SMARTHOME

SMARTHOME IMPLEMENTATA DA

Alessandro Cimarelli 1972070

Alessio Biancalana 1946866

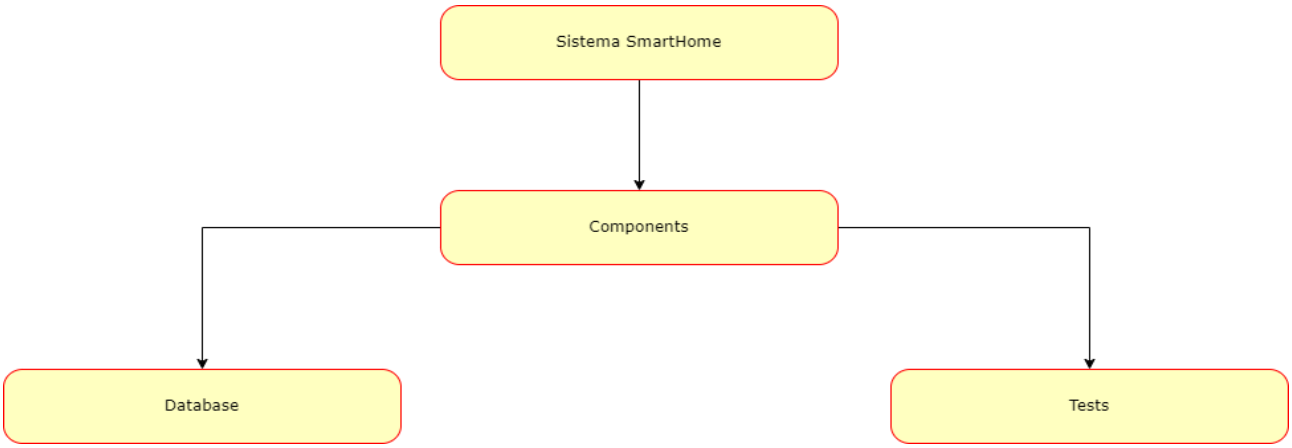
Gabriele Fabro 1934872

Sommario

DESCRIZIONE GENERALE

Il nostro progetto si concentra sulla creazione di un ecosistema di automazione domestica intelligente che offre agli utenti il completo controllo della propria casa. L'obiettivo è garantire un controllo efficiente del giardino, l'implementazione di un sistema di sicurezza sofisticato, la gestione precisa della temperatura e una dettagliata registrazione storica degli eventi. La versatilità della "SmartHome" si evidenzia attraverso la sua perfetta integrazione con dispositivi esistenti. Sia che si tratti di dispositivi smart già presenti in casa o di nuove aggiunte alla suite tecnologica, il sistema offre una compatibilità flessibile, consentendo agli utenti di massimizzare l'utilizzo dei loro dispositivi preferiti. Questo programma genera casualmente un numero preimpostato di azioni che vanno a generare un qualsiasi scenario possibile, con rappresentazioni il più possibili reali.

Sistema e ambiente operativo



USER REQUIREMENTS

Requisito 1: Gestione del giardino

Descrizione:

La funzione "Gestione giardino" offre il controllo automatizzato e manuale degli elementi del giardino, inclusa l'irrigazione e l'illuminazione esterna.

Specifica:

1. Gestione dell'irrigazione
2. Gestione dell'illuminazione esterna
3. Monitoraggio del sensore ambientale

Requisito 2: Gestione della sicurezza e sensori

Descrizione:

La funzionalità "Gestione della sicurezza" garantisce la sicurezza della casa attraverso videosorveglianza, sensori di movimento e allarmi intelligenti.

Specifica:

1. Videosorveglianza e sensore di movimento: Integrazione di telecamere di sorveglianza e sensori di movimento visualizzazione dello streaming video
2. Controllo di sicurezza
3. Recap sicurezza

Requisito 3: Gestione della temperatura

Descrizione:

La funzione "Gestione della temperatura" fornisce il controllo automatico e manuale della temperatura attraverso termostati intelligenti o sistemi di climatizzazione.

Specifica:

1. Accendi
2. Spegni
3. Cambia temperatura
4. Controllo remoto: possibilità di regolare la temperatura da remoto tramite un'interfaccia centralizzata come un dispositivo di controllo
5. Monitoraggio della temperatura: fornitura di monitoraggio in tempo reale delle temperature degli ambienti domestici

Requisito 4: Gestione dei dispositivi

Descrizione:

Gestione personalizzabile e modificabile di elettrodomestici, compresa accensione, spegnimento e personalizzazioni dell'utilizzo.

Specifica:

1. Accendi
2. Spegni
3. Programmazione oraria

L'elenco dei dispositivi smart che la casa mette a disposizione dell'utente:

- Televisione
- Lavastoviglie
- Lavatrice
- Macchina del caffè
- Microonde
- Coperte riscaldate
- Casse/Altoparlanti
- Sveglia
- Irrigatore
- Tosaerba
- Tapparelle elettriche

Requisito 5*: Gestione dei* luci

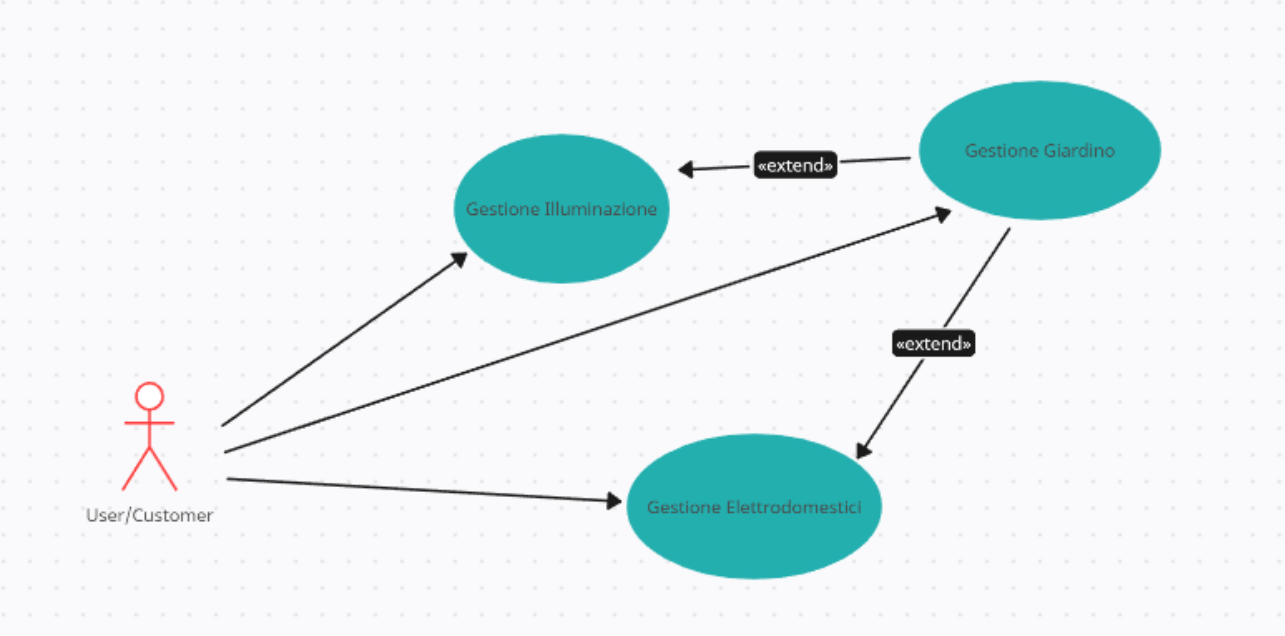
Descrizione:

Gestione personalizzabile e modificabile delle luci, compresa accensione, spegnimento e regolazione delle intensità e personalizzazioni del colore.

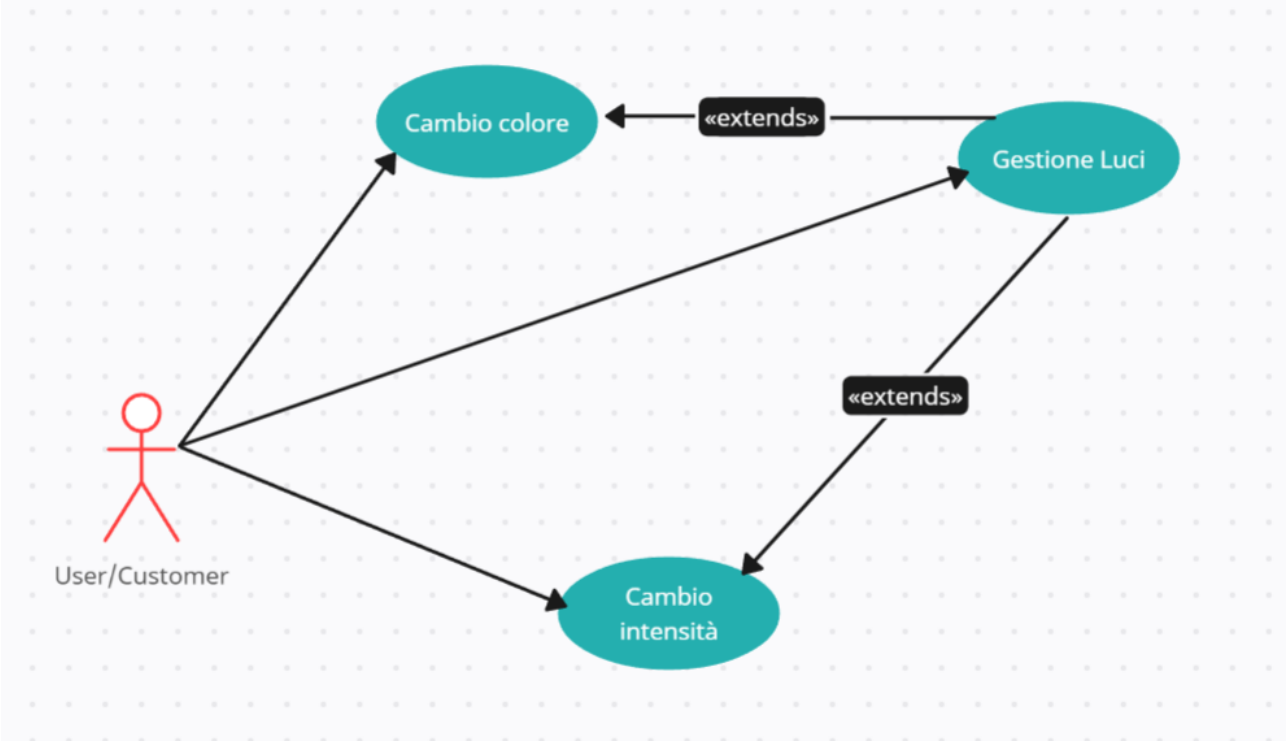
Specifica:

1. Accendi
2. Spegni
3. Cambia colore
4. Cambia intensità

Use Case dell'UML Giardino



Use Case dell'UML Luci



SYSTEM REQUIREMENTS

Requisito 1: Prestazioni

Descrizione: Prestazioni minime richieste dall'utente

Specifiche:

1. Tempo di risposta: il sistema deve garantire una risposta rapida, con un tempo massimo di 5 secondi per le richieste dell'utente.
2. Affidabilità: il sistema deve essere operativo almeno il 99% del tempo.

Requisito 2: Usabilità

Descrizione: Specifiche per l'interfaccia utente

Specifiche:

1. Interfaccia utente: l'interfaccia utente dovrebbe essere intuitiva e facile da usare per gli utenti senza particolari competenze tecniche.

Requisito 3: Gestione manutenzione

Descrizione:

Manutenzione periodica del software

Specifiche:

- **Aggiornamenti software**: gli aggiornamenti software dovrebbero essere rilasciati periodicamente, con notifiche chiare agli utenti in caso di manutenzione.
- **Manutenzione preventiva**: il sistema dovrebbe essere sottoposto a manutenzione preventiva ogni mese per garantirne la stabilità.

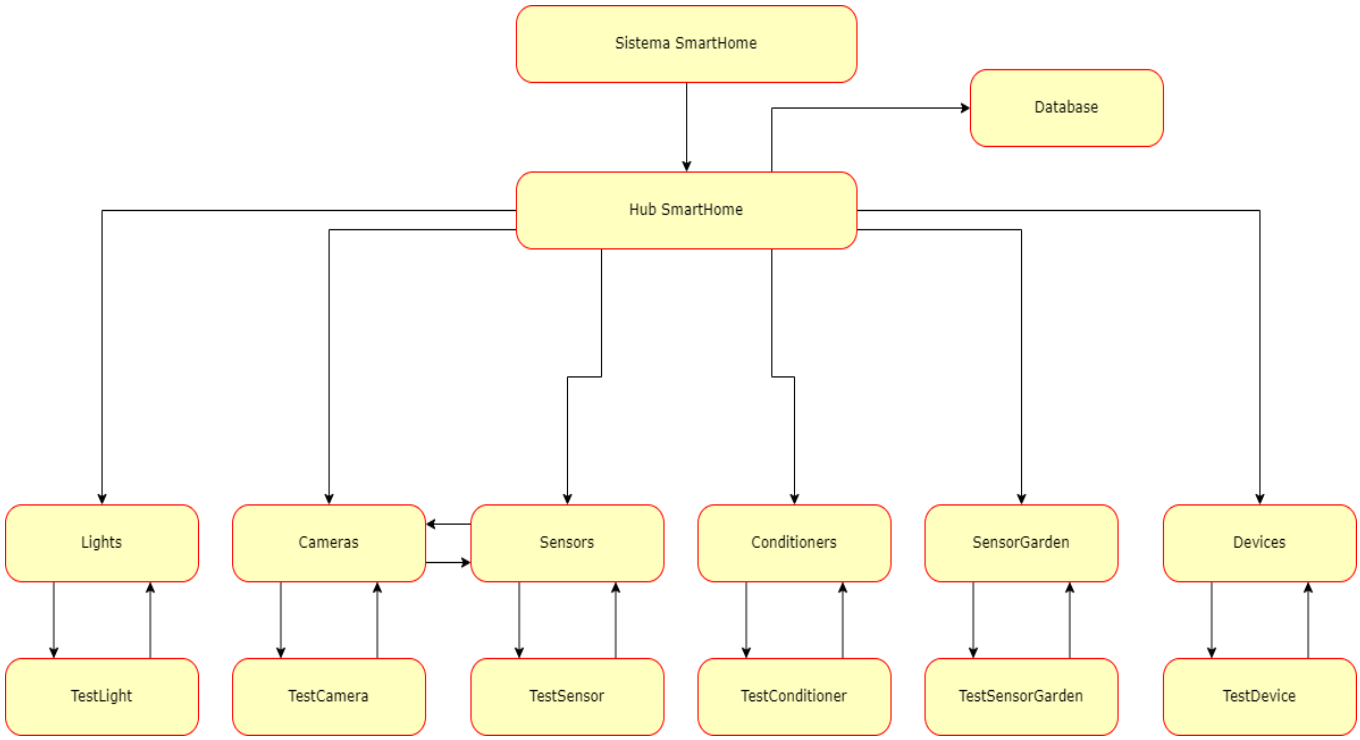
Requisito 4: Scalabilità

Descrizione: Migliora la scalabilità del sistema

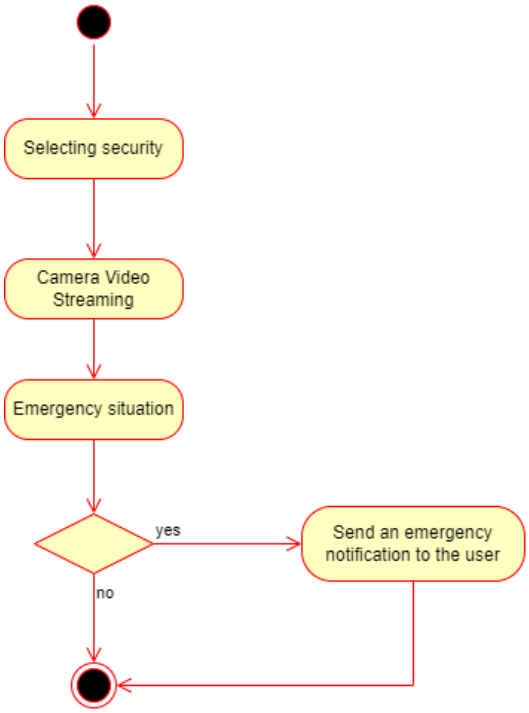
Specifiche:

1. Capacità di espansione: il sistema deve consentire l'aggiunta di nuovi dispositivi e funzionalità senza causare problemi di prestazioni.

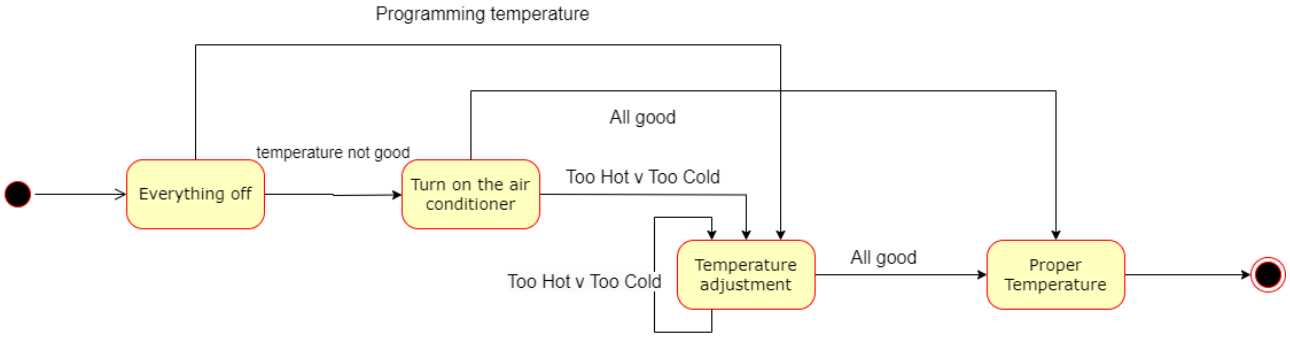
Diagramma architettura del sistema



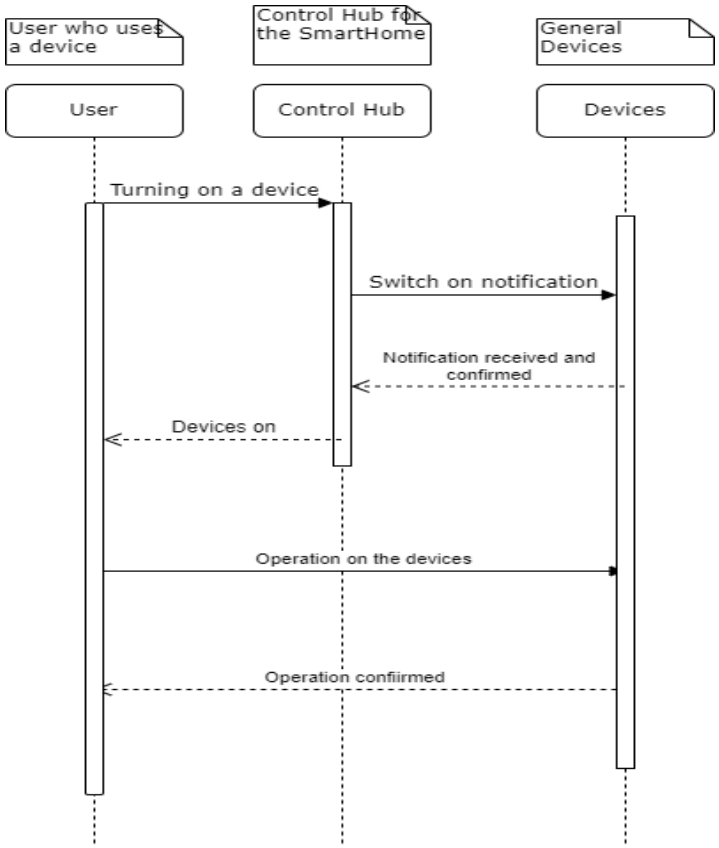
Activity Diagram UML Telecamera di Sorverglianza



State Diagram UML Condizionatore



Message Sequence Chart UML Dispotivi



IMPLEMENTAZIONE

Descrizione generale

Componenti del Sistema:

- Camera
- Conditioner
- Device
- Light
- Sensor
- SensorGarden

Funzionalità del Sistema:

Inizializzazione:

1. Inizializzazione del database e delle variabili necessarie.
2. Inizializzazione delle componenti del sistema con valori iniziali.
3. Impostazione del tempo di inizio.

Test Periodici:

Le componenti vengono testate periodicamente per rilevare il loro stato corrente e i dati rilevanti.

I risultati dei test vengono registrati nel database.

Monitoraggio e Log:

1. Durante i test, il sistema monitora il tempo di risposta per garantire che le risposte avvengano entro un limite di tempo prestabilito.
2. I dati rilevanti, inclusi gli stati delle componenti e i tempi di risposta, vengono registrati nel database per l'analisi e il monitoraggio.

Fine del Programma:

Dopo un numero predefinito di cicli di test, i dati raccolti vengono registrati nel database e il programma termina.

Conclusione:

In sintesi, l'implementazione del sistema prevede un ciclo di monitoraggio delle componenti, durante il quale vengono eseguiti test periodici per rilevare lo stato delle componenti e monitorare i tempi di risposta. I dati raccolti vengono registrati nel database per l'analisi e il monitoraggio delle prestazioni del sistema.

Pseudo-codice camera

Questa classe gestisce le telecamere di sorveglianza. Può iniziare a registrare con il metodo **setRecording(rec: booleano)** se il sensore rileva un movimento. E' inizializzata ad uno stato casuale e con il metodo **next()** passa ad un altro stato.

Classe Camera

Attributi:

state: camera_type, è lo stato che la camera può assumere

id: intero che rappresenta una camera specifica

recording: booleano, rappresenta lo stato della registrazione

Metodi:

Costruttore(id: intero, state: camera_type): Inizializza un oggetto Camera con l'id specificato e lo stato iniziale.

next(): Genera casualmente e restituisce il prossimo stato della fotocamera.

getState(): Restituisce lo stato attuale della fotocamera.

getId(): Restituisce l'ID della fotocamera.

setRecording(rec: booleano): Imposta lo stato di registrazione della fotocamera.

getRecording(): Restituisce lo stato corrente di registrazione della fotocamera.

setState(newState: camera_type): Imposta lo stato della fotocamera al valore specificato.

Enumerazione camera_type:

CameraON: La telecamera è accesa

WAITING : La telecamera è in attesa di registrare in caso il sensore rilevi un movimento

CameraOFF: La telecamera è spenta

Pseudo-codice Condizionatore

Questa classe gestisce i condizionatori. Può rilevare la temperatura della camera con il metodo **getTemperature()** e cambiarla con il metodo **modifyTemperature(newTemperature: intero)**. Il condizionatore viene inizializzato ad uno stato casuale e con il metodo **next()** passa ad un altro stato.

Classe Conditioner

Attributi:

state: conditioner_type è lo stato che il condizionatore può assumere

temperature: intero temperatura del condizionatore

id: intero che rappresenta un condizionatore specifico

Metodi:

Costruttore(id: intero, temperature: intero, state: conditioner_type): Inizializza un oggetto Conditioner con l'id, la temperatura e lo stato specificati.

next(): Genera casualmente e restituisce il prossimo stato del condizionatore.

getState(): Restituisce lo stato attuale del condizionatore.

modifyTemperature(newTemperature: intero): Modifica la temperatura del condizionatore.

getId(): Restituisce l'ID del condizionatore.

getTemperature(): Restituisce la temperatura attuale del condizionatore.

setState(newState: conditioner_type): Imposta lo stato del condizionatore al valore specificato.

Enumerazione conditioner_type:

ConditionerON: Il condizionatore si accende

ConditionerOFF: Il condizionatore si spegne

change_temperature: Cambio della temperatura

Pseudo-codice Dispositivi

Questa classe gestisce i dispositivi. Può programmare l'accensione del dispositivo in un certo range orario con il metodo **programmed_device(intervalloPrimo: intero, intervalloSecondo: intero** e puoi sapere anche quando con il metodo **getProgrammed**(). I dispositivi vengono inizializzati ad uno stato casuale e con il metodo **next()** passano ad un altro stato.

Classe Device

Attributi:

state: device_type , rappresenta uno stato del dispositivo

nome: nome_type, rappresenta il nome del dispositivo utilizzato

id: intero che rappresenta un dispositivo specifico

inizio: intero che rappresenta il periodo di inizio della programmazione

fine: intero che rappresenta il periodo di fine della programmazione

Metodi:

Costruttore(id: intero, state: device_type, nome: nome_type): Inizializza un oggetto Device con l'id, lo stato e il nome specificati.

next(): Genera casualmente e restituisce il prossimo stato del dispositivo.

programmed_device(intervalloPrimo: intero, intervalloSecondo: intero): Programma l'attivazione del dispositivo in un intervallo di tempo specificato.

getState(): Restituisce lo stato attuale del dispositivo.

getNome(): Restituisce il nome del dispositivo.

getId(): Restituisce l'ID del dispositivo.

setState(): Imposta lo stato del dispositivo a DeviceON.

setState(newState: device_type): Imposta lo stato del dispositivo al valore specificato.

getProgrammed(): Restituisce una tupla contenente l'inizio e la fine dell'intervallo programmato per il dispositivo.

Enumerazione device_type:

DeviceON : Accendi il dispositivo

DeviceOFF : Spegni il dispositivo

Programmed : Programma entro un certo range orario i dispositivi

Enumerazione nome_type: TV, LAVASTOVIGLIE, LAVATRICE, MACCHINA DEL CAFFE', MICROONDE, COPERTE RISCALDATE, CASSE, TAGLIAERBA

Pseudo-codice Luci

Questa classe gestisce le luci della casa. Può sia cambiare il loro colore con il metodo **setColor(newColor: light_color)** che la loro intensità con il metodo **setIntensity(newIntensity: intero).** Le luci vengono inizializzate ad uno stato casuale e con il metodo **next()** passano ad un altro stato.

Classe Light

Attributi:

state: light_type, rappresenta lo stato delle luci

color: light_color, rappresenta il colore delle luci

intensity: intero rappresenta l'intensità delle luci

id: intero che rappresenta una luce specifica

Metodi:

Costruttore(id: intero, state: light_type, color: light_color, intensity: intero): Inizializza un oggetto Light con l'id, lo stato, il colore e l'intensità specificati.

next(): Genera casualmente e restituisce il prossimo stato della luce.

setColor(newColor: light_color): Imposta il colore della luce al valore specificato.

getState(): Restituisce lo stato attuale della luce.

setIntensity(newIntensity: intero): Imposta l'intensità della luce al valore specificato.

getId(): Restituisce l'ID della luce.

getColor(): Restituisce il colore attuale della luce.

getIntensity(): Restituisce l'intensità attuale della luce.

setState(newState: light_type): Imposta lo stato della luce al valore specificato.

Enumerazione light_type:

LightON : Luci accese

LightOFF : Luci spente

change_intensity : Cambio dell'intensità

change_color : Cambio del colore

Enumerazione light_color: ROSSO, BLU, GIALLO , ROSA, ARANCIONE, BIANCO, VIOLA

Pseudo-codice Sensore

Questa classe gestisce i sensori di sicurezza. Possono rilevare un movimento con la funzione **setMovement(value: booleano) .** Se un movimento viene rilevato (*movement a true),*la telecamera di sorveglianza inizierà a registrare. I sensori vengono inizializzati ad uno stato casuale e con il metodo **next()** passano ad un altro stato.

Classe Sensor

Attributi:

state: sensor_type, rappresenta lo stato del sensore

id: intero che rappresenta una sensore specifico

movement: booleano rappresenta presenza di movimento

Metodi:

Costruttore(id: intero, state: sensor_type): Inizializza un oggetto Sensor con l'id e lo stato specificati.

next(): Genera casualmente e restituisce il prossimo stato del sensore.

getState(): Restituisce lo stato attuale del sensore.

getId(): Restituisce l'ID del sensore.

setMovement(value: booleano): Imposta il valore di movimento del sensore.

getMovement(): Restituisce il valore corrente di movimento del sensore.

setState(newState: sensor_type): Imposta lo stato del sensore al valore specificato.

Enumerazione sensor_type:

SensorON : I sensori vengono accessi

CHECKING : I sensori,accesi,controllano se c'è un movimento sospetto

SensorOFF : I sensori vengono spenti

Pseudo-codice Sensori del Giardino

Questa classe gestisce i sensori del giardino. Possono cambiare lo stato degli irrigatori in base alla temperatura e alla umidità rilevata dai sensori con i metodi **setTemperature(newTemperature: intero) e setHumidity(newHumidity: intero)** . I sensori del giardino vengono inizializzati ad uno stato casuale e con il metodo **next()** passano ad un altro stato.

Classe SensorGarden

Attributi:

state: sensorGarden_type, rappresenta lo stato del sensore del giardino

id: intero che rappresenta un sensore del giardino specifico

humidity: intero rappresenta l'umidità esterna

temperature: intero rappresenta la temperatura esterna

Metodi:

Costruttore(id: intero, state: sensorGarden_type, humidity: intero, temperature: intero): Inizializza un oggetto SensorGarden con l'id, lo stato, l'umidità e la temperatura specificati.

next(): Genera casualmente e restituisce il prossimo stato del sensore del giardino.

getState(): Restituisce lo stato attuale del sensore del giardino.

getId(): Restituisce l'ID del sensore del giardino.

getHumidity(): Restituisce l'umidità attuale rilevata dal sensore del giardino.

getTemperature(): Restituisce la temperatura attuale rilevata dal sensore del giardino.

setState(newState: sensorGarden_type): Imposta lo stato del sensore del giardino al valore specificato.

setTemperature(newTemperature: intero): Imposta la temperatura rilevata dal sensore del giardino al valore specificato.

setHumidity(newHumidity: intero): Imposta l'umidità rilevata dal sensore del giardino al valore specificato.

Enumerazione sensorGarden_type:

SensorGardenON : Sensori del giardino accesi

change_light : Cambio luci esterne

set_sprinklers : Gestione degli irrigatori in base alla temperatura esterna

SensorGardenOFF : Sensori giardino spenti

Descrizione Hub

Il **main**() è la funzione principale del programma che esegue il ciclo principale per il monitoraggio delle varie componenti del sistema. All'interno del ciclo, vengono eseguiti test e log delle varie componenti come fotocamera, condizionatore, dispositivo, luce, sensore e sensore del giardino. I test sono eseguiti utilizzando le funzioni di inizializzazione delle varie componenti (**initTest**()) e i loro stati vengono registrati nel database utilizzando le funzioni di logging (**log2db**). Infine, vengono eseguite delle operazioni di gestione del tempo (**nanos, monitorResponseTime, ecc**.) e di aggiornamento (**update_time**) prima di terminare il ciclo. Questo file coordina il funzionamento generale del sistema, inclusa l'inizializzazione delle componenti, il test periodico e il logging dei loro stati nel database.

Schema DB

```
-- Creazione della tabella Light
CREATE TABLE IF NOT EXISTS Light (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(20) NOT NULL,
    color VARCHAR(20) NOT NULL,
    intensity INT NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato della luce, il colore della luce, l'intensità, l'identificatore del processo e il tempo di inserimento del dato.

```sql
-- Creazione della tabella Camera
CREATE TABLE IF NOT EXISTS Camera (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(20) NOT NULL,
    recording INT NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato della telecamera, lo stato di registrazione, l'identificatore del processo e il tempo di inserimento del dato.

```sql
-- Creazione della tabella Conditioner
CREATE TABLE IF NOT EXISTS Conditioner (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(20) NOT NULL,
    temperature INT NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato del condizionatore, la temperatura, l'identificatore del processo e il tempo di inserimento del dato.

```sql
-- Creazione della tabella Device
CREATE TABLE IF NOT EXISTS Device (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(20) NOT NULL,
    nome VARCHAR(20) NOT NULL,
    inizio INT NOT NULL,
    fine INT NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato del dispositivi, il nome del dispositivo che stiamo utilizzando,il periodo di inizio della programmazione e il periodo di fine, l'identificatore del processo e il tempo di inserimento del dato.

```sql
-- Creazione della tabella Sensor
CREATE TABLE IF NOT EXISTS Sensor (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(15) NOT NULL,
    movement INT NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato del sensore, rilevazione del movimento, l'identificatore del processo e il tempo di inserimento del dato.

```sql
-- Creazione della tabella SensorGarden
CREATE TABLE IF NOT EXISTS SensorGarden (
    t INT NOT NULL,
    id INT NOT NULL,
    stato VARCHAR(15) NOT NULL,
    temperature INT NOT NULL,
    humidity INT NOT NULL,
    descr VARCHAR(30) NOT NULL,
    pid INT NOT NULL,
    temp VARCHAR(25) NOT NULL,
    PRIMARY KEY (t, pid)
);
```

Abbiamo t per l'iterazione dei test, l'id di riconoscimento, lo stato del sensore del giardino, la temperatura esterna, l'umidità esterna,descrizione dell'azione eseguita, l'identificatore del processo e il tempo di inserimento del dato.

```sql
CREATE TABLE IF NOT EXISTS LogActivity (
    name_activity varchar(20)NOT NULL,
    temp VARCHAR(25) NOT NULL,
    pid INT NOT NULL,
    PRIMARY KEY (name_activity, temp)
);
```

Creazione della tabella dell'attività eseguita con il suo nome, il tempo di inserimento e l'identificatore del processo.

```sql
CREATE TABLE IF NOT EXISTS SecurityRecap (
    temp VARCHAR(25) NOT NULL,
    pid INT NOT NULL,
    PRIMARY KEY (temp, pid)
);
```

Una tabella per il recap dei movimenti rilevati dai sensori e registrati dalle telecamere con tempo di inserimento del dato e l''identificatore del processo.

DESCRIZIONE CONNESSIONI REDIS

Utilizziamo le connessioni redis e tramite delle *get* e delle *set* facciamo comunicare tester e componenti per farsi passare i parametri da aggiornare. Comunicano tra di loro anche sensore e videocamera tramite la connessione Redis perchè il sensore passa alla connessione della videocamera quando c'è movimento cosi che inizi a registrare.

RISULTATI SPERIMENTALI

Table: Light

t	id	stato	color	intensity	pid	temp		
0	2062135979	change_intensity	PINK	20	606	2024-01-29 12:22:22		
1	2062135979	change_color	ORANGE	5	606	2024-01-29 12:22:22		
2	2062135979	LightOFF	ORANGE	5	606	2024-01-29 12:22:22		
3	2062135979	LightOFF	ORANGE	5	606	2024-01-29 12:22:22		
4	2062135979	LightOFF	ORANGE	5	606	2024-01-29 12:22:22		
5	2062135979	change_intensity	ORANGE	1	606	2024-01-29 12:22:22		
6	2062135979	LightON	ORANGE	1	606	2024-01-29 12:22:22		
7	2062135979	change_intensity	ORANGE	6	606	2024-01-29 12:22:22		
8	2062135979	change_color	YELLOW	6	606	2024-01-29 12:22:22		
9	2062135979	change_color	PINK	6	606	2024-01-29 12:22:22		

Table: Camera

t	id	stato	recording	pid	temp		
0	1863705823	CameraOFF	0	606	2024-01-29 12:22:22		
1	1863705823	CameraON	0	606	2024-01-29 12:22:22		
2	1863705823	CameraOFF	0	606	2024-01-29 12:22:22		
3	1863705823	CameraOFF	0	606	2024-01-29 12:22:22		
4	1863705823	CameraON	0	606	2024-01-29 12:22:22		
5	1863705823	Waiting	0	606	2024-01-29 12:22:22		
6	1863705823	Waiting	0	606	2024-01-29 12:22:22		
7	1863705823	CameraON	0	606	2024-01-29 12:22:22		
8	1863705823	CameraOFF	0	606	2024-01-29 12:22:22		
9	1863705823	CameraOFF	0	606	2024-01-29 12:22:22		

Table: Conditioner

t	id	stato	temperature	pid	temp		
0	1824747307	change_temperature	0	606	2024-01-29 12:22:22		
1	1824747307	change_temperature	4	606	2024-01-29 12:22:22		
2	1824747307	ConditionerON	4	606	2024-01-29 12:22:22		
3	1824747307	change_temperature	11	606	2024-01-29 12:22:22		
4	1824747307	ConditionerON	11	606	2024-01-29 12:22:22		
5	1824747307	ConditionerON	11	606	2024-01-29 12:22:22		
6	1824747307	ConditionerON	11	606	2024-01-29 12:22:22		
7	1824747307	ConditionerOFF	11	606	2024-01-29 12:22:22		
8	1824747307	ConditionerOFF	11	606	2024-01-29 12:22:22		
9	1824747307	change_temperature	8	606	2024-01-29 12:22:22		

Table: Device

t	id	stato	nome	inizio	fine	pid	temp		
0	1312277438	DeviceON	WASHING_MACHINE	0	0	606	2024-01-29 12:22:22		
1	1312277438	DeviceOFF	WASHING_MACHINE	0	0	606	2024-01-29 12:22:22		
2	1312277438	programmed	WASHING_MACHINE	5	20	606	2024-01-29 12:22:22		
3	1312277438	DeviceON	WASHING_MACHINE	5	20	606	2024-01-29 12:22:22		
4	1312277438	programmed	WASHING_MACHINE	21	21	606	2024-01-29 12:22:22		
5	1312277438	DeviceON	WASHING_MACHINE	21	21	606	2024-01-29 12:22:22		
6	1312277438	programmed	WASHING_MACHINE	11	9	606	2024-01-29 12:22:22		
7	1312277438	programmed	WASHING_MACHINE	17	3	606	2024-01-29 12:22:22		
8	1312277438	DeviceON	WASHING_MACHINE	17	3	606	2024-01-29 12:22:22		
9	1312277438	programmed	WASHING_MACHINE	12	15	606	2024-01-29 12:22:22		

Table: Sensor

t	id	stato	movement	pid	temp		
0	369753525	SensorON	0	606	2024-01-29 12:22:22		
1	369753525	SensorOFF	0	606	2024-01-29 12:22:22		
2	369753525	SensorON	0	606	2024-01-29 12:22:22		
3	369753525	SensorON	0	606	2024-01-29 12:22:22		
4	369753525	SensorOFF	0	606	2024-01-29 12:22:22		
5	369753525	CHECKING	0	606	2024-01-29 12:22:22		
6	369753525	SensorOFF	0	606	2024-01-29 12:22:22		
7	369753525	SensorOFF	0	606	2024-01-29 12:22:22		
8	369753525	CHECKING	1	606	2024-01-29 12:22:22		
9	369753525	SensorOFF	0	606	2024-01-29 12:22:22		

Table: SensorGarden

t	id	stato	temperature	humidity	descr	pid	temp		
0	2137607577	change_light	40	25	luci_accese	606	2024-01-29 12:22:22		
1	2137607577	change_light	43	25	luci_accese	606	2024-01-29 12:22:22		
2	2137607577	change_light	43	22	luci_accese	606	2024-01-29 12:22:22		
3	2137607577	change_light	45	23	luci_accese	606	2024-01-29 12:22:22		
4	2137607577	SensorGardenOFF	45	23	nothing_happened	606	2024-01-29 12:22:22		
5	2137607577	SensorGardenON	45	23	nothing_happened	606	2024-01-29 12:22:22		
6	2137607577	change_light	42	20	luci_accese	606	2024-01-29 12:22:22		
7	2137607577	SensorGardenOFF	42	20	nothing_happened	606	2024-01-29 12:22:22		
8	2137607577	SensorGardenON	42	20	nothing_happened	606	2024-01-29 12:22:22		
9	2137607577	SensorGardenON	42	20	nothing_happened	606	2024-01-29 12:22:22		

```
Table: LogActivity
name_activity  temp  pid
Test Sensor        2024-01-29 12:22:22  606
Test Camera        2024-01-29 12:22:22  606
Test Conditioner   2024-01-29 12:22:22  606
Test Device        2024-01-29 12:22:22  606
Test Light         2024-01-29 12:22:22  606
Test SensorGarden  2024-01-29 12:22:22  606
```

```
Table: SecurityRecap
temp  pid
2024-01-29 13:12:37  2297
```

Queste tabelle rappresentano ad ogni ripetizione di t per ogni componente, uno “Scenario” della nostra SmartHome. Nel caso dei test sopra eseguiti abbiamo alla ripetizione t=0 uno “Scenario” così composto:

- Cambiamo l’intensità delle luci (da 20 a 5);
- Spengiamo la telecamera di sorveglianza;
- Cambiamo la temperatura del condizionatore (da 0 a 4);
- Accendiamo la lavatrice;
- Accendiamo i sensori;
- E accendiamo le luci esterne;

Queste 6 azioni, una per ogni componente, va a formare un primo “Scenario”. Così via per ogni ripetizione di t, fino a creare 10 “Scenari” per la nostra SmartHome.

La tabella LogActivity registra semplicemente il “Log” dei vari Test eseguiti.

Infine SecurityRecap registra quando è avvenuto un movimento sospetto tramite l’interazione tra sensori e telecamere di sorveglianza.