DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Corso di Laurea

in Ingegneria Informatica Magistrale

Relazione Finale

Integrazione di funzionalità di Trigger-Action Programming nel Gemello Digitale di una Smart Home

Relatore: Prof.sa Daniela Fogli

Correlatore: Prof.sa Barbara Barricelli

Correlatore: Ing. Davide Guizzardi

Studente:

Giacomo Golino (719210)

Anno Accademico 2024/2025

Indice

Introduzione

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1. Titolo cap - 1

1.1 Introduzione

Da introdurre gli obiettivi e il contesto della tesi, ed eventualmente la sua struttura.

2. Stato dell'arte

In questo capitolo verranno analizzati e approfonditi i **metodi** e le **applicazioni** per la creazione di *automazioni* (chiamate anche *routine*) in ecosistemi *Internet of Things* (IoT). In particolare, verranno affrontate con maggiore attenzione le *smart home* come ecosistemi IoT, andando ad analizzare quali sono le principali piattaforme utilizzate per la creazione di *automazioni*.

Si definiscono ***automazioni*** o *routine* degli insiemi strutturati di istruzioni che, una volta impostati, consentono di eseguire automaticamente una serie di azioni al verificarsi di determinati eventi (o *trigger*). Tali azioni possono comprendere operazioni ripetitive, come l'accensione o lo spegnimento di dispositivi smart, oppure processi più complessi che coinvolgono diversi servizi, con l'obiettivo finale di semplificare e ottimizzare la gestione dell'ecosistema.

Le ***azioni*** (*action*) costituiscono la parte esecutiva di una routine: al verificarsi dell'evento scatenante (***trigger***), l'azione definita viene avviata per compiere l'operazione desiderata. Può trattarsi di semplici attività oppure di processi articolati che si integrano con servizi diversi.

All'interno di questo capitolo verranno quindi presentati:

- I concetti chiave relativi alle automazioni/routine e la loro importanza nel contesto IoT.
- Le principali piattaforme sul mercato che consentono di creare routine, nello specifico **Amazon Alexa**, **Google Home** e **Apple Casa**.
- Altri sistemi per il **Trigger-Action Programming** che si sono recentemente affermati e alcuni approcci proposti nella letteratura scientifica.

2.1 Concetti di Base

Le *automazioni* o *routine* nel contesto dell'IoT consistono in regole di tipo *trigger-action*, dove un evento scatenante (ad esempio, un orario predefinito, un comando vocale o una condizione ambientale rilevata da un sensore) provoca una o più azioni (come l'accensione di una luce, l'avvio di un elettrodomestico, l'invio di una notifica, ecc.). Questa logica di base, semplice da comprendere, si è rivelata estremamente potente e versatile in quanto:

- Riduce la necessità di interventi manuali da parte dell'utente, automatizzando **task** ripetitivi.
- Consente di *personalizzare* lo spazio domestico in base alle preferenze e alle abitudini di ognuno.
- Si presta a una gestione modulare: i vari eventi e azioni possono essere combinati per creare routine più avanzate.

Alcuni studi hanno evidenziato come la programmazione di tipo *trigger-action* (TAP) si adatti alla maggior parte delle esigenze di automazione espresse dagli utenti [1, 2], risultando intuitiva anche per chi non possiede competenze tecniche avanzate. In particolare, l'analisi di oltre 67.000 programmi condivisi su `IFTTT` e un test di usabilità condotto su 226 partecipanti conferma che, grazie alla semplicità di combinare in modo flessibile molteplici *trigger* e *action*, la curva di apprendimento rimane bassa, favorendo un'ampia adozione [1].

Nell'immagine qui di seguito (fig. 2.1) vengono mostrati alcuni esempi di quelli che potrebbero essere dei **trigger** e delle **action**.

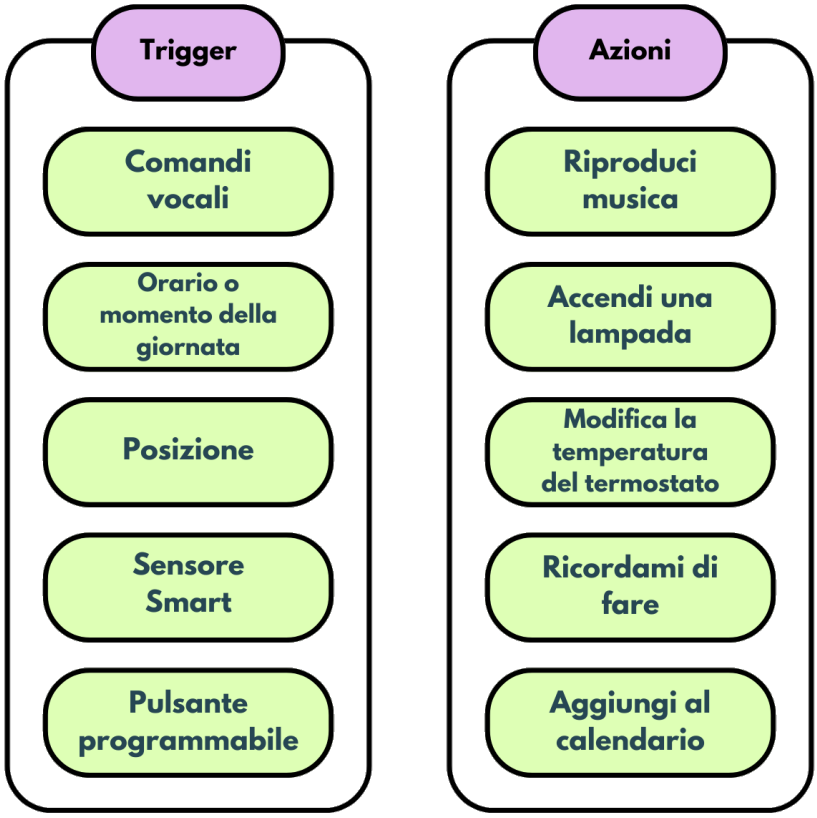

Figura 2.1: Esempio di trigger e azioni

2.1.1 Requisiti di una buona automazione

Per risultare efficace [3], un'automazione deve essere:

- **Facile da creare e gestire**: l'utente finale (che spesso non possiede competenze di programmazione) deve poter definire e modificare le routine in maniera intuitiva.
- **Affidabile**: deve funzionare in modo consistente nel tempo, senza errori o interruzioni non previste.
- **Adattabile**: dev'essere in grado di gestire modifiche nelle preferenze dell'utente, nelle caratteristiche dei dispositivi o nell'assetto del sistema.
- **Sicura**: in un ecosistema connesso, la *privacy* e la *sicurezza* di reti e dispositivi sono fondamentali.

2.2 Panoramica sulle piattaforme principali

Storicamente, le automazioni si basavano principalmente su *timer* o sensori semplici (come termostati o rilevatori di movimento) per poter funzionare. Al giorno d'oggi, invece, le piattaforme commerciali (e non) offrono soluzioni ben diverse rispetto a quelle di una volta. Nello specifico attualmente vengono utilizzate:

- Interfacce grafiche *drag-and-drop* per comporre in modo visivo le regole.
- Integrazione con *assistenti vocali* (es. Amazon Alexa, Google Assistant e Apple Siri) per impostare routine con frasi naturali.
- Utilizzo di tecniche di *machine learning* e *context awareness* per proporre automazioni "intelligenti— o per adattare le routine al comportamento dell'utente.

Il panorama degli strumenti per la creazione di automazioni è estremamente ampio. In particolare, i sistemi **Amazon Alexa**, **Google Home** e **Apple Casa** sono le principali piattaforme commerciali per la gestione domestica, ciascuno caratterizzato da specifiche architetture, protocolli di comunicazione e modelli di integrazione con dispositivi di terze parti.

2.2.1 Amazon Alexa

Amazon Alexa è un assistente vocale lanciato inizialmente su dispositivi *Echo*, poi rapidamente esteso a numerosi dispositivi di terze parti. Le *routine* di Alexa possono essere create tramite:

- L'app Alexa su smartphone.
- Interazione vocale diretta (nello specifico grazie alla sottoscrizione del servizio *Alexa+*, disponibile al momento solo negli USA).
- *Skill* aggiuntive, sviluppate da terze parti.

Inoltre, negli ultimi anni, Amazon ha introdotto:

- **Riconoscimento di suoni specifici**: ad esempio, la piattaforma è in grado di distinguere il suono di vetri rotti, avvisando di conseguenza l'utente. È

importante sottolineare che questa funzionalità è disponibile al momento solamente su **Echo smart speakers** e sui dispotivi **Echo smart displays**.

- **Hunches**: questa funzionalità, se attivata attraverso l'applicazione, consente ad Alexa di apprendere determinate routine ricorrenti, così da poter avvisare l'utente qualora tali azioni non vengano eseguite. Ad esempio, Alexa potrebbe imparare che ogni volta che un utente esce di casa, è solito spegnere tutte le luci e chiudere la porta. Nel caso in cui l'utente dovesse dimenticare di svolgere una o entrambe le azioni, il sistema lo notificherebbe, chiedendo se è necessario eseguire queste azioni.

- **Integrazioni con servizi esterni**: questa funzionalità consente l'integrazione con servizi di terze parti come calendari, servizi di musica in streaming o applicazioni di messaggistica.

2.2.2 Google Home

Google Home si basa sull'assistente vocale Google Assistant, sfruttando l'ampia rete di servizi Google. Le automazioni possono derivare dall'utilizzo di:

- Comandi vocali per la creazione e gestione delle *routine*. Gli utenti possono avviare la configurazione di una routine semplicemente pronunciando frasi come "Ok Google, crea una routine per...—, dopodiché Google Assistant guiderà attraverso i passaggi per impostare un *trigger* e una o più *action*. Questo metodo semplifica notevolmente l'automazione per chi preferisce un'interazione naturale senza dover passare obbligatoriamente da un'applicazione.
- L'app Google Home, che permette di definire *trigger* e *action* utilizzando dispositivi compatibili.
- L'integrazione con Google Calendar o Google Maps (permettendo, ad esempio, di attivare una routine se si è vicini a casa).

L'ecosistema Google si è evoluto introducendo:

- **Eventi geolocalizzati (geofencing)**: la routine si attiva o disattiva in base alla posizione dell'utente. Il *geofencing* è una tecnologia basata sulla geolocalizzazione che permette di definire un'area virtuale attorno a una posizione fisica (come casa ad esempio). Quando il dispositivo dell'utente entra o esce da questa zona predefinita, viene attivata una routine automatica. Ad esempio, si può configurare l'accensione delle luci smart quando si arriva a casa o la disattivazione del riscaldamento quando si lascia l'abitazione.
- **Riconoscimento vocale avanzato (Voice Match)**: l'assistente riconosce la voce di diverse persone e personalizza alcune automazioni (ad esempio, la musica preferita) in base al profilo.

2.2.3 Apple Casa

Apple Casa è l'applicazione di Apple per la gestione della domotica, basata sulla piattaforma HomeKit. Rispetto a soluzioni più aperte, punta molto sulla semplicità d'uso e su elevati standard di privacy e sicurezza. Di seguito sono riportate alcune delle principali caratteristiche di questa piattaforma:

- **App Casa su iOS**: qui si impostano automazioni basate sull'orario, sul rilevamento di un sensore, o sull'entrata/uscita da una determinata area geografica.
- **Supporto a scene e stanze**: l'utente può creare "scene— (sequenze di azioni) o associare dispositivi a stanze, semplificando la gestione.
- **Apertura verso standard di connettività**: Apple ha recentemente esteso il supporto a protocolli come Matter, garantendo più integrazione con prodotti non-Apple.

Le ultime versioni di iOS hanno introdotto:

- **Rilevamento della presenza via Apple Watch**: se l'utente indossa un Apple Watch, il sistema può capire se si trova effettivamente in casa (o nei dintorni).

2.3 Metodi e Applicazioni per la Creazione di Automazioni

Oltre alle piattaforme citate, esistono molteplici approcci al *Trigger-Action Programming*, come visto in [2, 4, 5]. Nei seguenti sottocapitoli verrà proposta un'analisi delle caratteristiche principali delle piattaforme più famose che permettono la *Trigger-Action Programming*.

2.3.1 Applicazioni per il Trigger-Action Programming

IFTTT

Acronimo di **If This Then That** è una delle prime piattaforme di automazione su larga scala, nota per la sua interfaccia intuitiva e la capacità di integrare servizi eterogenei senza richiedere competenze tecniche avanzate [1]. È possibile lavorare con questa piattaforma direttamente dal web, accedendo al sito: `ifttt.com`.
Gli utenti possono creare semplici regole *IF-THEN* (se accade un evento (*IF*), allora esegui un'azione (*THEN*)) selezionando trigger e azioni predefinite. Ad esempio, si può configurare l'invio automatico di un'email quando viene pubblicato un nuovo post su un blog. Tuttavia, rispetto ad altre soluzioni, IFTTT presenta limitazioni in termini di personalizzazione e controllo, e alcune funzionalità avanzate sono disponibili solo nella versione a pagamento.
In (fig. 2.2) viene mostrato un esempio della creazione di un'automazione che, al verificarsi di una specifica condizione meteo (*trigger*) invia una notifica all'utente (*action*).

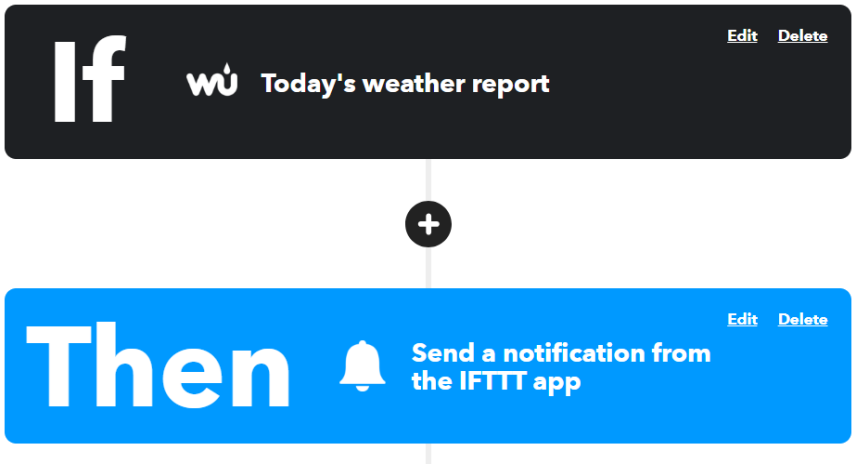

Figura 2.2: Esempio di creazione di una automazione con l'ausilio di IFTTT

Home Assistant

È una piattaforma open-source per la gestione avanzata della domotica, particolarmente apprezzata dagli utenti più esperti per la sua elevata configurabilità e il supporto a un'ampia gamma di dispositivi. È possibile reperire la documentazione di questa piattaforma sul sito: `www.home-assistant.io`.

Le automazioni possono essere create sia tramite un'interfaccia grafica (Graphic User Interface, GUI) che mediante codice YAML, consentendo logiche più sofisticate rispetto a soluzioni come IFTTT. Nell'esempio riportato in (fig. 2.3), è possibile vedere come è stato possibile programmare una sequenza di eventi condizionali, come l'accensione delle luci in una particolare stanza quando un sensore di un garage rileva l'apertura del suddetto, solo se il sole è già calato, mediante l'interfaccia grafica di Home Assistant.

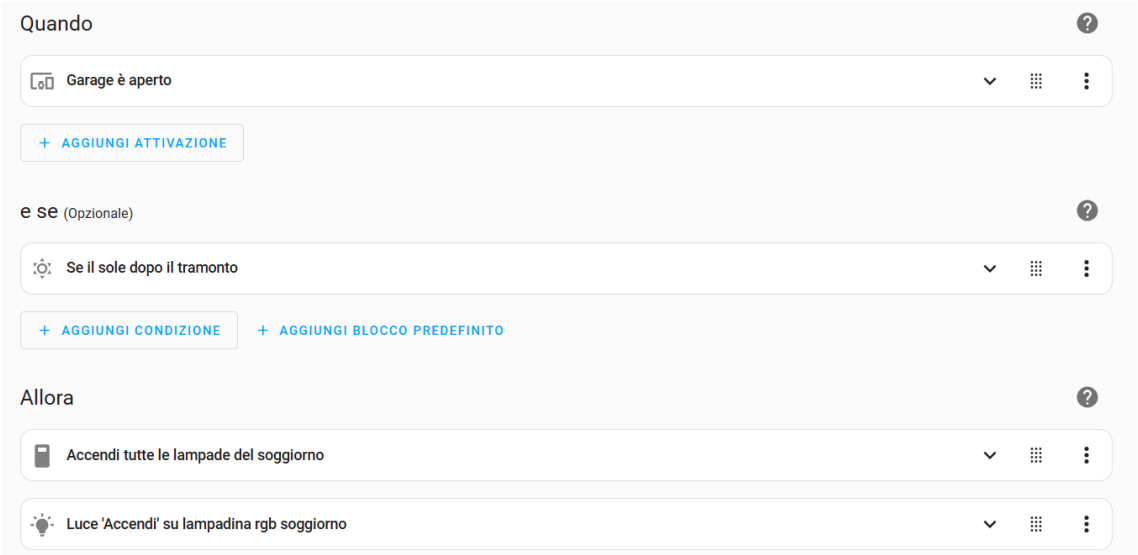

Figura 2.3: Esempio di creazione di un'automazione mediante GUI

La creazione di automazioni mediante codice YAML può risultare però più complessa per degli utenti non esperti, rispetto a quella tramite GUI. Nell'esempio di seguito riportato in (fig. 2.4) è illustrato come sia possibile definire un'automazione basata sulla geolocalizzazione degli utenti. In particolare, l'evento scatenante (`trigger`) si verifica quando uno dei dispositivi tracciati cambia stato da `not_home` a `home`, attivando così le azioni corrispondenti dopo un ritardo di un minuto.

```yaml
automation:
  triggers:
    - trigger: state
      entity_id:
        - device_tracker.paulus
        - device_tracker.anne_therese
      # Optional
      from: "not_home"
      # Optional
      to: "home"
      # If given, will trigger when the condition has been true for X time; you can also use days and
milliseconds.
      for:
        hours: 0
        minutes: 1
        seconds: 0
```

Figura 2.4: Esempio di codice YAML per la creazione di un'automazione

Node-RED

È un ambiente di sviluppo visuale basato su flussi, che consente di creare automazioni collegando elementi modulari chiamati "nodi—. Ogni nodo rappresenta un trigger, un'elaborazione o un'azione, e può essere connesso agli altri in un diagramma di flusso grafico. Questo approccio offre un controllo più avanzato rispetto a IFTTT e una maggiore semplicità rispetto alla configurazione testuale di Home Assistant. Ad esempio, un nodo può ricevere dati da un sensore di temperatura, elaborarli con una soglia personalizzata e inviare una notifica solo se viene superato un valore critico. Tuttavia, essendo una piattaforma più tecnica, richiede una certa familiarità con concetti di programmazione e rete.

È possibile reperire la documentazione per l'utilizzo di questa piattaforma presso il sito: `www.nodered.org`. In (fig. 2.5) si mostra un flusso di Node-RED che, dopo aver recuperato le condizioni meteorologiche e la temperatura corrente, interroga uno smartwatch per rilevare il numero di passi compiuti dall'utente durante la giornata; se tale valore è inferiore a 1000, il sistema invia automaticamente un'e-mail di notifica.

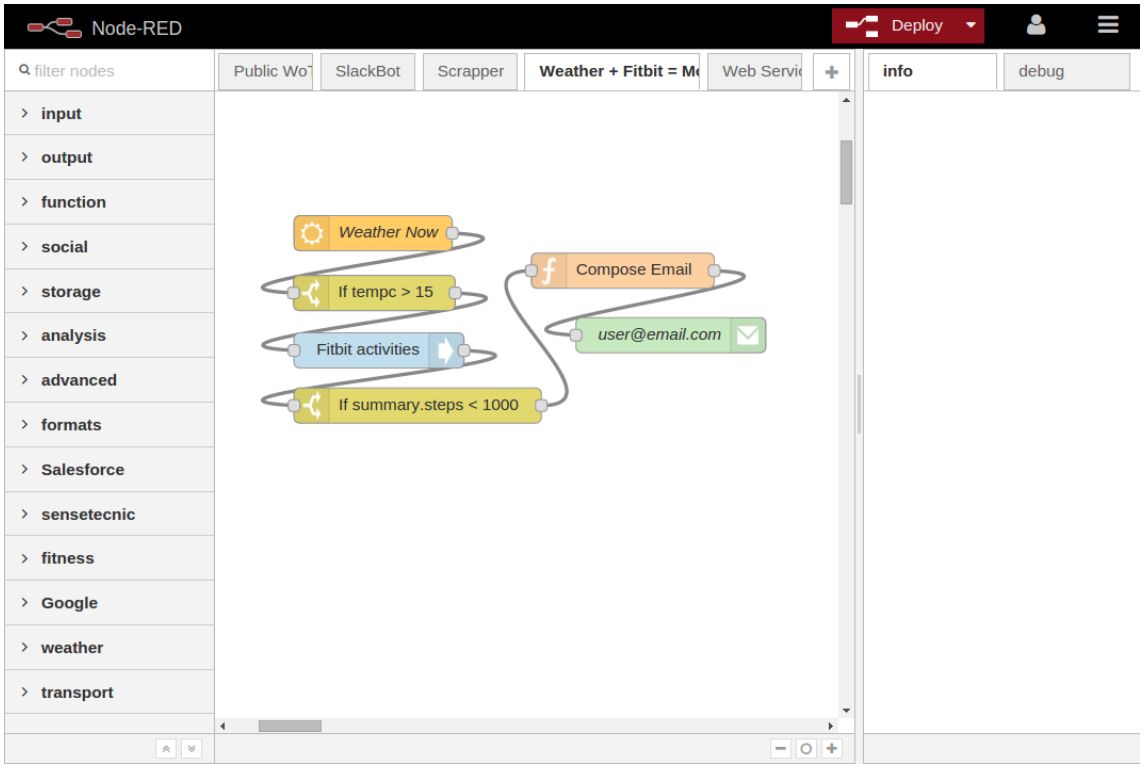

Figura 2.5: Esempio di generazione di un'automazione con Node-RED

2.3.2 Confronto fra le tre piattaforme

Per confrontare IFTTT, Home Assistant e Node-RED sono state considerate metriche analizzate in [6, 7, 8] come:

a) **Facilità d'uso e curva di apprendimento**
b) **Potenza espressiva delle regole**
c) **Modalità di esecuzione (cloud o locale)**
d) **Ampiezza delle integrazioni disponibili**
e) **Costo di adozione**

Studi recenti sottolineano come gli strumenti interamente visuali (es. IFTTT) favoriscano la rapidità di prototipazione, ma impongano limiti man mano che cresce la complessità logica [6]. Soluzioni a flusso come Node-RED colmano questo gap offrendo un controllo granulare senza richiedere la completa transizione a un approccio testuale; Home Assistant si colloca a metà strada grazie alla combinazione

di un'interfaccia grafica con la possibilità dell'utilizzo di YAML, mantenendo una soglia di ingresso relativamente accessibile ma consentendo automazioni avanzate [7, 8].

Criterio	IFTTT	Home Assistant	Node-RED
Facilità d'uso	Alta	Media	Media-bassa
Potenza logica	Bassa — singola regola	Alta — automazioni / script	Alta — flussi arbitrari
Esecuzione	Cloud	Locale / Cloud	Locale
Integrazioni	$\sim$ 700 servizi cloud	$\sim$ 2500 integrazioni	$\sim$ 3500 nodi
Costo	Freemium	Open-source	Open-source

Tabella 2.1: Confronto fra IFTTT, Home Assistant e Node-RED

In sintesi, *IFTTT* si distingue per la rapidità con cui consente di realizzare automazioni semplici; *Home Assistant* eccelle nei casi domestici complessi grazie all'ampio catalogo di integrazioni e alla combinazione di interfaccia grafica e configurazioni testuali; *Node-RED* offre la massima flessibilità architetturale, risultando ideale per scenari IoT multi-dominio che richiedono logiche di flusso elaborate.

2.3.3 Approcci nella letteratura scientifica

Negli ultimi anni, la ricerca si è concentrata su come rendere accessibile la creazione di routine e flussi di automazione anche a quegli utenti che non possiedono competenze di programmazione (*End-User Development*, EUD). In particolare, sono emersi due approcci complementari per aiutare l'utente nella definizione di regole e workflow: le *interfacce visuali* e le *interfacce conversazionali* [2].
In seguito si è poi deciso di provare a combinare i punti di forza di entrambi gli approcci dando vita alle *interfacce multimodali*, che consentono di sfruttare

simultaneamente modalità di input e output sia visuali sia conversazionali, offrendo così un'esperienza più ricca e flessibile per l'utente.

Interfacce Visuali

Le interfacce visuali hanno come principale vantaggio la rappresentazione esplicita dei *trigger* e delle *action*, tramite blocchi e connettori, o attraverso grafiche facilmente riconoscibili dall'utente finale. Paradigmi come il *flow-based programming* e il *block-based programming* permettono di trascinare e collegare componenti, rendendo chiari i passaggi di input e output e minimizzando gli errori di configurazione. Inoltre, rappresentazioni iconiche (ad esempio una lampadina o un termostato) possono semplificare l'interazione e offrire un riscontro immediato di ciò che si sta progettando.

Un semplice esempio per la creazione di un'automazione mediante *interfaccia visuale* è possibile vederla in (fig. 2.5).

Vantaggi Gli editor grafici offrono una *bassa soglia di ingresso* per gli utenti non programmatori: negli studi controllati il tempo di apprendimento medio è inferiore a 15 minuti [9]. Nel progetto, gli utenti coinvolti hanno definito la programmazione come «estremamente facile», segnalando un immediato senso di successo nella creazione di regole semplici [10].

Svantaggi. All'aumentare della complessità, i benefici iniziali si attenuano: l'espressione di condizioni composte o di regole multi-dispositivo fa crescere tempi di completamento e tassi d'errore [10].
Ambienti di sviluppo come Node-RED (vedasi 2.3.1) mostrano problemi di *scalabilità visiva*: poche decine di nodi superano lo spazio dello schermo, generando sovraccarico cognitivo e rendendo difficile comprendere le cause di un'azione [11].
Ulteriori limiti emersi riguardano:

- L'ambiguità tra eventi e stati, che confonde gli utenti [11].
- La mancanza di supporto multi-utente e di possibilità di simulazione, evidenziata da Caivano et al. nel confronto Atooma/IFTTT/Tasker [12].
- L'assenza di meccanismi di risoluzione di conflitti: loop, collisioni e ridondanze possono produrre comportamenti inattesi [13, 14].

In sintesi, le interfacce visuali rimangono efficaci per regole semplici e utenti alle prime armi, ma necessitano di funzioni di astrazione, filtraggio e analisi per restare usabili in scenari reali e dinamici.

Interfacce Conversazionali

Parallelamente all'approccio visuale, diversi studi hanno approfondito l'efficacia delle cosiddette *interfacce conversazionali* [2], che consentono la definizione di routine mediante interazione in linguaggio naturale con assistenti vocali (ad es. Amazon Alexa, Google Assistant, Apple Siri). In questo caso, la creazione di una regola o routine avviene attraverso un dialogo guidato, dove l'utente può specificare trigger e azioni utilizzando frasi colloquiali [2, 15].

L'utilizzo del linguaggio naturale può apparire più intuitivo rispetto a un'interfaccia a blocchi, specialmente per utenti poco esperti, poiché non richiede di apprendere una sintassi formale o di manipolare elementi grafici. Tuttavia, la sfida maggiore risiede nella corretta interpretazione delle richieste (*Natural Language Understanding, NLU*) e nella necessità di meccanismi di conferma o riepilogo, che riducano il rischio di errori o ambiguità.

Viene riportato di seguito un esempio di un'interazione vocale creata partendo dal sistema analizzato in [2].

Esempio di una possibile interazione vocale:

> **Utente**: "Alexa, voglio creare una nuova routine.—
> **Assistente**: "Va bene, come desideri attivare la routine? Puoi impo-

stare un comando vocale, un giorno e orario specifici, oppure l'alba o il tramonto.—

Utente: "Imposta il comando vocale.—

Assistente: "Hai scelto di attivare la routine tramite un comando vocale. Dimmi la frase che vuoi usare per avviarla.—

Utente: "Accendi luci salotto.—

Assistente: "Perfetto, hai impostato il comando vocale 'Accendi luci salotto.' Ora, desideri aggiungere una o più azioni alla routine?—

Utente: "Sì, aggiungi il comando di accendere le luci del salotto.—

Assistente: "Aggiungo l'azione 'accensione luci salotto.' Vuoi aggiungere altro o salvare la routine?—

Utente: "Salva routine.—

Assistente: "La tua routine è stata salvata con il comando vocale 'Accendi luci salotto.' Puoi modificarla in qualsiasi momento.—

In questo esempio, si evidenzia come l'assistente debba fornire indicazioni chiare per ciascun passaggio, ripetere o confermare le scelte dell'utente e limitare la complessità di ogni richiesta per evitare sovraccarico cognitivo. Questo approccio è coerente con i principi di progettazione conversazionale e con la cosiddetta *Miller's Law* [16], che evidenzia come la capacità di memorizzazione e comprensione degli utenti sia limitata a un numero ristretto di elementi per volta.

Vantaggi

L'interazione in linguaggio naturale consente un accesso "senza mani" e a bassa soglia cognitiva: negli esperimenti con smart speaker *speech-only*, il tasso di completamento delle routine semplici ha raggiunto il 94 % con un punteggio SUS medio di 78/100 [2, 11, 17] . Jarvis (assistente conversazionale progettato per la gestione di sistemi IoT complessi tramite linguaggio naturale) [11] mostra che la formulazione vocale riduce il tempo medio di espressione di una regola del 30 % rispetto alla costruzione grafica in Node-RED (2.3.1) [11].

L'impiego di chatbot basati su modelli linguistici di grandi dimensioni (es. RuleBot++) migliora la comprensione da parte del sistema delle richieste effettuate dall'utente [15].

Inoltre gli assistenti vocali risultano inclusivi per utenti con ridotta capacità visiva o con mani occupate, ampliando l'accessibilità del controllo domestico [18].

Svantaggi

- **Ambiguità semantiche**: la stessa frase può essere interpretata in modi diversi; nelle prove di Jarvis oltre il 20 % dei comandi liberi ha richiesto chiarimenti [11].
- **Recupero da errori**: senza un supporto visivo, gli utenti faticano a capire che cosa non sia stato compreso e come riformulare; ParlAmI riporta interazioni interrotte nel 15 % dei casi [19].
- **Scalabilità logica**: con regole che includono più di due trigger e tre azioni aumentano i turni di dialogo; RuleBot++ registra un +42 % di turni di chiarimento rispetto a regole elementari [15].
- **Privacy e fiducia**: nei diari d'uso longitudinali di Sciuto et al. il 38 % degli utenti ha limitato i comandi sensibili per timore di ascolto continuo [18].

In sintesi, le interfacce conversazionali semplificano l'accesso e rendono naturale la definizione di automazioni semplici; per scenari complessi occorrono strategie di disambiguazione, conferma e sintesi per mantenere affidabilità ed efficienza.

Interfacce Multi-modali

Per far fronte alle problematiche che possono nascere dall'utilizzo di un'interfaccia puramente conversazionale o visuale è possibile andare ad utilizzare quelle che vengono definite *interfacce multi-modali*. Con questo termine si identificano quei dispositivi che permettono di interagire con il sistema sfruttando simultaneamente

più canali comunicativi, come ad esempio voce e visualizzazione di un'interfaccia grafica. [5, 19]

Un'evoluzione di questo filone è **RuleBot** ++ [15], un chatbot basato su ChatGPT (GPT-4) integrato in un'interfaccia multi-modale: l'utente può impartire i comandi a voce o via chat, visualizzare la regola generata sul display (o in app) e, se necessario, correggerla toccando i singoli parametri. Nei test di usabilità (16 partecipanti) RuleBot ++ ha fatto registrare un punteggio SUS medio di 78/100 e un *time-on-task* inferiore del 35 % rispetto alla GUI di Home Assistant. [17]

In (fig. 2.6) è possibile vedere un ulteriore esempio di interfaccia multi-modale utilizzata per la definizione di una nuova routine tramite l'ausilio di un dispositivo *Amazon Alexa*. [5]

Figura 2.6: Esempio di un sistema multi-modale utilizzato per la generazione di routine

Una recente analisi sperimentale condotta in [5] ha evidenziato i vantaggi dell'utilizzo di soluzioni multi-modali per la creazione di routine all'interno di ecosistemi IoT, in quanto:

- Le interfacce multi-modali risultano più apprezzate dagli utenti rispetto alle interazioni *voice-only*, specialmente per la possibilità di visualizzare in tempo reale le scelte già effettuate e selezionare i comandi su schermo.
- Le interfacce conversazionali puramente vocali mantengono un elevato grado di accessibilità, ma richiedono una maggiore attenzione e memoria da parte dell'utente per seguire l'intero dialogo ed evitare di perdersi tra le opzioni disponibili.

- Aspetti come la dipendenza dal brand (ad esempio, la familiarità con Alexa o con Google Assistant) e il livello di esperienza con i dispositivi smart possono influire significativamente sulla percezione di usabilità e soddisfazione, suggerendo che la personalizzazione dell'interfaccia e la coerenza del dialogo siano fattori chiave.

Vantaggi

- **Integrazione voce e display**: quando input vocali e riscontri grafici/tattili convivono, i limiti del canale singolo si riducono sensibilmente: l'utente può parlare per avviare la routine e, nello stesso tempo, verificare a colpo d'occhio ciò che il sistema ha compreso.
- **Maggiore usabilità del sistema**: come mostrato in [5], l'utilizzo di una soluzione multimodale ha portato ad un incremento del tasso di completamento dei compiti assegnati agli utenti. Nello specifico si è passati da un 88 % (solo voce) ad un 97 % dei task completati e ad innalzo del punteggio SUS da 74/100 a 83/100. Tutto ciò conferma che la doppia modalità migliora sia efficacia sia soddisfazione. [17]
- **Maggiore controllo**: i partecipanti allo studio hanno apprezzato di poter usare il display per rivedere o correggere parametri complessi, lasciando alla voce i comandi rapidi («Alexa, salva»).[5]
- **Accessibilità amplificata**: la ridondanza dei canali favorisce pubblici diversi: persone con disabilità visive possono affidarsi alla sintesi vocale, mentre chi opera in ambienti rumorosi ricorre all'interfaccia touch.
- **Supporto "intelligente— alla correzione**: integrando un *Large Language Model* (LLM), come nel caso di *RuleBot ++*, l'interfaccia è in grado di proporre suggerimenti o versioni alternative della routine, riducendo errori sintattici e velocizzando la rifinitura delle regole [15].

Svantaggi

- **Aumento della complessità dell'UI**: alcuni utenti hanno segnalato confusione sul "dove guardare— quando la creazione di una routine visualizzava contemporaneamente conferme testuali e vocali [2].
- **Costi e frammentazione**: display intelligenti e robot aumentano i costi di adozione e introducono differenze tra ecosistemi (Amazon vs Google) che richiedono percorsi di progettazione paralleli [2].
- **Gestione dei conflitti multimodali**: ParlAmI riporta episodi di competizione tra input vocali e tocchi simultanei, con necessità di strategie di arbitraggio per evitare comandi duplicati [19].
- **Maggiore carico cognitivo in scenari complessi**: quando la regola supera tre azioni, il passaggio continuo voce a touch e viceversa aumenta il tempo medio di completamento del 22 % rispetto alla sola interazione touch [2].

3. Architettura del sistema

In questo capitolo verrà descritta e approfondita l'architettura del sistema oggetto dell'elaborato.

Il *Gemello Digitale* in questione è progettato per la gestione intelligente di abitazioni, con una particolare attenzione alla sostenibilità energetica e all'ottimizzazione dei consumi.

3.1 Introduzione all'architettura

L'architettura è concepita per favorire una gestione ottimizzata e sostenibile delle risorse energetiche e degli apparati domestici. Uno dei principali obiettivi del sistema è la creazione di un'interazione semplice e intuitiva tra l'utente e l'ambiente domestico intelligente.

Una rappresentazione grafica dettagliata di tale architettura viene mostrata in (fig. 3.1).

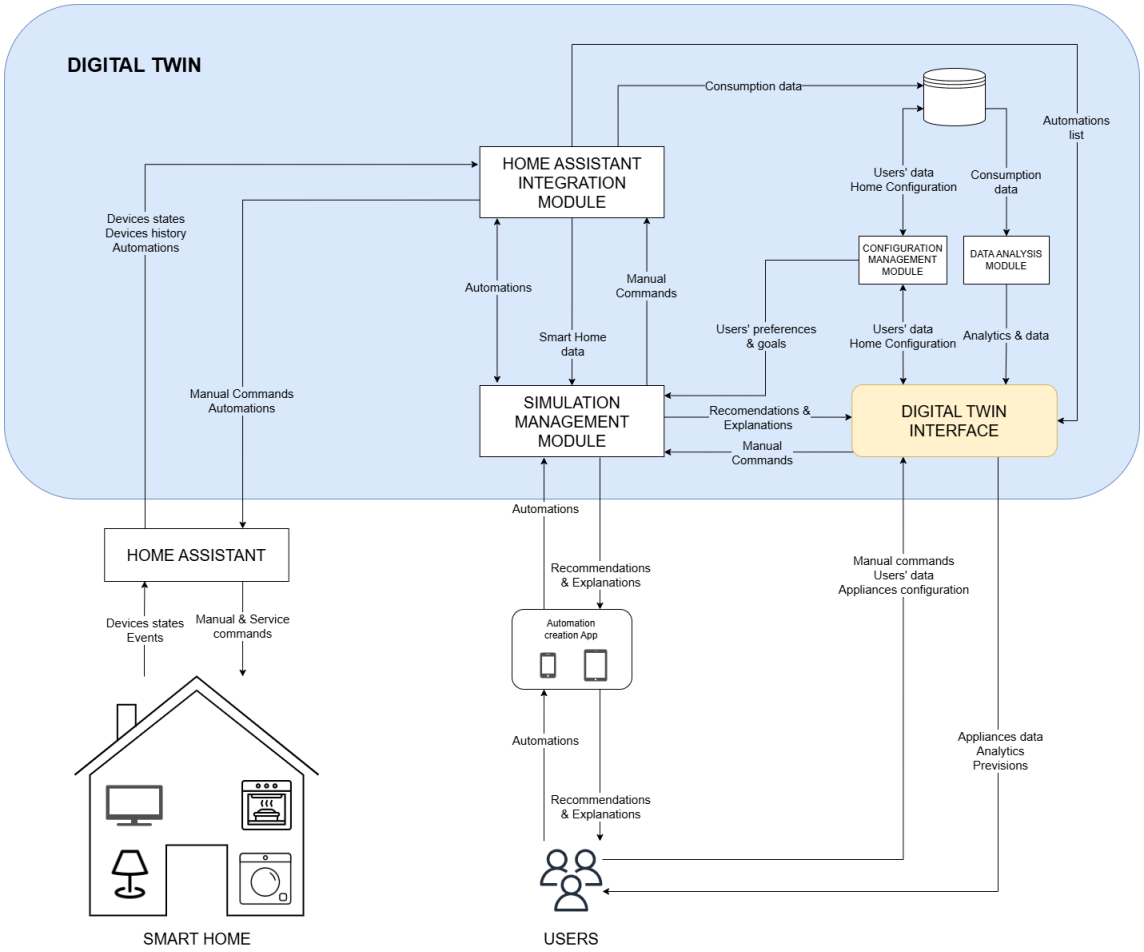

Figura 3.1: Rappresentazione dell'architettura presa in analisi

Tale architettura consente agli utenti di realizzare automazioni domestiche attraverso un'interfaccia grafica (GUI). Questa interfaccia, sulla base di controlli e previsioni relativi ai consumi energetici, restituisce un feedback all'utente indicando chiaramente se l'automazione proposta è consentita oppure se genera conflitti con altre automazioni precedentemente impostate o, ancora, se rischia di eccedere il limite massimo di consumo energetico, andando a suggerire possibili modifiche all'automazione che si stava cercando di creare.

3.2 Digital Twin Interface

La *Digital Twin Interface* costituisce la componente del sistema che permette all'utente finale di interagire con il Gemello Digitale sviluppato in questa tesi. A differenza della *Automation Creation App* conversazionale progettata dal CNR e attualmente eseguita come applicazione esterna, la Digital Twin Interface espone una applicazione web tradizionale che offre viste analitiche e strumenti operativi per comprendere lo stato della smart home, controllarne i dispositivi e integrare le nuove funzionalità di generazione delle automazioni descritte nel capitolo successivo.

L'interfaccia attuale è organizzata in quattro sezioni principali:

- **Dashboard** — riassume lo stato energetico della casa mostrando mappe interattive degli ambienti, indicatori sui dispositivi attivi e l'esito dei suggerimenti energetici ricevuti dal *Data Analysis Module*. Da questa vista l'utente può inviare comandi manuali che vengono inoltrati al *Simulation Management Module* (fig. 3.6).
- **Consumption** — espone grafici storici e predittivi dei consumi energetici (fig. 3.14) alimentati dalle API del *Data Analysis Module*. L'utente può cambiare intervalli temporali, confrontare dispositivi e ricevere notifiche quando le stime superano le soglie definite nel *Configuration Management Module*.
- **Automations** — elenca le automazioni presenti in Home Assistant, permettendo di attivarle o disattivarle, avviarne l'esecuzione e consultare i suggerimenti prodotti dal Simulation Management Module quando vengono rilevati conflitti o consumi eccessivi (fig. 3.8). In questa sezione verrà integrato il nuovo builder visuale implementato in questa tesi, eliminando la dipendenza dalla sola Automation Creation App.
- **Configuration e User area** — permettono di modificare i dati personali, impostare le preferenze dell'utente e configurare mappe, dispositivi, gruppi e piano energetico. I dati inseriti vengono salvati nel *Configuration Management Module* e resi disponibili agli altri moduli del sistema.

Oltre alle funzionalità rivolte all'utente, la Digital Twin Interface gestisce la comunicazione con Home Assistant, con il Simulation Management Module e con i servizi Rulebot:

- invia i comandi manuali e le nuove automazioni al Simulation Management Module, che dopo la simulazione li inoltra al *Home Assistant Integration Module* per l'esecuzione effettiva (fig. 3.7);
- sincronizza lo stato dei dispositivi e delle automazioni leggendo i dati esposti dall'Home Assistant Integration Module e li rende disponibili alla pagina Automations;
- integra nella stessa interfaccia la vista dell'applicazione Rulebot esistente, fornendo all'utente un flusso continuo fra generazione, simulazione e attuazione delle regole.

La Automation Creation App rimane oggi un canale complementare per gli utenti che prediligono l'interazione conversazionale e continua a essere eseguita come componente esterna. L'obiettivo di questa tesi è integrare progressivamente quelle funzionalità nella Digital Twin Interface, rendendo il builder visuale il punto di accesso unico alla creazione delle automazioni: il flusso sarà interamente web, dalla definizione della regola all'invio ai moduli di simulazione e all'implementazione su Home Assistant.

In sintesi, la Digital Twin Interface è il punto di contatto tra utente e architettura software e verrà ulteriormente estesa nel capitolo successivo per completare il percorso di progettazione e attivazione delle automazioni.

3.3 Simulation Management Module

Il *Simulation Management Module* rappresenta un componente centrale nell'architettura del gemello digitale di una smart home, svolgendo un ruolo chiave nel garantire che le automazioni proposte dagli utenti siano efficaci, sicure e coerenti con le preferenze personali e gli obiettivi di ciascun utente.

Questo modulo gestisce principalmente la simulazione e la validazione preventiva delle automazioni definite dagli utenti, al fine di prevenire eventuali conflitti e massimizzare l'efficienza operativa della smart home.

Nello specifico il processo inizia quando l'utente definisce una nuova automazione, la quale viene inviata direttamente al *Simulation Management Module* (fig. 3.2)

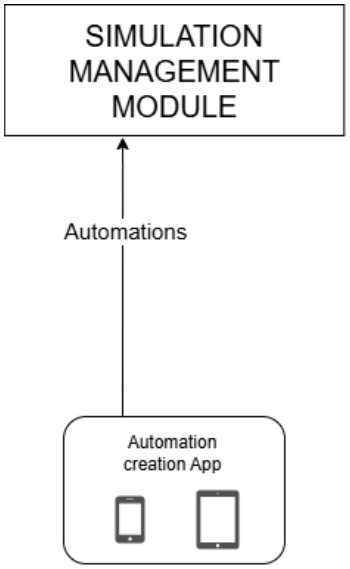

Figura 3.2: Invio delle automazioni dall'utente al Simulation Management Module

A questo punto, il modulo procede ad acquisire dal *Home Assistant Integration Module* le automazioni che son già state definite all'interno della Smart Home, insieme ai dati relativi agli stati (come consumi, temperature, ecc.) di tutti i dispositivi collegati.

È possibile visualizzare la porzione di architettura che si occupa di queste operazioni in (fig. 3.3)

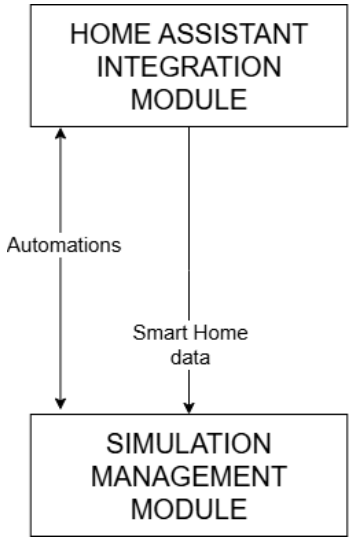

Figura 3.3: Acquisizione da parte del Simulation Management Module delle automazioni e dei dati della Smart Home

Parallelamente, il *Simulation Management Module* procede ad acquisire tutte le informazioni sugli obiettivi e le preferenze impostate dall'utente tramite il *Configuration Management Module*, garantendo così il rispetto delle sue esigenze personali (fig. 3.4).

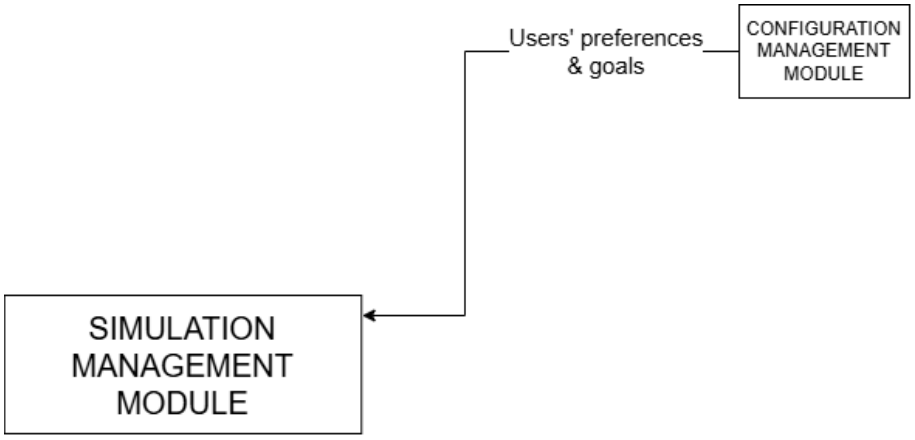

Figura 3.4: Acquisizione da parte del Simulation Management Module delle preferenze dell'utente

Una volta raccolti tutti questi dati, il *Simulation Management Module* esegue una

simulazione dell'automazione che l'utente aveva intenzione di implementare nel sistema. Tale simulazione non solo verifica il corretto funzionamento dell'automazione stessa, ma anche l'interazione con eventuali automazioni preesistenti. Un obiettivo fondamentale del modulo è quello di individuare possibili conflitti che potrebbero sorgere, ad esempio automazioni che si contraddicono o che, se attivate contemporaneamente, potrebbero causare un uso inefficiente delle risorse energetiche o problemi di comfort per gli utenti.

Al termine della simulazione, il modulo produce una serie di suggerimenti e spiegazioni dettagliate destinate all'utente, evidenziando eventuali migliorie possibili o segnalando i conflitti riscontrati durante l'analisi. Queste permettono agli utenti di prendere decisioni informate riguardo all'adozione, alla modifica o alla rimozione delle automazioni proposte, contribuendo così a migliorare continuamente la gestione energetica e funzionale della propria abitazione (fig. 3.5).

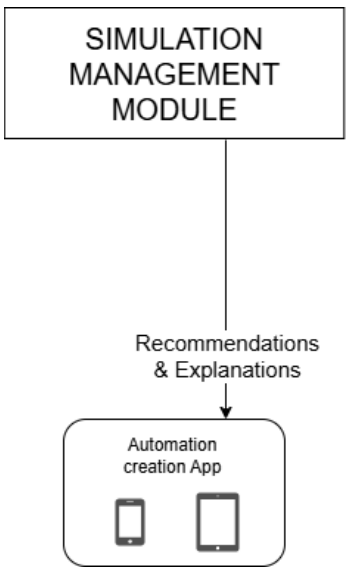

Figura 3.5: Generazione e invio di suggerimenti e spiegazioni da parte del Simulation Management Module all'utente

Nel caso invece in cui la simulazione confermi l'assenza di problemi, il *Simulation Management Module* inoltra automaticamente le automazioni approvate al *Home*

Assistant Integration Module, che provvede poi all'effettiva implementazione e attivazione delle automazioni all'interno della smart home (fig. 3.3). In questo modo, il modulo assicura un passaggio fluido ed efficiente dalla simulazione teorica alla reale implementazione delle automazioni, minimizzando il rischio di imprevisti operativi o inefficienze.

I comandi forniti dall'utente vengono inizialmente acquisiti dalla *Digital Twin Interface*, che provvede poi a inoltrarli al *Simulation Management Module* (fig. 3.6). Quest'ultimo li utilizzerà per eseguire i necessari controlli e simulazioni, come descritto precedentemente per l'implementazione di nuove automazioni.

Figura 3.6: Ricezione da parte dell'interfaccia del Digital Twin dei comandi dati dall'utente

Successivamente, al termine delle simulazioni, il *Simulation Management Module* inoltra i comandi validati al *Home Assistant Integration Module* (fig. 3.7), il quale li applica concretamente nella Smart Home fisica.

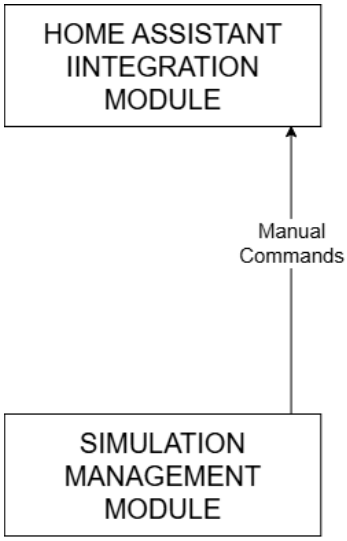

Figura 3.7: Invio dei comandi utente da parte del Simulation Management Module al Home Assistant Integration Module

Nel mentre al *Digital Twin Interface* vengono inviati tutti i suggerimenti e spiegazioni generati durante la fase di simulazione (fig. 3.8). Questo permetterà quindi di andare ad integrarli all'interno della GUI che l'utente utilizza.

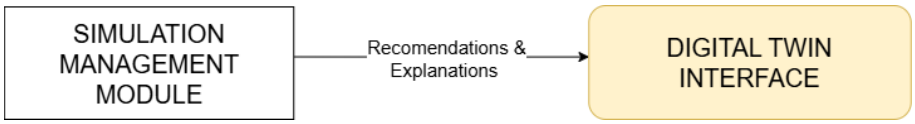

Figura 3.8: Invio dei suggerimenti generati durante la fase di simulazione all'interfaccia del Digital Twin

In sintesi, il *Simulation Management Module* rappresenta un elemento fondamentale per assicurare che le automazioni definite dagli utenti siano sempre ottimali, prive di conflitti e perfettamente allineate agli obiettivi di sostenibilità, efficienza e comfort tipici di una smart home evoluta.

3.4 Home Assistant Integration Module

Il *Home Assistant Integration Module*, rappresenta una componente centrale dell'architettura del Gemello Digitale della Smart Home. Questo modulo svolge una serie di funzioni essenziali che garantiscono la comunicazione tra il sistema di gestione smart home, rappresentato da *Home Assistant*, e il resto dell'architettura del Gemello Digitale.

Innanzitutto, il modulo ha il compito di raccogliere in maniera continua i dati provenienti dai dispositivi smart integrati nell'ambiente domestico tramite *Home Assistant*. Questi dati, oltre ad includere gli stati attuali di ogni dispositivo, forniscono anche informazioni inerenti a come i dispositivi sono stati impostati dall'utente (o da alcune automazioni), ad esempio come la temperatura impostata mediante un termostato o la luminosità di lampade.

Oltre ad acquisire le informazioni descritte, l'*Home Assistant Integration Module* recupera anche i dati di tutte le automazioni configurate in *Home Assistant*. Questa operazione assicura che il sistema disponga di un quadro completo e aggiornato delle automazioni attive.

La sezione di architettura che si occupa di queste funzionalità è mostrata in (fig. 3.9).

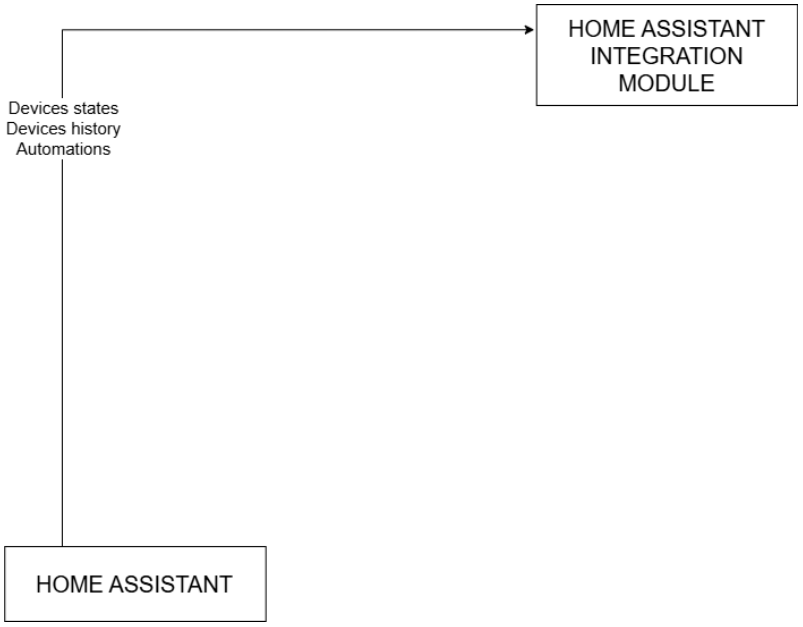

Figura 3.9: Ricezione dello stato dei dispositivi, del loro storico e delle automazioni

Parallelamente allo stato corrente, il modulo acquisisce anche uno storico dettagliato dei consumi energetici generati dai vari dispositivi, come elettrodomestici, luci intelligenti, climatizzatori e sistemi di riscaldamento. Questo registro permette non solo di avere una panoramica chiara delle abitudini di utilizzo degli utenti, ma è anche fondamentale per analisi future e per la realizzazione di previsioni accurate riguardo al consumo energetico domestico.

Una volta acquisiti, questi dati di consumo vengono inoltrati a un database specificamente predisposto (fig. 3.10). Tale database non rappresenta semplicemente un archivio passivo, ma costituisce una componente che rende disponibili le informazioni raccolte per successive analisi predittive. Grazie a queste informazioni dettagliate e cronologicamente ordinate, il sistema può identificare pattern, rilevare anomalie nei consumi e suggerire strategie di ottimizzazione energetica agli utenti.

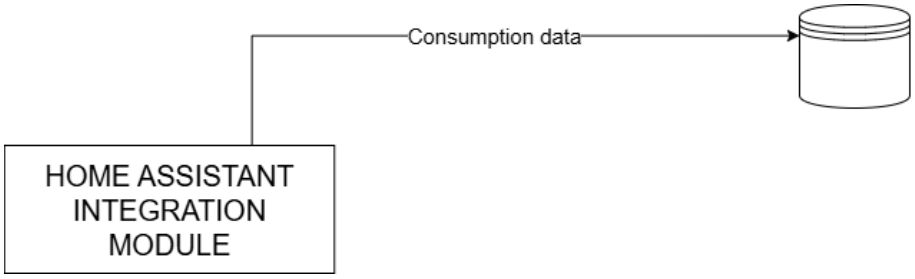

Figura 3.10: Invio dei dati di consumo al database

Oltre alla raccolta e archiviazione dei dati di consumo, il *Home Assistant Integration Module* mantiene costantemente aggiornata una lista di tutte le automazioni configurate in *Home Assistant*. Questa lista dettagliata e aggiornata è cruciale per garantire che la *Digital Twin Interface* disponga sempre delle informazioni più recenti circa le automazioni attive (fig. 3.11). Ciò consente agli utenti del Gemello Digitale di visualizzare chiaramente tutte le automazioni esistenti, gestirle e modificarle in tempo reale.

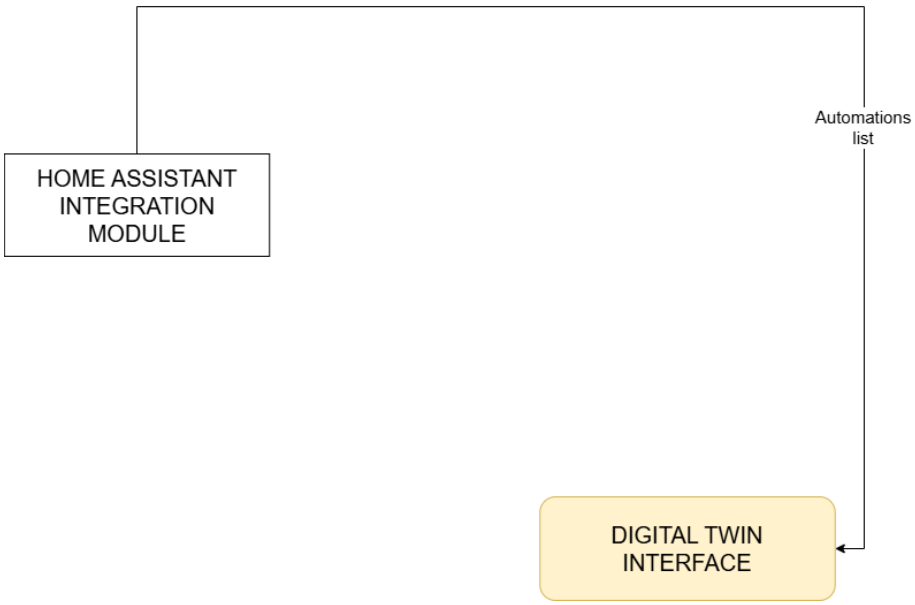

Figura 3.11: Invio delle automazioni alla Digital Twin Interface

Infine il modulo funge anche da intermediario per la comunicazione verso il sistema

Home Assistant. Nello specifico, riceve dal *Simulation Management Module* le automazioni e i comandi manuali generati dagli utenti in fase di simulazione e verifica. Questi comandi, dopo essere stati validati attraverso le simulazioni, vengono trasmessi a *Home Assistant* per la loro implementazione effettiva nell'ambiente domestico reale (fig. 3.12). In questo modo, il modulo assicura che l'applicazione delle automazioni simulate sia accurata e coerente con le intenzioni degli utenti, riducendo il rischio di errori o conflitti operativi.

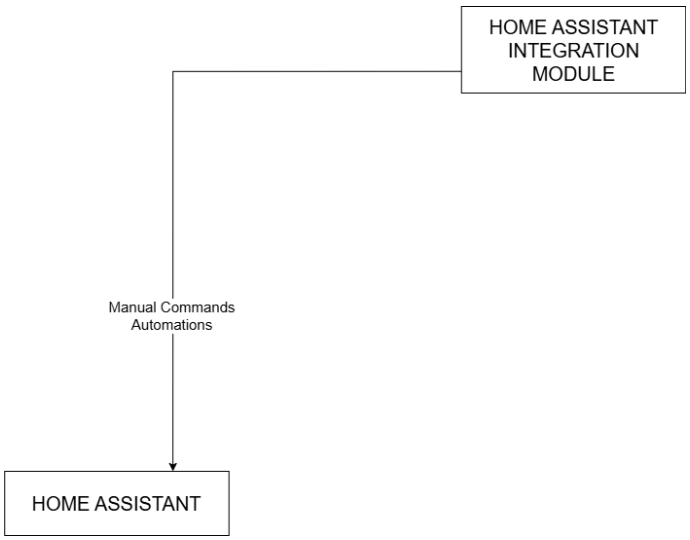

Figura 3.12: Invio dei comandi e delle automazioni a Home Assistant

Attraverso queste funzionalità integrate e coordinate, il *Home Assistant Integration Module* gioca un ruolo fondamentale nel mantenere l'intera architettura del Gemello Digitale sincronizzata, aggiornata e perfettamente funzionante, garantendo un'elevata affidabilità e una user experience ottimale.

3.5 Configuration Management Module

Il modulo di gestione della configurazione svolge la funzione di raccogliere e conservare in modo coerente tutte le preferenze espresse dagli utenti, gli obiettivi definiti e le specifiche impostazioni di configurazione che questi hanno precedentemente inserito nel sistema. Tali informazioni vengono memorizzate all'interno del database,

garantendo così la disponibilità continua e l'aggiornamento tempestivo dei dati rilevanti.

Per poter modificare tali dati è possibile accedervi tramite l'interfaccia del Digital Twin (fig. 3.13).

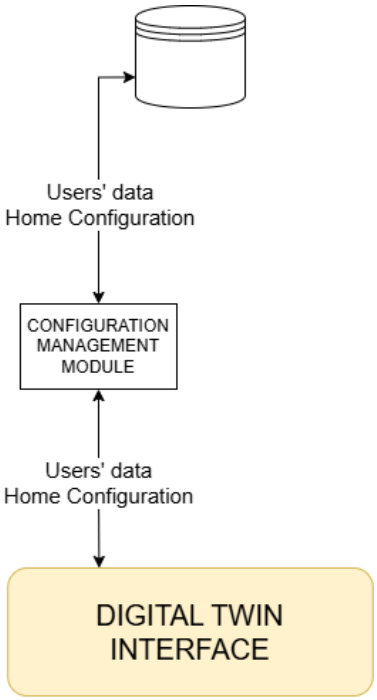

Figura 3.13: Porzione di architettura che si occupa dell'accesso e modifica delle configurazioni dell'utente

3.6 Data Analysis Module

Il *Data Analysis Module* rappresenta una componente fondamentale all'interno dell'architettura presa in esame, svolgendo il ruolo cruciale di raccogliere, elaborare e interpretare i dati energetici generati dagli utenti e dai dispositivi domestici. Attraverso processi di analisi avanzati, questo modulo permette di ottenere informazioni approfondite e utili per ottimizzare la gestione energetica della casa, migliorando l'efficienza e la sostenibilità complessiva del sistema.

In particolare, il modulo svolge tre funzioni primarie:

1. **Acquisizione e validazione dei dati**: estrae dal database tutti i dati storici di consumo energetico provenienti dai dispositivi integrati in *Home Assistant*.
2. **Elaborazione e modellazione**: i dati vengono arricchiti con variabili contestuali (ad esempio il giorno della settimana e la fascia oraria) e sottoposti a pipeline di *feature engineering* per alimentare diversi modelli di Machine Learning (ML) e Deep Learning (DL) impiegati per analisi descrittive, predittive e prescrittive.
3. **Servizi di esposizione**: rende disponibili, tramite API REST, le previsioni fatte alla *Digital Twin Interface* e al *Simulation Management Module*, consentendo così la verifica della sostenibilità e il supporto alle decisioni operative.

Tutti questi processi vengono eseguiti dalla seguente porzione di architettura (fig. 3.14).

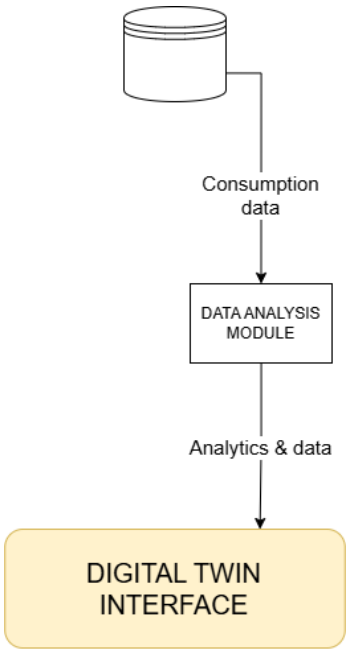

Figura 3.14: Porzione di architettura che si occupa di effettuare le predizioni sui consumi

In particolare, per la previsione dei consumi futuri è stato adottato un approccio basato su modelli di *Long Short-Term Memory* (LSTM), che risultano particolarmente efficaci nell'analisi di dati temporali grazie alla loro capacità di apprendere dipendenze sequenziali anche su intervalli temporali lunghi [20]. I modelli sono stati allenati su dati storici di consumo energetico provenienti dall'ambiente domestico reale, arricchiti con informazioni contestuali come il giorno della settimana e la fascia oraria. Le predizioni generate vengono poi impiegate per valutare, in fase preventiva, l'impatto delle automazioni sulla sostenibilità energetica dell'abitazione, supportando l'utente nel prendere decisioni più consapevoli e informate. Per la selezione dei modelli più performanti è stata condotta una procedura di ottimizzazione iperparametrica mediante *grid search*, in grado di individuare le combinazioni ottimali dei parametri principali del modello in base all'accuratezza predittiva. Tutti i dettagli di questa metodologia sono descritti in [20].

3.7 Smart Home e Home Assistant

La *Smart Home* rappresenta l'ambiente fisico reale in cui vengono applicate e sperimentate le automazioni create e gestite dal sistema. Questo ambiente comprende una varietà di dispositivi domestici intelligenti, tra cui elettrodomestici, sistemi di illuminazione, climatizzazione, sensori ambientali e dispositivi di sicurezza. Questi dispositivi comunicano costantemente con il sistema centrale tramite *Home Assistant*, che funge da ponte essenziale tra gli utenti, il Gemello Digitale e l'ambiente fisico.

Home Assistant svolge principalmente due ruoli fondamentali. In primo luogo, raccoglie continuamente i dati provenienti dai dispositivi domestici, registrandone lo stato corrente (ad esempio acceso, spento, temperatura rilevata, consumo energetico) e notificando eventuali cambiamenti di stato in tempo reale al sistema centrale. In secondo luogo, *Home Assistant* riceve i comandi generati dall'utente attraverso l'interfaccia digitale o elaborati dal Gemello Digitale e li inoltra direttamente ai dispositivi appropriati. Ciò assicura che le automazioni desiderate dagli utenti siano correttamente implementate, garantendo così un'efficiente interazione e un elevato grado di integrazione tra il mondo fisico reale e la sua controparte digitale.

Questa gestione bidirezionale dei dati e dei comandi consente di mantenere sincronizzati e aggiornati costantemente entrambi gli ambienti, permettendo al Gemello Digitale di simulare fedelmente e prevedere con precisione le condizioni operative della Smart Home, migliorando così la qualità e l'efficienza complessiva della gestione domestica (fig. 3.15).

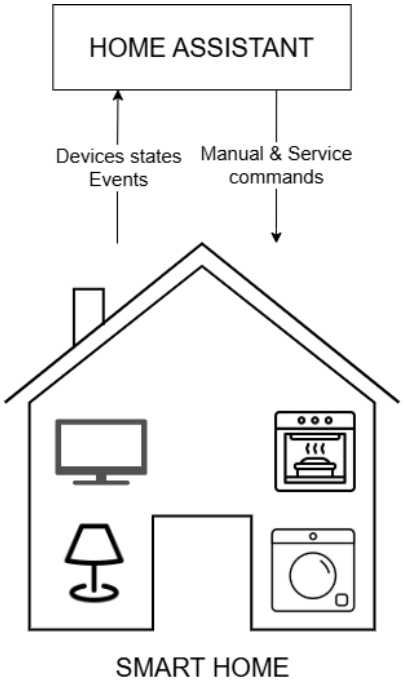

Figura 3.15: Smart Home fisica

3.8 Definizione delle automazioni da parte dell'utente

Sebbene l'architettura descritta copra in maniera approfondita aspetti cruciali quali la simulazione preventiva delle automazioni, l'integrazione con *Home Assistant* e la gestione avanzata dei dati energetici, rimane ancora incompleta la parte riguardante la definizione delle automazioni da parte dell'utente finale.

Al momento, l'interfaccia grafica presentata consiste in un prototipo che consente principalmente l'attivazione e la disattivazione intuitiva di automazioni predefinite tramite uno slider, nonché la visualizzazione di suggerimenti energetici generati a seguito delle simulazioni preventive. Tuttavia, essa non permette ancora agli utenti di creare da zero nuove automazioni personalizzate, definendo in autonomia le condizioni specifiche di attivazione (trigger), le azioni conseguenti e i dispositivi coinvolti.

La progettazione e implementazione completa di questa componente, che costituirà

un elemento fondamentale del sistema, sarà trattata nel prossimo capitolo. Tale capitolo approfondirà in dettaglio come gli utenti potranno interagire con una GUI dedicata, semplice e intuitiva, per definire autonomamente le proprie automazioni.

Questo sviluppo contribuirà significativamente a estendere le capacità del sistema, valorizzando ulteriormente l'interazione utente-sistema e promuovendo una gestione domestica ancora più flessibile.

4. Progettazione e implementazione del sistema

In questo capitolo vengono illustrate nel dettaglio le scelte progettuali e le attività di implementazione che hanno portato alla realizzazione del Gemello Digitale presentato nel Capitolo 3. L'obiettivo è mostrare come le esigenze degli utenti finali siano state tradotte in linee guida progettuali e, successivamente, in componenti software funzionanti.

4.1 Requisiti e principi guida

La definizione dei requisiti è partita dall'analisi dei tre attori principali del sistema: utenti domestici, ricercatori che monitorano i consumi e sviluppatori che mantengono l'infrastruttura. Per conciliare esigenze differenti sono state adottate le seguenti linee guida:

- **Chiarezza delle azioni**: la GUI deve distinguere nettamente tra consultazione dei dati e controllo dei dispositivi per ridurre il rischio di attivazioni involontarie.
- **Supporto alle decisioni**: ogni suggerimento energetico deve essere accompagnato da una spiegazione sintetica (ad esempio il risparmio stimato o il conflitto rilevato dal Simulation Management Module).
- **Progressive disclosure**: funzionalità avanzate (configurazione di mappe, integrazione con Home Assistant, personalizzazione delle soglie) sono raggiungibili senza sovraccaricare la dashboard principale.
- **Tracciabilità dei dati**: ogni componente deve indicare la fonte delle informazioni mostrate (API Home Assistant, servizi predittivi, Rulebot) per facilitare il debugging.

4.2 Processo di progettazione dell'interfaccia

Il processo UX si è articolato in tre iterazioni: raccolta casi d'uso, prototipazione low-fidelity e definizione del design system.

4.2.1 Casi d'uso e scenari

Sono stati individuati i principali journey che gli utenti devono poter eseguire:

1. Verificare in pochi secondi se il consumo corrente è sostenibile.
2. Creare o modificare un'automazione ricevendo feedback immediati sui conflitti.
3. Aggiornare le preferenze personali, ad esempio cambiando il peso dato al risparmio energetico o alla sicurezza.

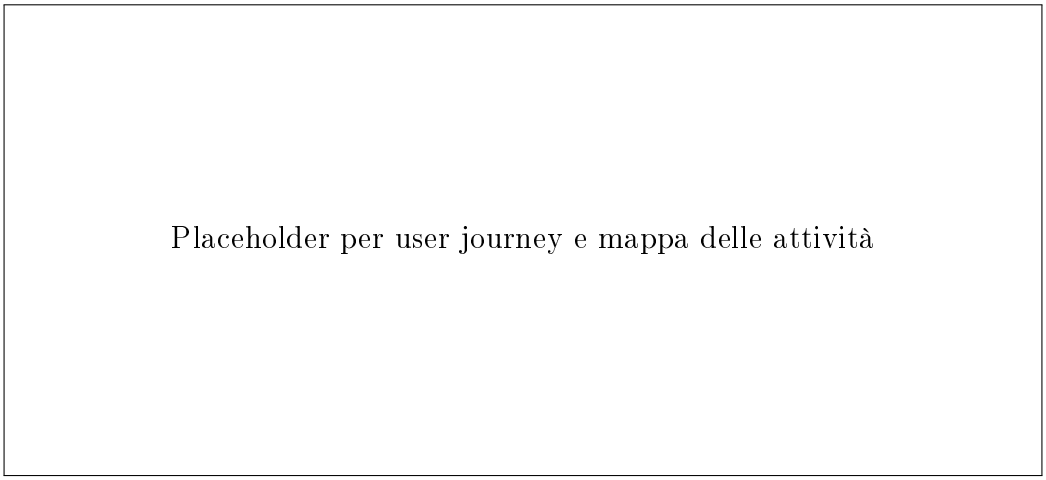

Figura 4.1: [TODO] User journey principale con punti di contatto digitali

Questi scenari hanno motivato l'introduzione di sezioni dedicate (Dashboard, Consumption, Automations, Configuration) e la scelta di mantenere visibile un menù laterale persistente, che consente di passare rapidamente da una vista all'altra senza perdere contesto.

4.2.2 Prototipazione e design system

Durante la fase di prototipazione sono stati creati wireframe a bassa fedeltà per validare la disposizione delle informazioni. Successivamente è stato definito un design system basato su card, tipografia Sans e palette neutra con accenti verde lime per richiamare il tema della sostenibilità.

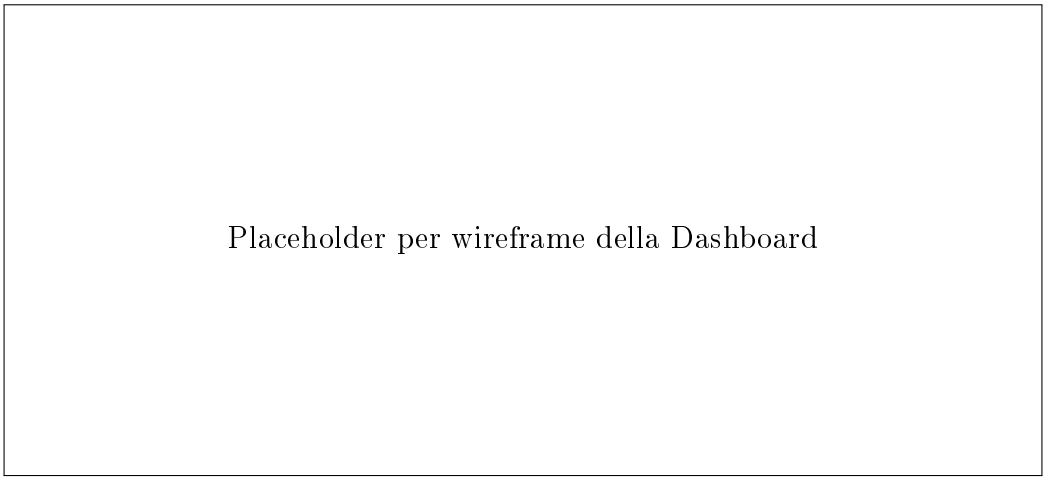

Figura 4.2: [TODO] Wireframe a bassa fedeltà della Dashboard

Le principali decisioni di UI sono state:

- **Uso estensivo di card** per raggruppare indicatori omogenei e rendere la dashboard scalabile su risoluzioni differenti.
- **Tipografia uniforme** per favorire la leggibilità in modalità chiara e scura, coerentemente con quanto implementato dal tema React.
- **Color coding** (verde = stato attivo, grigio = inattivo, giallo = warning) che aiuta l'utente a capire a colpo d'occhio lo stato delle automazioni.

4.3 Architettura applicativa front-end

L'interfaccia è stata implementata con React e Inertia.js sopra Laravel. La scelta è stata dettata dalla necessità di riutilizzare l'ecosistema Laravel per l'autenticazione e, allo stesso tempo, ottenere una SPA dinamica.

4.3.1 Strutturazione dei componenti

Il codice è suddiviso in tre categorie principali:

1. **Pages** (es. `Dashboard3.jsx`, `Automation.jsx`) che raccolgono e orchestrano i dati provenienti dai provider contestuali.
2. **Components** riutilizzabili (card, grafici, tabelle) che implementano lo stile definito dal design system.
3. **Context Providers** come `DeviceContextRefresh` che sincronizzano elenco dispositivi e stato energetico.

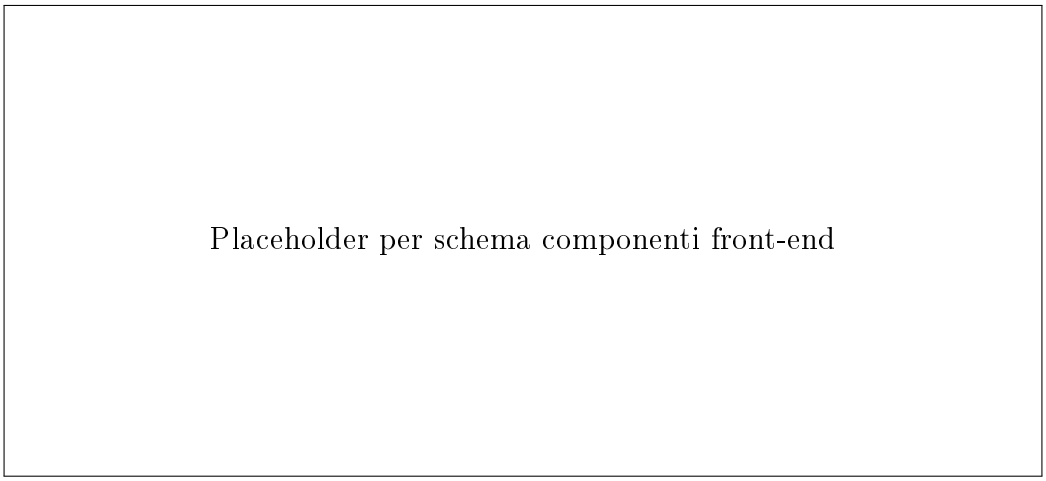

Figura 4.3: [TODO] Diagramma della gerarchia di componenti React

Questa strutturazione consente di limitare il re-rendering: le Pages ricevono i dati via context e li propagano verso componenti puramente presentazionali, mentre le interazioni (toggle di un'automazione, caricamento di mappe) invocano helper comuni come `apiFetch` o `callService`.

4.4 Integrazione con i servizi backend

Per mantenere coerenti i dati tra Gemello Digitale e Home Assistant sono state implementate API dedicate:

- Endpoint REST in Laravel (`/maps`, `/user`, `/automation`) protetti tramite

Sanctum.

- Web service Rulebot per la creazione conversazionale di regole, incorporato via iframe ma accessibile anche tramite API per la sincronizzazione dello stato.

La scelta di mantenere un backend Laravel monolitico, anziché microservizi separati, permette di condividere meccanismi di caching, logging e validazione dell'input. Inoltre facilita l'orchestrazione con il Simulation Management Module, che necessita di serializzare lo stesso set di dati sia verso la Digital Twin Interface sia verso Home Assistant.

4.5 Implementazione dell'interfaccia

In questa sezione si descrive come le linee guida sono state tradotte in codice.

4.5.1 Dashboard

La pagina `Dashboard3.jsx` combina mappe interattive (`AnimateMap2`) e indicatori sintetici. La sezione di destra utilizza componenti come `WhiteCard` per mostrare potenza istantanea, emissioni e numero di dispositivi attivi. L'uso di gauge semicircolari è stato preferito ai grafici a barre per offrire un feedback immediato: i test con utenti hanno evidenziato che la lettura del colore (verde/giallo/rosso) è più rapida dell'interpretazione di valori numerici.

4.5.2 Sezione Automations

Il componente `AutomationTable` elenca le automazioni utilizzando card espandibili e un drawer laterale. Durante la progettazione sono state valutate due alternative: una tabella tradizionale e una timeline. È stata scelta la tabella con filtri perché permette di integrare controlli diretti (switch, pulsante delete) senza sovraccaricare l'utente. Il drawer mantiene visibile la lista mentre si esaminano i singoli dettagli, riducendo i click necessari.

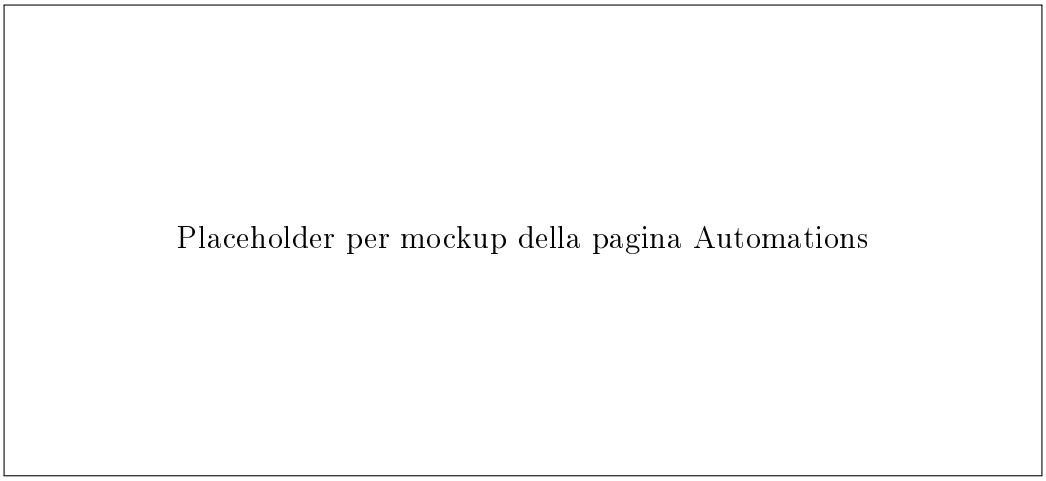

Figura 4.4: [TODO] Mockup ad alta fedeltà della pagina Automations

4.5.3 Configurazione e preferenze

La pagina di configurazione riutilizza gli stessi componenti per offrire coerenza visiva. L'utente può caricare mappe, associare dispositivi e definire l'energy plan. Le preferenze personali sono rappresentate tramite carta trascinabile per rispettare il principio di manipolazione diretta discusso nel Capitolo 2.

4.6 Implementazione del builder di automazioni

Il builder descritto nel Capitolo 3 richiede un'interazione ricca: selezione di trigger, condizioni, azioni e verifica tramite simulazione.

4.6.1 Motivazioni progettuali

- **Editor modulare**: trigger, condizioni e azioni sono rappresentati da card riordinabili (`Reorder` di Framer Motion). Ciò consente di aggiungere facilmente nuovi tipi senza riscrivere l'interfaccia.
- **Validazione in tempo reale**: i campi obbligatori vengono evidenziati non appena l'utente li lascia vuoti, riducendo gli errori in fase di simulazione.
- **Trasparenza della simulazione**: una volta inviato il modello al Simulation Management Module, i risultati vengono integrati nella stessa vista tramite

componenti `ToastNotification` per non interrompere il flusso.

4.6.2 Flusso di implementazione

1. L'utente definisce la regola nella UI; il builder genera un JSON conforme al formato richiesto dal Simulation Management Module.
2. Il backend Laravel inoltra la richiesta al servizio di simulazione e attende la risposta asincrona.
3. I risultati vengono mostrati nella UI e, in caso di esito positivo, trasmessi al Home Assistant Integration Module per l'attuazione.

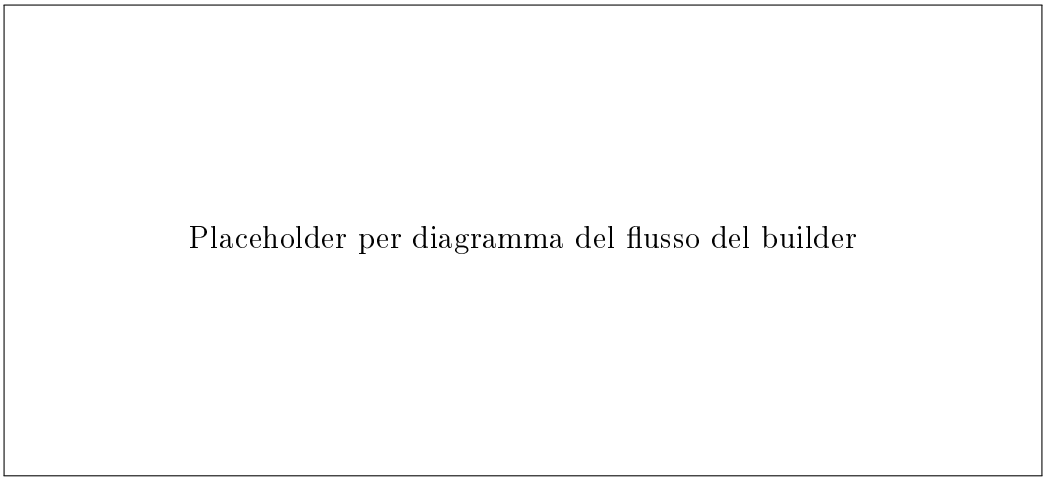

Figura 4.5: [TODO] Diagramma del flusso di creazione e simulazione delle automazioni

4.7 Strategie di verifica e iterazione

La verifica del sistema è stata organizzata su due livelli:

- **Test funzionali** automatizzati su Laravel per garantire la correttezza degli endpoint e la protezione dei dati sensibili.
- **Sessioni di revisione con utenti pilota**, durante le quali sono state raccolte metriche qualitative (tempo per completare un'azione, numero di errori) e osservazioni sulle preferenze di visualizzazione. I feedback hanno portato, ad esempio, all'introduzione del pulsante di refresh dell'iframe Rulebot e alla

suddivisione della dashboard in colonne per garantire la leggibilità anche su schermi ridotti.

4.8 Sintesi

La progettazione del Gemello Digitale ha richiesto un equilibrio tra esigenze eterogenee: trasparenza verso l'utente finale, integrazione tecnica con Home Assistant e Rulebot e mantenibilità del codice. Le scelte descritte in questo capitolo forniscono il contesto necessario per comprendere le funzionalità che saranno approfondite nei capitoli successivi.

Conclusioni

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Bibliografia

[1] Blase Elyse Ur, Elyse McManus Ho, Melwyn Pak Yong Ho e Michael L. Littman. «Practical Trigger-Action Programming in the Smart Home». In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. New York, NY, USA: Association for Computing Machinery, 2014, pp. 803–812. ISBN: 9781450324731. DOI: 10.1145/2556288.2557420. URL: https://doi.org/10.1145/2556288.2557420.

[2] Barbara Rita Barricelli, Alessandro Bondioli, Daniela Fogli, Letizia Iemmolo e Angela Locoro. «Creating Routines for IoT Ecosystems through Conversation with Smart Speakers». In: *International Journal of Human–Computer Interaction* 40.20 (2024), pp. 6109–6127. DOI: 10.1080/10447318.2023.2247845. URL: https://doi.org/10.1080/10447318.2023.2247845.

[3] Michaela R. Reisinger, Sebastian Prost, Johann Schrammel e Peter Fröhlich. «User requirements for the design of smart homes: dimensions and goals». In: *Journal of Ambient Intelligence and Humanized Computing* 14 (2023), pp. 15761–15780. DOI: 10.1007/s12652-021-03651-6. URL: https://doi.org/10.1007/s12652-021-03651-6.

[4] Fulvio Corno, Luigi De Russis e Alberto Monge Roffarello. «From Users' Intentions to IF-THEN Rules in the Internet of Things». In: *ACM Transactions on Information Systems* 39.4 (2021), 53:1–53:33. DOI: 10.1145/3447264. URL: https://doi.org/10.1145/3447264.

[5] Barbara Rita Barricelli, Daniela Fogli, Letizia Iemmolo e Angela Locoro. «A Multi-Modal Approach to Creating Routines for Smart Speakers». In: *Proceedings of the 2022 International Conference on Advanced Visual Interfaces (AVI '22)*. Association for Computing Machinery, 2022, pp. 1–5. DOI: 10.1145/3531073.3531168. URL: https://doi.org/10.1145/3531073.3531168.

[6] McKenna McCall, Eric Zeng, Faysal Hossain Shezan, Mitchell Yang, Lujo Bauer, Abhishek Bichhawat, Camille Cobb, Limin Jia e Yuan Tian. «Towards Usable Security Analysis Tools for Trigger-Action Programming». In: *Proceedings of the Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*. A cura di Patrick G. Kelley e Apu Kapadia. Anaheim, CA: USENIX Association, ago. 2023, pp. 301–320. ISBN: 978-1-939133-36-6. URL: https://www.usenix.org/system/files/soups2023-mccall.pdf.

[7] Likewin Thomas, M. K. Manoj Kumar, S. D. Shiva Darshan e B. S. Prashanth. «Towards Comprehensive Home Automation: Leveraging the IoT, Node-RED, and Wireless Sensor Networks for Enhanced Control and Connectivity». In: *Engineering Proceedings*. Vol. 59. 2023, p. 173. DOI: 10.3390/engproc2023059173.

[8] Anders Peter Aavild, Aleksander Rosenkrantz de Lasson, Christian Moesgaard, Erik Christensen, Sergio Moreschini, David Hastbacka, Davide Taibi e Michele Albano. «Distributed Home Automation with Home Assistant». In: *Proceedings of the 14th International Conference on the Internet of Things (IoT '24)*. 2024, pp. 206–212. DOI: 10.1145/3703790.3703828.

[9] Giuseppe Desolda, Carmelo Ardito e Maristella Matera. «Empowering End Users to Customize Their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools». In: *ACM Transactions on Computer-Human Interaction* 24.2 (2017), 12:1–12:52. DOI: 10.1145/3057859. URL: https://doi.org/10.1145/3057859.

[10] Joëlle Coutaz e James L. Crowley. «A First-Person Experience with End-User Development for Smart Homes». In: *IEEE Pervasive Computing* 15.2 (2016), pp. 26–36. DOI: 10.1109/MPRV.2016.34. URL: https://doi.org/10.1109/MPRV.2016.34.

[11] André Sousa Lago, João Pedro Dias e Hugo Sereno Ferreira. «Managing non-trivial internet-of-things systems with conversational assistants: A prototype and a feasibility experiment». In: *Journal of Computational Science* 51 (2021),

p. 101324. DOI: 10.1016/j.jocs.2021.101324. URL: https://doi.org/10.1016/j.jocs.2021.101324.

[12] Danilo Caivano, Daniela Fogli, Rosa Lanzilotti, Antonio Piccinno e Fabio Cassano. «Supporting end users to control their smart home: Design implications from a literature review and an empirical investigation». In: *The Journal of Systems & Software* 144 (2018), pp. 295–313. DOI: 10.1016/j.jss.2018.06.035. URL: https://doi.org/10.1016/j.jss.2018.06.035.

[13] Fulvio Corno, Luigi De Russis e Alberto Monge Roffarello. «Loops, Inconsistencies and Redundancies in Trigger-Action Rules for Smart Environments». In: *Proceedings of the 5th International Conference on Internet of Things Design and Implementation (IoTDI '20)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 259–270. DOI: 10.1145/3386910.3397254. URL: https://doi.org/10.1145/3386910.3397254.

[14] Xiang Chen, Yuncong Li, Earlence Fernandes e Atul Prakash. «Understanding Rule Preventions, Collisions, and Unexpected Chains in Trigger-Action Programming». In: *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P '21)*. IEEE, 2021, pp. 1154–1171. DOI: 10.1109/SP40001.2021.00091. URL: https://doi.org/10.1109/SP40001.2021.00091.

[15] Simone Gallo, Fabio Paternò e Alessio Malizia. «A Conversational Agent for Creating Automations Exploiting Large Language Models». In: *Personal and Ubiquitous Computing* 28.5 (2024), pp. 931–946. DOI: 10.1007/s00779-024-01825-5. URL: https://doi.org/10.1007/s00779-024-01825-5.

[16] George A. Miller. «The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information». In: *Psychological Review* 101.2 (1994), pp. 343–352. DOI: 10.1037/0033-295X.101.2.343. URL: https://doi.org/10.1037/0033-295X.101.2.343.

[17] John Brooke. «SUS: A Quick and Dirty Usability Scale». In: *Usability Evaluation in Industry*. A cura di Patrick W. Jordan, Bruce Thomas, Bernard A. Weerdmeester e Ian L. McClelland. CRC Press, 1996, pp. 189–194.

[18] Andrea Sciuto, Arnita Saini, Jodi Forlizzi e Jason I. Hong. «"Hey Alexa, What's Up?": A Mixed-Methods Study of In-Home Conversational Agent Usage». In: *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18)*. ACM, 2018, pp. 857–868. DOI: 10.1145/3196709.3196772.

[19] Evropi Stefanidi, Michalis Foukarakis, Dimitrios Arampatzis, Maria Korozi, Asterios Leonidis e Margherita Antona. «ParlAmI: A Multimodal Approach for Programming Intelligent Environments». In: *Technologies* 7.1 (2019), p. 11. DOI: 10.3390/technologies7010011. URL: https://doi.org/10.3390/technologies7010011.

[20] Davide Guizzardi, Barbara Rita Barricelli e Daniela Fogli. «A User-in-the-loop Digital Twin for Energy Consumption Prediction in Smart Homes». In: *AXAI 2025: Workshop on Adaptive eXplainable AI*. Department of Information Engineering, University of Brescia. CEUR-WS. Brescia, Italy, 2025.