori che usano AngularJS, Angular Material é un UI Component Framework che implementa le specifiche del Material Design all'interno dell'applicazione. Angular Material é stata usata per creare l'interfaccia grafica sia della nostra webApp, sia dell'applicazione per Android e Ios. Per la spiegazione e l'analisi di questo Framework, prenderemo in esempio la costruzione della nostra web app, in quanto un comportamento pressoché uguale abbiamo usato per tutti e due i tipi di interfaccia. Qui di seguito vedremo il codice dei componenti piú importanti di Proximity System.

5.1 Accessibilitá del prodotto

Un prodotto scritto con l'ausilio di Angular Material, che segue i canoni del Material Design, permette a qualsiasi utente di usare l'applicazione nel modo migliore possibile. La UI infatti deve essere il piú semplice possibile. I canoni principali che un buon programmatore o un designer deve seguire per avere questi risultati sono:

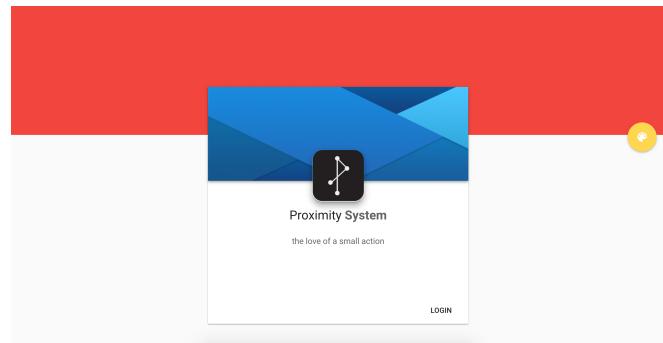
1. L'interfaccia deve essere responsive e scalabile;
2. Deve esserci un forte contrasto tra le componenti principali;
3. Indicare (se strettamente necessario) cosa l'oggetto fa se interagisce con l'utente;
4. Creare spesso dialog di conferma azione;
5. Ripetere le strutture gerarchiche presenti nel Material Design.

5.2 Flex

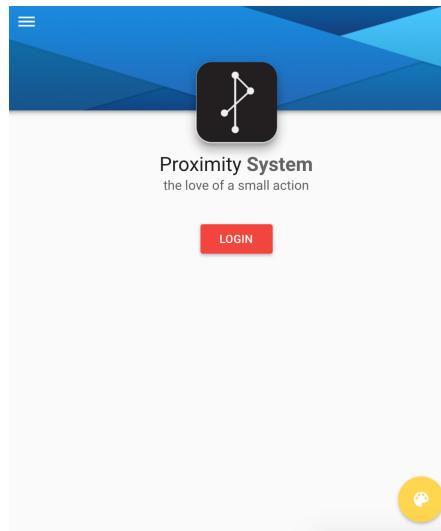
Angular Material ha una gestione molto comoda del Layout della pagina, infatti attraverso comandi come "flex" e "hide" permette la gestione della pagina in modo da cambiare anche drasticamente la grafica per regolarsi a seconda della grandezza. "Hide-xs", per esempio, permette la comparsa dei vari elementi solo se la larghezza della pagina é di 600px o inferiore.

Qui di seguito riportiamo il codice della nostra Home per quanto riguarda la grafica mobile:

```
1 <div hide-gt-sm>
2   <md-toolbar class="image2_md-tall_md-whiteframe-z2">
3     <md-button hide-gt-sm ng-click="toggleSidenav('left')" class="md-icon-button"
4       aria-label="Settings">
5       <md-icon class="md-alt-text_white" md-svg-icon="img/hamburger.svg">
6       </md-icon>
7     </md-button>
8   </md-toolbar>
```



(a) *Home page della webapp desktop.*



(b) *Home page della webapp mobile.*

Figura 5.1: Home Responsive

```

9 <div class="height1" layout="row" layout-align="center_start">
10   
11 </div>
12 <div layout="row" layout-align="center_start">
13   <span class="_md-headline">Proximity <strong class="gray">System</strong></span>
14 </div>
15 <center><span class="gray"> the love of a small action</span></center>
16 <div layout="column" layout-align="center_center">
17   <p flex> </p>
18   <md-button class="md-raised_md-primary_white" ng-if="user.username===''" href="#/login">Login </md-button>
19   <md-button class="md-raised_md-primary_white" ng-if="user.username!==''" href="#/login">Logout </md-button>
20 </div>
21 <md-fab-speed-dial hide-gt-md ng-hide="demo.hidden" md-direction="up"
22   md-open="demo.isOpen" class="md-scale_md-fab-bottom-right"
23   ng-class="{ '_md-hover-full': demo.hover }" ng-mouseenter="demo.isOpen=true"
24   ng-mouseleave="demo.isOpen=false">
25   <md-fab-trigger>
26     <md-button aria-label="menu" class="md-fab">
27       <md-icon style="fill:_white" md-svg-icon="img/palette.svg"></md-icon>
28     </md-button>
29   </md-fab-trigger>
30   <md-fab-actions>
31     <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme1')">
32       <md-icon style="fill:_blue" md-svg-icon="img/paint.svg"></md-icon></md-button>
33     <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme')">
34       <md-icon style="fill:_red" md-svg-icon="img/paint.svg"></md-icon></md-button>
35     <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme2')">
36       <md-icon style="fill:_green" md-svg-icon="img/paint.svg"></md-icon></md-button>
37   </md-fab-actions>
38 </md-fab-speed-dial>
39 </div>
40
41

```

Codice 5.1: webserver/angular/templates/home.html

5.3 Gestione dei Temi

Abbiamo parlato nel capitolo precedente dell'importanza dei colori all'interno del Material Design. Importanza che viene risaltata all'interno di Angular Material attraverso l'uso dei moduli. Infatti attraverso ”\$mdThemeProvider” si possono aggiungere temi, che vanno a cambiare sia il primary, sia l'accent color, diversi da quello di default¹.

Di seguito riportiamo il codice dei nostri tre temi:

```

1 $mdThemingProvider.theme('altTheme')
2   .primaryPalette('red')
3   .accentPalette('amber')
4 $mdThemingProvider.theme('altTheme1')
5   .primaryPalette('cyan')
6   .accentPalette('deep-orange')
7 $mdThemingProvider.theme('altTheme2')
8   .primaryPalette('green')
9   .accentPalette('deep-purple')
10 $mdThemingProvider.setDefaultTheme('altTheme');
11 $mdThemingProvider.alwaysWatchTheme(true);

```

Codice 5.2: /webserver/angular/js/routes.js

¹il colore di default è il blu

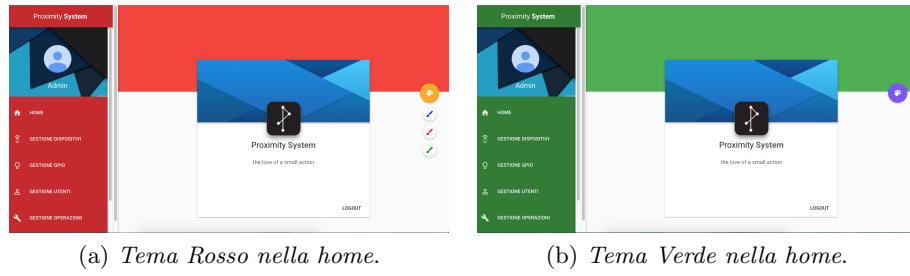


Figura 5.2: Gestione Temi

5.4 Componenti Principali

5.4.1 Sidenav

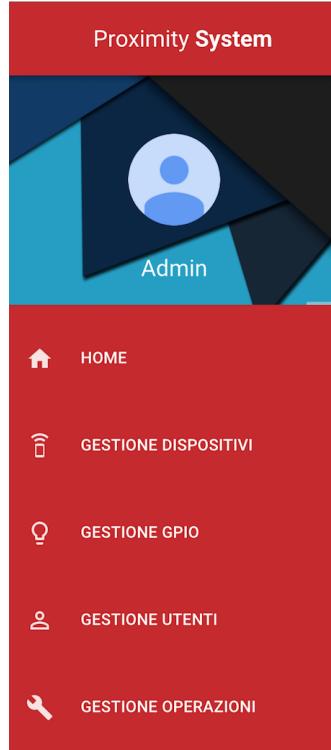


Figura 5.3: Sidenav Proximity System

É un componente che può essere chiuso e aperto in maniera programmata, essa funge da barra di navigazione all'interno della nostra applicazione. La Sidenav, se necessario, può contenere anche informazioni sull'utente o informazioni riguardanti l'applicazione. La Bottom Navigation, per quanto riguarda la parte mobile, non è stata implementata perché non era ancora disponibile all'interno del framework quando è stata creata la grafica per Proximity System. Codice della Sidenav usata all'interno della nostra webapp:

```

1 <md-sidenav ng-if="user.username != ''" class="height_md-whiteframe-z2_md-sidenav-left"
2   md-component-id="left" md-is-locked-open="$mdMedia('gt-sm')">
3     <md-toolbar class="md-whiteframe-z2_md-hue-2" md-scroll-shrink>
4       <center><p class="white">Proximity <strong>System</strong></p></center>
5     </md-toolbar>
```

```

6   <md-content class="height">
7     <md-toolbar class="image" layout="column" layout-align="start_center">
8       <span flex></span>
9       <center>
10      <p class="white">{{user.firstname}}</p>
11    </center>
12  </md-toolbar>
13  <md-toolbar class="height_md-hue-2">
14    <md-list class="contact_list">
15      <md-list-item ng-repeat="nav1_in_nav">
16        <md-button class="contactButton_no-hover_larghezza" href="{{nav1.link}}"
17          ng-class="md-raised" ng-click="toggleSidenav('left')">
18          <md-icon class="position_one" md-svg-icon="{{nav1.icon}}></md-icon>
19          <p class="position_white_contact_list">{{nav1.name}}</p>
20        </md-button>
21      </md-list-item>
22    </md-list>
23  </md-toolbar>
24 </md-content>
25 </md-sidenav>

```

Codice 5.3: webserver/angular/index.html

5.4.2 Toolbar

In italiano "Barra degli strumenti", è una componente sopraelevata rispetto ai contenuti dell'applicazione ed è situata all'interno della nostra applicazione. Essa può racchiudere i comandi elementari dell'app, come per esempio l'apertura e la chiusura della sidenav, oppure funzioni leggermente più complesse, come la ricerca all'interno di un contenuto. In Angular Material, per richiamarla, si fa riferimento alla direttiva "md-toolbar". Per cambiare l'altezza di quest'ultima, invece, usiamo "md-medium-tall" o "md-tall". Ecco un esempio di codice con la ricerca del GPIO all'interno di essa, presente nel nostro codice:

```

1 <md-toolbar ng-show="!showSearch" class="colortoolbar">
2   <div class="md-toolbar-tools_md-whiteframe-z2">
3     <md-button hide-gt-sm ng-click="toggleSidenav('left')"
4       class="md-icon-button" aria-label="Settings">
5       <md-icon class="md-alt-text" md-svg-icon="img/hamburger.svg"></md-icon>
6     </md-button>
7     <h2>
8       <span class="gray">Gestione GPIO</span>
9     </h2>
10    <span flex></span>
11    <md-button class="md-icon-button" ng-click="showSearch_=!showSearch">
12      <md-icon class="gray" md-svg-icon="img/search.svg"></md-icon>
13    </md-button>
14  </div>
15 </md-toolbar>
16 <md-toolbar ng-show="showSearch">
17   <div class="md-toolbar-tools">
18     <md-button hide-gt-sm ng-click="toggleSidenav('left')"
19       class="md-icon-button" aria-label="Settings">
20       <md-icon class="md-alt-text" md-svg-icon="img/hamburger.svg"></md-icon>
21     </md-button>
22     <h2>
23       <span hide-sm hide-xs class="white">Gestione GPIO</span>
24     </h2>
25     <span flex></span>
26     <div class="toolbar-input">
27       <md-input-container>
28         <label class="white">Cerca GPIO...</label>
29         <input class="white" ng-model="searchInput">
30       </md-input-container>
31     </div>

```

```

32   <md-button class="md-icon-button" ng-click="showSearch_=_!showSearch">
33     <md-icon class="white" md-svg-icon="img/search.svg"></md-icon>
34   </md-button>
35 </div>
36 </md-toolbar>
```

Codice 5.4: webserver/angular/templates/gpio.html

5.4.3 Card



Figura 5.4: Card GPIO Proximity System

È il componente che ho usato di più all'interno di Proximity System, sia per la web app, sia per l'app in Ionic. La Card si è rivelata utilissima, in quanto poteva contenere tutti i tipi di informazioni di dimensioni variabili che avevamo. Per richiamarla in Angular Material si fa riferimento alla direttiva "md-card". Dentro di essa, poi, è possibile inserire tante altre direttive a seconda del contenuto che si ha a disposizione, come per esempio "md-card-title" o "md-card-title-media". Ecco qui un esempio di codice preso da una delle nostre card in Proximity System:

```

1  <div ng-if="users.length" layout="row" class="md-padding_sfondo"
2    layout-xs="column" layout-wrap>
3    <md-card flex-xs flex-gt-xs="50" ng-repeat="user_in_users/orderBy:'_id'">
4      <md-toolbar class="md-whiteframe-z2">
5        <div layout="row" layout-align="center_center">
6          <p class="white">{{user.username}}</p>
7        </div>
8      </md-toolbar>
9      <md-card-title>
10        <md-card-title-media>
11          <div>
12            
13          </div>
14        </md-card-title-media>
15        <md-card-content>
16          <strong class="skyblue">Nome:</strong> {{user.firstname}}<br>
17          <strong class="red"> Cognome:</strong> {{user.lastname}}<br>
18          <strong class="orange"> Permessi:</strong> {{user.permission}}<br>
19        </md-card-content>
20      </md-card-title>
21      <md-card-actions layout="row" layout-align="end_center">
22        <md-menu>
23          <md-button class="md-raised_md-primary" ng-click="$mdOpenMenu() ">
24            <span class="white">azioni</span></md-button>
25          <md-menu-content>
26            <md-menu-item>
27              <md-button data-ng-if="!user.block" data-ng-click="blockUser(user)">
28                <md-icon class="md-alt-text" md-svg-icon="img/block.svg"></md-icon>
29                Blocca
30              </md-button>
31              <md-button data-ng-if="user.block" data-ng-click="blockUser(user)">
32                <md-icon md-svg-icon="img/person.svg"></md-icon>
33                Sblocca
34              </md-button>
35            </md-menu-item>
```

```

36      <md-menu-divider></md-menu-divider>
37      <md-menu-item>
38          <md-button ng-click="showAdd(user)">
39              <md-icon md-svg-icon="img/penna.svg"></md-icon>
40              Modifica
41          </md-button>
42      </md-menu-item>
43  </md-menu-content>
44  </md-menu>
45  </md-card-actions>
46</md-card>
47</div>

```

Codice 5.5: webserver/angular/templates/gpio.html

5.4.4 Raised Button

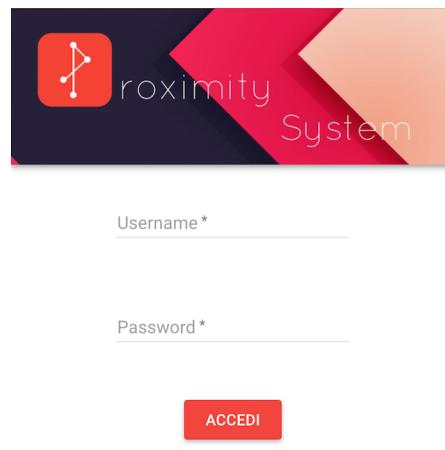


Figura 5.5: Raised Button in una Card

Come abbiamo visto precedentemente nel Material Design, questo tipo di bottone aggiunge un livello di elevazione rispetto l'interfaccia, ciò li rende facilmente individuabili in un UI piena di elementi; esempio di codice in Angular Material:

```
1 <md-button class="md-raise"> Raised Button </md-button>
```

Codice 5.6: esempio Raised Button

5.4.5 Flat Button

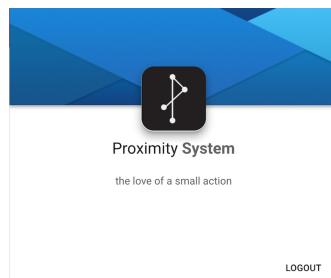


Figura 5.6: Flat Button di logout in una Card

Il button Flat invece viene usato in Angular Material per avere un pulsante a livello di interfaccia; esso è il bottone di "default" all'interno di questo framework; esempio di codice in Angular:

```
1 <md-button class="md-Primary"> Raised Button </md-button>
```

Codice 5.7: esempio Flat Button

5.4.6 Float Button



Figura 5.7: Float Button

Il bottone principale della nostra web app, ha un proprio codice CSS da rispettare, qui di seguito riportato:

```
1 .md-button .md-fab{  
2   line-height: 5.6 rem;  
3   min-width:0;  
4   width: 5.6 rem;  
5   height: 5.6 rem;  
6   border-radius: 50%;  
7 }
```

Codice 5.8: CSS Float Button

5.4.7 Button md-fab-speed-dial



Figura 5.8: md-fab-speed-dial

All'interno della nostra applicazione, abbiamo utilizzato un tipo particolare di bottone, che racchiude all'interno di esso altre opzioni. Questo bottone viene chiamato md-fab-speed-dial e qui sotto ne riportiamo il codice:

```
1 <md-fab-speed-dial hide-xs hide-sm hide-gt-md ng-hide="demo.hidden"  
2   md-direction="down" md-open="demo.isOpen" class="md-scale_md-fab-bottom-right_move-up1"  
3   ng-class="{ 'md-hover-full': demo.hover }" ng-mouseenter="demo.isOpen=true"  
4   ng-mouseleave="demo.isOpen=false"  
5     <md-fab-trigger>
```

```
6   <md-button aria-label="menu" class="md-fab_md-accent">
7     <md-icon style="fill:_white" md-svg-icon="img/palette.svg"></md-icon>
8   </md-button>
9 </md-fab-trigger>
10 <md-fab-actions>
11   <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme1')">
12     <md-icon style="fill:_blue" md-svg-icon="img/paint.svg"></md-icon></md-button>
13   <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme')">
14     <md-icon style="fill:_red" md-svg-icon="img/paint.svg"></md-icon></md-button>
15   <md-button class="md-fab_md-raised_md-mini" ng-click="setTheme('altTheme2')">
16     <md-icon style="fill:_green" md-svg-icon="img/paint.svg"></md-icon></md-button>
17 </md-fab-actions>
18 </md-fab-speed-dial>
```

Codice 5.9: codice FAB

6. Screenshoot Ionic App

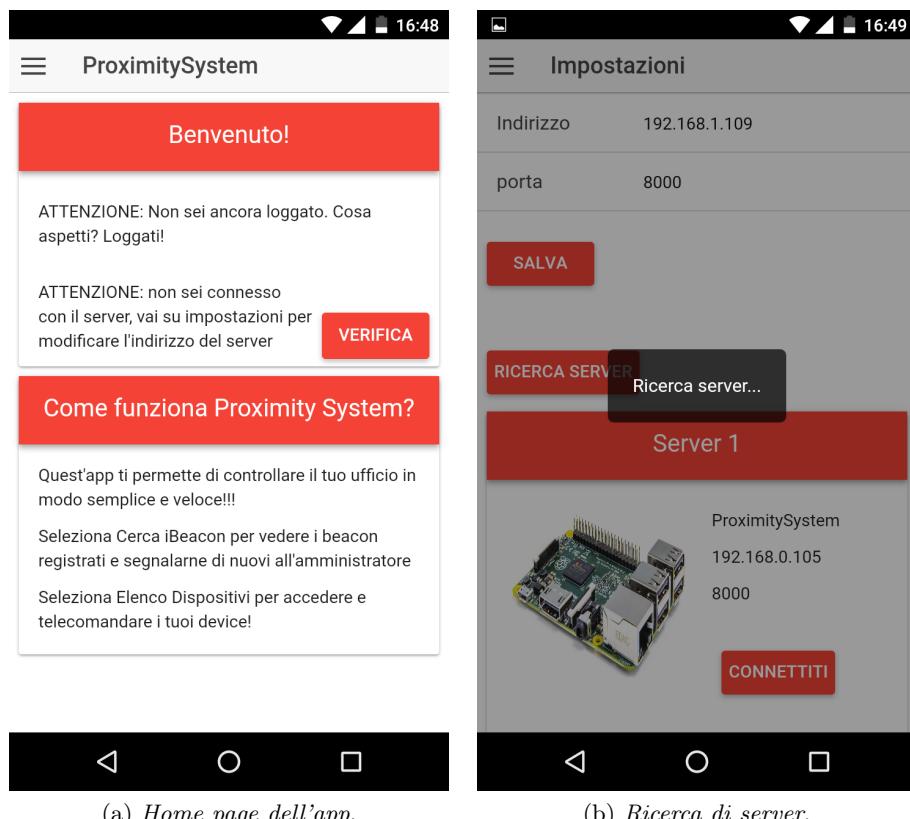


Figura 6.1: Connessione iniziale con il server di Proximity System.

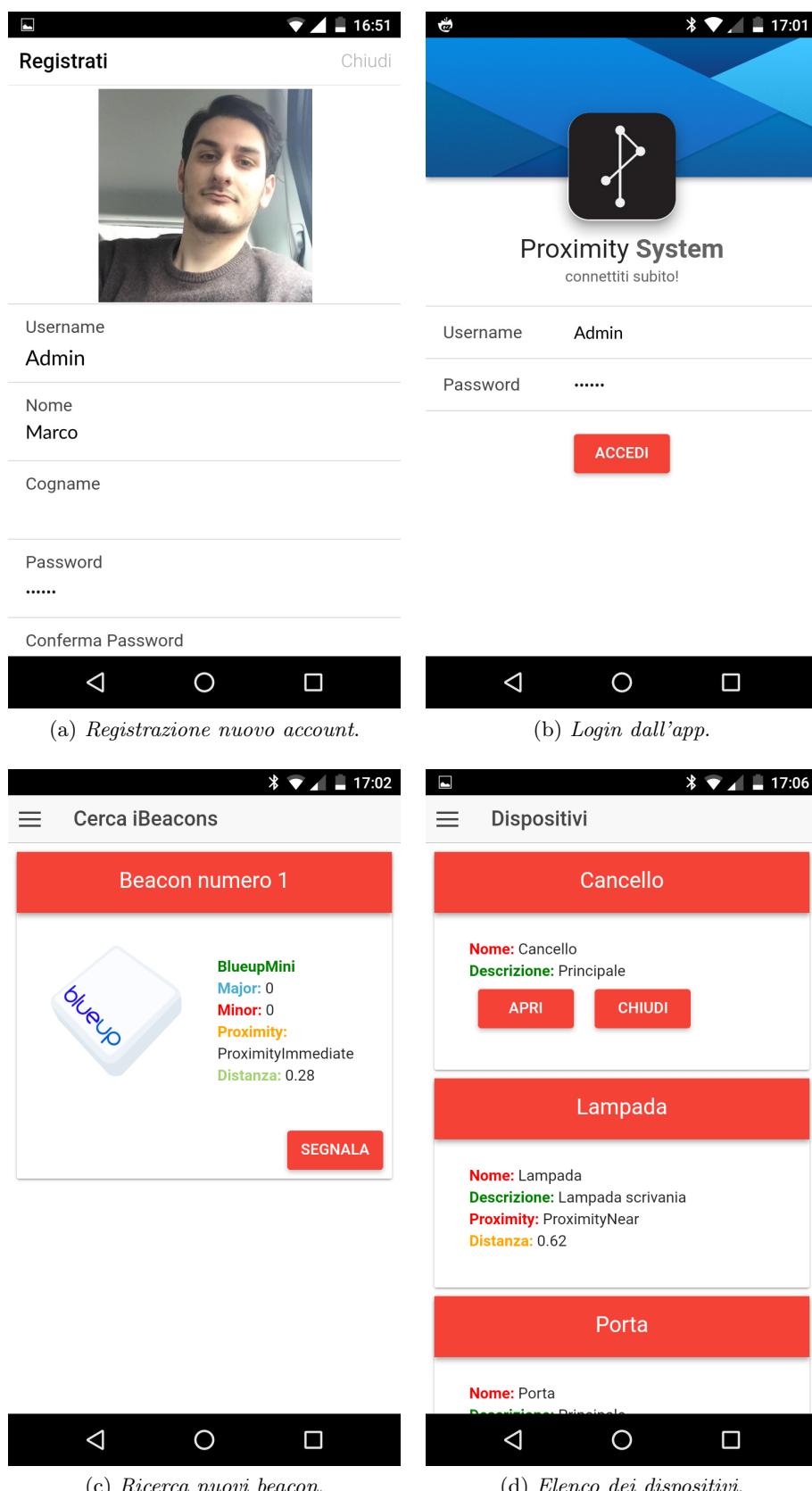


Figura 6.2: Alcune delle pagine principali dell'app.

7. Conclusioni

Questo lavoro é il risultato finale di un percorso piú che positivo per me durato tre anni. Ció mi ha permesso di crescere non solo umanamente ma anche professionalmente. Ho avuto modo di entrare nel mondo del lavoro come stagista presso l'azienda E-xtrategy di Monsano e lí ho conosciuto persone stupende che mi hanno aiutato, insieme al mio collega Saverio, nella creazione del nuovo prodotto "Proximity System".

Ho avuto modo di apprendere nuove tecniche lavorative a me, fino ad allora, sconosciute e di sperimentare l'importanza di lavorare in team. Team, che se pur piccolo, é stato molto coeso, creativo e collaborativo. Ho avuto modo di esprimere nel corso di questi tre mesi di Stage le mie idee liberamente, anzi mi é stato possibile potenziarle, migliorarle grazie anche e soprattutto alle intuizioni ed esperienze degli altri. Tutti insieme abbiamo scelto di creare Proximity System, un prodotto della domotica poco costoso e facile da usare, da inserire in un mercato che, se pur di nicchia, é comunque in forte espansione.

In questa prima fase abbiamo pensato di rivolgerci per lo piú a famiglie ed installatori, ciò non toglie che in un non lontano futuro possano rientrare anche tra il target dei nostri clienti: settori commerciali e studi professionali.

Grazie a Proximity Sistem é stato interessante anche osservare verso che cosa sta andando il mercato e quali sono i nuovi mezzi pubblicitari. Grazie ad un'app dedicata ed alla precisione del sistema che utilizza gli iBeacon per identificare i punti di interesse, é oggi possibile veicolare contenuti legati ad una micro localizzazione anche indoor, al fine di fornire informazioni utili, stimolare l'azione degli utenti, farli interagire e tutto a costi non troppo elevati.

Per tutto questo un ringraziamento particolare va in particolare alla sig.ra Silvia Morresi, che ci ha aiutato soprattutto nella parte economica, al dott. Francesco Strazzullo, nostro tutor, a tutta l'azienda E-xtrategy ed al Prof. Francesco De Angelis che ci ha seguito per tutto il nostro percorso.

A. Installazione Proximity System

A.1 Backend e WebApp

In questa sezione vedremo come installare e avviare Proximity System. Iniziamo con il backend scaricando il codice da GitHub

```
1 git clone https://github.com/e-xtrategy/unicam-beacon-server.git
```

Spostiamoci dentro la cartella webserver e installiamo Node tramite NVM

```
1 cd unicam-beacon-server/webserver  
2 nvm install
```

Installiamo i moduli per il backend tramite npm e per la webapp tramite bower

```
1 npm install  
2 npm install -g bower  
3 cd angular  
4 bower install  
5 cd ..
```

Assicuriamoci che il demone di MongoDB sia in esecuzione e lanciamo il seguente comando per popolare il database con i dati iniziali

```
1 node test/config/setup_tests.js
```

Prima di far partire il webserver dobbiamo settare una variabile d'ambiente per far capire a Node se ci troviamo in un PC o in un RaspberryPI

```
1 export NODE_ENV=development
```

Development indica che ci troviamo in fase di sviluppo e quindi su un PC. Se ci trovassimo su un RaspberryPi potremmo settarla su production o, più semplicemente, non settarla. Ora avviamo il webserver con il comando

```
1 npm start
```

Se ci troviamo in un PC e non vogliamo eseguire il webserver con i permessi di root è anche possibile avviare il tutto tramite

```
1 node server
```

Ora il nostro backend resterà in ascolto sulla porta 8000. Attenzione al fatto che il codice funziona soltanto sui sistemi operativi basati su Debian. Per far girare il programma su altri OS come Windows o Mac dobbiamo rimuovere il modulo rpi-gpio con npm

```
1 npm remove rpi-gpio
```

e commentare la prima riga di codice di

```
1 /unicam-beacon-server/webserver/app/services/gpio.js
```

A.2 App

Passiamo all'app. Iniziamo scaricando il codice dal repository su GitHub

```
1 cd
2 git clone https://github.com/e-xtrategy/unicam-ionic-beacon-app.git
```

Assicuriamoci di utilizzare la giusta versione di Node

```
1 cd app
2 nvm install
```

Installiamo Ionic, Cordova, Bower e tutte le dipendenze del progetto

```
1 npm install -g ionic cordova bower
2 bower install
3 ionic state restore
```

Passiamo all'esecuzione dell'app. Per fare ciò è necessario avere l'ambiente di sviluppo configurato correttamente. Per Android, quindi, sarà necessario aver installato correttamente Android Studio o più generalmente l'SDK, mentre per iOS Xcode. Adesso possiamo lanciare rispettivamente i seguenti comandi per android e iOS

```
1 ionic run android
2 ionic emulate ios
```

References

- [1] Trello. Trello. <https://trello.com/>.
- [2] Google. Angularjs. <https://angularjs.org/>.
- [3] Google. Angular material. <https://material.angularjs.org>.
- [4] Ionic. Ionic. <https://ionicframework.com/>.
- [5] Radigan Dan. Kanban. <https://www.atlassian.com/agile/scrum>.
- [6] Radigan Dan. Scrum. <https://www.atlassian.com/agile/scrum>.
- [7] Allan Afuah, Christopher L Tucci, and Francesco Virili. *Modelli di e-business: acquisire vantaggi competitivi con organizzazioni internet-based*. McGraw-Hill, 2002.
- [8] Lubbers Peter, Albers Brian, and Salim Frank. *HTML5 Tecniche professionali*. Hoepli.
- [9] Bacceli Micelli Cammisa, Matrisciano. *Diritto e tecniche amministrative vol.B*. Scuola & Azienda.
- [10] Alexander Osterwalder and Yves Pigneur. *Creare modelli di business efficace per ispirare chi deve creare o innovare un modello di business, Fag, Assago (MI)*, 2012.
- [11] Bacceli Micelli Cammisa, Matrisciano. *Diritto e tecniche amministrative vol.C*. Scuola & Azienda.
- [12] Domenico Ursino and Cristiano Antonio Capanna. Angularjs: un framework di frontiera per la realizzazione di siti web.