POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Progettazione *user-centered* di sistemi a regole in ambienti intelligenti

Relatori
Fulvio Corno
Luigi De Russis

Candidato
Rosalba Castro

Dicembre 2014

Ringraziamenti

Non è facile citare e ringraziare in poche righe tutte le persone che hanno reso possibile la realizzazione di questo lavoro di tesi: chi con un supporto morale o materiale, chi con consigli e suggerimenti, chi con parole di incoraggiamento, chi anche solamente ascoltandomi pur senza comprendere ciò che stessi dicendo.
La mia gratitudine va innanzitutto al Prof. Fulvio Corno e al Dott. Luigi De Russis per la loro guida, i loro preziosi consigli e il loro supporto durante i mesi in cui ho svolto il lavoro di tesi. È stato per me un grande piacere e un'occasione di crescita collaborare con il gruppo e-Lite del Politecnico di Torino.
Desidero inoltre ringraziare i colleghi e le colleghe che ho incontrato lungo tutto il mio percorso universitario, con cui ho condiviso momenti di studio ma anche di divertimento, ansie e preoccupazioni ma anche gioie e soddisfazioni. Con molti di loro ho instaurato sinceri rapporti di amicizia che andranno ben oltre il periodo universitario.
Il sostegno e la vicinanza dei miei amici, recenti e di vecchia data, sono stati per me fondamentali in questi anni da studentessa fuori sede a più di 1500 km di distanza da casa.
Ma il mio più grande ringraziamento va ai miei genitori, che mi hanno sempre incoraggiata e sostenuta, e alle mie sorelle Francesca e Irene, che mi hanno costantemente consigliata e aiutata senza mai smettere di credere in me. Senza di loro probabilmente, anzi sicuramente, non ce l'avrei fatta!

Grazie a tutti!
Rosalba

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Contesto generale

Negli ultimi decenni si è assistito a una crescente introduzione nelle abitazioni e nei luoghi di lavoro di tecnologie innovative aventi come obiettivo primario quello di rendere tali ambienti più confortevoli, più sicuri e più efficienti. Le nuove tecnologie dell'informazione e della comunicazione permettono di sviluppare servizi e applicazioni che consentono di controllare a distanza, attraverso un'interfaccia di comando, i dispositivi presenti nell'ambiente, fornendo in tal modo un valido ausilio allo svolgimento delle attività quotidiane. La tendenza è quella di integrare le varie tecnologie usate dagli elettrodomestici ed dai sistemi di comunicazione e di controllo che si trovano all'interno delle comuni abitazioni. Per far ciò è necessario aggiungere intelligenza[1] e funzioni di comunicazione a tutti i dispositivi che presentano una connessione a una rete informatica, come la rete Internet, alla quale l'abitazione deve essere collegata in modo che questi strumenti digitali possano scambiarsi dei dati e fornire dei servizi. Un ambiente, opportunamente progettato e tecnologicamente attrezzato, che metta a disposizione dell'utente delle apparecchiature e dei sistemi in grado di svolgere

[1] Con il termine intelligenza, riferito a un dispositivo, si vuole indicare la capacità dello stesso di riprodurre, almeno per alcuni aspetti, l'intelligenza umana. Si tratta di dispositivi tecnologicamente avanzati, di grande utilità perché in grado di fornire servizi che rispondono ai bisogni delle persone.

funzioni parzialmente o totalmente autonome viene chiamato "*ambiente intelligente*". Quando l'ambiente intelligente è un'abitazione, il termine usato è quello di "*smart home*" o "*casa intelligente*". Le funzioni degli impianti tradizionali si attivano tramite comuni dispositivi (interruttori, regolatori, cronotermostati, ecc.) che non sono in grado di interagire tra loro e non possono essere gestiti da un'unità centrale. Le interfacce di una casa intelligente invece consentono all'utente di inviare comandi e ricevere informazioni da parte di un sistema di controllo connesso ai diversi componenti dell'impianto. Questi possono avere ruoli molto differenti tra loro: esecuzione di specifici comandi impartiti dall'utente, monitoraggio di parametri ambientali e consumi energetici, gestione di regolazioni e carichi, generazione di segnalazioni e così via. All'esecuzione di un qualsiasi comando, un feedback viene inviato immediatamente all'utente, il quale può avere il totale controllo dello stato del sistema anche da remoto. Il controllo da remoto può avvenire, oltre che per mezzo dei tradizionali PC, anche per mezzo dei dispositivi mobile (*smartphone* e *tablet*). Recentemente infatti, l'enorme diffusione di *smartphone* e *tablet* ha fatto sì che questi dispositivi abbiano assunto un ruolo rilevante nel controllo dell'ambiente e l'uso di applicazioni mobile in questo contesto sembra affascinare molti utenti. Le interfacce grafiche dei programmi di gestione di ambienti intelligenti su dispositivi mobile dovrebbero avere quindi caratteristiche che garantiscano ad ogni tipologia di utente di poter utilizzare volentieri l'applicazione senza particolari difficoltà.
I sistemi che possono rivelarsi di più facile comprensione e utilizzo per la gestione della casa sono i sistemi a regole. Se un utente vuole impartire alla sua abitazione intelligente delle regole affinché essa possa semplificargli attività della vita quotidiana, è necessario un linguaggio facilmente comprensibile per l'utente e facilmente traducibile in comandi. Il linguaggio basato su regole nella forma "*IF condizione, THEN azione*" (*se ... allora ...*) potrebbe essere quello che più si adatta a soddisfare a questi compiti.

1.1.1 Approfondimento: la casa intelligente

Prima di parlare approfonditamente di casa intelligente è necessario introdurre il concetto di *domotica*. Il termine *domotica* è un neologismo che deriva dal francese "*domotique*", a sua volta contrazione della parola latina "*domus*" (casa, o dal greco "*domos*", con lo stesso significato) e di

"*automatique*"(automazione). Tramite l'utilizzo combinato di elettrotecnica, elettronica, informatica e telecomunicazioni la domotica studia e realizza sistemi integrati per l'automazione di processi e controlli attraverso i quali è quindi possibile ottenere una migliore qualità della vita, maggiore sicurezza e soprattutto un notevole risparmio dei consumi energetici. Essa rende intelligenti apparecchiature ed impianti, che acquisiscono la capacità di svolgere funzioni autonome o programmate dall'utente secondo reazioni a parametri ambientali. Tramite un sistema di acquisizione costituito da vari sensori (per es. di luminosità, temperatura, presenza di gas, umidità, movimento, ecc.), le informazioni sullo stato della casa vengono convogliate a un sistema informatico di controllo che monitora lo stato dei parametri ambientali ed eventualmente provvede a intervenire, azionando automaticamente alcuni dispositivi mediante segnali di comando. Il sistema informatico di controllo è generalmente un *gateway* che permette lo scambio di messaggi tra dispositivi che utilizzano protocolli di comunicazione diversi.
Negli ultimi anni il modello più utilizzato e sviluppato per case intelligenti è quello schematizzato in figura 1.1. Come si può notare in figura la ca-

Figura 1.1. Schema generale di una casa intelligente.

sa intelligente consiste nell'integrazione di differenti sistemi automatizzati, applicazioni e dispositivi integrati in un singolo nodo centrale che è capace di far dialogare i sistemi tra di loro e verso l'esterno. Il nodo centrale conferisce al sistema l'intelligenza del sistema stesso ed è un middleware, tipicamente chiamato *Home Gateway*. Un *Home Gateway* ha la peculiarità di rendere possibile la creazione di ambienti intelligenti più complessi e la comunicazione tra dispositivi disomogenei (come un qualsiasi gateway) e tra gateway stessi. Negli ambienti intelligenti si possono infatti avere delle problematiche di interoperabilità a causa della presenza, nello stesso ambiente, di molti sistemi intelligenti e di dispositivi appartenenti a sistemi diversi che non possono interagire per via delle diverse tecnologie e dei diversi protocolli utilizzati per la comunicazione. Per interagire quindi in una casa intelligente è essenziale avere uno strato di software, un middleware, che agisca da ponte tra le diverse tecnologie e supporti complesse interazioni tra i dispositivi. Superato questo problema una casa intelligente permette una gestione coordinata, integrata e digitalizzata di impianti domestici (quali per es. climatizzazione, energia o impianti di sicurezza), e ciò porta a un miglioramento in termini di sicurezza, comfort, risparmio energetico ed inoltre ad una migliore adattabilità alle esigenze degli utenti nella gestione della casa. Nel sistema intelligente preso in considerazione in questo lavoro di tesi l'*Home Gateway* che rappresenta l'intelligenza del sistema è il middleware *Dog*, sviluppato dal gruppo di ricerca e-Lite del Politecnico di Torino.

1.2 Obiettivi della tesi

L'obiettivo principale della tesi è lo studio, la progettazione e la realizzazione di un'applicazione mobile per la definizione di regole che consentano la gestione combinata dei dispositivi presenti in un *smart home*. È attraverso le regole che l'utente può dare al sistema intelligente direttive su come gestire gli impianti della casa.

Per orientare il lavoro, è stato fondamentale iniziare da un'approfondita lettura e da un'attenta analisi della letteratura riguardante interfacce per la composizione di regole in ambienti intelligenti, in particolare quelle rivolte a utenti non esperti.

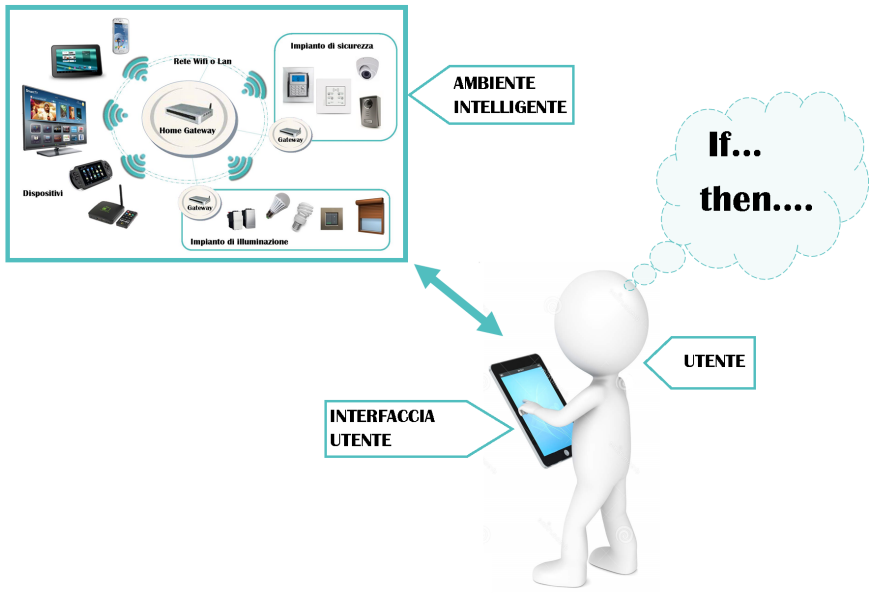

Figura 1.2. Schema generale del contesto in cui si colloca l'interfaccia utente da realizzare.

Per rispondere alle esigenze di tali utenti, nello sviluppo dell'applicazione oggetto del presente lavoro, si è cercato di:

- *Utilizzare una grammatica semplice e intuitiva per la creazione delle regole.* Gli utenti devono poter pensare nel loro linguaggio naturale ad una regola da creare e riuscire facilmente a riportare la loro idea sull'applicazione.

- *Creare un'interfaccia mobile chiara, facile e piacevole da usare per la composizione di regole.*

- *Utilizzare una metodologia di progettazione incentrata sull'utente (user-centered).* Per rispondere alle esigenze degli utenti, anche dei meno esperti, minimizzare le difficoltà di utilizzo e realizzare un prodotto finale chiaro, semplice e intuitivo, risulta, infatti, vantaggioso che le caratteristiche dell'interfaccia siano definite e validate coinvolgendo l'utente stesso.

- *Creare una modalità di composizione delle regole interattiva.* I dispositivi mobile si prestano ad essere utilizzati come strumento di interazione per la creazione delle regole. Sfruttando tale caratteristica, si è ideata, in aggiunta alla modalità classica di composizione delle regole, una seconda modalità di composizione che consente anche l'interazione dell'utente con l'ambiente circostante.

La Figura 1.2 illustra schematicamente il contesto in cui si colloca l'interfaccia utente da realizzare. In particolare è evidenziata l'interazione tra l'interfaccia utente, il sistema "*ambiente intelligente*" e l'utente. L'utente impartisce alla sua *smart home* dei comportamenti, che il sistema sarà tenuto a prendere in considerazione, attraverso la definizione di regole nella forma "*If ... Then ...*" programmate in un'applicazione mobile.

1.3 Scelte effettuate e requisiti

L'interfaccia utente è stata realizzata per sistema operativo Android 4.x e ottimizzata per un dispositivo con schermo di 10 pollici.
Per realizzare l'applicazione è stato utilizzato l'ambiente di sviluppo *Eclipse* con l'installazione dell'*Android Developer Tools* (*ADT*). La parte algoritmica e le parti dinamiche dell'interfaccia grafica sono state quindi implementate in Java mentre gli elementi statici dell'applicazione in XML.
Per quanto riguarda la modalità "interattiva" di composizione delle regole è necessario che l'ambiente intelligente acquisisca le interazioni che l'utente effettua con esso e le renda disponibili, nell'interfaccia utente, per il loro utilizzo. A tale scopo, è stata utilizzata la tecnologia *WebSocket* per permettere la comunicazione tra l'interfaccia utente e il sistema, e in particolare le *API WebSocket* fornite dal middleware *Dog*.
Il software realizzato durante la tesi sarà rilasciato con licenza di tipo *open source* (*Apache License 2.0*).

1.4 Struttura della tesi

Nei prossimi capitoli verranno approfonditi alcuni aspetti necessari a comprendere i motivi e le modalità che hanno portato alla realizzazione di

un'interfaccia per le regole. Si descriveranno le fasi di prototipazione, progettazione e di implementazione del progetto e si presenteranno i risultati ottenuti. In dettaglio:

- Nel capitolo 2 "*Regole e interfacce per smart home*", si approfondiranno alcuni concetti accennati in questo primo capitolo e sarà presentato lo stato dell'arte.
- Nel capitolo 3 "*Soluzioni tecniche adottate*", verrà introdotto il sistema di riferimento e descritto il middleware *Dog*, realizzato dal gruppo e-Lite del Politecnico di Torino. Inoltre, verranno brevemente illustrate alcune caratteristiche del sistema operativo Android, utilizzato per la realizzazione dell'interfaccia mobile.
- Nel capitolo 4 "*Progettazione fase 1: prototipazione*", si presenteranno i prototipi di interfaccia creati e le varie scelte progettuali comuni ai vari prototipi.
- Nel capitolo 5 "*Progettazione fase 2: studio utenti*", verrà descritto il test cui sono stati sottoposti alcuni utenti, in fase di progettazione, per confrontare i prototipi di interfaccia realizzati. Verranno inoltre presentati i risultati del test e le modifiche progettuali fatte alla luce di tali risultati.
- Nel capitolo 6 "*Implementazione*", si presenteranno le scelte implementative, si parlerà di come è stata realizzata l'interfaccia utente e si discuterà anche dell'organizzazione del codice e del suo possibile riutilizzo.
- Nel capitolo 7 "*Conclusione e sviluppi futuri*", verranno elencati alcuni sviluppi futuri che il progetto descritto in questa tesi potrà avere.

Capitolo 2

Regole e interfacce per *smart home*

Una *smart home* è un'abitazione dotata di una rete tecnologicamente avanzata che collega tra loro sensori, dispositivi e apparecchiature domestiche monitorabili, accessibili e controllabili a distanza.
Affinché un utente possa gestire al meglio la sua casa intelligente e adattarla ai suoi bisogni e ai bisogni delle persone che vivono all'interno dell'abitazione, è importante che l'utente possa sempre controllare il sistema e stabilire quali attività delegargli. Le modalità che consentono all'utente di svolgere questo compito sono state oggetto di studio negli ultimi decenni. In letteratura molti autori fanno riferimento a un approccio che permette la composizione di regole da parte degli utenti stessi. A tale scopo si ricercano modi per creare interfacce semplici da gestire, facili da usare, intuitive, e poco invasive.
Nei successivi paragrafi verrà introdotta la logica di tali regole (paragrafo 2.1) e le caratteristiche che una applicazione user-centered deve avere (paragrafo 2.2). Di seguito verranno analizzate le più recenti ricerche condotte in quest'ambito (paragrafo 2.3).

2.1 Regole: il modello *ECA*

Molti sistemi per la gestione di *smart home* impiegano generalmente un linguaggio basato su regole per delegare attività l'ambiente. Un linguaggio di

questo tipo implica tipicamente che le istruzioni date al sistema dall'utente siano nella forma: "Quando qualcosa accade, se certe condizioni sono soddisfatte, allora fa qualcos'altro". In letteratura molti concordano nel ritenere che l'approccio basato su regole risulta molto espressivo perché imita i processi utilizzati dagli utenti quando devono svolgere azioni della quotidianità. Poiché le persone generalmente vedono negli eventi del mondo fisico delle relazioni di casualità, su tali relazioni dovrebbe basarsi la composizione delle regole.
Il modello di composizione delle regole basato su eventi, condizioni e azioni, è chiamato *ECA* (*Event-Condition-Action*) e applica tale causalità a sistemi informatici guidati da eventi. Il modello *ECA* fu originariamente utilizzato per specificare le regole delle basi di dati attive, ovvero regole che vengono attivate in modo automatico al verificarsi di specifici eventi sulla base di dati e, verificate alcune condizioni, vengono eseguite azioni sulla base di dati stessa. Tramite le regole di tipo *ECA* si compiono quindi azioni in risposta a eventi interni o esterni al database. Successivamente, questa tipologia di regole è stata adottata per altri sistemi *event-driven*.
Nel modello *ECA* una regola si compone di 3 elementi:

1. L'**evento** che attiva la regola. Un evento, in una *smart home*, corrisponde di solito a una operazione di cambiamento di uno stato o a un messaggio inviato autonomamente da un dispositivo. In un modello generale potrebbe anche trattarsi di un evento temporale, ad esempio: "ogni giorno alle ore 8 del mattino". Molto spesso il termine *trigger* (evento scatenante) è usato al posto del termine evento.

2. La **condizione** (o un insieme di condizioni) che impone ulteriori vincoli per l'esecuzione della regola. Quando si è verificato l'evento scatenante, è possibile valutare una o più condizioni facoltative. Se nessuna condizione è specificata, la regola verrà eseguita quando accade l'evento. Se è specificata una condizione, questa viene prima valutata, e, solo se il valore restituito è corretto, la regola sarà effettivamente eseguita. Se invece sono specificate più condizioni, l'intero insieme delle condizioni deve essere verificato per far scattare la regola.

3. L'**azione** che si deve compiere quando l'evento si è scatenato e le condizioni sono state verificate.

Il significato di una regola *ECA* è sostanzialmente: "Quando qualche evento si verifica, controlla le condizioni eventualmente presenti e, se esse sono verificate, esegui le azioni specificate". Linguaggi di regole basati su un modello con eventi, condizioni e azioni, e sue varianti sono ampiamente usati per permettere agli utenti un controllo indiretto di *smart home* (come in:[1], [2], [3], [4], [5], [6], [7], [8]). Le regole *ECA* possono, senza grandi difficoltà, essere tradotte in espressioni comprensibili per gli utenti in modo da permettere l'apprendimento e la modifica indipendente.

2.2 Interfacce e dispositivi

2.2.1 L'approccio *user-centered*

Con l'acronimo HCI (*Human-Computer Interaction*) ci si riferisce a dei metodi d'interazione tra gli utenti e i computer finalizzati alla progettazione e allo sviluppo di sistemi interattivi che siano usabili, affidabili e che supportino e facilitino le attività umane.
Nel campo dell'informatica, il termine interfaccia grafica, o anche GUI (*Graphical User Interface*), si utilizza per indicare un' interfaccia utente che permetta l'interazione con l'utente attraverso metafore grafiche di immediata comprensibilità. Per tutte le interfacce uomo-computer risultano fondamentali i concetti di usabilità ed accessibilità, i quali garantiscono a diverse tipologie di utente un uso "amichevole" del dispositivo. Per raggiungere tale scopo risulta essenziale la creazione di interfacce *user-centered*, ovvero incentrate sull'utente.
In generale, l'obiettivo della metodologia di progettazione *user-centered* è quello di tenere presente i bisogni, le esigenze e le attività degli utenti. Il tipico approccio utilizzato per soddisfare tali intenti prevede il coinvolgimento degli utenti nelle varie fasi di progettazione e verifica, in modo da poter avere feedback da parte degli utilizzatori dell'interfaccia per migliorarla in itinere. Nella fase di progettazione vengono sottoposti agli utenti mockup e/o prototipi in modo da testare le loro reazioni e ottenere da queste suggerimenti e indicazioni utili per ottenere una migliore interfaccia. Tale processo di progettazione, che utilizza feedback attivi da parte degli utenti, produce un numero significativo di vantaggi: le applicazioni finali risultano più facili da capire e da utilizzare; si ottiene così, a posteriori, un riscontro positivo da parte degli utenti, se ne migliora la soddisfazione nell'utilizzo e

si riduce la richiesta di spiegazioni e supporto.
Potrebbe sembrare che l'approccio *user-centered* complichi il processo di sviluppo del sistema, a causa della necessità di rispondere ai feedback degli utenti con continue modifiche del software. Nonostante ciò, molti studi su questa filosofia di progettazione sostengono che i benefici che si possono ottenere sono evidenti. Un design centrato sull'utente promuove infatti la comunicazione tra il team di progettazione e gli utenti, permette di individuare i problemi nelle prime fasi di sviluppo dell'applicazione, quando è molto più conveniente implementare delle modifiche ed inoltre permette di avere un buon livello di ergonomia. Il livello di ergonomia[1] esprime la qualità del rapporto tra l'utente e il mezzo fisico utilizzato. Nell'ambito delle *smart home* è essenziale raggiungere un buon livello di ergonomia, per cui un approccio di tipo *user-centered* acquisisce un ruolo fondamentale negli scenari domestici. Molti dei dispositivi che si trovano in casa sono interattivi, come lettori DVD, gli impianti hi-fi, la lavatrice, i sistemi di riscaldamento; essi molto spesso sono "sottoutilizzati" perché visti come difficili da capire ed utilizzare. Se a ciò si aggiunge la rapida e dilagante diffusione di dispositivi collegati online, il potenziale di complessità e le problematiche relative al loro utilizzo da parte dell'utente domestico crescono. Inoltre gli utenti, che nell'ambiente domestico ricercano un po' di relax e sollievo dallo stress e dagli impegni lavorativi, potrebbero non volere trascorrere troppo tempo a leggere lunghi manuali per far funzionare applicazioni complesse.

2.2.2 I dispositivi mobile

All'interno del settore della gestione di ambienti domestici, gli ormai diffusissimi smartphone e tablet stanno assumendo un ruolo sempre più centrale. Le ampie possibilità di connessione e la forte portabilità hanno fortemente contribuito a rendere tali dispositivi di primaria importanza anche all'interno di una *smart home*. Le applicazioni all'interno di questo settore sono in continua crescita e rappresentano uno dei punti di maggiore interesse per le aziende produttrici di software e hardware per *smart home*.
Nonostante sia assodato che non possa esistere un unico dispositivo che

[1] L'ergonomia è quella scienza che si occupa dell'interazione tra gli elementi fisici di un sistema e la funzione per cui vengono progettati, allo scopo di migliorare la soddisfazione dell'utente e l'insieme delle prestazioni del sistema.

soddisfi ogni utente e ogni contesto di utilizzo, gli smartphone e i tablet sembrano essere, per le loro caratteristiche, i dispositivi che più si prestano alla gestione di *smart home*.
Rispetto a dispositivi più tradizionali quali i computer, gli smartphone e i tablet vantano caratteristiche che ne consentono una maggiore versatilità ed accessibilità per la maggior parte degli utenti.

Vantaggi	- Leggeri e facili da trasportare. - Non richiedono dispositivi di input aggiuntivi. - Accessibili sempre e da qualunque posto.
Svantaggi	- Poca precisione nella selezione. - Inserimento dati più lento. - *drag and drop* più complesso per l'utente.

Tabella 2.1. Utilizzo di dispositivi mobile: vantaggi e svantaggi rispetto ai dispositivi tradizionali (PC).

L'utilizzo di dispositivi mobile può avere svariati vantaggi ma anche diversi svantaggi; nella Tabella 2.1 ve ne sono schematizzati alcuni. Un primo vantaggio deriva dal fatto che si tratta di dispositivi portatili; in quanto tali, sono leggeri e facili da trasportare. Inoltre non richiedono ulteriori dispositivi per l'inserimento di input, non necessitano di tastiera o mouse aggiuntivi e quindi non serve spazio aggiuntivo oltre a quello richiesto dallo schermo stesso. Se da un lato, per tali ragioni, questa caratteristica risulta vantaggiosa, dall'altro l'inserimento dei dati potrebbe risultare più lento rispetto all'inserimento attraverso una tastiera fisica. Inoltre alcune operazioni, come ad esempio il *drag and drop*, risultano, per l'utente, un po' più complicate da effettuare con il touch screen rispetto all'utilizzo di un mouse; il puntamento con il touch screen può essere più impreciso se effettuato senza l'uso di dispositivi di input specializzati. Inoltre gli smartphone presentano schermi di dimensioni inferiori rispetto ai tablet; ciò può, per alcuni utenti, aumentare le difficoltà di selezione, può rendere più difficile la lettura e limitare la quantità di informazioni che possono essere presentate in qualsiasi momento.
Un importante punto a favore dei dispositivi mobile è invece quello di essere

accessibili sempre e da qualunque posto. Tale caratteristica riesce a soddisfare l'esigenza degli utenti di accedere agli stessi servizi in qualsiasi luogo si trovino, garantendo loro di essere sempre connessi e aggiornati.

Nell'ambito della creazione di regole in una casa intelligente, il vantaggio principale derivante dall'utilizzo di uno smartphone o di un tablet è quello di poter utilizzare il dispositivo stesso come strumento di interazione tangibile per la creazione delle regole, oltre a poter controllare la casa in qualsiasi posto l'utente si trovi tramite un'interfaccia grafica. Quest'approccio risulta essere funzionale perché permette all'utente di delegare alla casa tramite il dispositivo i settaggi voluti, sfruttando gli spostamenti e le azioni fatte all'interno della casa.

2.3 Stato dell'arte

In letteratura la maggior parte delle applicazioni per la creazione delle regole sviluppate per l'ambiente domestico utilizzano il modello *ECA* descritto nel paragrafo precedente, o sue varianti, e sono applicazioni "*context-aware*" ovvero sensibili al contesto. Il sistema più famoso per far ciò è iCAP [1], un ambiente di sviluppo basato su regole che supporta gli utenti finali nella creazione di applicazioni senza necessità di utilizzare linguaggi di programmazione. Esso permette agli utenti di creare una grande varietà di regole (dette anche *context-aware application*, selezionando menù e trascinando elementi grafici (oggetti, attività, luoghi, persone e istanti temporali) (Figura 2.1). iCAP supporta tre comuni tipi di regole:

- Semplici regole *if-then*, in cui una data azione è eseguita solo quando una certa condizione è soddisfatta.

- Azioni basate su relazioni, ovvero si sviluppano regole costruite sulla base di relazioni personali, spaziali e temporali, così da essere più vicine al modo di pensare umano. iCAP può permettere la creazione di regole del tipo: "Quando il mio coinquilino è nella sua stanza, ricordami di chiedere se può andare a fare la spesa"; questa regola implica che il sistema abbia conoscenza su delle relazioni personali (il mio coinquilino), spaziali (la stanza del mio coinquilino è nel mio stesso appartamento),temporali (il mio coinquilino è entrato nella sua stanza qualche minuto prima).

- Personalizzazione dell'ambiente, è l'ambiente a soddisfare le diverse preferenze dei suoi abitanti.Per esempio, se ad un utente piace la luce soffusa e la musica classica e ad un'altro utente un ambiente luminoso e la musica rock, per soddisfare le esigenze di entrambi gli utenti, l'ambiente deve considerare le loro preferenze e modificarsi in realazione a questo.

Da studi effettuati dagli ideatori di iCAP su un campione di possibili utenti finali, è stato evidenziato che la grande maggioranza di essi è portato a pensare, e quindi a progettare un'applicazione, in termini di semplici regole "*if-then*" (solo un quarto dei partecipanti ha creato regole più complesse contenenti costrutti booleani), e che l'ambiente visivo è tanto più appropriato quanto più è semplice e familiare.

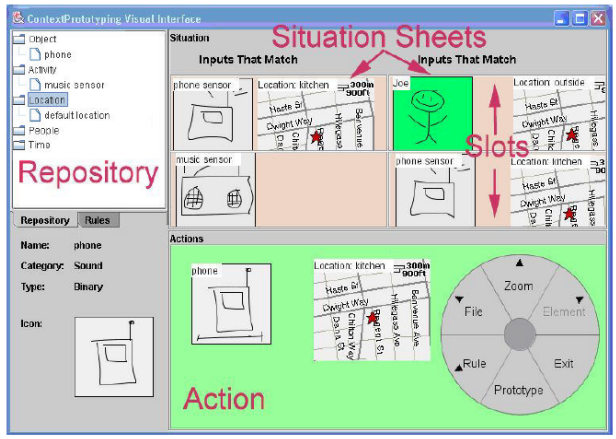

Figura 2.1. Interfaccia basata su regole del sistema iCAP.

Il lavoro e i risultati ottenuti da iCAP sono stati considerati da studi successivi in ambiente domotico. Lo studio di Bonino et al.[2] delinea un'interfaccia visiva per il web (Figura 2.2), indirizzata ad utenti con un livello base di conoscenza informatica, utilizzando la metafora del *drag-and-drop*. La grammatica proposta è simile a quella di iCAP, vengono usate regole con una struttura *"if-then"*, ma a contrario di iCAP viene fatta distinzione tra eventi e condizioni. Essa è basata su quattro parole chiave:

- SE (IF): esprime l'evento che farà scattare la regola;

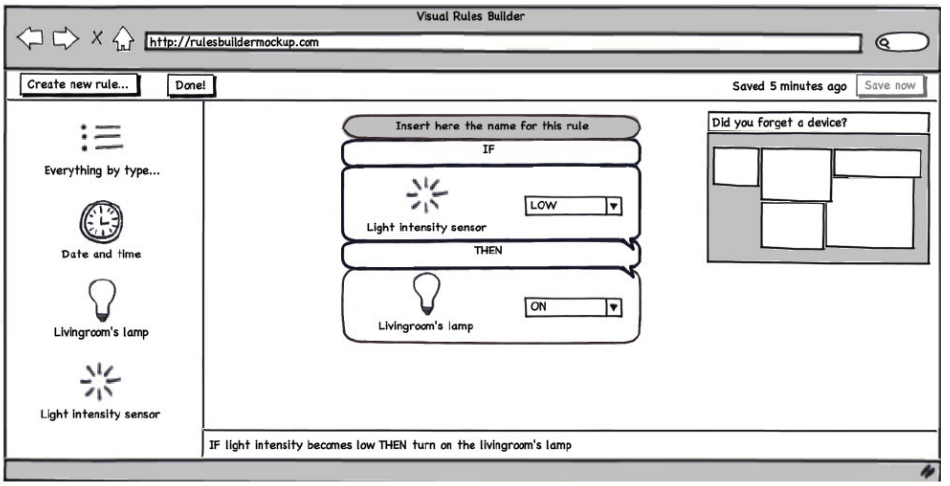

Figura 2.2. Interfaccia *drag and drop* gruppo e-Lite del Politecnico di Torino.

- ALLORA (THEN): indica una serie di azioni da eseguire al verificarsi dell' evento;
- QUANDO (WHEN): definisce una o più condizioni che limitano l'evento (nel caso vi siano più vincoli questi devono essere soddisfatti tutti contemporaneamente);
- OPPURE SE (OR IF): è una congiunzione per ripetere più volte la parte IF-WHEN.

Le prime due parole chiave sono obbligatorie, mentre le altre sono opzionali. Una regola è quindi sempre composta da un evento (IF) che deve verificarsi e da una o più azioni (THEN) che devono essere eseguite; inoltre opzionalmente possono esser presenti una o più condizioni (WHEN) che limitano l'evento e più eventi da valutare insieme all'evento principale (OR IF). Quando un dispositivo è coinvolto nella creazione di una regola, esso assume un comportamento diverso in base a dove viene inserito, coerentemente con il tipo di dispositivo: mentre le azioni supportano unicamente i dispositivi controllabili, gli eventi e le condizioni supportano dispositivi controllabili, eventi temporali e sensori. Le regole composte usando questa grammatica seguono un linguaggio naturale il più vicino possibile a quello

dell'utente.

Dello stesso argomento si sono occupati Nakagawa et al. [3], i quali hanno proposto la specifica di regole *ECA* con l'obiettivo di realizzare una piattaforma in cui gli sviluppatori possono facilmente creare algoritmi per realizzare regole sempre più complesse. Nakagawa et al. sostengono che la loro proposta fornisca flessibilità nel descrivere contesti arbitrari e che per far ciò sono importanti sia le "*semantics of context*" sia il "*composite context*". Per le semantiche del contesto hanno utilizzato le W4H, ovvero le cinque dimensioni semantiche di: identità (chi → *who*), posizione (dove → *where*), tempo (quando → *when*), attività (cosa → *what*) e profilo del dispositivo (come → *how*).
Il contesto composto viene costruito utilizzando eventi definiti dall'utente ed è formato da più contesti unitari (come ad es. periodi di tempo massimi, distanze da un luogo, budget, etc). Sono disponibili due topologie di regole *ECA*: JOIN-type, in cui dal contesto composto viene generata una azione, e FORK-type, in cui da una singolo contesto si diramano più azioni.

L'approccio *ECA* è stato utilizzato anche nel progetto (ultimato a Settembre 2013) della POSTECH, l'Università di Scienze e Tecnologia di Pohang, per la realizzazione di una *Energy-smart home* in Korea[4].

Una ricerca affine è quella di Garcìa-Herranz et al. [5], che si pone come obiettivo quello di permettere agli utenti di controllare e programmare il loro ambiente in modo uniforme e indipendente dai dispositivi usati. Nei loro studi precedenti ([6] e [7]) hanno sottolineato come un sistema *rule-based* rispondesse bene alla necessità di avere un linguaggio facile da usare per gli utenti e capace di descrivere e associare contesti e azioni. Inoltre, analizzando il problema del tempo, ovvero studiando quando è necessario controllare il sistema affinché questo funzioni nel modo previsto, hanno inizialmente optato per un approccio *event-based*: solo un cambiamento nell'ambiente poteva scatenare un altro cambiamento nell'ambiente stesso. Oltre a ciò, si sono occupati dell'invasività che un sistema automatico può avere sull'utente (cercando di perseguire la regola del "non disturbare") e della "modularizzazione" del sistema per permettere differenti personalizzazioni dell'ambiente. Nel loro linguaggio, ogni regola ha una struttura *ECA*, ma la loro regola può essere composta da un solo evento scatenante (*trigger*),

da più condizioni, e da una sola azione. Ad ogni regola è poi associato un peso variabile che serve a discriminare se bisogna attivare o disattivare la regola. A partire da questo linguaggio basato su regole per descrivere azioni associate al contesto, nel loro più recente lavoro [5] Garcìa-Herranz et al. hanno cercato di mantenere un linguaggio base semplice e, al contempo, di "isolare la complessità" per elementi che la richiedono, per esempio nel caso delle azioni che dipendono dal tempo. Per quanto riguarda il linguaggio base hanno condotto uno studio utente per misurare l'adeguatezza della struttura trigger-conditions-actions. Lo studio è stato fatto su un campione di 30 utenti di lingua spagnola sia con esperienza in programmazione e sia senza esperienza. Dai risultati emerge che:

- anche se le performance dei programmatori nella comprensione delle regole sono state migliori, non c'è stata una significativa differenza tra i 2 gruppi; le performance negative erano da attribuire alla difficoltà degli utenti a memorizzare i nomi degli elementi di una casa che non era a loro familiare;

- sia programmatori che non programmatori sono riusciti a distinguere, in modo abbastanza naturale, gli "eventi scatenanti" (*triggers*) dalle condizioni. Ma, poiché le parole spagnole per "when" e "if" sono semanticamente vicine (come lo sono in molte lingue), ciò può portare ad errori negli utenti che non hanno familiarità con linguaggi di programmazione.

Per quanto riguarda la complessità delle azioni che dipendono dal tempo, essi hanno isolato i vincoli temporali in uno speciale tipo di azione: il TIMER.

I TIMER sono usati per creare comportamenti complessi, definire eventi composti o eventi basati su politiche di consumo. I TIMER "isolano la complessità", riescono a dare flessibilità e potere espressivo, non interferendo con la semplicità richiesta dagli utenti. Un timer è composto da 4 differenti parti: *ending time, on-running rules, on-load rules e on-finished rules*. In figura 2.3 è mostrata la struttura del linguaggio base e la descrizione dell'estensione TIMER. Come si può notare in figura 2.3, i TIMER agiscono tra regole e non all'interno di regole. Da tutto ciò possono essere

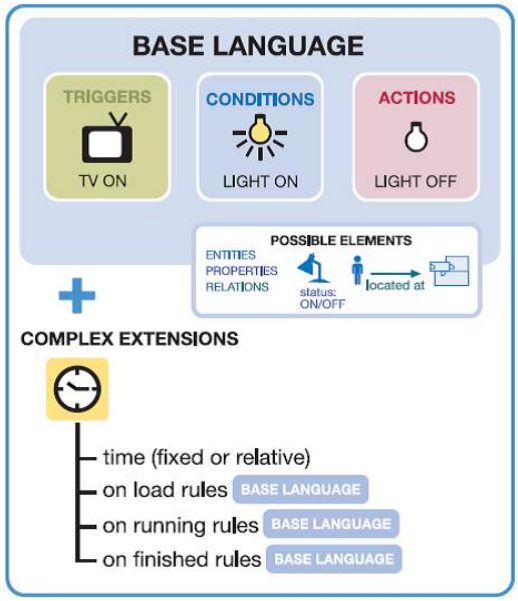

Figura 2.3. Struttura del linguaggio ed estensione TIMER del gruppo di ricerca di Garcìa-Herranz del Politecnico di Madrid.

create differenti interfacce utente per trattare diversi scenari di interazione e gradi di competenza. Al momento sono state implementate 3 tipi di interfacce per la creazione di regole :

- un'interfaccia *drag and drop*,
- una GUI semplice basata su parole chiave,
- un'interfaccia basata su menù.

Una recentissima ricerca della Brown University e della Carnegie Mellon University [8] ha mostrato che la programmazione *trigger-action*, ovvero le dichiarazioni "*if this, then that*" ("se questo, allora quest'altro") funzionano bene per la programmazione di applicazioni nelle *smart home*. In questo studio è stata valutata la praticità dell'uso di costrutti *"if . . . then . . . "*, ed in particolare:

- se questo tipo di programmazione riesce o meno ad esprimere la maggior parte dei comportamenti;

- se la personalizzazione è ancora necessaria per la programmazione di case intelligenti;

- se è facile per un utente medio imparare ad usare la programmazione *trigger-action* in pochi minuti e che tipo di complessità può avere l'interfaccia.

Analizzando i programmi realizzati da vari campioni di utenti e osservando il loro approccio a questo metodo, è stato dedotto che la programmazione riesce ad esprimere certamente la maggior parte delle richieste degli utenti riguardo a cosa vogliono che la loro casa faccia. Sorprendentemente è stato visto che nessuno dei comportamenti desiderati avrebbe richiesto un linguaggio di programmazione troppo complesso a patto che esistano *triggers* e *actions* sufficientemente espressivi. Inoltre né sesso, né età, né esperienza precedente di programmazione sono dati rilevanti nell'approccio alla programmazione. Ciò suggerisce che gli utenti possono essere facilmente istruiti ad utilizzare programmi per la gestione di *smart home*. E' stato appurato che, anche se le regole *"if...then..."* più comunemente utilizzate potrebbero essere inserite in un app store in modo da essere condivise, gli utenti preferiscono specificare essi stessi la combinazione desiderata piuttosto che ricercarla in un elenco. Inoltre un elenco di combinazioni potrebbe non essere esaustivo e non soddisfare i desideri della totalità degli utenti. Infine sottoponendo agli utenti due interfacce di complessità differenti ("*simple interface*" con un solo trigger e una sola azione; "*complex interface*" con trigger multipli e azioni multiple) non si sono rilevate differenze nell'accuratezza e nella velocità d'uso tra le due.

Sempre nell'ambito delle applicazioni per la definizione di regole per le *smart home*, un contributo interessate viene fornito da un recente studio di Lee et al. [9], in cui è definito il sistema GALLAG Strip. Così come iCAP, GALLAG Strip fornisce un facile strumento di programmazione visiva che permette a utenti che non hanno mai scritto del codice, di creare applicazioni *context-aware*. Esso però, a differenza di iCAP, è stato realizzato per ambiente mobile e ha la particolarità di essere uno strumento di programmazione tangibile, ovvero in esso l'interfaccia consente la programmazione anche attraverso l'interazione fisica degli utenti con sensori e oggetti. GALLAG Strip è anche un sistema di programmazione con dimostrazione, in altre parole l'utente mostra all'app ciò che egli vuole programmare; per far

ciò sono disponibili due modi: il *recording mode*, in cui il programma percepisce e registra gli eventi che accadono in base alle azioni dell'utente (Figura 2.4), e l'*edit mode*, che permette all'utente di modificare o aggiungere manualmente informazioni.

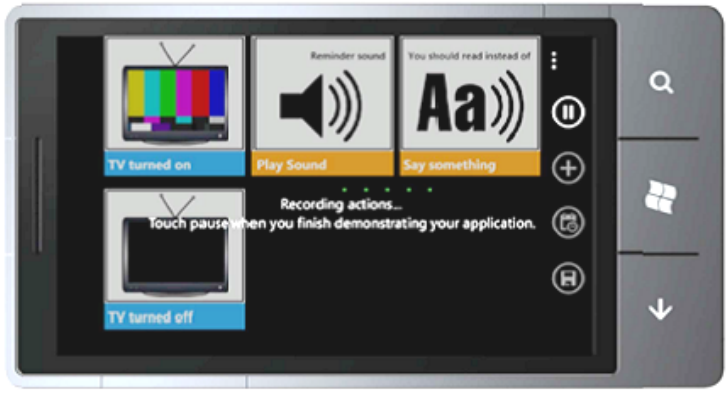

Figura 2.4. Interfaccia del sistema GALLAG Strip.

Un'applicazione GALLAG è rappresentata attraverso l'*application strip*, una sequenza di tre tipi diversi di frame: *action*, *response* e *time-data*. I frame "azione"(in blu in Figura 2.4) rappresentano le azioni dell'utente, ovvero gli eventi del modello *ECA*, i frame "risposta" (in arancione in Figura 2.4) rappresentano le azioni che il sistema deve compiere e sono settate dagli utenti, ovvero le azioni del modello *ECA*, i frame "tempo-data" indicano un particolare istante di tempo o una data in cui le azioni si devono eseguire. Test fatti sull'usabilità del sistema hanno evidenziato la capacità degli utenti di realizzare applicativi con complessità crescente in poco tempo e in alcuni casi gli utenti meno esperti nella programmazione sono stati più veloci di utenti più esperti. è stato condotto anche un esperimento per capire come l'approccio della programmazione tangibile può essere considerata dagli utenti rispetto ad altri approcci più tradizionali, quali interfacce con menù. Al gruppo di partecipanti al test, ingegneri e non ingegneri, sono stati quindi dati 3 diversi tipi di interfacce:

- *mobile-tangible* (MT), i partecipanti programmavano attraverso l'interfaccia GALLAG Strip vista prima;

- *mobile-menu* (MM), i partecipanti utilizzavano una GUI basata su menù, su un dispositivo mobile;

- *stationary-menu* (SM), i partecipanti programmavano attraverso una GUI basata su menù al PC e non potevano quindi muoversi nell'ambiente.

Dai risultati, per quanto riguarda la facilità d'utilizzo, tra gli ingegneri la MT ha ottenuto i punteggi migliori mentre la SM è stata ritenuta quella meno facile. Per i non ingegneri, invece, la MT risultava la soluzione più complessa rispetto a SM e MM. Inoltre la difficoltà riscontrata con le proposte MM e MT è stata la piccola dimensione dello schermo del dispositivo usato e la scomodità di tenere in mano il dispositivo nella fase d'interazione con i sensori soprattutto per i non ingegneri.

Figura 2.5. Esempi dei cubi di Media Cube.

Per quanto riguarda altri strumenti di programmazione tangibile nello sviluppo di applicazioni *context-aware* molti altri studi hanno preso piede nell'ultimo decennio. Tra questi, i meno recenti sono Media Cubes di Blackwell e Hague [10] e SiteView di Beckmann e Dey [11], mentre PiP (Pervasive interactive Programming) di Chin et al. [12], [13],[14], e HomeMaestro di Salzberg [15] risalgono a studi degli ultimi 5 anni.

Media Cubes è un linguaggio di programmazione in cui gli elementi sintattici sono dei cubi fisici che contengono superfici di rilevamento e di comunicazione in modo da consentire l'interazione con la casa e la percezione di altri cubi nelle vicinanze. Le facce di ogni cubo raffigurano (Figura 2.5) o le operazioni che è possibile svolgere o etichette con valori (per es. lo stato del dispositivo). I programmatori che usano Media Cube devono sistemare i cubi l'uno accanto all'altro e registrare la sequenza di operazioni.

L'interazione tra i dispositivi e i cubi può avvenire anche affiancando il cubo al dispositivo stesso.

SiteView è un sistema per la creazione e visualizzazione di regole per il controllo automatico. Esso utilizza un metodo di interazione tangibile e intuitivo per la composizione delle regole e mostra lo stato interno del sistema migliorandone la comprensione da parte dell'utente. Con SiteView la creazione della regola avviene manipolando degli "*interactors*", oggetti fisici concreti che permettono la rappresentazione delle condizioni rilevate (per es. pomeriggio) o delle azioni automatiche (per es luce accesa), e ponendoli all'interno del WIM (*world-in-miniature*) ovvero l'ambiente attivo. Un esempio di costruzione di una regola è illustrato nella figura 2.6.

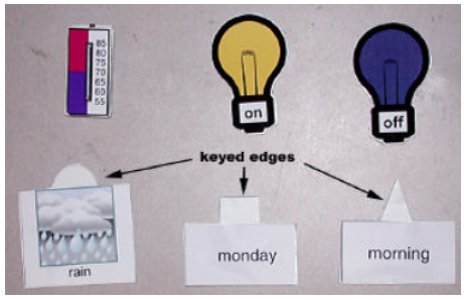

Figura 2.6. Costruzione di una regola con SiteView.

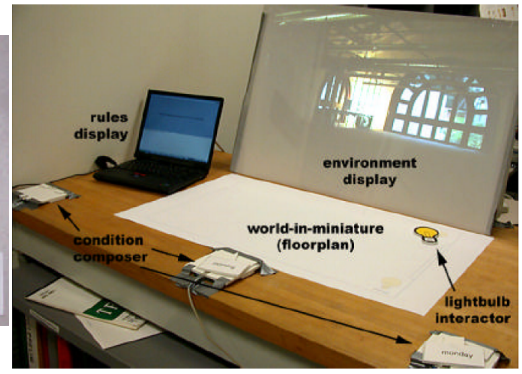

Figura 2.7. SiteView.

Per migliorare la trasparenza sono forniti all'utente, durante la programmazione, dei feedback espliciti sullo stato interno del sistema di controllo. Ciò avviene attraverso il *rule display*, che mostra le regole di controllo applicabili nelle condizioni in cui è l'utente, e attraverso l'*environment display*, che mostra come sarà l'ambiente sotto le condizioni e le azioni specificate in quel momento.
Come si può vedere nella figura 2.7, il sistema è quindi composto da *interactors*, *world-in-miniature* e *condition composer*, che forniscono supporto per la creazione della regola, ed *environment display* e *rule display*, per i feedback sullo stato del sistema.

PiP è un tool di programmazione end-user in cui gli utenti possono combinare i dispositivi e i servizi da questi offerti per creare i loro "dispositivi virtuali" chiamati MAps (*Meta Appliaces/Appliactions*). Una MAps ha delle proprietà primitive e contiene una collezione di regole che determinano il comportamento dei dispositivi collegati e quindi dell'ambiente. Le regole sono viste come unione di due differenti tipi di componenti:

- *Antecedent*, corrispondono agli eventi nel modello *ECA* e possono essere espresse tramite un "*if*";
- *Consequent*, corrispondono alle azioni nel modello *ECA* e possono essere espresse tramite "*then*".

Una regola può avere da 0 a n "antecedenti" e da 1 a n "conseguenti"; una MAps può contenere da 0 a n regole e queste possono essere combinate come si vuole anche senza seguire un ordine logico. In PiP le MAps vengono gestite tramite un'interfaccia *drag and drop* chiamata PiPView e l'utente può informare il sistema su quello che vuole fare in 3 differenti modi:

- attraverso l'interazione fisica con i dispositivi stessi
- usando una interfaccia utente su un pannello di controllo
- attraverso la combinazione dei 2 metodi precedenti.

Dai test effettuati PiP si è rivelato utile e facile da usare e si è visto che gli utenti in genere preferiscono l'interazione fisica con il sistema piuttosto che l'interfaccia statica.

HomeMaestro, invece, è una piattaforma per definire in modo intuitivo i comportamenti di elettrodomestici presenti in casa, permettendo agli utenti di automatizzare alcune operazioni quotidiane muovendosi nell'ambiente con l'uso di uno smartphone. Esso consiste essenzialmente di:

- un'interfaccia tangibile per impostare le regole che andranno ad attivare un particolare elettrodomestico;
- un "*rule marketplace*" dal quale si possono scaricare facilmente regole per la casa o condividere le proprie.

L'utente ha a disposizione due tipi d'interfacce, una mobile e una basata sul web. Egli può selezionare da una lista uno tra gli elettrodomestici presenti in casa in modo da scegliere, mediante un'altra lista, i comandi da fargli eseguire e le domande da sottoporgli (per esempio, ad un termostato, "qual è la temperatura corrente?"). Le regole possono essere create tramite un'interfaccia *drag and drop* oppure registrate muovendosi per la casa e interagendo con gli elettrodomestici. Una volta create, le regole vengono memorizzate ed è possibile successivamente disattivarle o cancellarle.
Nella modalità "*Record*" si registrano semplici regole nello stile "*When/Then*" e le componenti *when* possono essere combinate solo con la relazione booleana AND, non con OR oppure NOT. HomeMaestro è stato usato da Microsoft per testare la semplicità e l'intuitività del loro sistema HomeOS su Windows Phone.

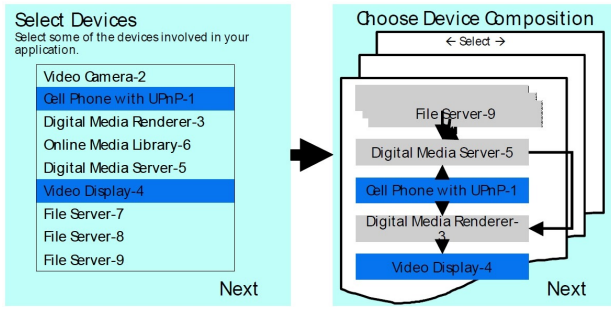

Figura 2.8. Framework del Nokia Research Center di Cambridge.

Della programmazione end-user che riguarda l'uso di dispositivi mobile in *smart home* si è occupato lo studio di Wisner e Kalofonos in [16]. Nel framework proposto l'utente, in una prima interfaccia, seleziona i dispositivi che vorrebbe poter usare contemporaneamente per creare un task (o manualmente o selezionando una composizione di dispositivi fornita dal sistema) e in seguito sceglie tra una lista di eventi e azioni ciò che intende fare con i dispositivi composti precedentemente (Figura 2.8). L'idea di fondo è quella di guidare l'utente nello specificare il comportamento desiderato.

L'approccio di seguire l'utente passo passo per la composizione della regola è in parte anche utilizzato da DiamondHelp [17][18]. Si tratta di una

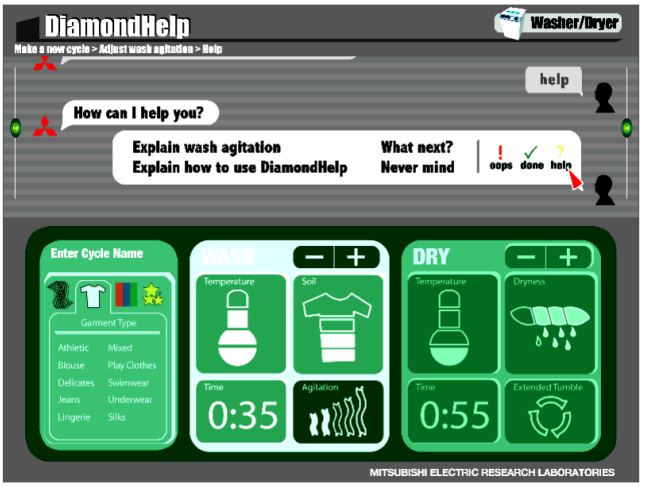

Figura 2.9. DiamondHelp.

interfaccia utente grafica e "intelligente" basata sul paradigma della *manipolazione diretta*. Il punto di partenza di DiamondHelp è la consapevolezza che esiste una differenza sostanziale tra funzioni semplici, di uso comune negli utenti, e funzioni più complesse, generalmente usate occasionalmente, che richiedono la programmazione e la personalizzazione dei dispositivi. La programmazione con DiamondHelp aiuta gli utenti soprattutto nel creare funzioni complesse come un termostato programmabile per un'intera settimana che si adatta alle esigenze di una famiglia. DiamondHelp si basa sul paradigma della "conversazione collaborativa" ovvero, in base alla conoscenza dell'utente, le istruzioni possono essere create in autonomia o tramite un "tutoraggio". L'utente può essere dunque aiutato attraverso istruzioni dettagliate fornite passo-passo dal programma, ma può scegliere in qualunque momento di proseguire da solo. L'interfaccia del programma (Figura 2.9) presenta infatti una sezione, con i fumetti in stile "chat online", da usare se si vuole un aiuto intelligente, e una sezione in cui si possono gestire autonomamente i propri dispositivi.

La Tabella 2.2 presenta un sommario delle soluzioni analizzate fino ad ora.

Nome	Modello *ECA*	Interfaccia tangibile	Mobile	Web/ Desktop	Altre informazioni
iCAP	parzialmente (EA)	no	no	sì	
e-Lite	sì	no	no	sì	
Nakagawa	sì	no	no	sì	uso delle cinque dimensioni semantiche W4H e due topologie di regole: Jointype e FORKtype
Garcìa-Herranz	sì	no	no	sì	aggiunta TIMER tra regole
GALLAG Strip	parzialmente (EA)	sì	sì	sì	applicazione vista come sequenza dei frame azione, risposta e tempodata
Media Cube	no	sì	no	no	
SiteView	parzialmente	sì	no	sì	quando un utente specifica un set di condizioni, il sistema mostra le regole che matchano con il set specificato

Continua alla pagina successiva...

Continua dalla pagina precedente...

PiP	parzialmente (EA)	si	sì	sì	
HomeMaestro	parzialmente (EA)	sì	sì	sì	modalità "record" con regole stile "when/then"
DiamondHelp	no	no	no	sì	paradigma della "conversazione collaborativa", le istruzioni possono essere fornite in autonomia o tramite tutoraggio

Tabella 2.2: Riepilogo caratteristiche sistemi di composizione regole

Capitolo 3

Soluzioni tecniche adottate

Prima di esporre nei dettagli le fasi di progettazione e di sviluppo dell'interfaccia realizzata è necessario introdurre le soluzione tecniche adottate. In questo lavoro di tesi è stato utilizzato un middleware[1] per *smart home*, sviluppato dal gruppo di ricerca e-Lite del Politecnico di Torino, chiamato *Dog*. Maggiori dettagli del sistema in questione sono presentati nel paragrafo successivo.
Come accennato in precedenza, l'interfaccia è stata realizzata per un dispositivo mobile avente come sistema operativo Android 4.x; alcuni dettagli implementativi si trovano nel paragrafo 3.2.

3.1 Sistema di riferimento

Per la realizzazione di un'interfaccia nel campo delle *smart home* è importante avere conoscenza dell'ambiente con cui l'applicazione deve interfacciarsi. Come già introdotto nel paragrafo 1.1.1, per permettere l'interazione di dispositivi diversi all'interno di un sistema intelligenti è necessario avere uno strato di software che permetta la cooperazione tra diverse tecnologie

[1] Il middleware è un *software* di connessione che consiste di un insieme di servizi e/o di ambienti di sviluppo di applicazioni distribuite che permettono a più entità (processi, oggetti ecc.), residenti su uno o più elaboratori, di interagire attraverso una rete di interconnessione a dispetto di differenze nei protocolli di comunicazione, architetture dei sistemi locali, sistemi operativi ecc ...

e diversi dispositivi all'interno dell'ambiente. Per questo lavoro di tesi è stato scelto il middleware *Dog*, sviluppato dal gruppo di ricerca e-Lite del Politecnico di Torino.

3.1.1 *Dog*

Dog è un middleware strutturato su un'ontologia (*DogOnt* [19]), esso sfrutta il *framework OSGi* per coordinare l'attivazione dinamica di moduli, per fungere da *hot-plug* per nuovi componenti e per reagire a guasti verificatisi [20]. Tali caratteristiche di base, insieme a *DogOnt*, permettono l'integrazione di reti diverse, la creazione di scenari *inter-network*, il supporto ad una intelligenza basata sulla logica e l'accesso a differenti dispositivi e applicazioni domotiche attraverso un'unica rappresentazione neutrale dal punto di vista tecnologico.

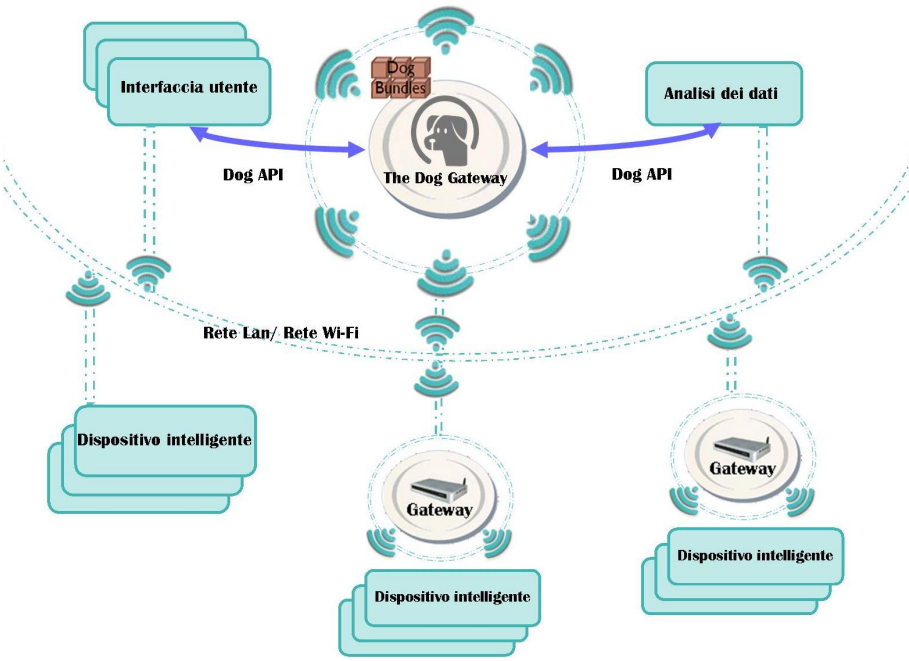

Figura 3.1. Rappresentazione di un ambiente intelligente che utilizza *Dog*.

Dog ha quindi due finalità:

1. trasformare dei protocolli di basso livello in delle comunicazioni *event-based* di livello applicativo, astraendo dall'effettiva tipologia di dispositivo,

2. fornire delle API grazie alle quali vari applicativi possono accedere all'ambiente intelligente secondo modalità ben definite.

In figura 3.1 la rappresentazione di un ambiente intelligente che utilizza *Dog*.
Tra i requisiti di progettazione rispettati da *Dog*, presentati in [21], vi sono:

- versatilità,
- supporto ad una intelligenza avanzata,
- accessibilità da applicazioni esterne.

Versatilità
La versatilità è stata ottenuta utilizzando, come già accennato, il *framework OSGi*. L'*OSGi Alliance*, precedentemente conosciuta come *Open Service Gateway initiative*, a partire dal 1999 ha cercato di indentificare una serie di specifiche per la realizzazione di una piattaforma di servizi basata su Java in grado di essere controllata da remoto. Oggi la piattaforma aperta *OSGi* rappresenta il principale standard per componenti software destinati a dispositivi collegati in rete, come telefoni mobile digitali, dispositivi embedded, gateway, PC desktop e server di fascia alta, mainframe compresi. Il fulcro di questa piattaforma è appunto il *framework OSGi*. Esso è un *framework* Java general purpose in cui i componenti, chiamati *bundle*, possono essere installati e modificati a *runtime*, e ciò lo rende un sistema altamente flessibile e con una elevata modularità. Il *framework OSGi* è inoltre specifico nel settore della domotica e permette quindi di utilizzare dei pattern robusti e già collaudati per risolvere alcune problematiche tipiche dei sistemi automatizzati.

Supporto ad un'intelligenza avanzata

Per ottenere l'intelligenza è necessario che il sistema abbia un modello di rappresentazione della conoscenza. *DogOnt* è il modello formale per la rappresentazione di un ambiente domotico, che consiste in un'ontologia, espressa in *OWL* (*Web Ontology Language*), il linguaggio standard nel settore del *Semantic Web*. Grazie alla rappresentazione tramite ontologie e a delle strutture adeguate al ragionamento, *DogOnt* è in grado di affrontare i problemi di interoperabilità tra dispositivi, consentendo di descrivere:

- dov'è situato un dispositivo domotico;
- le capacità e le funzionalità di un dispositivo;
- le specifiche tecnologiche necessarie per interfacciarsi con un dispositivo;
- le possibili configurazioni che un dispositivo può assumere;
- come è strutturata l'abitazione (piani, stanze, muri);
- gli elementi architetturali e di arredamento presenti nell'ambiente.

Queste informazioni possono poi essere sfruttate da sistemi basati sull'inferenza per fornire le funzionalità avanzate richieste al *smart home*. L'intelligenza del sistema è quindi realizzata modellando formalmente l'ambiente, attraverso l'ontologia di *DogOnt*, e definendo meccanismi di ragionamento appropriati.

Accessibilità da applicazioni esterne

L'accessibilità da parte di sistemi esterni è fornita attraverso delle API che permettono l'interazione con varie interfacce utente e con eventuali sistemi di analisi dei dati. Le API fornite sono sia per applicazioni basate su REST ma anche per applicativi che utilizzano *WebSocket* [20].

3.1.2 Comunicazione con *WebSocket*

Le *WebSocket* rappresentano una tecnologia che consente una comunicazione bidirezionale e full-duplex tra un client e un server. Essa viene utilizzata

per effettuare comunicazioni bidirezionali in tempo reale ed infatti prevede l'esistenza di un canale di comunicazione sempre attivo, a bassa latenza utilizzabile da entrambi sia in scrittura che in lettura. L'utilizzo del protocollo *WebSocket* lato client è possibile attraverso specifiche API. Esse consentono di aprire e chiudere una connessione, ricevere e inviare messaggi, e reagire in seguito a particolari eventi. Il *middleware Dog*, nel suo livello di comunicazione con l'esterno, include i bundle che defiscono le interfacce di programmazione (indipendenti dalla tecnologia) per permettere l'accesso ai servizi forniti da *Dog* ad applicazioni esterne. L'interfacciamento può avvenire sia per mezzo di un endpoint REST oppure tramite WebSocket, attraverso le due API presenti in questo livello, rispettivamente *API REST* e *API WebSoket*. Nell'applicazione sviluppata per questo lavoro di tesi è stata utilizzata l'*API WebSocket*. Questa API è in grado di fornire una connessione asincrona e bidirezionale tra il *gateway* e gli end-user devices (per es. tablet e smartphone) e inoltre si basa su un protocollo standard per il Web ampiamente diffuso in grado di fornire supporto a buona parte delle piattaforme software esistenti. Le *API WebSocket*, così come le *API REST*, possono permettere di:

1. recuperare la configurazione dell'edificio;

2. inviare comandi ai dispositivi connessi a *Dog*;

3. gestire le informazioni sulla struttura dell'edificio, ovvero la divisione in camere, appartamenti, ecc.;

4. acquisire lo stato dei dispositivi.

Inoltre, poiché la tecnologia *WebSocket* fornisce una comunicazione *full-duplex*, le relative API permettono anche la gestione di eventi asincroni (chiamati comunemente "notifiche"[2]) che possono provenire da qualsiasi dispositivo gestito da *Dog*.

[2]Le notifiche sono dei feedback trasmessi dal sistema ogni volta che, all'interno dell'ambiente intelligente, viene eseguita un'operazione o un comando.

3.2 Android

Per lo sviluppo dell'applicazione destinato ai terminali mobile, è stato utilizzato Android 4.x. Android è una piattaforma *Open-Source*, realizzata da Google, per dispositivi mobile ([22]). E' un vero e proprio stack che comprende diverse componenti, dal sistema operativo (Linux) ad una *virtual machine*, chiamata *Dalvik*, per l'esecuzione delle applicazioni. Più di recente, nella versione Android 4.4 *KitKat* è stato introdotto, in via sperimentale[3] un nuovo *runtime system software*: *Art* (*Android RunTime*); *Art* ha sostituito definitivamente la *virtual machine Dalvik* con il rilascio di *Android 5.0 Lollipop*. Vista la politica *open-source* adottata, gli sviluppatori possono liberamente studiare e apportare modifiche a tutti gli strumenti della piattaforma stessa e quindi le applicazioni non sono soggette ad alcun tipo di limitazione. Inoltre un'altra caratteristiche importanti di Android è quella di permettere agli sviluppatori di applicazioni mobile di poter adattare l'applicazione, oltre che sugli Smartphone, anche su Tablet, e-reader, TV e Media Player. Queste peculiarità, insieme ad altri fattori commerciali, hanno permesso ad Android di diventare il sistema operativo più utilizzato in ambito mobile.

3.2.1 Caratteristiche generali di un'app

Un'applicazione Android si compone di due parti:

1. una parte dinamica costituita da codice, scritto in Java, per la gestione degli eventi.

2. una parte statica, scritta in XML, che definisce le caratteristiche che non cambiano durante l'esecuzione dell'applicazione, come la disposizione del testo.

Essa è un' applicazione *Java-based* in quanto si sviluppa in Java e viene compilata in un insieme di risorse e *bytecode* (un "codice intermedio") Java, ma il *bytecode* finale e la *virtual machine* sulla quale verrà eseguito non è Java. La *Dalvik Virtual Machine* utilizzata è stata progettata e realizzata

[3]Si può abilitare il runtime manualmente attraverso l'apposita funzione in Settings > developer options.

con il principale obiettivo di essere ottimizzata per dispositivi mobile, essa infatti permette una maggiore ottimizzazione della memoria in dispositivi con bassa capacità. La *Dalvik VM* non esegue *bytecode* Java ma un qualcosa che si può ottenere da esso e che prende il nome di *Dalvik bytecode*. Le applicazioni per Android si sviluppano in Java sfruttando i tool di sviluppo classici come *Eclipse*, *NetBeans* e ora *Android Studio* per poi trasformare il *bytecode* Java in *Dalvik bytecode*. Su un dispositivo Android non girerà alcun *bytecode* Java ma un *bytecode* le cui specifiche sono descritte dal formato DEX (*Dalvik EXecutable*). Per realizzare l' applicazione è stata utilizzata la piattaforma integrata *Eclipse* con l'installazione del ADT (*Android Developer Tools*). Dal punto di vista dello sviluppo di una app, gli elementi fondamentali forniti da Android sono:

- *activity*,
- *intent* e *intent filter*,
- *broadcast intent receiver*,
- *service*,
- *content provider*.

Una *activity* corrisponde a una schermata di un'applicazione di Android, ossia un qualcosa che visualizza delle informazioni o permette l'inserimento di dati da parte dell'utente. La creazione delle *activity* avviene attraverso la descrizione di *view* e la definizione delle modalità con cui le diverse view si passano le informazioni.

Gli *intent* e gli *intent filter* svolgono un ruolo importante nel permettere uno sfruttamento migliore delle risorse e il riutilizzo di operazioni comuni a più applicazioni. Gli *intent* vengono utilizzati nella comunicazione tra più *activity*. Ogni *activity* può dichiarare un insieme di *intent* che è in grado di esaudire attraverso gli *intent filter*. Gli *intent filters* sono un metodo per mostrare al resto del sistema le azioni che un'applicazione può compiere in modo da poterle fare utilizzare a qualunque altra applicazione, essi offrono numerosi benefici tra cui:

- Massimizzare il riutilizzo e la modularità dei componenti. Le applicazioni infatti si possono specializzarsi su servizi singoli e in caso di necessità interagire tra loro per fornire un servizio migliore.

- Adattare al meglio le applicazioni alle esigenze dell'utente. Una applicazione che richiede un determinato servizio può non essere a conoscenza delle applicazioni correntemente installate sul dispositivo in quanto sarà il meccanismo degli *Intent Filters* a individuare i programmi che potranno accogliere la richiesta dell'utente e far decidere all'utente quale applicazione utilizzare.

Un *broadcast intent receiver* è la componente del sistema Android in grado di reagire a determinati eventi come per esempio attivare un'applicazione, visualizzare una notifica, iniziare a vibrare o altro ancora. Esso è in grado di attivarsi a seguito del lancio di un particolare *intent* che è detto appunto di broadcast.

I *service* vengono utilizzati quando è necessario implementare delle funzionalità *long running* (di esecuzione prolungata) non direttamente legate ad aspetti visuali. Essi sono quindi dei componenti in grado di garantire l'esecuzione di alcuni task in background in modo indipendente da quello con cui l'utente sta interagendo in quel momento e quindi da ciò che è visualizzato nel display.

Un *content provider* è un contenitore di dati condivisibili globalmente tra più applicazioni. Esso permette di offrire ai propri client un'interfaccia per l'esecuzione delle operazioni di CRUD (*Create*, *Retrieve*, *Update*, *Delete*) su un particolare insieme di entità. Essendo Android un sistema Linux-based, ogni app ha un suo userid, la sua directory "data" (/data/data/nomepackage) e un suo spazio protetto di memoria. Per questo motivo gli applicativi Android hanno bisogno dei *content provider* per comunicare tra loro. I processi possono segnalarsi al sistema come *content provider* di un insieme di dati. Quando le informazioni sono richieste, sono richiamate da Android grazie ad un insieme specifico di API che permettono di agire sul contenuto secondo delle specifiche predefinite.

Capitolo 4

Progettazione fase 1: prototipazione

L'accettazione di un programma da parte degli utenti dipende, molto spesso, dalla qualità delle interfacce grafiche messe a disposizione dell'utilizzatore. Molti sono i fattori da considerare al fine di realizzare un'interfaccia utente di buona qualità e, a tale scopo, la fase di progettazione rappresenta una delle fasi fondamentali. È in questa fase, infatti, che vengono individuati i requisiti che il programma che si sta per realizzare deve possedere e si stabilisce cosa esso deve fare e come si deve fare. In questo capitolo, saranno esposti i passaggi che hanno portato alla scelta e alla realizzazione dell'interfaccia utente creata. In particolare, il paragrafo 4.1 tratta i requisiti generali che si sono scelti per il sistema in esame; nel paragrafo 4.2 vengono descritti due iniziali prototipi di interfaccia; nel paragrafo 4.3 vengono introdotte altre scelte progettuali comuni ai due prototipi e che sono state poi implementate in questo lavoro di tesi.

4.1 Requisiti generali

Per rispondere alle esigenze di tutti gli utenti, in particolare i meno esperti, e minimizzare le difficoltà di utilizzo è necessario che l'applicazione soddisfi alcuni requisiti. I requisiti, nell'ambito della progettazione di applicazioni, riguardano un qualcosa che il prodotto deve fare o una caratteristica che esso deve possedere. In particolare, per la realizzazione dell'applicazione oggetto

di questa tesi, si è cercato di utilizzare una grammatica semplice e intuitiva per la creazione delle regole (descritta nel successivo paragrafo 4.1.1) e si è cercato di realizzare un'interfaccia mobile chiara, facile e piacevole da usare (i dettagli nel paragrafo 4.1.2).

4.1.1 Grammatica delle regole

Come messo in evidenza nel capitolo 2, nell'ambito delle *smart home*, la costruzione delle regole attraverso la sintassi "*if... then...*", con la distinzione tra eventi (o *trigger*), condizioni e azioni (ovvero una sintassi *ECA*), si è rivelata di facile comprensione e utilizzo anche per utenti non esperti. La scelta riguardante la grammatica da utilizzare è ricaduta quindi su una sintassi di questo tipo e, in particolare, è stata utilizzata quella ideata dal gruppo di ricerca e-Lite ([2], [20]) con alcune modifiche.

In Figura 4.1 è schematizzata la grammatica su cui si basa la creazione di una regola. Questa grammatica [20] si fonda su 3 parole chiavi fisse (di colore rosso in Figura 4.1): *IF*, *THEN*, *WHEN*. Le prime due sono obbligatorie mentre la terza è opzionale. La parola chiave *IF* introduce l'evento che farà scattare la regola. La parola chiave *WHEN* definisce un vincolo che limita l'evento, il vincolo che deve essere soddisfatto viene indicato come condizione. La parola chiave *THEN* esprime l'azione da eseguire al verificarsi delle condizioni specificate precedentemente.

In Figura 4.1 sono presenti anche altre 3 parole chiavi: *OR IF*, *AND WHEN*, *AND THEN* (di colore blu in Figura 4.1), le quali possono permettere l'aggiunta di altri blocchi "event", "condition", "action". La parola chiave *OR IF* può essere utilizzata per ripetizioni dell'*IF*; ciò permette la composizione di regole che presentano delle disgiunzioni. In una regola possono esistere più vincoli che limitano l'evento o gli eventi, in tal caso si avranno più condizioni da soddisfare simultaneamente; in Figura 4.1 l'aggiunta di condizioni multiple avviene nel loop attorno alla parola chiave *AND WHEN*. Infine è possibile far eseguire contemporaneamente più azioni; in Figura 4.1 l'aggiunta di azioni avviene nel loop attorno alla parola chiave *AND THEN*.

Per fare in modo che, utilizzando questa grammatica, le regole formate siano sempre valide, ogni dispositivo coinvolto nella creazione di un regola ha un comportamento differente a seconda del blocco in cui è inserito. Quindi per mantenere una certa coerenza nella regola si ha che:

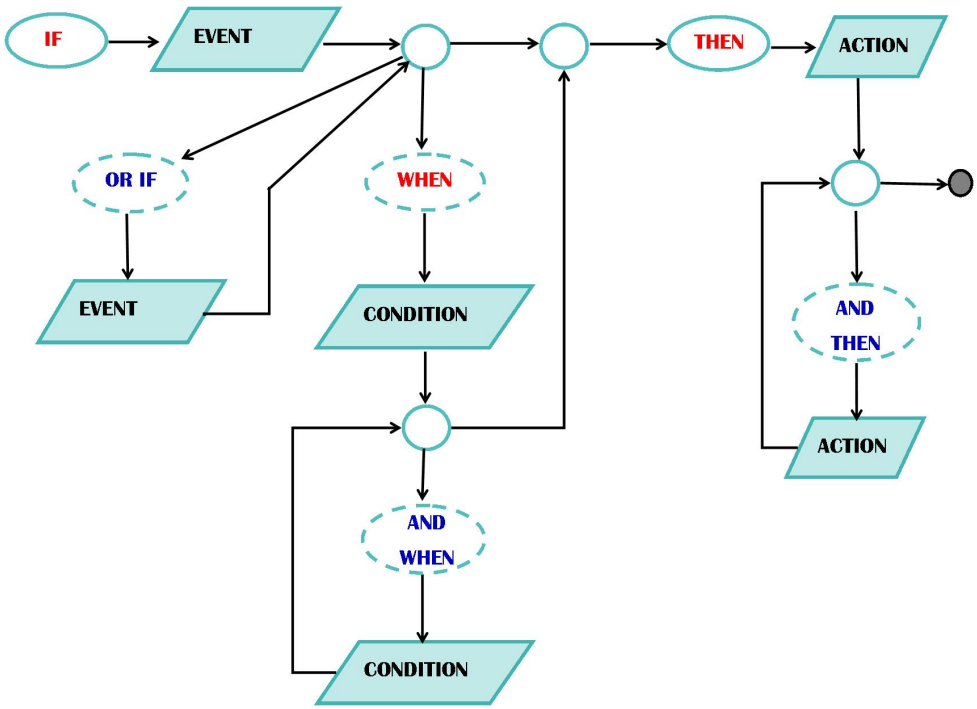

Figura 4.1. Schema della grammatica utilizzata.

- il blocco “evento” interpreta gli eventi generati da dispositivi controllabili, orologio e sensori.
- il blocco “condizione” accetta solamente dispositivi controllabili, non accetta quindi i sensori.
- il blocco “azione” supporta dispositivi controllabili, orologio e sensori.

Questa grammatica fa sì che la composizione di una regola risulti semplice anche agli utenti con un livello base di alfabetizzazione informatica; le conoscenze che servono, infatti, sono solo quelle riguardanti i dispositivi della casa, ad esempio come utilizzare un determinato elettrodomestico. Il linguaggio utilizzato è molto vicino al linguaggio parlato dagli utenti e quindi le regole create sembrano essere auto-esplicative. Inoltre una struttura di

questo tipo sembra essere sufficientemente espressiva per permettere la gestione della maggior parte delle situazioni che un utente potrebbe delegare al sistema.

4.1.2 Requisiti di progetto per una GUI Android

L'utilizzo di strumenti di modellazione, come quelli forniti dal *plugin ADT*, facilita, sul piano operativo, la creazione di interfacce utente. Essi infatti permettono la creazione di schermate anche sofisticate con poche operazioni. Tuttavia i problemi in questo ambito sono dovuti a motivi più strutturali. Un primo motivo è quello di costruire un'interfaccia che sia al tempo stesso gradevole, funzionale, coerente, facile da apprendere e da usare in contesti operativi reali. Per ottenere un risultato soddisfacente, occorre progettare con attenzione il flusso informativo e l'organizzazione grafica dei contenuti. Nella creazione dell'interfaccia utente si è cercato di fare attenzione a soddisfare queste caratteristiche, cercando di analizzare le scelte fatte alla luce di tre parole chiave:

- *facilità d'uso* - l'interfaccia deve permettere all'utente di esprimersi nel modo più naturale possibile;
- *intuitività* - l'interfaccia deve richiedere all'utente un basso sforzo nell'apprendere l'utilizzo;
- *chiarezza* - l'interfaccia deve permettere all'utente di trovare con immediatezza tutto quello che cerca all'interno di essa.

L'unione di queste tre caratteristiche permette di soddisfare anche la caratteristica di *usabilità*, primo obiettivo di una progettazione centrata sull'utente. Cercando il più possibile di soddisfare questi requisiti, inizialmente sono stati creati quattro *mockup*[1] (gli schizzi sono mostrati in Figura 4.2) e, successivamente, dalla fusione delle loro caratteristiche, si è arrivati a

[1] Un *mockup* è una rappresentazione statica del progetto da realizzare; che ha la funzione di rappresentare nel dettaglio i vari contenuti, dimostrare le funzionalità base in maniera statica e mostrare anche il lato grafico del progetto. A differenza del prototipo, essendo un'immagine statica, è più veloce da realizzare; la differenza con un prototipo cartaceo consiste nella quantità di dettagli presenti.

crearne due. Si sono quindi realizzati i prototipi cartacei delle due interfacce scelte, che verranno di seguito descritte, e si è fatto un test utente per individuare quale tra le due risultasse la più facile, chiara e intuitiva.

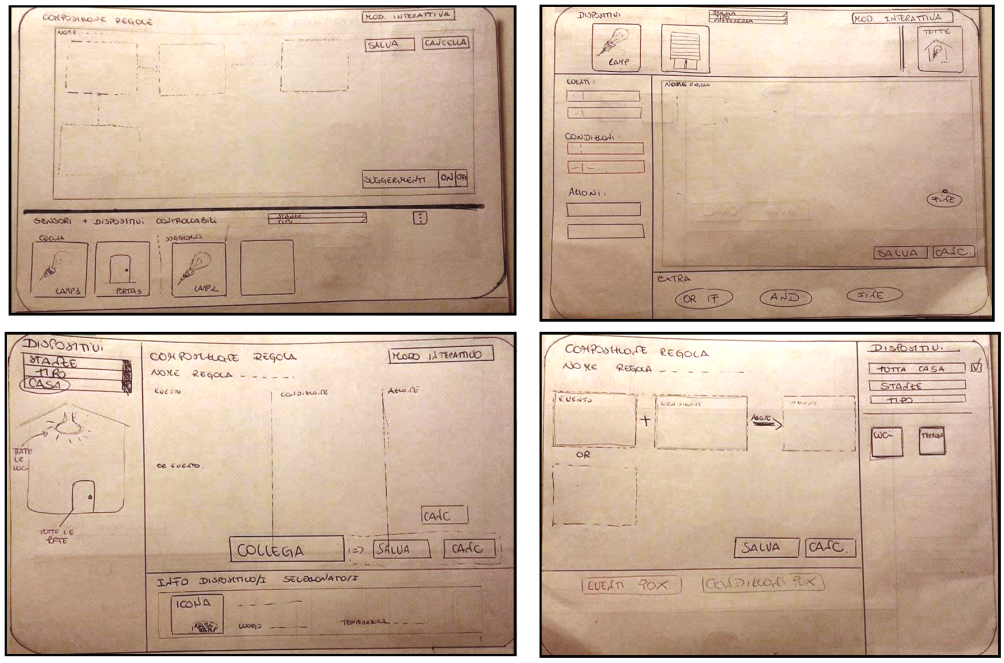

Figura 4.2. I 4 *mockup* iniziali.

Un altro problema nella creazione di GUI Android potrebbe essere legato al fatto che i terminali mobile variano ampiamente in termini di:

- *dimensioni fisiche*, quindi variano di conseguenza la leggibilità e l'interattività;
- *risoluzione dello schermo*, varia quindi la quantità di informazione complessivamente presentabile;
- *orientamento*, varia quindi la struttura delle pagine e l'organizzazione logica dei contenuti.

Queste differenze fanno sì che non sia possibile realizzare una singola interfaccia e adattarla ad ogni tipologia di dispositivo. L'interfaccia realizzata

nel lavoro di tesi è ottimizzata per essere visualizzata su un tablet da 10" con orientamento *portrait*. Tuttavia la scelta progettuale di utilizzare i *fragment* è stata ideata per consentire di apportare facilmente modifiche alla struttura dell'interfaccia e permettere possibili configurazioni alternative per gestire al meglio la diversità dei vari dispositivi.

4.2 Prototipazione: due modalità di interazione

La prototipazione consente di generare e organizzare le idee su come un'interfaccia utente può essere progettata, valutando in anticipo la qualità di una soluzione, simulando comportamenti e funzionalità di un prodotto che può essere completamente diverso da quello finale. Essa viene vista (ISO 13407: *Human-centered design process* [23]) come un punto fondamentale della progettazione *user-centered*. Secondo la norma ISO 13407 un prototipo è "*una rappresentazione di un prodotto o di un sistema, o di una sua parte, che, anche se in qualche modo limitata, può essere utilizzata a scopo di valutazione*" e quindi la creazione di prototipi è importante per:

- avere decisioni di progetto più esplicite;
- permettere l'esplorazione di diversi *design concept*;
- ottenere dei *feedback* nelle fasi preliminari;
- permettere una più semplice valutazione di progetti alternativi;
- testare la qualità e completezza delle specifiche.

Sebbene non sia molto intuitivo testare un'interfaccia utente senza l'uso del dispositivo su cui l'interfaccia verrà realizzata, lo scopo dei prototipi su carta è quello di massimizzare il *feedback* minimizzando lo sforzo che si farebbe se fosse portata a termine un'interfaccia che non è usabile. Usando i prototipi cartacei si possono effettuare dei test utente per avere informazioni sul "successo" dell'interfaccia e scegliere con più accuratezza ciò che si dovrà implementare. A tale scopo, nella prima fase della progettazione sono stati realizzati dei prototipi cartacei delle interfacce in esame ed è stato somministrato un test di usabilità a 12 partecipanti. Di seguito sono descritti i due prototipi di interfaccia realizzati. La sostanziale differenza tra le due interfacce consiste nella disposizione degli elementi all'interno dell'interfaccia e

nella modalità di selezione dei componenti grafici: un'interfaccia permette la composizione mediante trascinamento, l'altra mediante tocco. Di seguito la prima interfaccia sarà indicata con l'espressione "*interfaccia drag & drop*" mentre la seconda con l'espressione "*interfaccia click*".

4.2.1 Interfaccia *drag & drop*

Il primo prototipo di interfaccia realizzato è stato pensato per utilizzare come modalità di interazione con l'utente il trascinamento, meglio conosciuto con il nome di "*drag and drop*". Il *drag and drop* consiste nel cliccare su un oggetto virtuale (quale ad esempio un'icona) per trascinarlo (in inglese: *drag*) in un'altra posizione, dove viene rilasciato (in inglese: *drop*).

La figura 4.3 mostra il prototipo cartaceo dell'interfaccia utilizzato per il test di usabilità.

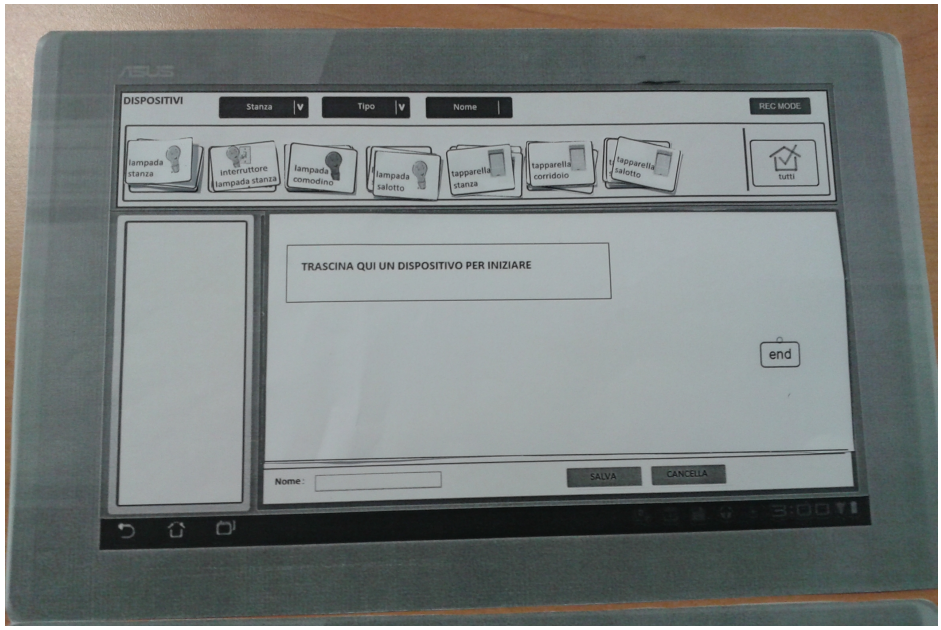

Figura 4.3. Prototipo cartaceo dell'interfaccia di composizione mediante l'operazione di *drag & drop*

In questa interfaccia i dispositivi sono raffigurati con un'icona e un nome; sono visibili in alto e possono essere raggruppati per stanza, per tipologia

di oggetto oppure in ordine alfabetico. L'icona "TUTTI", in alto a destra, è stata pensata per dare l'opportunità di selezionare tutti i dispositivi della *smart home*. L'area centrale dell'interfaccia è vista come la zona di composizione della regola, il posto dove l'utente può trascinare i vari elementi per creare la sua regola.
L'utente sceglie il dispositivo e lo trascina nell'area centrale. A questo punto nella colonna di sinistra compaiono alcuni blocchi che contengono eventi, condizioni e azioni possibili per il dispositivo selezionato. I blocchetti avranno:

(a) un colore diverso a seconda che siano eventi, condizioni o azioni;

(b) una descrizione;

(c) un'immagine che esprima in modo illustrato il significato del blocchetto.

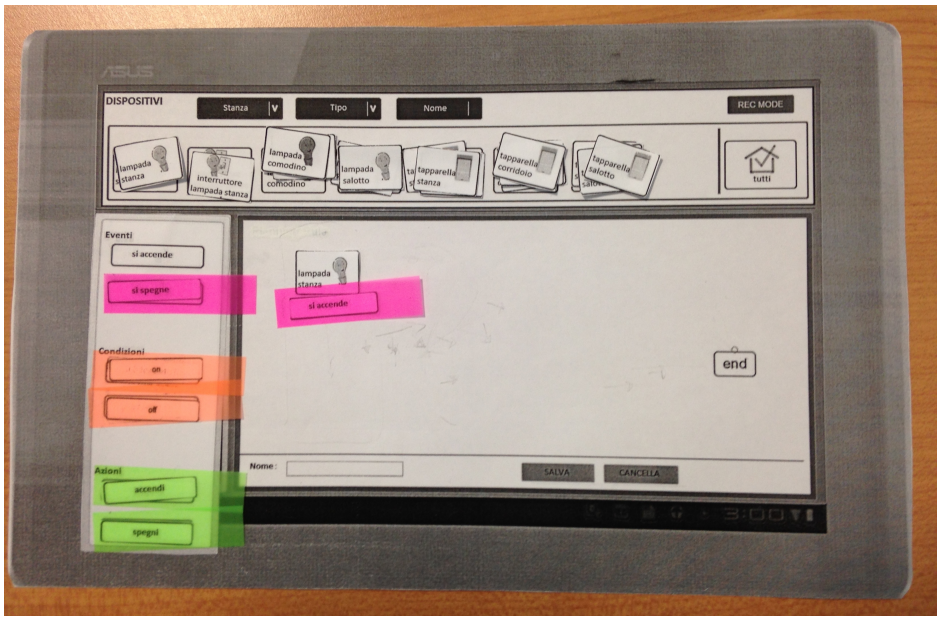

Figura 4.4. Esempio di utilizzo del prototipo cartaceo dell'interfaccia *drag & drop*

Trascinando il blocchetto evento/condizione/azione sopra l'icona del dispositivo posto nella zona di composizione e rilasciandolo in quel punto, si creerà in automatico un collegamento tra il dispositivo e il blocco aggiunto, inoltre verrà inserito un pallino di uscita dal blocco che poi l'utente utilizzerà per unire tanti blocchi e creare la regola.
Nella figura 4.4 vi è un esempio della scelta di un evento, quale "si accende", associato al dispositivo "lampada stanza". Successivamente, quando l'utente avrà posto tutte le componenti della regola nella zona di composizione, si dovranno collegare i vari blocchetti e legare il blocchetto dell'azione al blocco "FINE". Per collegare più blocchi eventi si dovrà utilizzare il blocco "OR", per collegare azioni o condizioni il blocco "AND". Per quando riguarda i blocchetti "OR" e "AND", essendo questi intrinsecamente legati alla grammatica utilizzata, si è pensato di evitare all'utente l'aggiunta manuale del blocchetto esatto e di inserirli in automatico quando l'utente collega tra di loro due o più blocchi della stessa tipologia. Per disegnare il collegamento tra due blocchi basta toccare sul cerchietto presente nel blocco, a questo

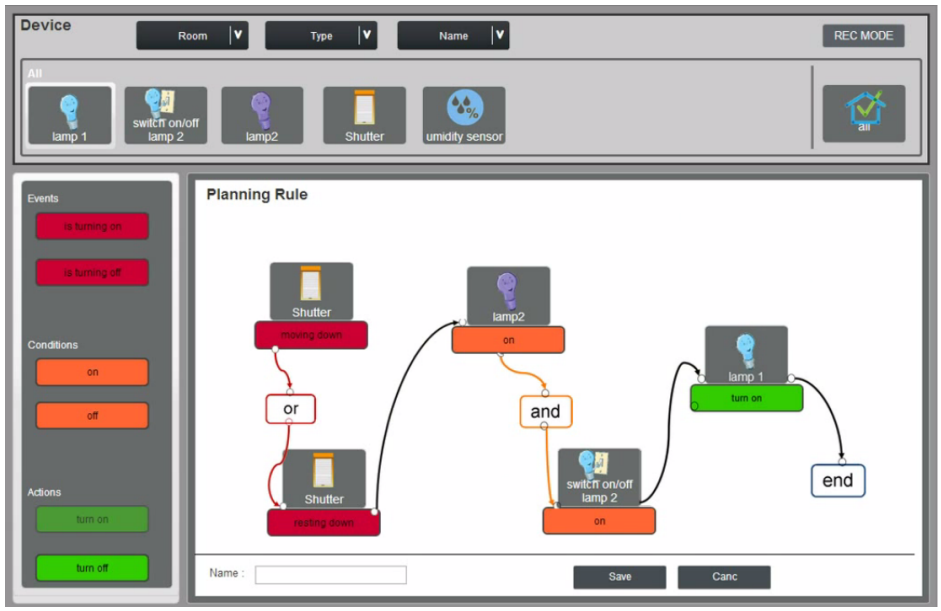

Figura 4.5. Esempio di regola completa composta con il prototipo dell'interfaccia *drag & drop* realizzato al PC.

punto comparirà una freccia la cui testa deve essere collegata a un altro blocco. Alla fine la regola sarà simile a una catena di composizione che inizia con un blocco evento e finisce con un "FINE".
In figura 4.5 è mostrato uno *screenshot* del prototipo *drag & drop*, realizzato al PC, in cui è mostrato l'output finale della composizione.

4.2.2 Interfaccia con l'operazione di selezione

Il secondo prototipo di interfaccia realizzato utilizza un'altra modalità di interazione con i contenuti dell'interfaccia, la selezione. L'operazione di selezione corrisponde alla più nota operazione di *click*, ovvero l'operazione di premere il pulsante del mouse; in questo caso, avendo un dispositivo *touch*, l'operazione è quella di far *touch* sull'elemento opportuno.
La figura 4.6 mostra la schermata iniziale sul prototipo cartaceo utilizzato per il test di usabilità.

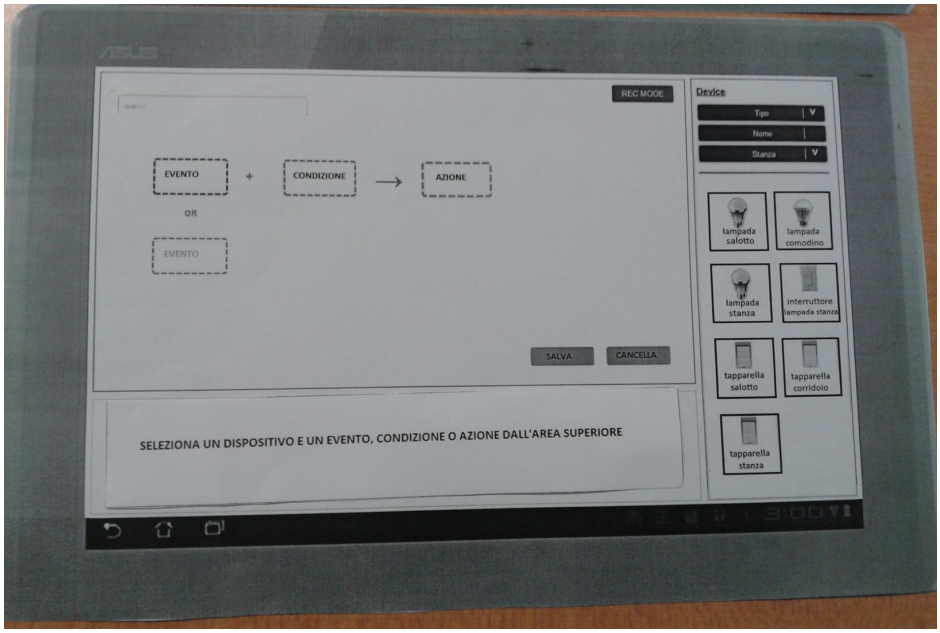

Figura 4.6. Prototipo cartaceo dell'interfaccia di composizione mediante l'operazione di selezione

L'interfaccia presenta nella zona centrale lo schema della regola tratteggiato e nella di parte destra i dispositivi; in una fase iniziale la parte di destra risulta disabilitata. Quando l'utente clicca su una regione tratteggiata, ad esempio quella per l'evento, viene abilitata l'area di destra ed egli può scegliere un dispositivo. A questo punto, nella sezione in basso, compariranno gli eventi possibili per quel dispositivo. L'utente sceglie cosa fare e nella zona di composizione, al posto del tratteggio, comparirà una "scheda" che contiene un'icona e la descrizione dell'evento. Sotto la scheda dell'evento appena scelto verrà mostrata un'altra regione tratteggiata per permettere l'aggiunta di un ulteriore evento. Come si nota dalla figura 4.7, dove è mostrata l'aggiunta dell'evento "la tapparella del corridoio si sta alzando", sotto la scheda dell'evento inserito vi è la parola chiave "OR"(OPPURE) per indicare che l'aggiunta di eventi può avvenire solamente in disgiunzione.

Successivamente l'utente può cliccare sul rettangolo della condizione; il

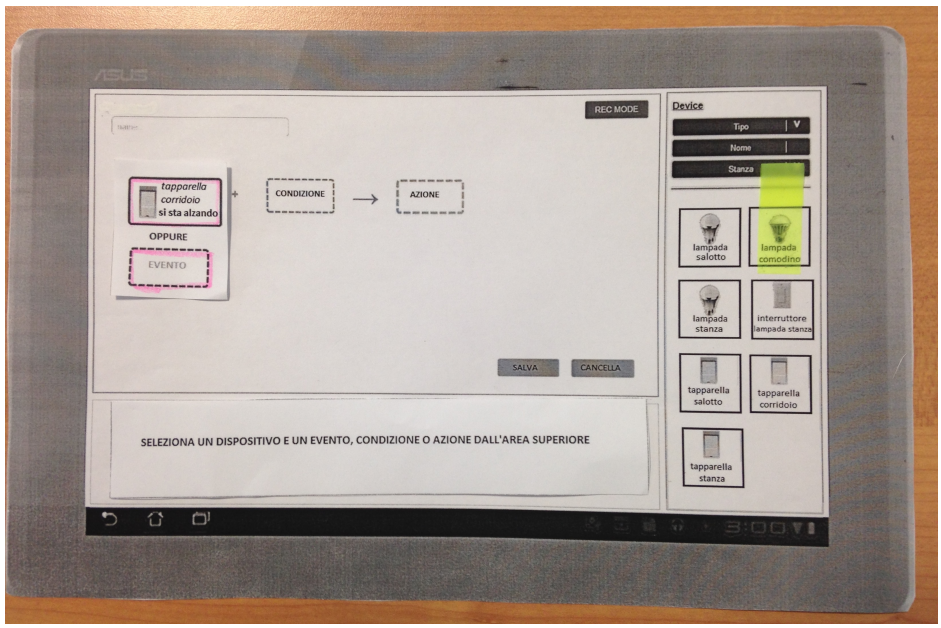

Figura 4.7. Esempio di utilizzo del prototipo cartaceo dell'interfaccia *click*

processo è uguale a prima ma nello spazio in basso compariranno le condizioni possibili per il dispositivo selezionato, stessa cosa succede se si vuole aggiungere un'azione. Quando l'utente sceglie la condizione desiderata, si crea la "scheda" della condizione e in questo caso, come anche nel caso delle azioni, allo schema tratteggiato si aggiungerà in automatico un ulteriore + in verticale sotto le condizioni e un altro rettangolo tratteggiato per aggiungere altre eventuali condizioni/azioni. In figura 4.8 è mostrato uno *screenshot* di una regola realizzata con il prototipo al PC dell'interfaccia *click*. Come si può notare, la regola creata ha due eventi legati in "OR", due condizioni legate dal segno "+" (in "AND") e un'azione.

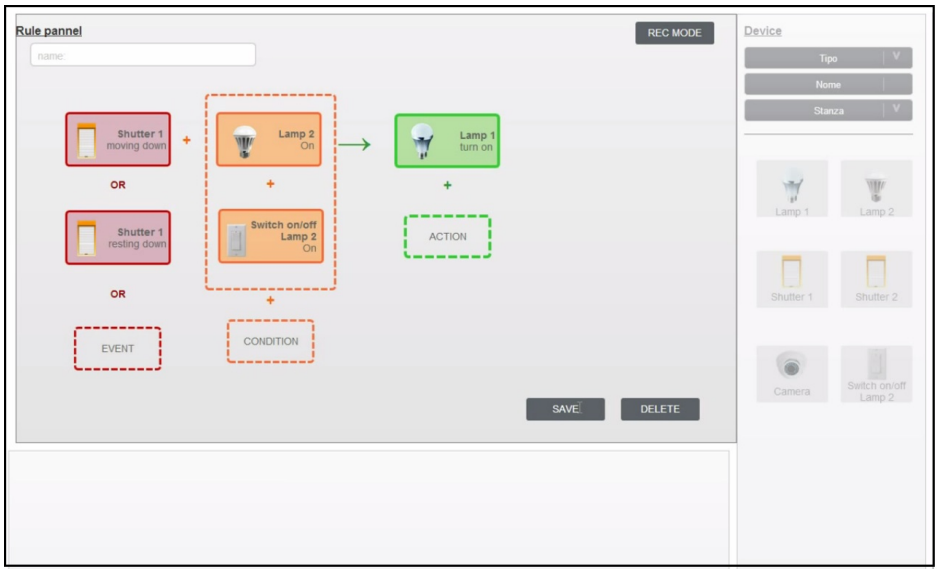

Figura 4.8. Esempio di regola completa composta con il prototipo dell'interfaccia *click* realizzato al PC.

4.3 Ulteriori scelte progettuali

4.3.1 Introduzione alla modalità “registrazione interattiva”

Dalle figure precedenti si può notare la presenza del tasto “REC MODE” nella zona di composizione. Esso è stato pensato per permettere la registrazione degli eventi/condizioni/azioni che l’utente vuole inserire in una regola; esso permette quindi di costruire la regola “dimostrando” al sistema cosa vorrebbe facesse al verificarsi di certe situazioni. Anche per la modalità interativa sono state pensate varie composizioni degli elementi grafici e varie possibilità di composizione della regola con questa modalità (Figura 4.9).

Cliccando su “REC MODE”, l’utente chiede al sistema di notificare tutti

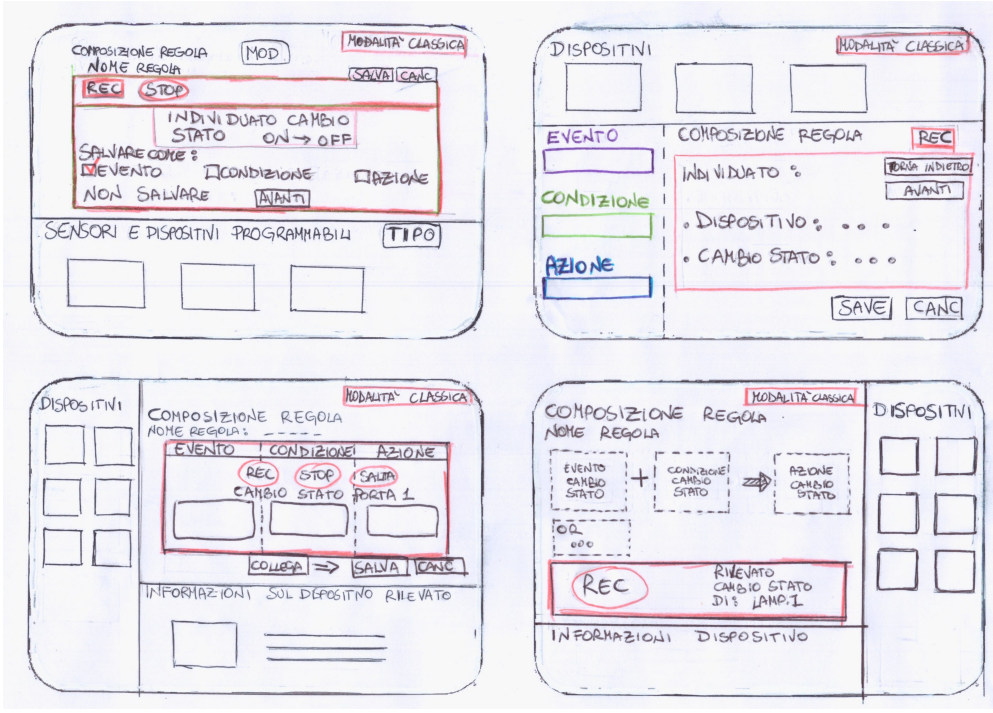

Figura 4.9. I 4 *mockup* iniziali della modalità di registrazione interattiva.

i cambiamenti che esso riceve. All'utente compaiono quindi i *feedback* dall'ambiente ed egli sceglie quale ruolo attribuire ad ogni singola interazione, se di evento, condizione o azione, per la composizione della regola. L'idea originale era quella di permettere la composizione di un'intera regola, registrando tutte le interazioni dell'utente con l'ambiente. Tuttavia questo modo di operare avrebbe potuto creare problemi, si pensi al caso in cui si volesse aggiungere un elemento che non dipende da una interazione dinamica con l'ambiente circostante, ad esempio una misura di temperatura fissa, oppure un vincolo temporale. Permettendo agli utenti di completare la regola solamente con la registrazione, quindi vedendo questa modalità come interamente parallela alla modalità di aggiunta "classica" (quella descritta nei prototipi precedenti), si sarebbe rischiato di limitare troppo il tipo e la quantità di regole componibili. Quindi si è pensato di creare una modalità mista in cui l'utente ha la possibilità di aggiungere alcune parti della regola attraverso la modalità di registrazione interattiva e altre con la modalità classica.

4.3.2 Timer e Calendar

Dall'analisi dello stato dell'arte è emerso che un altro punto, essenziale soprattutto per una gestione di regole semanticamente più complesse, è l'individuazione di un giusto modo per trattare i diversi tipi di proposizioni temporali esistenti nel linguaggio comune. Della gestione del tempo, come si è visto, si sono occupati gli studi di Garcìa-Herranz [5], ideando un tipo di azione aggiuntiva chiamata "TIMER", e anche il gruppo e-Lite, introducendo nelle precedenti applicazioni realizzate, un orologio e un calendario come dei dispositivi fisici.
Nell'applicazione realizzata si è scelto di utilizzare degli oggetti che permettessero di distinguere la contemporaneità dalla posteriorità e dall'anteriorità e, dopo aver analizzato varie idee, si è deciso di inserire il "timer" e il "calendar" come dispositivi aggiuntivi a quelli forniti dalla casa intelligente. I due dispositivi aggiuntivi possono essere utilizzati per inserire eventi e condizioni secondo le regole della grammatica scelta, quindi può essere aggiunto un solo evento relativo al tempo e un solo evento relativo al calendario, la stessa cosa vale per le condizioni. Ciò significa che in una regola si può fissare un solo istante di tempo o una sola data in *OR* a tutti gli altri eventi, e, nel caso delle condizioni, un solo intervallo orario e un solo intervallo di giorni

in *AND* a tutte le altre condizioni da verificare per far scattare la regola. Se ad esempio si vuole creare una regola in cui se si verifica un evento in due distinti intervalli di tempo (per esempio delle settimane di mesi diversi) si vuole attivare una data azione, si devono necessariamente comporre due regole diverse inserendo una condizione diversa.

Capitolo 5

Progettazione fase 2: studio utenti

Come detto precedentemente, in una progettazione *user-centered* sono tenuti in grande considerazione le opinioni che l'utente ha dell'interfaccia e i suoi suggerimenti per migliorarla. A tale scopo, è di grande importanza effettuare un test nella fase successiva alla prototipazione da sottoporre a un campione di partecipanti, ben rappresentativo della popolazione di utenti. Nei paragrafi successivi verranno descritti gli obiettivi e le modalità di svolgimento del test somministrato per confrontare le due interfacce. Verranno analizzati i risultati del test e descritte le modifiche progettuali fatte alla luce dei risultati ottenuti.

5.1 Test: confronto tra due approcci

5.1.1 Obiettivi

Lo scopo del test è quello di valutare l'usabilità delle due diverse interfacce per la composizione di regole domotiche mostrate in 4.4, 4.7, paragonandone la facilità d'uso, l'intuitività e la chiarezza. Nello specifico, si vogliono ottenere indicazioni su possibili elementi dell'interfaccia che ostacolano l'esatta composizione delle regole. Si vuole valutare in modo particolare:

- il modello concettuale/semantico, se questo viene compreso dai partecipanti al test o se vi sono problemi di interpretazione dei concetti o dei termini utilizzati;

- la funzionalità dell'applicazione, se l'interfaccia ha tutto ciò che serve per svolgere i compiti per la quale è stata progettata, se mancano elementi che ne faciliterebbero l'utilizzo da parte degli utenti.

- la fruibilità nella navigazione dell'app e nello svolgimento dei compiti o se gli utenti si orientano con facilità nell'esplorare l'applicazione e la trovano gradevole da utilizzare;

- la terminologia, se gli utenti capiscono a cosa servono gli elementi grafici presenti e se questi sono particolarmente esplicativi;

- l'apprendibilità, se si apprende l'utilizzo dell'interfaccia in modo veloce e intuitivo.

Si è cercato quindi di rispondere alle seguenti domande di ricerca:

- Quanto gli utenti comprendono la composizione con trascinamento e quanto quella con selezione? Quali sono le problematiche?

- Gli utenti si orientano all'interno dell'applicazione?

- Quanto facilmente riescono a comporre le regole e che difficoltà incontrano nelle due rispettive interfacce?

- Le interfacce contengono tutto quello che serve? Manca qualcosa?

- L'uso degli elementi grafici (icone - frecce) viene compreso facilmente? Le etichette sono abbastanza esplicative?

- C'è un appropriato bilanciamento tra facilità di utilizzo e facilità di apprendimento?

- La modalità di registrazione viene ritenuta utile?

5.1.2 Materiali e metodi

Il test è stato somministrato ad un gruppo di 12 partecipanti, compresi tra i 20 e i 40 anni di età, con differente formazione di studi ed esperienza nell'uso di dispositivi mobile e nella programmazione.
I 12 utenti si possono dividere in due gruppi:

- Utenti che hanno frequentato un percorso di studi di Ingegneria o hanno conoscenza nella programmazione (successivamente chiamati anche *utenti tecnici*)
- Utenti con studi non ingegneristici e nessuna esperienza nella programmazione (successivamente chiamati anche *utenti non tecnici*).

Utente	Caratteristiche	
	Età	Sesso
tecnico 1	30	M
tecnico 2	24	M
tecnico 3	25	F
tecnico 4	26	F
tecnico 5	24	F
tecnico 6	25	M
non tecnico 1	30	M
non tecnico 2	24	M
non tecnico 3	24	F
non tecnico 4	27	F
non tecnico 5	28	F
non tecnico 6	24	M

Tabella 5.1. Le caratteristiche dei partecipanti al test

Nella composizione dei due gruppi, entrambi formati da sei partecipanti, si è cercato di far sì che essi fossero omogenei per quanto riguarda l'età e il sesso, in modo da evitare la presenza di *bias* che potessero alterare i risultati. Le caratteristiche (età, sesso e percorso di studi) dei partecipanti al test sono mostrate in Tabella 5.1 .

Nella somministrazione del test si possono individuare 3 fasi che vengono riportate di seguito.

In una prima fase si è cercato di mettere l'utente a suo agio, è stata illustrata le modalità di esecuzione del test, è stato spiegato come è composta una regola domotica nel sistema preso in esame e sono state presentate le due interfacce cartacee che i partecipanti avrebbero dovuto utilizzare nella fase successiva. Per spiegare in cosa consistesse il test e illustrarne gli obiettivi, veniva letta una breve introduzione in modo da poter fornire a tutti i partecipanti le stesse informazioni nello stesso modo. Successivamente veniva somministrato un questionario anonimo per ricavare dati sui partecipanti, quali età, professione, frequenza di utilizzo del tablet ed esperienza nell'uso di applicazioni mobile e di applicazioni mobile. Inoltre si chiedeva il consenso per audioregistrare il test.

Nella seconda fase, il partecipante al test ha sperimentato le due interfacce attraverso dei prototipi cartacei, ha compilato un questionario valutativo per ogni interfaccia e ha espresso i sui pareri sulle interfacce. Questa seconda fase si può suddividere in due ulteriori parti: una prima, in cui veniva fatta testare l'interfaccia *drag &drop* e una seconda, in cui veniva fatta testare l'interfaccia con *click*. L'ordine con cui venivano testate le due interfacce è stato scelto a priori in modo che lo stesso numero di partecipanti al test provasse prima l'una e poi l'altra e viceversa. Questo modus operandi ha evitato possibili penalizzazioni nel giudizio alle interfacce derivanti dall'ordine di esecuzione del test. Per ciascuna interfaccia testata, è stato chiesto ai partecipanti di eseguire la composizione di due regole utilizzando il prototipo cartaceo. Si chiedeva di immaginare che il foglio di carta fosse un tablet reale e di comporre le seguenti regole:

- "All'accensione della lampada del comodino, spegni la lampada della stanza"
- "Quando la tapparella del salotto si sta alzando oppure la tapparella del corridoio si sta alzando, e la lampada del salotto è accesa, spegnila".

Le due regole hanno complessità differente: la prima contiene un solo evento da cui scaturisce una sola azione; la seconda contiene due eventi, una condizione e un'azione conseguente. Gli utenti potevano scegliere liberamente

quale regola comporre per prima. Quando il partecipante al test interagiva con l'interfaccia cartacea venivano fatti comparire i menù e gli altri elementi grafici relativi all'interfaccia che l'utente stava provando. Nell'interfaccia in cui l'utente doveva comporre le regole mediante le operazioni di *click* veniva chiesto appunto di toccare i pulsanti, mentre nell'interfaccia basata su operazioni di trascinamento e composizione "dinamica" veniva chiesto di trascinare gli elementi e di disegnare con una matita delle frecce per collegare i vari blocchetti. Dopo aver composto le due regole, veniva chiesto di continuare a sperimentare l'interfaccia componendo, se si desiderasse, qualche altra regola. Successivamente, veniva somministrato un questionario per ricavare opinioni sulle due interfacce e stabilire quale tra le due l'utente considerasse più semplice da utilizzare e di quale avrebbe fatto uso più volentieri. Infine, veniva chiesto ai partecipanti quali fossero, secondo il loro parere, i punti di forza e di debolezza di ciascuna interfaccia e quali aspetti potessero essere migliorati.

Nella fase finale, per ogni interfaccia provata, veniva mostrato al PC un video che illustrava come si sarebbe svolta la composizione delle regole nei due casi sperimentati sul cartaceo poco prima, attraverso un prototipo interattivo. Alla luce dei video esplicativi, si chiedeva quindi ai partecipanti se le opinioni relative alle due interfacce, espresse precedentemente, avessero subito cambiamenti o fossero rimaste invariate.

Il partecipante veniva seguito nello svolgimento del test e aiutato senza influenzarne la prestazione. La prova era audio-registrata per tener traccia delle difficoltà e delle osservazioni fatte dai partecipanti.

5.2 Risultati

In questo paragrafo vengono illustrati i risultati dei questionari sottoposti ai partecipanti. Per quanto confrontare le due interfacce sono state rivolte ai partecipanti le seguenti domande:

- "Quale tra le due interfacce ritieni più semplice da utilizzare?"
- "Quale tra le due interfacce utilizzeresti più volentieri?"

Come emerge dai grafici, creati a partire dai risultati ottenuti (Figura 5.1) non si sono riscontrate differenze tra tecnici e non tecnici sulla netta preferenza per l'interfaccia basata su operazioni di *click*. Sia programmatori che non programmatori sono stati concordi nell'affermare, in più della maggioranza dei casi, che l'interfaccia basata sul *click* è semplice, chiara e facilmente comprensibile.

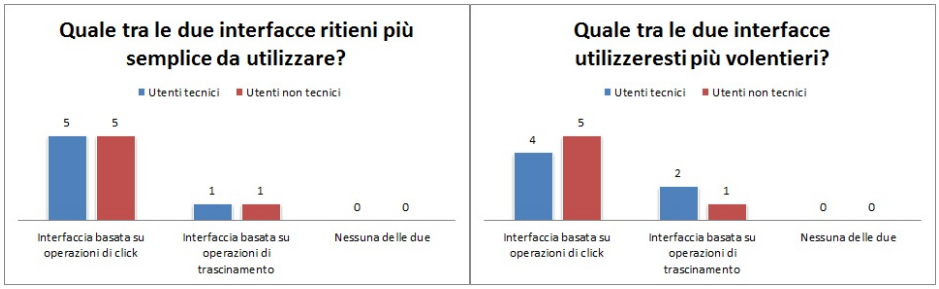

Figura 5.1. Confronto delle valutazioni date dagli utenti (distinti in tecnici e non tecnici) alle due interfacce.

Nelle figure successive sono mostrati i questionari somministrati ai partecipanti con i relativi risultati riscontrati; i risultati vengono raggruppati per interfaccia e per ordine di effettuazione del test. La distinzione tra tecnici e non tecnici è messa in evidenza mediante un colore diverso. La Figura 5.2 mostra le risposte al questionario date dai partecipanti relative all'interfaccia *drag and drop*. Il colore rosso indica che l'interfaccia *drag & drop* è stata testata dopo aver già sperimentatato l'altra interfaccia, mentre il colore verde indica che l'interfaccia *drag & drop* è stata testata come prima.

La Figura 5.3 mostra invece le risposte al questionario date dai partecipanti relative all'interfaccia basata sul *click*, anche in questo caso il colore rosso indica che questa interfaccia è stata testata come seconda, mentre il colore verde indica che è stata testata per prima.

Come si evince dalle risposte date, i partecipanti al test hanno compreso abbastanza facilmente sia la composizione mediante trascinamento che quella mediante *click*. Tutti sono riusciti a comporre le regole date sia con la prima interfaccia sia con la seconda; in particolare, soprattutto attraverso l'interfaccia *click*, le regole sono state composte senza particolari problemi. Molti partecipanti inoltre si sono cimentati nella composizione di regole personali, riuscendo facilmente nell'intento.

1) Penso che l'interfaccia provata sia facile da utilizzare.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
		X X O	X X X O O O O	X O

2) Penso che la struttura dell' interfaccia risulti semplice e comprendibile.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
	O	X X	X X X X O O O O	O

3) Non è stato difficile apprendere il funzionamento dell'applicazione.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
	X	X X O O	X X O O	X O O

4) Ho composto la regola 1 (breve) senza particolari problemi.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
		X O O	X X O O O	X X X O

5) Ho composto la regola 2 (lunga) senza particolari problemi.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
X		X X O	X O O	X X O O O

Legenda:
colore verde : interfaccia testata per prima
colore rosso : interfaccia testata per seconda

X : utenti tecnici (che hanno frequentato un percorso di studi di Ingegneria e hanno conoscenze di programmazione)
O : utenti non tecnici (con studi non ingegneristici e nessuna esperienza nella programmazione)

Figura 5.2. Questionario somministrato agli utenti dopo aver testato l'interfaccia *drag & drop*.

1) Penso che l'interfaccia provata sia facile da utilizzare.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
		X	X X O	X X X O O O O O

2) Penso che la struttura dell' interfaccia risulti semplice e comprendibile.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
		X	X X O O	X X X O O O O

3) Non è stato difficile apprendere il funzionamento dell'applicazione.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
	X	X	X O O O	X X X O O O

4) Ho composto la regola 1 (breve) senza particolari problemi.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
	X		X O O O	X X X X O O O

5) Ho composto la regola 2 (lunga) senza particolari problemi.

Sono fortemente in disaccordo	Sono in disaccordo	Non sono né d'accordo né in disaccordo	Sono d'accordo	Sono fortemente d'accordo
		O	X X X O O	X X X O O O

Legenda:
colore verde : interfaccia testata per prima
colore rosso : interfaccia testata per seconda

X : utenti tecnici (che hanno frequentato un percorso di studi di Ingegneria e hanno conoscenze di programmazione)
O : utenti non tecnici (con studi non ingegneristici e nessuna esperienza nella programmazione)

Figura 5.3. Questionario somministrato agli utenti dopo aver testato l'interfaccia *click*.

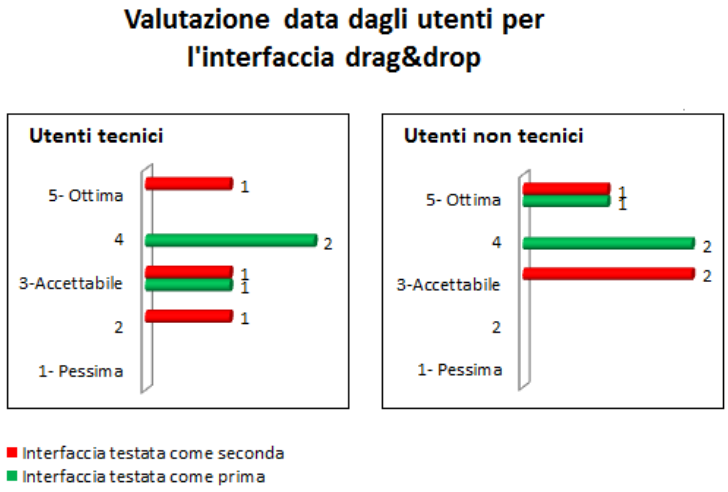

Figura 5.4. Valutazione data dagli utenti attraverso una scala quantitativa(da 1 a 5) all'interfaccia basata su operazioni di *drag & drop*.

A ciascun partecipante si chiedeva, inoltre, di dare una valutazione complessiva all'interfaccia, con una votazione da 1 a 5. Dall'analisi dei risultati, mostrati nelle figure 5.4, 5.5, si può notare che tutti i partecipanti, senza

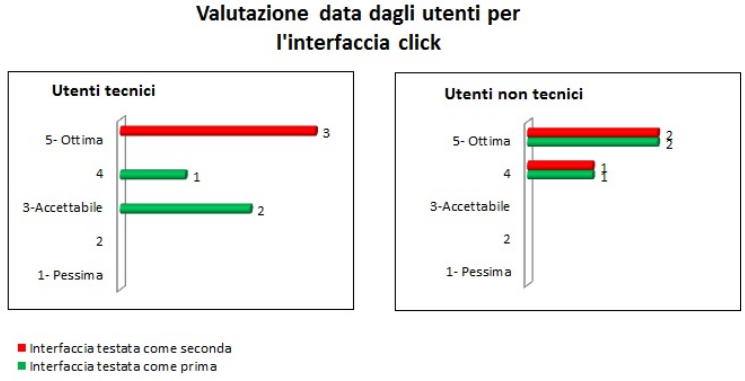

Figura 5.5. Valutazione data dagli utenti attraverso una scala quantitativa(da 1 a 5) all'interfaccia basata su operazioni di *click*.

distinzione hanno valutato accettabile o più che accettabile le due interfacce. Si può notare inoltre che la votazione numerica data all'interfaccia *drag & drop* se questa veniva testata dopo quella di *click* è inferiore rispetto a quella attribuita a quest'ultima; al contrario, anche quando l'interfaccia di *click* è stata testata come seconda, i punteggi attribuiti ad essa sono stati sempre superiori rispetto a quelli dati all'interfaccia *drag & drop*. In definitiva, entrambi i gruppi hanno manifestato una netta preferenza per l'interfaccia basata su operazioni di *click* e, in più della maggioranza dei casi, sono stati concordi nell'affermare che l'interfaccia basata sul *click* sia semplice, comprensibile e facile da utilizzare. La composizione mediante trascinamento, pur essendo ritenuta semplice e facile da usare, ha suscitato qualche perplessità. In molti hanno sottolineato che, anche se il trascinamento rientra più nell'ottica *touch* e quindi più vicino alle operazioni che si fanno con un *tablet*, può indurre a commettere più errori rispetto al semplice *click*, ad esempio di rilasciare un oggetto nel posto non corretto oppure di creare errori muovendo in modo sbagliato gli elementi. Dall'altra parte, con il *click*, l'operazione è più istantanea e immediata e offre minori possibilità di commettere errori. Un altro parere condiviso da molti partecipanti al test è stato quello di preferire uno schema della regola preimpostato piuttosto di avere libertà nella composizione della stessa nel piano di lavoro. Anche questo parere quindi va a favore dell'interfaccia con composizione tramite *click*. La possibilità, offerta dall'interfaccia *drag & drop*, di spostare i blocchetti degli eventi, delle condizioni e delle azioni a piacere nello schermo non è stato ritenuto essenziale.
Riguardo alla modalità di registrazione interattiva, si è chiesto ai partecipanti se ritenessero utile tale modalità e se la utilizzerebbero. Molti hanno risposto di considerarla utile per la creazione di regole semplici; 6 utenti (3 tecnici e 3 non tecnici), pur non escludendo la possibilità di poterla utilizzare in futuro, al momento non l'hanno ritenuta necessaria e non la utilizzerebbero; gli altri 6 utenti invece si sono mostrati entusiasti di questo modo di interagire con la casa intelligente e ne farebbero uso.

5.3 Modifiche progettuali alla luce del test

La maggior parte dei commenti e dei pareri negativi sono stati dati per l'interfaccia *drag & drop*. Il principale problema emerso consiste nel collegamento tra i blocchetti nella zona di composizione; la modalità di collegamento tramite freccia non risultava immediata.
Alla luce dei risultati ottenuti, l'interfaccia *drag & drop* è stata abbandonata e nella successiva fase di programmazione ci si è concentrati sull'interfaccia basata sulle operazioni di *click*.

Dalle, seppur poche, critiche mosse dagli utenti all'interfaccia basata su operazioni di *click*, si sono potuti ricavare degli spunti per migliorarla. In particolare si sono rilevate tre principali necessità:

- la necessità di avere uno schema iniziale della regola (quello mostrato nella zona di composizione) che sottolineasse anche la possibilità di inserire più condizioni e più azioni.

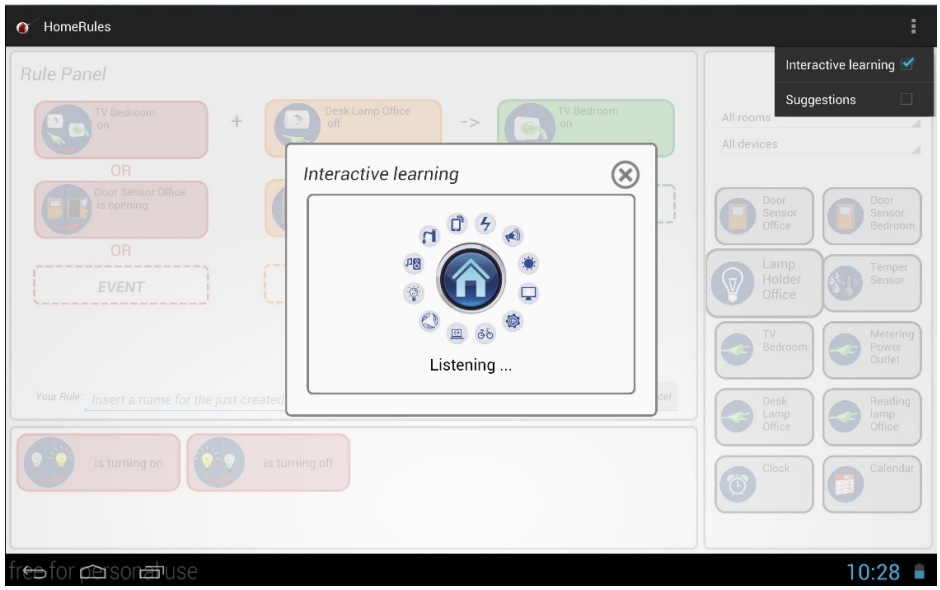

Figura 5.6. L'interfaccia finale con l'opzione "Interactive learning" attiva. Fase 1- Applicazione in ascolto di una interazione dell'utente con la casa.

- la necessità di avere dei suggerimenti durante tutta la composizione e spiegazioni aggiuntive su come comporre la regola (paragrafo 5.3.1).
- un modo più esplicativo (rispetto al tasto "REC MODE") o diverso per passare alla modalità di registrazione interattiva, di seguito definita anche interactive learning (paragrafo 5.3.2).

5.3.1 Interactive learning

Inizialmente, come visto nei prototipi, il tasto per passare alla modalità di registrazione interattiva è stato posto nell'area centrale. Successivamente, alla luce delle opinioni date dai partecipanti al test, tale tasto è stato eliminato e si è scelto di far aggiungere parti di regola con la modalità interattiva selezionando l'opzione dal menù dell'interfaccia (opzione "Interactive learning").

In Figura 5.6 si può vedere cosa accade selezionando la modalità di interactive learning. Abilitando l'opzione, nella zona centrale dello schermo

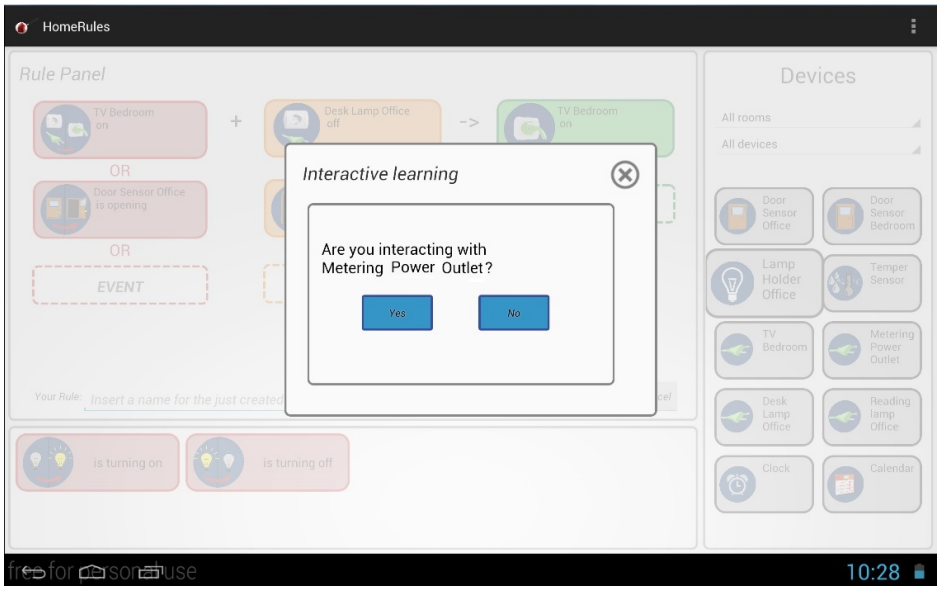

Figura 5.7. L'interfaccia finale con l'opzione "Interactive learning" attiva. Fase 2- Conferma del dispositivo con cui l'utente ha interagito.

compare un'area semitrasparente (per permettere di vedere in trasparenza la composizione della regola) e l'applicazione si mette in ascolto dei movimenti dell'utente.
Quando l'utente interagisce con un dispositivo, l'applicazione notifica all'utente con quale dispositivo si sta interagendo e chiede se è proprio quello il dispositivo che si vuole usare per aggiungere una parte regola (Figura 5.7). Se l'utente conferma il dispositivo allora l'applicazione mostra la notifica che ha ricevuto dall'ambiente e dà la possibilità all'utente di scegliere se aggiungere quella notifica come evento, come azione o come condizione (Figura 5.8). A quel punto verrà creato un blocchetto che sarà messo nella zona di composizione.
In qualsiasi momento l'utente può scegliere di chiudere l'aggiunta di parti di regola in modo interattivo premendo sulla "X" della videata oppure disabilitando l'opzione dal menù.

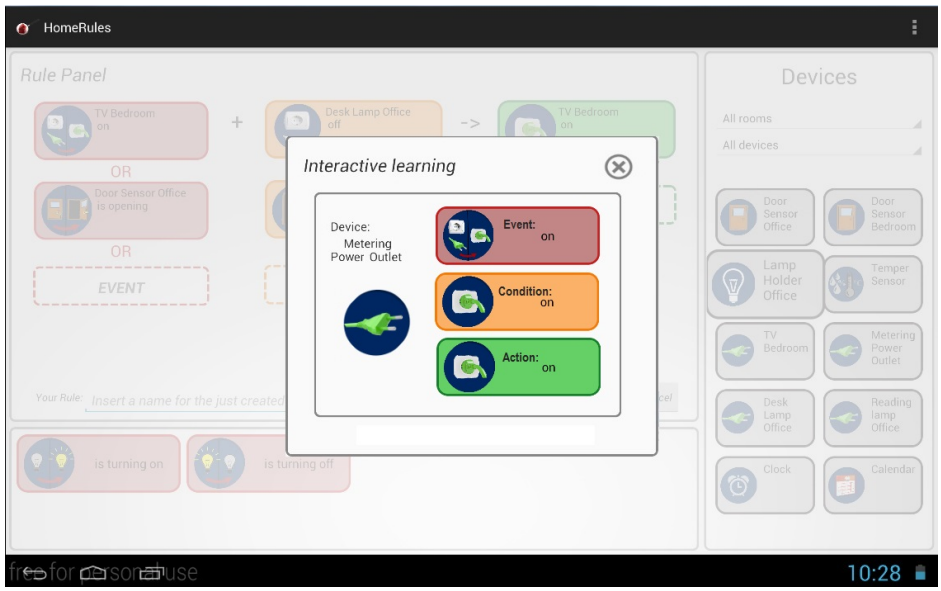

Figura 5.8. L'interfaccia finale con l'opzione "Interactive learning" attiva. Fase 3- Visualizzazione della notifica ricevuta.

5.3.2 Suggerimenti

Analizzando lo stato dell'arte, è emersa, soprattutto in [17], [18], l'importanza di guidare gli utenti nella composizione delle regole attraverso una sorta di "tutoraggio".
Quest'idea è stata supportata anche dai risultati del test. I prototipi di interfaccia contenevano un solo suggerimento nelle schermata iniziale che indicava all'utente come approcciarsi all'uso dell'interfaccia stessa. Dal test è emerso che, soprattutto gli utenti non tecnici, cercavano spiegazioni o suggerimenti anche nei passaggi successivi e quindi avrebbero preferito avere anche, almeno per le prime volte, una modalità che li guidasse passo passo. Alla luce di ciò, si sono scelte due possibilità di supporto da fornire all'utente.

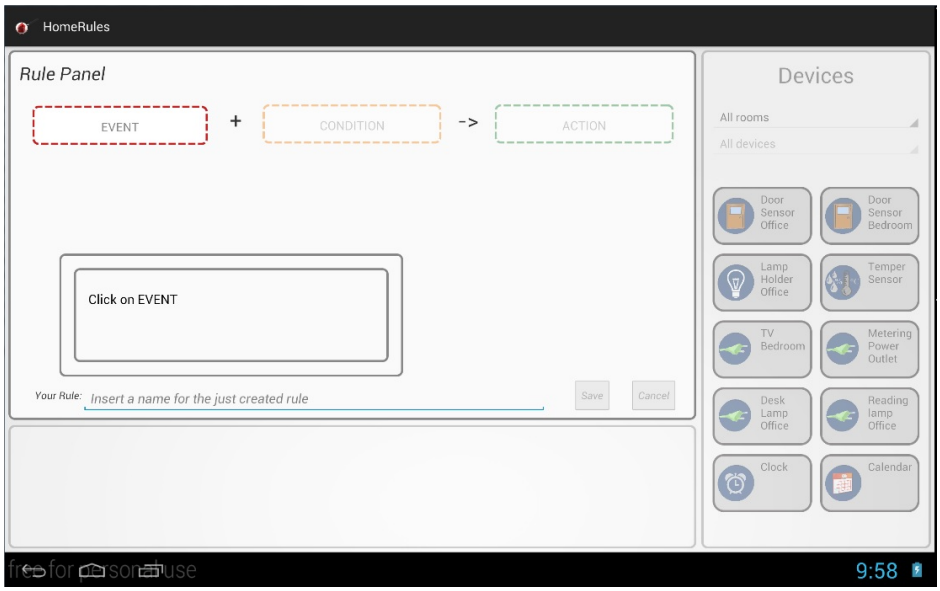

Figura 5.9. L'interfaccia finale nella modalità guidata.

La prima possibilità di aiuto è stata pensata per guidare passo passo l'utente nella creazione di una intera regola semplice. Si tratta di una modalità alternativa alla creazione della regola. L'interfaccia è stata quindi modificata per consentire all'utente di fare delle scelte forzate (Figura 5.9).

Si chiede di aggiungere un evento (ovvero viene chiesto di cliccare sul rettangolo evento, poi di selezionare un dispositivo, successivamente di scegliere un evento tra quelli possibili) e un'azione (chiedendo quindi di selezionare il rettangolo azione, scegliere un dispositivo e scegliere l'azione) e infine di scegliere un nome per la regola e salvare. All'utente viene anche data, opzionalmente, la possibilità di aggiungere una condizione (Figura 5.10).

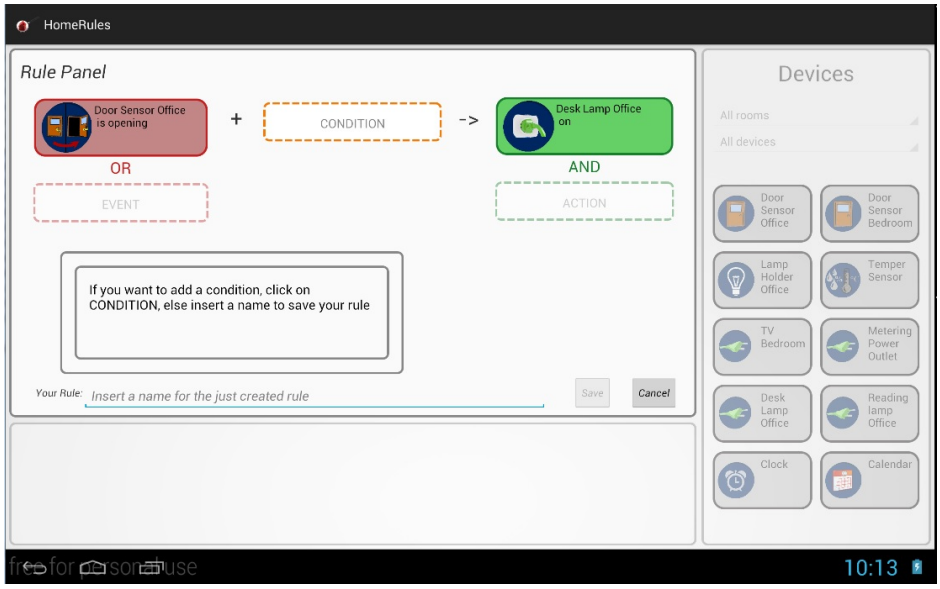

Figura 5.10. L'interfaccia finale nella modalità guidata, aggiunta opzionale di una condizione.

La seconda possibilità di ottenere suggerimenti è stata pensata per dare un supporto e un aiuto all'utente in qualsiasi momento egli ne abbia bisogno, anche solo le prime volte che l'utente si approccia all'uso dell'applicazione. I suggerimenti, in questo caso, sono visti come un'opzione, opzione "Suggestions", dell'interfaccia classica (Figura 5.11). L'utente può abilitarli da menù; compariranno delle scritte che descrivono le possibilità di selezione che ha l'utente in qualsiasi parte della composizione si trovi. I suggerimenti possono essere visualizzati anche quando l'opzione di "Interactive learning" è abilitata.

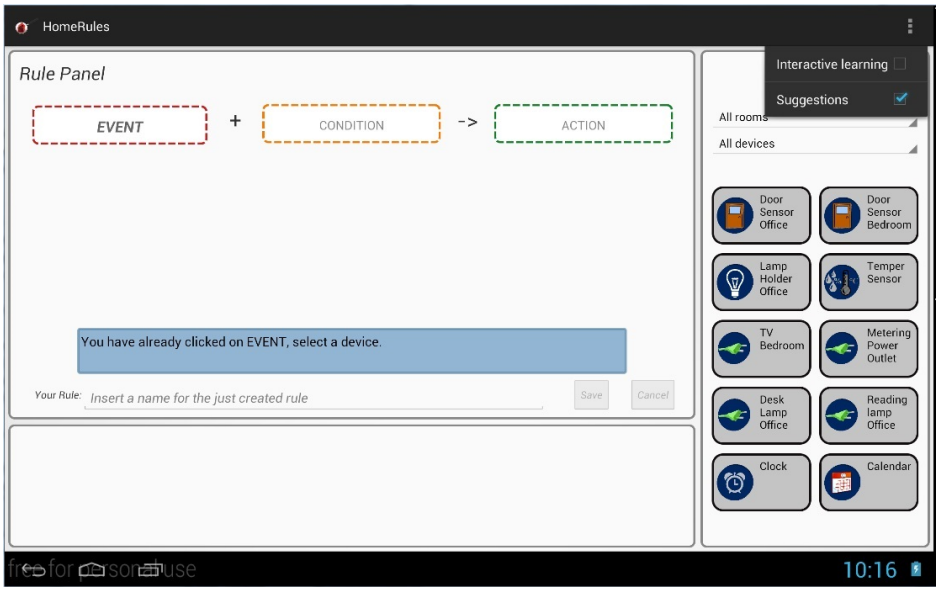

Figura 5.11. L'interfaccia finale nella modalità classica con l'opzione "Suggestions" attiva.

Capitolo 6

Implementazione

Alla fase di progettazione, descritta nei due precedenti capitoli, fa seguito quella di implementazione, il cui scopo è quello di realizzare un'applicazione funzionante. In questo capitolo vengono esposte e motivate le scelte

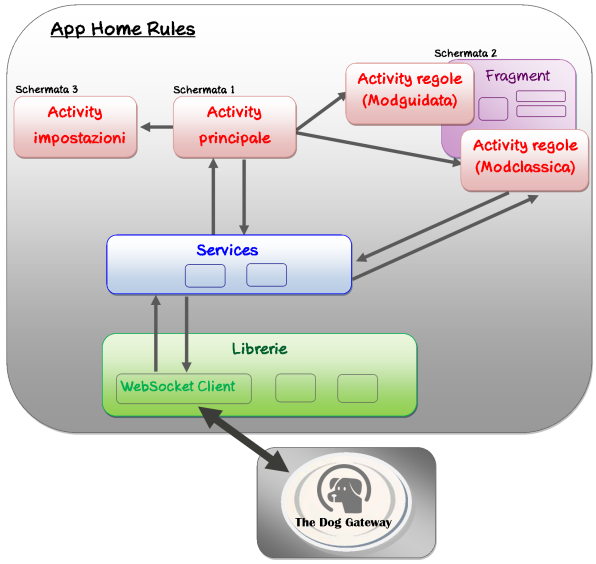

Figura 6.1. Schema dell'architettura complessiva dell'applicazione realizzata.

implementative fatte e sono illustrate alcune caratteristiche dell'applicazione finale e la sua architettura complessiva (di cui è mostrato uno schema in Figura 6.1).
In dettaglio, nel paragrafo 6.1 verrà descritta la struttura generale dell'app; nel paragrafo 6.2 sarà presentato il *Manifest* dell'applicazione; nel paragrafo 6.3 verrà descritto come è stato implementato il passaggio dei dati tra *activity*, *fragment* e *service*; nel paragrafo 6.4 sarà illustrata la struttura dati scelta per le regole; infine nel paragrafo 6.5 si parlerà delle possibilità di riutilizzo del codice creato.

6.1 La struttura dell'applicazione

L'applicazione realizzata è stata denominata *Home Rules* e si compone di tre interfacce principali: la schermata di "*welcome*", la schermata per i settaggi della connessione con *Dog* e la schermata per la composizione delle regole.
La struttura di base e le parti statiche delle interfacce sono implementate

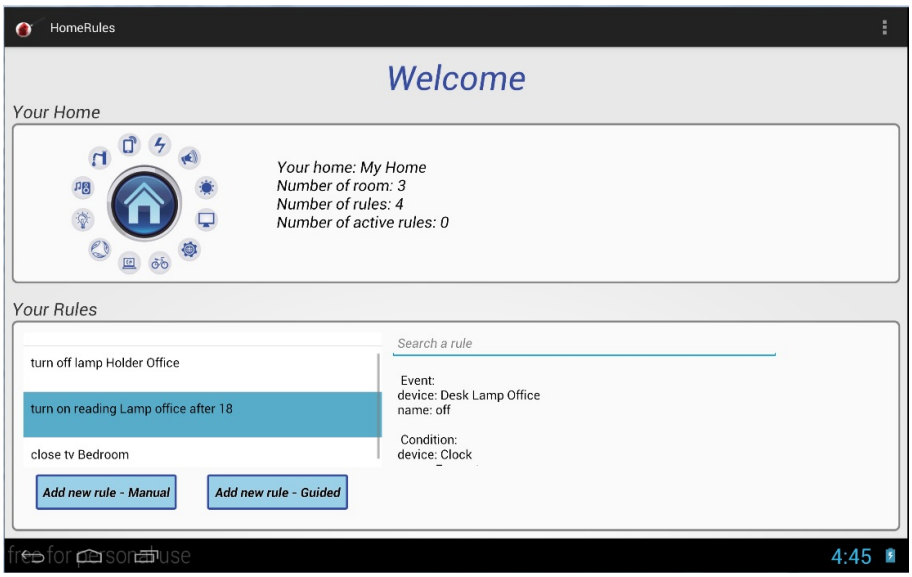

Figura 6.2. La schermata principale di "*welcome*".

in codice XML, le parti dinamiche invece sono state scritte in Java, come tutta la parte algoritmica. Di seguito vengono analizzate nei dettagli le strutture delle varie interfacce dell'applicazione:

La schermata principale

La schermata principale di "*welcome*" (Figura 6.2) è quella che l'utente vede appena l'applicazione viene aperta. Essa può essere suddivisa

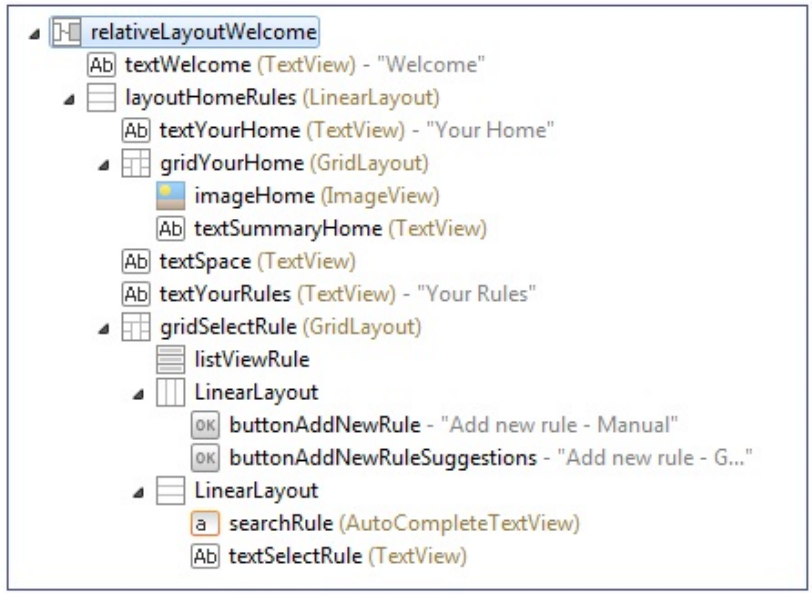

Figura 6.3. Schema degli elementi presenti nella schermata principale

orizzontalmente in due aree: nella prima area, denominata "*Your Home*", vengono mostrate le informazioni sulla *smart home*; nella seconda area, denominata "*Your Rules*", vengono mostrate le regole create, le informazioni su ogni regola e la possibilità di aggiungerne di nuove. Come si può notare dallo schema degli elementi presenti nella schermata in Figura 6.3, per le due aree appena descritte si è scelto di utilizzare due *gridLayout*, rispettivamente *gridYourHome* e *gridYourRule*, all'interno di un *LinearLayout* (*LayoutHomeRules*). Il primo *gridLayout* contiene un'immagine e una *textView* con le informazioni

sullo stato della *smart home*; tali informazioni vengono aggiornate mediante codice, quando queste sono disponibili o subiscono modifiche. Il secondo *gridLayout* contiene una *listView* (*listViewRule*) che mostra tutte le regole create per la *smart home*; un'*autoCompleteTextView* che permette la ricerca per nome delle regole presenti nella lista; una *textView* che dà la possibilità di visualizzare le informazioni riguardanti una regola selezionata dalla lista; infine due *button* che consentono di accedere alle schermate di composizione della regola.

La schermata per i settaggi della connessione con Dog

La seconda schermata, presentata in Figura 6.4, è quella delle impo-

Figura 6.4. La schermata per i settaggi della connessione con *Dog*

stazioni. Questa schermata ha una grafica molto semplice, contiene una zona con due *editText* che permette all'utente di inserire le informazioni necessarie per stabilire una comunicazione con *Dog*. Esse comprendono l'indirizzo del server e il numero massimo di tentativi di connessione che si devono effettuare nel caso in cui il server non risponda.

Questa schermata è la prima che viene visualizzata dopo l'installazione dell'applicazione, la prima volta che questa viene avviata. È possibile accedere a tale schermata dal menù della schermata principale, nel caso in cui l'utente volesse cambiare i parametri di connessione.

La schermata per la composizione delle regole

La schermata di composizione delle regole è la schermata fondamentale dell'applicazione. È l'interfaccia che è stata oggetto di programmazione dettagliata e di cui si è parlato nei precedenti capitoli.

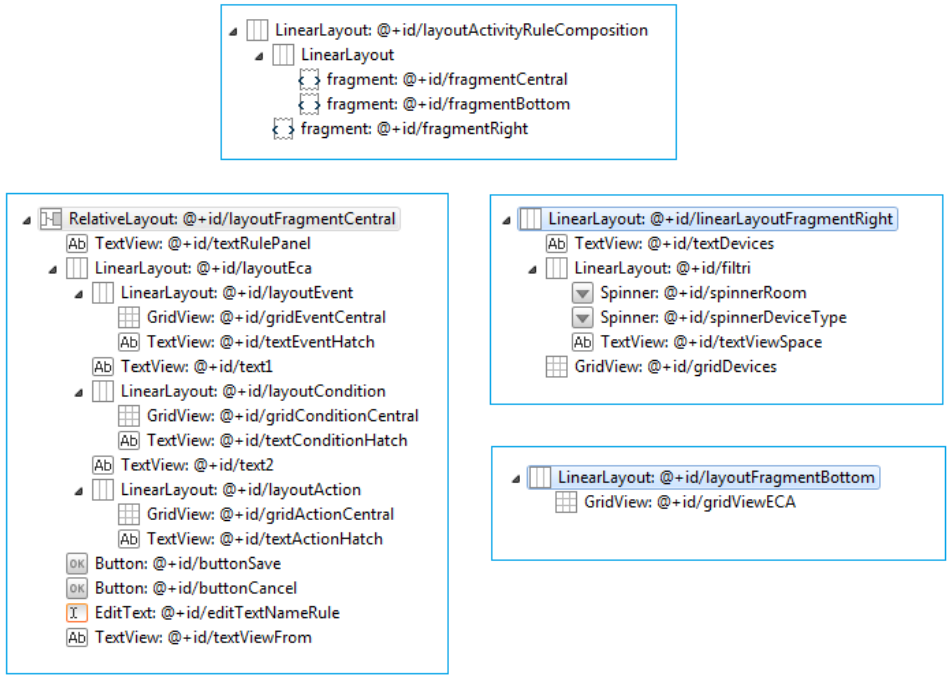

Figura 6.5. Gli *outline* dei file xml della schermata per la composizione delle regole.

In figura 6.5 è mostrato l'*outline* dei file xml principali che la compongono, mentre in Figura 6.6 vi è il layout grafico della schermata.

Figura 6.6. Layout grafico della schermata per la composizione delle regole.

Come si può notare, l'interfaccia è composta da tre *fragment*:

- un *fragment* posto centralmente nello schermo, che contiene la zona di composizione della regola;
- un *fragment* posto sulla destra, che contiene gli elementi grafici per la gestione dei dispositivi;
- un *fragment* posto in basso, che permette la visualizzazione degli eventi, delle condizioni e delle azioni relative al dispositivo selezionato.

In dettaglio, il *fragment* centrale, di composizione della regola, contiene: tre *gridView* che permettono la visualizzazione delle componenti della regola che l'utente compone; tre *textView* che consentono di selezionare rispettivamente l'evento, la condizione o l'azione; un'*editText* che permette all'utente l'inserimento del nome da dare alla regola; infine i *button* "*Save*" e "*Cancel*" che servono rispettivamente a permettere il salvataggio della regola, oppure a cancellare tutti elementi già inseriti.
Nel *fragment* di destra sono mostrati in una *gridView* tutti i dispositivi

presenti nella *smart home*; inoltre sono presenti due *spinner* che permettono di ricercare i dispositivi rispettivamente in base alla stanza di appartenenza o in base al tipo di dispositivo.

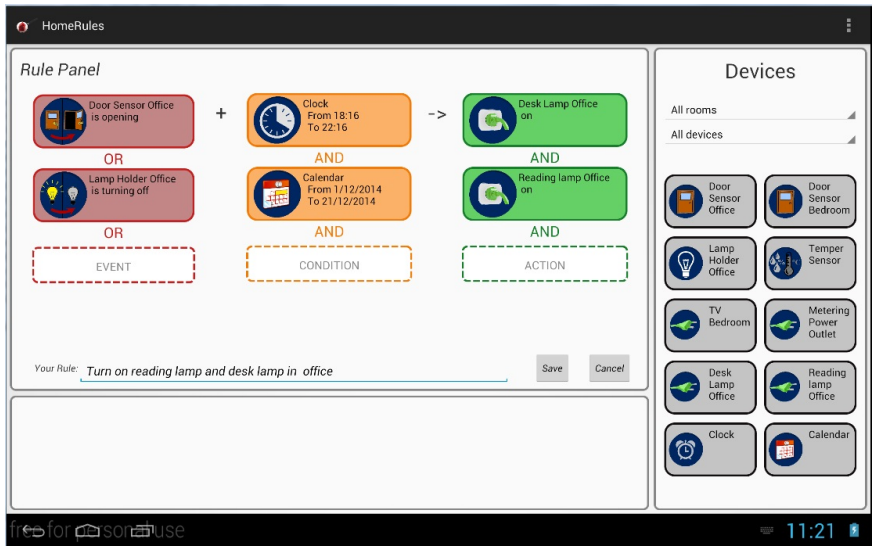

Figura 6.7. La schermata di composizione delle regole con un esempio di regola composta.

Il *fragment* in basso contiene una *gridView* che viene aggiornata in base al dispositivo selezionato dal *fragment* di destra e all'evento, alla condizione o all'azione scelti nel *fragment* centrale.
La Figura 6.7 mostra l'interfaccia di composizione delle regole; nell'area di composizione della regola si può vedere un esempio di regola composta.

6.1.1 Approfondimento: *gridView* e *adapter*

Di grande utilità per la creazione delle interfacce è stato l'uso delle *gridView* e degli *adapter*, a cui è dedicato questo approfondimento.
L'*adapter* è un componente collegato ad una struttura dati di oggetti Java (ad esempio gli *array*, le *Collections* o i risultati di *query*) e che incapsula il meccanismo di trasformazione di questi oggetti in altrettante *view*da mostrare su un layout. Una *gridView* è un particolare tipo di *adapterView*,

ovvero una componente visuale che è collegata ad un *adapter* e raccoglie tutte le *view* prodotte dall'*adapter*, per mostrarle all'interno di una griglia. Nell'applicazione, dovendo mostrare contenuti di vario genere, è stato opportuno creare diversi *adapter* e gestire le *gridView* in modo che esse, quando necessario, possano essere aggiornate settando l'*adapter* corretto. Ad esempio nella *gridView* per la gestione di eventi, condizioni e azioni, presente in basso nella schermata di composizione delle regole, vengono aggiornati gli *adapter* in base al dispositivo selezionato dal *fragment* di destra e all'evento, alla condizione o all'azione scelti nel *fragment* centrale.
Le *gridView*, come tutti gli *adapterView*, hanno un altro ruolo molto importante che è quello della gestione degli eventi. La gestione degli eventi avviene mediante il meccanismo dei *listener*. Nell'applicazione, sugli elementi della *gridView* è stato settato il *setOnItemClickListener*, che permette la gestione del *click* sugli elementi presenti. La classe *OnItemClickListener*, infatti, viene utilizzata allo scopo di gestire il *click* e il metodo *onItemClick*, interno ad essa, contiene il codice da attivare alla selezione di ogni elemento. Considerando, ad esempio, il *fragment* in basso della schermata di composizione delle regole, gli elementi di quella *gridView* reagiscono in modo diverso al click, a seconda della loro tipologia. Per gli eventi e le condizioni che richiedono la definizione di parametri aggiuntivi, è prevista la visualizzazione di un *dialogFragment* al *click* dell'utente. Nelle Figure 6.8, 6.9, 6.10, 6.11, 6.12, 6.13 sono mostrati alcuni *dialogFragment* custom creati.

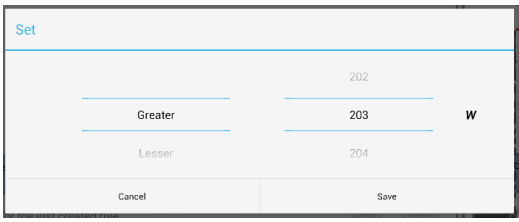

Figura 6.8. *DialogFragment* per il settaggio di un valore

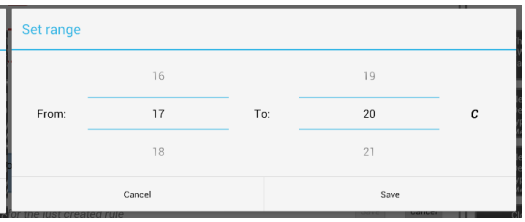

Figura 6.9. *DialogFragment* per il settaggio di un range di valori

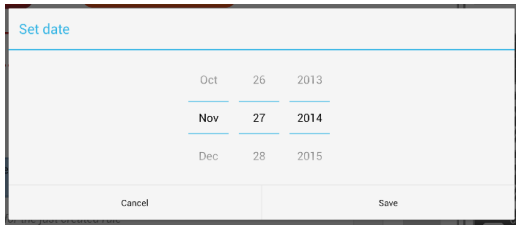

Figura 6.10. *DialogFragment* per il settaggio della data

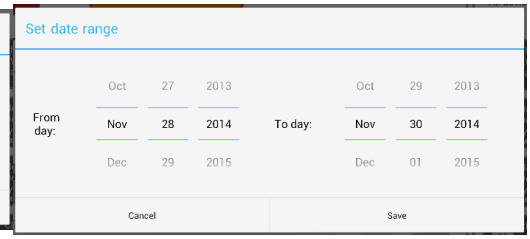

Figura 6.11. *DialogFragment* per il settaggio di un intervallo tra due date.

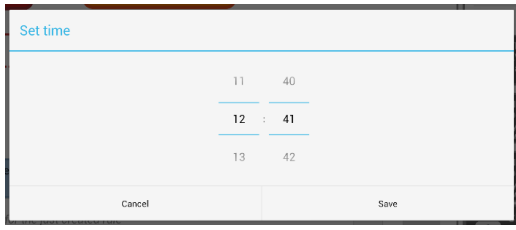

Figura 6.12. *DialogFragment* per il settaggio dell'orologio.

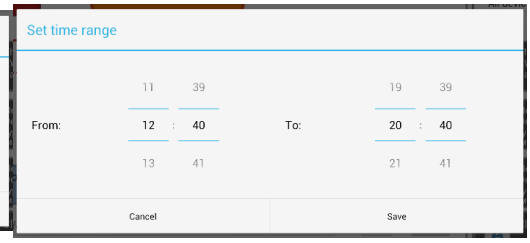

Figura 6.13. *DialogFragment* per il settaggio di un intervallo di tempo.

6.2 Uno sguardo al *Manifest*

Il file *AndroidManifest* è il punto di partenza nello sviluppo di un'applicazione per *Android*. Esso è il file che contiene tutte le informazioni specificate durante la creazione del progetto, l'icona, la versione dell'applicazione, inoltre, all'interno di questo file, sono elencate le *activity* e i *service* dell'applicazione e vi è la dichiarazione dei permessi di cui necessita per funzionare correttamente.

La figura 6.14 mostra l'*outline* del *manifest* dell'applicazione realizzata. L'applicazione è composta da quattro *activity* e due *services*. Le *activity* sono di seguito elencate e descritte:

- *La MainActivityHomeRules* è l'*activity* principale, ad essa è associato il layout della schermata di avvio dell'applicazione (Figura 6.2).
- La *SettingActivity* è l'*activity* a cui è collegata la schermata per l'inserimento dei parametri di configurazione della connessione con *Dog*. Al primo avvio dell'applicazione, dopo l'installazione, questa *activity*

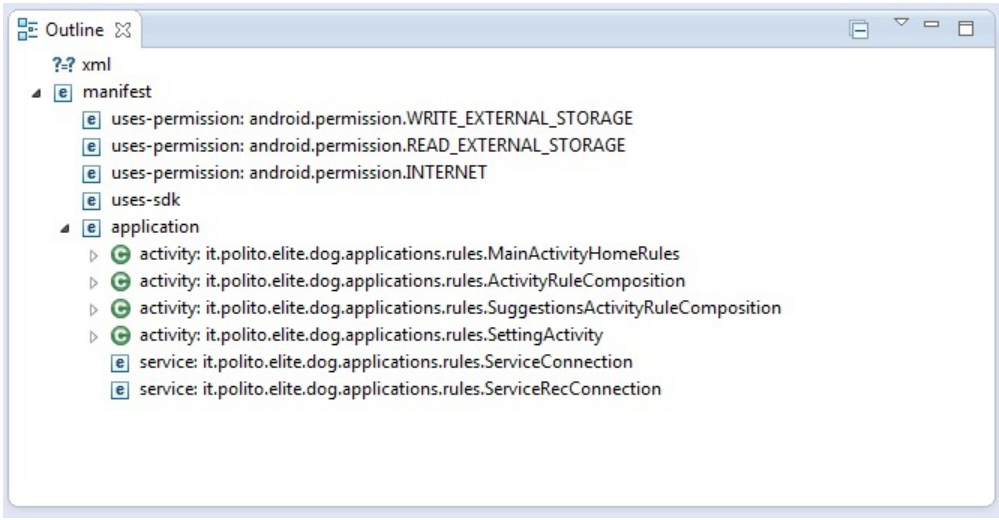

Figura 6.14. L'*outline* del *Manifest* dell'applicazione.

è quella che viene lanciata per prima; una volta che l'utente avrà configurato i parametri di connessione, la SettingActivity non sarà più lanciata nei successivi avvii dell'applicazione, a meno che l'utente non scelga di modificare i parametri di configurazione attraverso l'opzione "*Setting*" dal menù della schermata principale. La scelta di lanciare o meno la *SettingActivity* all'avvio può essere fatta nell'*activity* principale, mediante l'utilizzo delle *SharedPreferences*. La classe *android.content.SharedPreferences* permette agli sviluppatori di salvare dei settaggi applicativi in un file e di utilizzarli nell'applicazione stessa.

- L'*ActivityRuleComposition* è l'*activity* che gestisce la composizione della regola nella modalità di composizione classica. Essa viene lanciata quando l'utente clicca su "add new rule - manual", scegliendo di aggiungere una regola con modalità non guidata.

- La *SuggestionsActivityRuleComposition* è l'*activity* per la gestione della composizione della regola nella modalità guidata. Essa viene lanciata quando l'utente clicca su "add new rule - guided", scegliendo di aggiungere una regola con la modalità facilitata.

L'*ActivityRuleComposition* e la *SuggestionsActivityRuleComposition* utilizzano lo stesso *layout* xml e gli stessi *fragment,* ma si differenziano per l'interazione con i *fragment* stessi e per la presenza di elementi grafici differenti. La *SuggestionsActivityRuleComposition* presenta sempre un *popup* di spiegazione; tale modalità è guidata: l'utente non può scegliere liberamente dove cliccare, molti elementi sono disattivati e abilitati solamente quando è il caso; inoltre questa *activity* non presenta un menù. L'*ActivityRuleComposition* dà invece all'utente la possibilità di creare la regola liberamente, cliccando dove preferisce, senza avere un percorso obbligato da seguire. Essa presenta un menù, da cui si può abilitare la modalità di registrazione interattiva e i suggerimenti. Nella Figura 6.15 è mostrata la schermata con entrambe le opzioni abilitate.

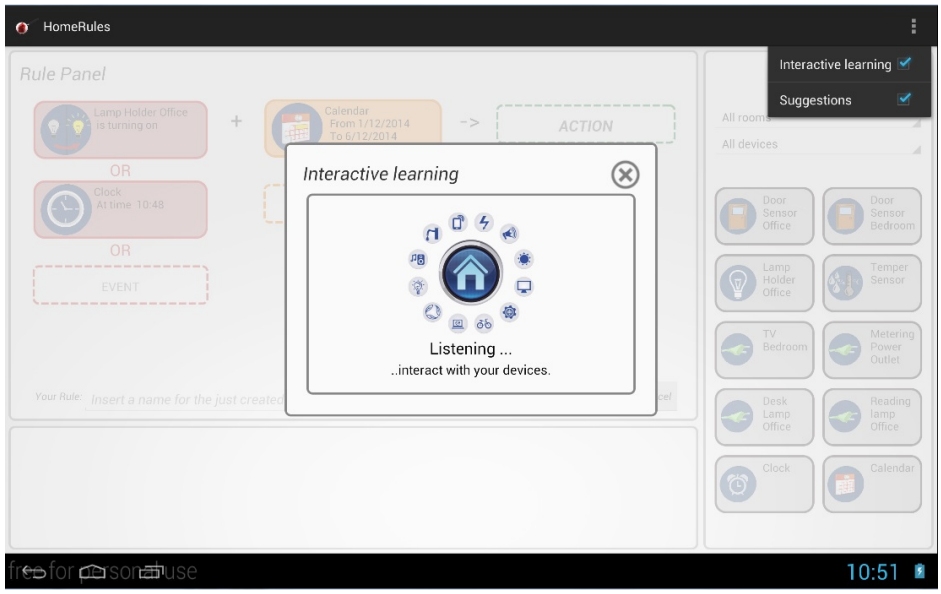

Figura 6.15. La schermata di composizione delle regole con le opzioni di "*Interactive Learning*" e di "*Suggestions*" abilitate.

Di seguito vengono elencati e descritti i *services*:

- Il *ServiceConnection*, è il servizio che viene lanciato all'avvio dell'applicazione; esso effettua la connessione con *Dog* e richiede la configurazione della casa. Quando al servizio saranno notificate le informazioni richieste, quali ad esempio i dispositivi presenti nella casa o il numero di stanze, esso le invierà all'*activity* principale.

- Il *ServiceRecConnection*, è il servizio che viene lanciato quando l'utente sceglie di aggiungere una parte della regola in modalità interattiva. Quando viene attivata la modalità interattiva, tale servizio richiede a *Dog* la ricezione di tutte le notifiche dei dispositivi presenti nella *smart home* e rimane in ascolto delle risposte del sistema.

Nel *Manifest* sono presenti anche i permessi richiesti dall'applicazione realizzata; essi sono:

- *READ EXTERNAL STORAGE*, ovvero il permesso per la lettura dalla microSD esterna;

- *WRITE EXTERNAL STORAGE*, ovvero il permesso per la scrittura sulla microSD esterna;

- *INTERNET*, ovvero il permesso per accedere ed utilizzare la connessione Internet.

I permessi di lettura e di scrittura su memoria sono stati richiesti per consentire il salvataggio delle regole create. L'applicazione, nella versione realizzata, salva i dati in locale, creando un file di tipo JSON[1] che contiene tutte le regole che l'utente ha creato per la sua *smart home*. All'apertura dell'app, il file viene letto e vengono ricavate le informazioni sulle regole presenti; successivamente, ogni volta che una nuova regola viene creata, il file JSON viene aggiornato. Il permesso per accedere ed utilizzare la connessione Internet è fondamentale per la connessione del dispositivo con *Dog* e quindi con la *smart home*.

[1] JSON è l'acronimo di *JavaScript Object Notation*. Si tratta di un formato adatto ad immagazzinare varie tipologie di informazioni e soprattutto a scambiare queste informazioni tra applicazioni client/server.

Nella Figura 6.14 è presente, inoltre, la riga *uses-sdk*, ciò perché nel *Manifest* è stata inserita la dichiarazione dell' *<user-sdk>*. Questo elemento dichiara le diverse compatibilità per android grazie all'utilizzo di *androd:minSdkVersion* e *android:TargetSdkVersion*. Nell'applicazione realizzata l'*SDK* minimo per l'app è *Android 4.0* (*API Level 14*), mentre come la versione più aggiornata su cui l'app può essere installata è *Android 4.4* (*API Level 19*) .

6.3 Scambio di dati tra *fragment*, *activity* e *service*

Come si è visto, l'applicazione consta di quattro *activity* che comunicano, si richiamano e si susseguono tra di loro e di due servizi che, una volta attivati, vengono eseguiti in background e mandano le informazioni alle *activity*.
Tutto ciò richiede un meccanismo che permetta il passaggio di dati e informazioni tra le *activity* e tra le *activity* e i *service*. Nello sviluppo dell'applicazione si è ricorso agli *intent* per consentire la comunicazione tra le *activity* e tra i fragment, nel caso in cui l'*activity* sia composta da più fragment, e per permettere l'avvio dei servizi.
Per ricevere le risposte che il servizio invierà, nell'*activity* è utilizzato un *BroadcastReceiver*. Esso è in ascolto di determinati messaggi, chiamati *Intents Broadcast*, i quali vengono spediti attraverso il metodo *sendBroadcast()*. Il servizio manda quindi un *Intent Broadcast* per notificare all'*activity* che sono arrivate delle notifiche da parte del server *Dog*; l'*activity* implementa il metodo astratto *onReceiver* per rispondere alla notifica ricevuta.
Poiché i dati da scambiare tra *activity*, *fragment* e *service* sono oggetti complessi creati appositamente per l'applicazione, si è dovuto far ricorso alla classe *Parcelable*. Nelle classi degli oggetti custom si è dovuto quindi implementare il metodo *writeToParcel()*, che riceve un oggetto *parcel* nel quale vengono trasferiti gli attributi dell'oggetto custom, e inserire un campo statico chiamato *CREATOR*, che serve per ricostruire la classe che implementa *Parcelable*.

6.4 Struttura dati delle regole e output JSON

La struttura dati di una regola è schematizzata in Figura 6.16. La classe *Rule*, rispecchiando quelle che erano state le scelte progettuali di una sintassi *ECA*, presenta un array di eventi (*EventOfRule*), un array di condizioni (*ConditionOfRule*) e un array di azioni (*ActionOfRule*). La classe *Rule* contiene tutti i metodi di lettura e scrittura delle strutture presenti e implementa la classe *Parcelable*.

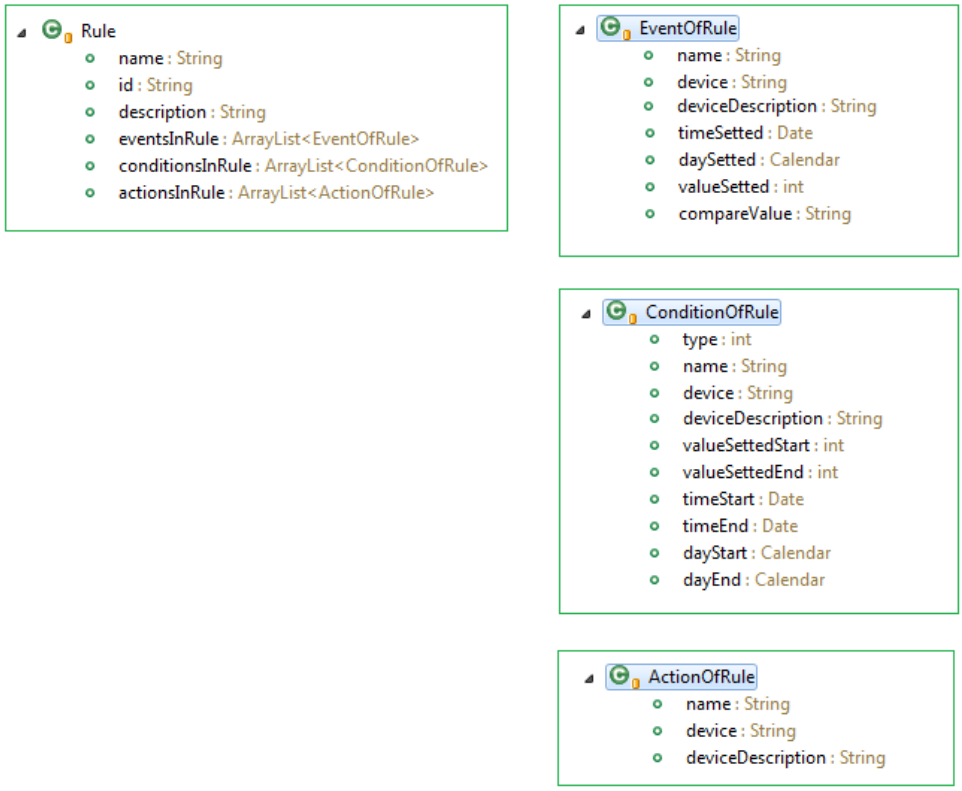

Figura 6.16. La struttura dati di una regola.

La struttura della regola è stata riprodotta nel file JSON, seguendo fedelmente la struttura dati appena presentata. La Figura 6.17 mostra un esem-

```
{
  "Home": {
    "Rules": [
      {
        "nameRule": "turn on reading lamp and desk lamp",
        "eventsInRule": [
          {
            "valueSettedInEvent": 210,
            "deviceInEvent": "MeteringPowerOutlet_4",
            "compareValue": "Lesser",
            "nameEvent": "newActivePowerValue"
          }
        ],
        "descriptionRule": "create in 2014-11-30 at 17:13",
        "conditionsInRule": [
          {
            "timeSettedEndInCondition": "12:10",
            "timeSettedStartInCondition": "08:10",
            "nameCondition": "From...to",
            "deviceInCondition": "Clock"
          },
          {
            "nameCondition": "close",
            "deviceInCondition": "DoorSensor_86"
          }
        ],
        "idRule": "2014-11-30 17:13",
        "actionsInRule": [
          {
            "deviceInAction": "MeteringPowerOutlet_7",
            "nameAction": "on",
          },
          {
            "deviceInAction": "MeteringPowerOutlet_5",
            "nameAction": "on",
          }
        ]
      },
      {
        "nameRule": "close tv Bedroom",
        "eventsInRule": [
          {
            "deviceInEvent": "LampHolder_4",
            "nameEvent": "is turning on"
          }
        ],
        "descriptionRule": "create in 2014-11-30 at 16:41",
        "idRule": "2014-11-30 16:41",
        "actionsInRule": [
          {
            "deviceInAction": "MeteringPowerOutlet_3",
            "nameAction": "off",
          }
        ]
      },
    ],
    "NameHome": "My Home",
    "RoomNumber": 3
  }
```


Figura 6.17. File JSON con le regole presenti nella casa.

pio di file JSON generato dall'applicazione. Alla base della rappresentazione JSON si hanno due tipologie di strutture:

- un insieme di proprietà (ovvero coppie nome-valore), che viene definito oggetto ed è descritto tra parentesi graffe ({ }),
- un elenco ordinato di valori, che è definito array ed è descritto fra parentesi quadre ([ ]).

Premesso ciò, nella struttura JSON creata, "*Home*" è un oggetto che possiede 2 proprietà (*NameHome* e *RoomNumber*) e un array di regole (*Rules*). Ogni regola ha 4 proprietà (*nameRule*, *descriptionRule* e *idRule*,) e può avere 3 array (*eventsInRule*, *conditionsInRule*, *actionsInRule*). Infatti, gli array *eventsInRule* e *actionsInRule* saranno sempre presenti perché il salvataggio di una regola avviene solamente se è presente almeno un evento e un'azione, la presenza invece dell'array *conditionsInRule* dipende dalla regola creata. Gli array *eventsInRule*, *conditionsInRule*, *actionsInRule* hanno al loro interno delle proprietà, ovvero coppie nome-valore, che variano in base al dispositivo e all'evento, alla condizione o all'azione salvata. Nella Figura 6.17 vi è un esempio di file json con la descrizione delle regole presenti nella casa.

La scelta del formato di salvataggio dei dati è ricaduta su JSON per la facilità di scrittura e di analisi del formato stesso e, soprattutto, per favorire una futura possibilità di passaggio dei dati dall'applicazione a un server remoto.

6.5 Organizzazione e riusabilità del codice

Nel realizzare l'applicazione si è posta particolare attenzione a produrre un codice leggibile, ben documentato e riutilizzabile. Il riuso è uno dei più importanti fattori di qualità di un software. Per creare un'applicazione quanto più riutilizzabile si è cercato di introdurre una certa modularità all'interno del progetto. Ogni funzionalità offerta dall'applicazione è stata pensata e implementata in modo da essere indipendente il più possibile dalle altre. Si è cercato infatti di creare metodi ben strutturati, che possano essere riutilizzati anche in altri contesti senza doverne stravolgere la struttura. Per quanto riguarda la parte grafica, l'applicazione è stata realizzata in modo da consentire di apportare eventuali cambiamenti al suo aspetto, in

modo semplice. è possibile infatti modificare gli *shape* presenti nella cartella *drawable* del progetto o crearne di nuovi e cambiare gli stili degli elementi testuali, modificando il file *styles.xml* nella cartella *value*.

Capitolo 7

Conclusione e sviluppi futuri

Con il presente lavoro ci si è posti l'obiettivo di studiare, progettare e realizzare un'applicazione mobile per la definizione di regole per la gestione dei dispositivi presenti in una *smart home*.
Una prima parte del lavoro è stata dedicata ad effettuare un'approfondita lettura e un'attenta analisi della letteratura riguardante interfacce per la composizione di regole in ambienti intelligenti, in particolare quelle rivolte ad utenti non esperti. In seguito, il lavoro si è incentrato sulla progettazione di un'interfaccia mobile chiara, semplice e piacevole da usare per la composizione di regole, cercando di soddisfare le esigenze di tutti gli utenti, sia esperti che, soprattutto, meno esperti. In aggiunta alla modalità classica di composizione delle regole, si è inoltre ideata una seconda modalità di composizione che consente anche l'interazione dell'utente con l'ambiente circostante, denominata "*Interactive learning*". Focus della tesi è stato l'utilizzo di una metodologia di progettazione incentrata sull'utente (*user-centered*), che prevede il coinvolgimento degli utenti nelle varie fasi di progettazione e verifica. Seguendo tale metodologia, nella fase di progettazione è stato effettuato, attraverso l'utilizzo di prototipi e *mockup*, un test di usabilità che ha consentito di confrontare due proposte di interfacce, una basata su operazioni di trascinamento (*drag & drop*), l'altra basata su operazioni di selezione (*click*). I partecipanti al test, comprendenti utenti tecnici e non tecnici in pari numero, hanno manifestato una netta preferenza per l'interfaccia basata sul click, considerandola più semplice, comprensibile e facile da utilizzare. Alla luce di tali risultati, l'interfaccia *drag & drop* è stata

abbandonata e, nella successiva fase di implementazione, ci si è concentrati sullo sviluppo dell'interfaccia basata sulle operazioni di *click*.
Il lavoro svolto ha portato al raggiungimento degli obiettivi prefissati, schematizzati nella Tabella 7.1.

Obiettivi prefissati
Utilizzare una grammatica semplice e intuitiva per la creazione delle regole.
Creare un'interfaccia mobile chiara, facile e piacevole da usare per la composizione di regole.
Utilizzare una metodologia di progettazione incentrata sull'utente (user-centered).
Creare una modalità di composizione delle regole interattiva.

Tabella 7.1. Obiettivi prefissati e raggiunti.

Facendo un confronto con i lavori presenti in letteratura, l'applicazione realizzata presenta molte delle caratteristiche che sono allo stato attuale dell'arte. Nella tabella 7.2 alcune delle caratteristiche dell'applicazione sviluppata, denominata *Home Rules*, sono confrontante con quelle di altri sistemi di composizione delle regole. Tra le caratteristiche più salienti dell'applicazione sviluppata si annoverano: l'essere progettata per dispositivi mobile, la presenza di un modello di struttura delle regole di tipo *ECA*, l'utilizzo di un'interfaccia tangibile (modalità interattiva). La presenza di una modalità di registrazione interattiva è un importante punto di forza dell'applicazione, in quanto rappresenta una delle più recenti innovazioni nell'ambito della creazione delle regole per sistemi intelligenti.

Nome	Modello *ECA*	Interfaccia tangibile	Mobile	Web/ Desktop	Altre informazioni
Home Rules	sì	sì	sì	no	le regole possono essere costruite in autonomia o con modalità guidata, nella modalità non guidata sono presenti anche le opzioni di "Interactive Learning" ed i suggerimenti
iCAP	parzialmente (EA)	no	no	sì	
e-Lite	sì	no	no	sì	
Nakagawa	sì	no	no	sì	uso delle cinque dimensioni semantiche W4H e due topologie di regole: Join-type e FORK-type
Garcìa-Herranz	sì	no	no	sì	aggiunta TIMER tra regole
GALLAG Strip	parzialmente (EA)	sì	sì	sì	applicazione vista come sequenza dei frame azione, risposta e tempodata

Continua alla pagina successiva...

Continua dalla pagina precedente...

Media Cube	no	sì	no	no	
SiteView	parzialmente	sì	no	sì	quando un utente specifica un set di condizioni, il sistema mostra le regole che matchano con il set specificato
PiP	parzialmente (EA)	si	sì	sì	
HomeMaestro	parzialmente (EA)	sì	sì	sì	modalità "record" con regole stile "when/then"
DiamondHelp	no	no	no	sì	paradigma della "conversazione collaborativa", le istruzioni possono essere fornite in autonomia o tramite tutoraggio

Tabella 7.2: Riepilogo caratteristiche sistemi di composizione regole aggiornato con *Home Rules*

7.1 Sviluppi futuri

L'applicazione progettata ha le potenzialità per diventare la base di partenza per altre applicazioni per la gestione e il controllo di case intelligenti. Grazie alla sua struttura modulare, essa ben si presta ad eventuali miglioramenti, modifiche e funzionalità aggiuntive.
Alcune idee di miglioramenti che potrebbero essere apprezzati dagli utenti comprendono:

- Possibilità di modificare e cancellare regole già esistenti, al momento solo consultabili, nella schermata di welcome;
- Fornire una visualizzazione delle regole su un calendario in modo da permettere all'utente di visualizzare le sue regole avendo anche un riferimento temporale;
- Prevedere una schermata con una mappa tridimensionale della casa che mostra dove sono collocati i dispositivi e le informazioni su di essi;
- Permettere l'abilitazione delle regole o la disattivazione delle stesse in particolari occasioni. Ad esempio, si potrebbe prevedere la disattivazione di alcune regole nel fine settimana;
- Possibilità di arricchire l'applicazione con impostazioni di personalizzazione di stili e colori.
- Migliorare le prestazioni della modalità interattiva, ad esempio tramite l'aggiunta di comandi vocali.

Un test di utilizzo dell'applicazione su un numero più vasto di utenti potrebbe consentire di valutare il successo che l'applicazione riscuote, verificare se i risultati del test di usabilità effettuati sui prototipi vengono confermati una volta che l'applicazione è stata realizzata ed avere suggerimenti da parte degli utenti stessi che possono contribuire a migliorare ulteriormente il prodotto finale.

Bibliografia

[1] Dey Anind K., Timothy Sohn, Sara Streng, and Justin Kodama. "iCAP: Interactive Prototyping of Context-Aware Applications." In Pervasive Computing, 254-71. Springer, 2006. http://link.springer.com/chapter/10.1007/11748625_16.

[2] Bonino Dario, Fulvio Corno, and Luigi De Russis. "A User-Friendly Interface for Rules Composition in Intelligent Environments." In Ambient Intelligence-Software and Applications, 213-17. Springer, 2011. http://link.springer.com/chapter/10.1007/978-3-642-19937-0_27.

[3] Nakagawa T., C. Doi, K. Ohta, and H. Inamura. "Customizable Context Detection for ECA Rule-Based Context-Aware Applications." ICMU, May 30 (2012).

[4] Xiao Jin, and Raouf Boutaba. "The Design and Implementation of an Energy-Smart Home in Korea." *Journal of Computing Science and Engineering* 7, no. 3 (September 30, 2013).

[5] Garcìa-Herranz Manuel, Pablo A. Haya, and Xavier Alamán. "Towards a Ubiquitous End-User Programming System for Smart Spaces." *J. UCS* 16, no. 12 (2010).

[6] Garcìa-Herranz Manuel, Pablo A. Haya, Abraham Esquivel, Germán Montoro, and Xavier Alamán. "*Semi-Automation in Perceptive Environments: A Rule-Based Agents Proposal*." VII Cong. Int. Interacción Persona-Ordenador, 2006 http://www.ii.uam.es/~montoro/publications/AIPO06.pdf.

[7] Garcìa-Herranz Manuel, Pablo A. Haya, Abraham Esquivel, Germán Montoro, and Xavier Alamán. "Easing the Smart Home: Semi-Automatic Adaptation in Perceptive Environments." *J. UCS* 14, no. 9 (2008).

[8] Ur Blase, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. "Practical Trigger-Action Programming in the Smart Home." 803-12. ACM Press, 2014. doi:10.1145/2556288.2557420.

[9] Lee Jisoo, Luis Garduño, Erin Walker, and Winslow Burleson. "A Tangible Programming Tool for Creation of Context-Aware Applications." 391. ACM Press, 2013. doi:10.1145/2493432.2493483.

[10] Blackwell Alan F., and Rob Hague. "Designing a Programming Language for Home Automation." In *Proceedings of the 13th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2001)*, 85-103, 2001. http://www.ppig.org/papers/13th-blackwell.pdf.

[11] Beckmann Chris, and Anind Dey. "Siteview: Tangibly Programming Active Environments with Predictive Visualization." In *Adjunct Proceedings of UbiComp*, 167-68, 2003. http://intel-research.net/Publications/Berkeley/070920031125_143.pdf.

[12] Chin Jeannette S., Vic Callaghan, and Graham Clarke. "A Pervasive Computing Programming Approach for Non-Technical Users." In *Pervasive Computing and Applications, 2006 1st International Symposium on*, 235-40. IEEE, 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4079146.

[13] Chin Jeannette S., Vic Callaghan, and Graham Clarke. "An End-User Programming Paradigm for Pervasive Computing Applications." In *Pervasive Services, 2006 ACS/IEEE International Conference on*, 325-28. IEEE, 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1652254.

[14] Pena Rios, Anasol C., Jeannette Shiaw-Yuan Chin, and Victor L. Callaghan. "A Web Based Approach to Virtual Appliance Creation, Programming and Management." In *Intelligent Environments (IE), 2010 Sixth International Conference on*, 174-77. IEEE, 2010. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5673933.

[15] Shaun Salzberg. HomeMaestro. http://www.shaunsalzberg.com/medialab/homemaestro.

[16] Paul Wisner and Dimitris N. Kalofonos. "A Framework for End-User Programming of Smart Homes Using Mobile Devices." In *Proceedings of the 4th IEEE Consumer Communications and Networking Conference CCNC*. Vol. 7. Citeseer, 2007. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.1251&rep=rep1&type=pdf.

[17] Rich Charles, Candy Sidner, Neal Lesh, Andrew Garland, Shane Booth, and Markus Chimani. "DiamondHelp: A Graphical User Interface Framework for Human-Computer Collaboration." In *IEEE International Conference on Distributed Computing Systems Workshops*, 514-19. 2005. http://www.ae.uni-jena.de/alenmedia/dokumente/iwsawc05_preprint_diamondhelp_pdf.pdf.

[18] Rich Charles, and Candace L. Sidner. "DiamondHelp: A Generic Collaborative Task Guidance System." *AI Magazine* 28, no. 2 (2007).

[19] Dario Bonino and Fulvio Corno. "DogOnt - Ontology Modeling for Intelligent Domotic Environments." In *International Semantic Web Conference*, number 5318 in LNCS, pages 790-803. Springer-Verlag, October 2008.

[20] De Russis, Luigi. "Interacting with Smart Environments: Users, Interfaces, and Devices." Politecnico di Torino, 2014. http://porto.polito.it/2536887/1/main_thesis.pdf

[21] D. Bonino, E. Castellina, and F. Corno. "The dog gateway: enabling ontology-based intelligent domotic environments." In *Consumer Electronics, IEEE Transactions on*, 54(4):1656-1664, 2008.

[22] Android Open Source Project http://source.android.com/source/index.html

[23] ISO 13407: *Human-centered design process* http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197 http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=52075