



Documentazione Tecnica

Server Cloud, Monitor Consumi e AI locale

Progetto Smart Office - IoT and 3D Intelligent Systems

Arfaoui Bilel

Macellaro Vincenzo

Puzone Felicia



Panoramica

Il server principale funge come punto centrale dell'intero sistema, gestisce l'occupazione delle stanze, fornisce agli utenti Admin un'interfaccia per gestire lo spazio reso da loro disponibile ed elabora i dati necessari per l'AI e bot telegram.

Il Monitor di consumi è processo che traccia la potenza impiegata dalle sessioni attive, permettendo di costruire report e prendere provvedimenti in modo automatico se viene superata una soglia prefissata.

L'AI locale è stata pensata per evitare possibili rallentamenti in caso di mancata raggiungibilità del server AI remoto.

Obiettivi

1. Fornire un servizio back-end per la gestione della registrazione utenti, autenticazione tramite login, occupazioni delle stanze e personalizzazione della sessione.
2. Fornire un'interfaccia WEB semplice ed intuitiva, a supporto degli utenti Admin che vogliono gestire il loro spazio.
3. Monitorare i consumi delle occupazioni che hanno luogo in un'area geografica specifica, intervenendo in modo equamente distribuito in caso di consumi eccessivi.
4. Tracciare i consumi degli edifici e zone, e renderli disponibili per la lettura tramite bot Telegram.
5. Rendere il sistema resiliente ai malfunzionamenti e fornire un servizio prestante.

Specifiche

Tutte le componenti risiedono all'interno di un'istanza EC2 t2.micro con sistema operativo Windows Server 2022, fornita dal free-tier di AWS.

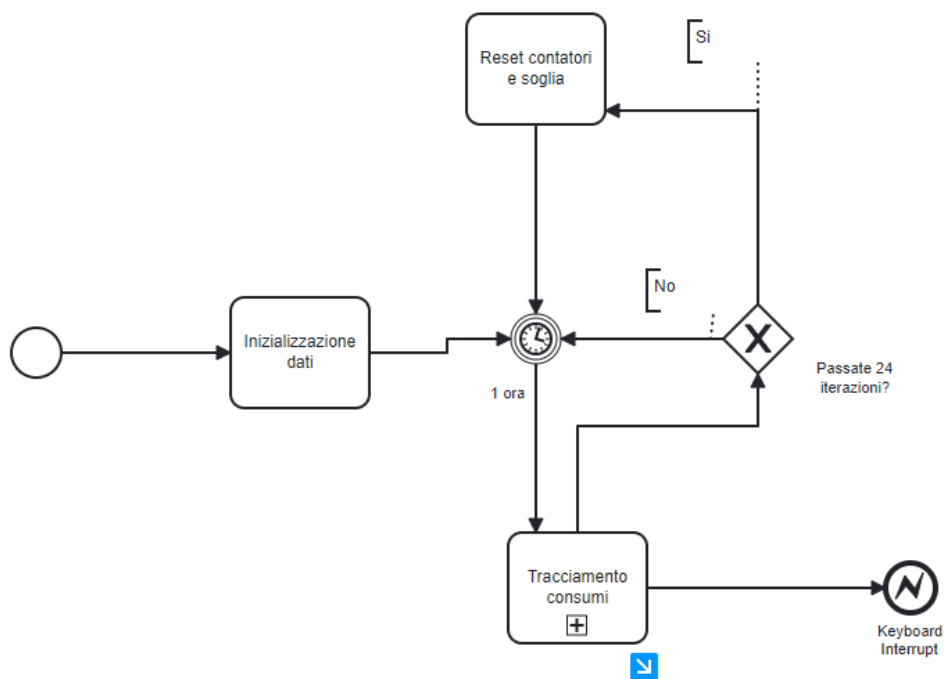
Monitor Consumi

Processo principale

Lo script del Monitor è scritto in Python e viene messo in esecuzione dal tecnico della zona, inizializzando i parametri del tracciamento che sono:

- Temperatura meteo attuale
- Città e Stato della zona
- Soglia massima dei consumi totali giornali
- Potenza termostato
- Percentuale di consumo aggiuntivo per gradi di scostamento dalla temperatura interna a quella esterna
- Timestamp d'inizio e fine
- Contatore iterazioni

Una volta messo in esecuzione, grazie alla libreria **scheduler** eseguirà la funzione **hour_count()** all'inizio di ogni ora, da inizio al tracciamento dei consumi. Dopo 24 iterazioni, che corrispondono a un intero giorno solare, fa un reset dei parametri. Il processo continuerà ad elaborare indefinitamente se non in caso di chiusura manuale da parte dei tecnici.



Rilevamento consumi

Per ogni iterazione il processo controlla se stanno avendo o hanno avuto luogo sessioni dei digital twin. Una volta rilevati calcola i consumi del termostato e LED per ognuna di esse, per poi ottenere un totale dei consumi e confrontarli con la soglia impostata all'inizio.

Formule per il calcolo dei consumi:

$$\Delta T = \text{Temperatura}(\text{esterna}) - \text{Temperatura}(\text{interna}) \text{ [C}^\circ\text{]}$$

$$\text{Consumo}(\text{intensita` LED}) = (\text{internsita`} * 15) \text{ [Watt]}$$

$$\text{possibili valori intensita`} = [1, 2, 3]$$

$$\text{Consumo}(\text{Termostato}) = \text{Potenza} * (1 + \Delta T * 0,03) \text{ [Watt]}$$

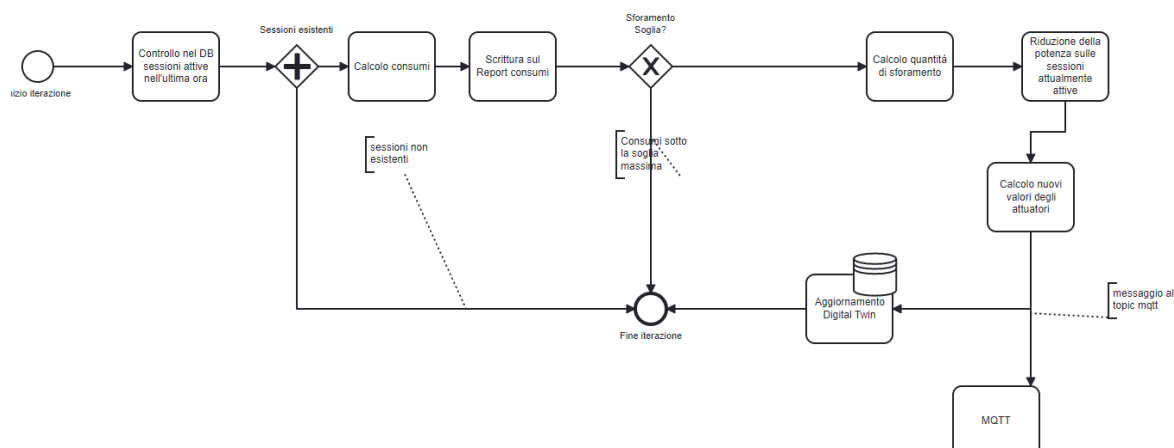
Gestione degli sprechi

Se i consumi superano la soglia, allora si ricava lo sfioramento di potenza e si applica una riduzione equamente distribuita su tutte le sessioni attive. Il processo calcola i nuovi valori degli attuatori, li comunica su MQTT e aggiorna il digital twin.

Formule per il calcolo nuovi valori degli attuatori:

$$\text{Riduzione} = \frac{\text{Sforamento}}{N \text{ Sessioni Attive}}$$

$$\text{Nuova Temperatura} = \left(\frac{(\text{Consumo Sessione} - \text{riduzione})}{(\text{Potenza Termostato})} - 1 \right) * \left(\frac{1}{0,03} \right)$$



Server Cloud

Il server Cloud, è una applicazione scritta in linguaggio di programmazione Python, segue il paradigma MVC (Model View Controller) grazie alla libreria Flask, fornisce i seguenti servizi:

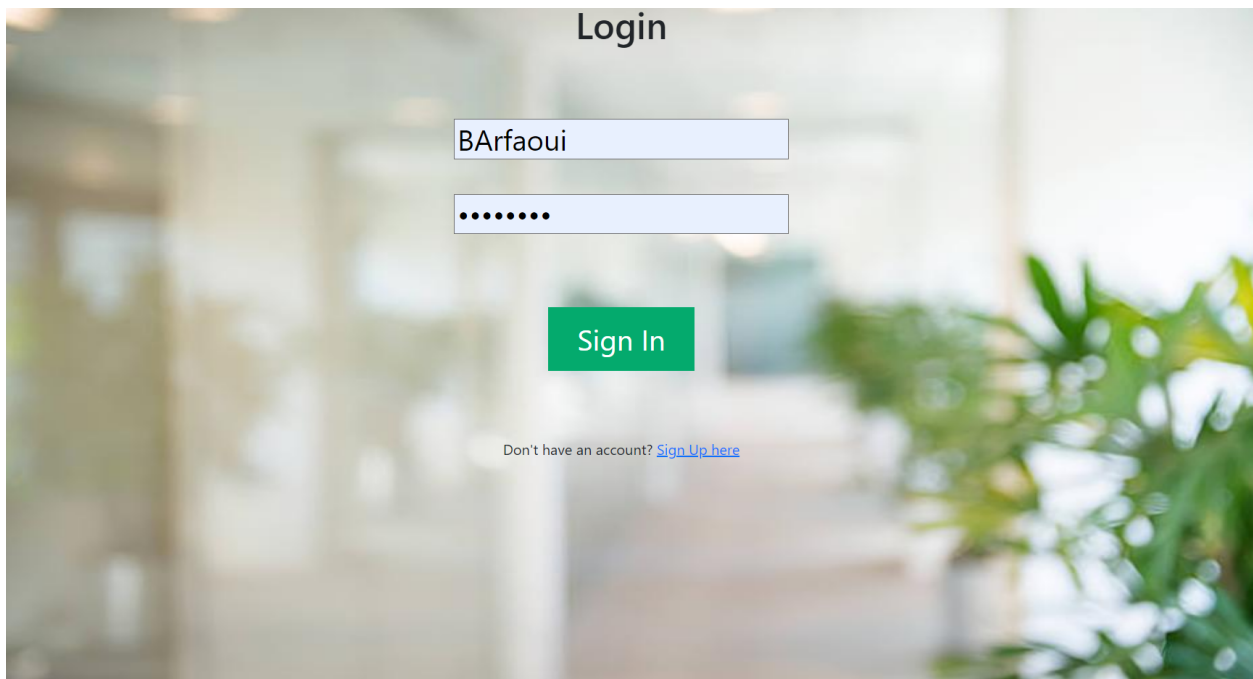
- Occupazione stanze
- Personalizzazione della sessione del DigitalTwin
- Interfaccia di gestione stanze
- Interfaccia di gestione edifici
- Interfaccia di gestione zone
- Visualizzazione dashboard zone, edifici e stanze
- Visualizzazione credenziali telegram

Route

/ [GET] /home [GET] /login [GET]

Descrizione: Route che indirizzano verso la schermata principale.

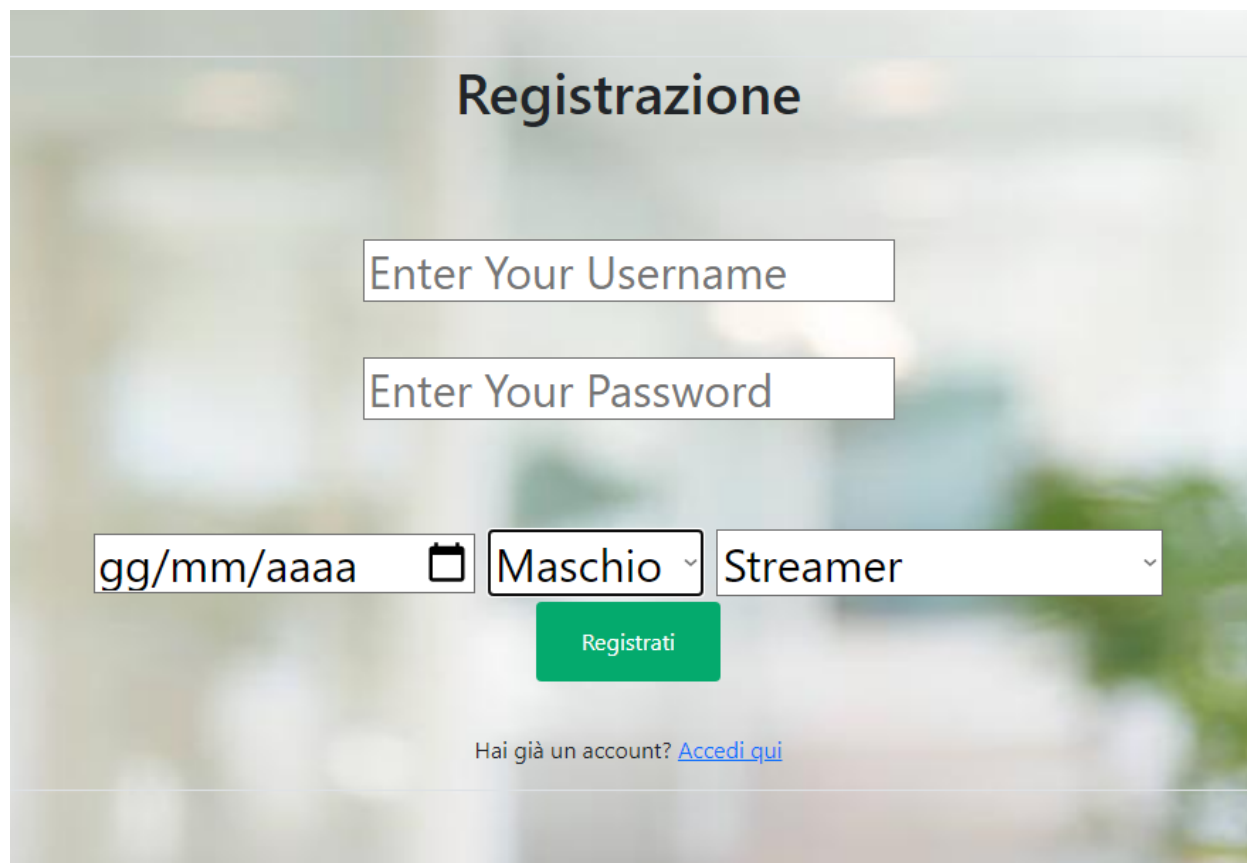
View:



/register [GET]

Descrizione: Route che indirizza verso la schermata di registrazione.




View:

A screenshot of a web registration form titled "Registrazione". The form is centered on a blurred background. It contains two text input fields: "Enter Your Username" and "Enter Your Password". Below these are three input fields: a text field with "gg/mm/aaaa" and a calendar icon, a dropdown menu with "Maschio" and a downward arrow, and another dropdown menu with "Streamer" and a downward arrow. A green button labeled "Registrati" is positioned below these fields. At the bottom, there is a link: "Hai già un account? [Accedi qui](#)".

Registrazione

Enter Your Username

Enter Your Password

gg/mm/aaaa  Maschio  Streamer 

Registrati

Hai già un account? [Accedi qui](#)

/login [POST]

Descrizione: Route che gestisce il meccanismo di autenticazione degli utenti. Il server prevede come Input i seguenti parametri:

Headers:{'Content-ID:Stringa'}

Parametri(Input): **Form {'username':Integer, 'password':String}**

Una volta ricevuti, il server controllerà se le credenziali sono corrette. In caso di esito negativo, l'utente riceverà un messaggio d'errore. Altrimenti il server fornirà i dati necessari in base al tipo di utente e la piattaforma che utilizza.

Risposta per APP Android {Content-ID:"LOGIN-APP"}

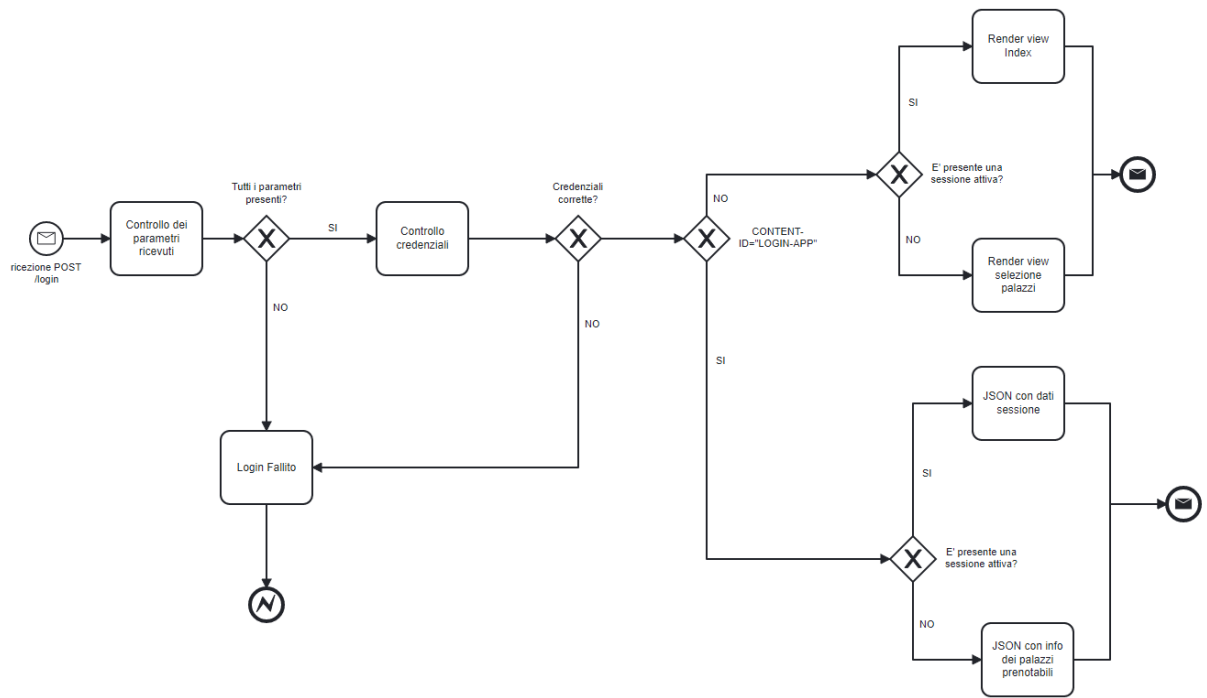
Login fallito : {logged_in: False}

Login con successo

```
{ logged_in: True,
  outcome: "Login",
  username:String,
  buildings: List
  token:String
}
```

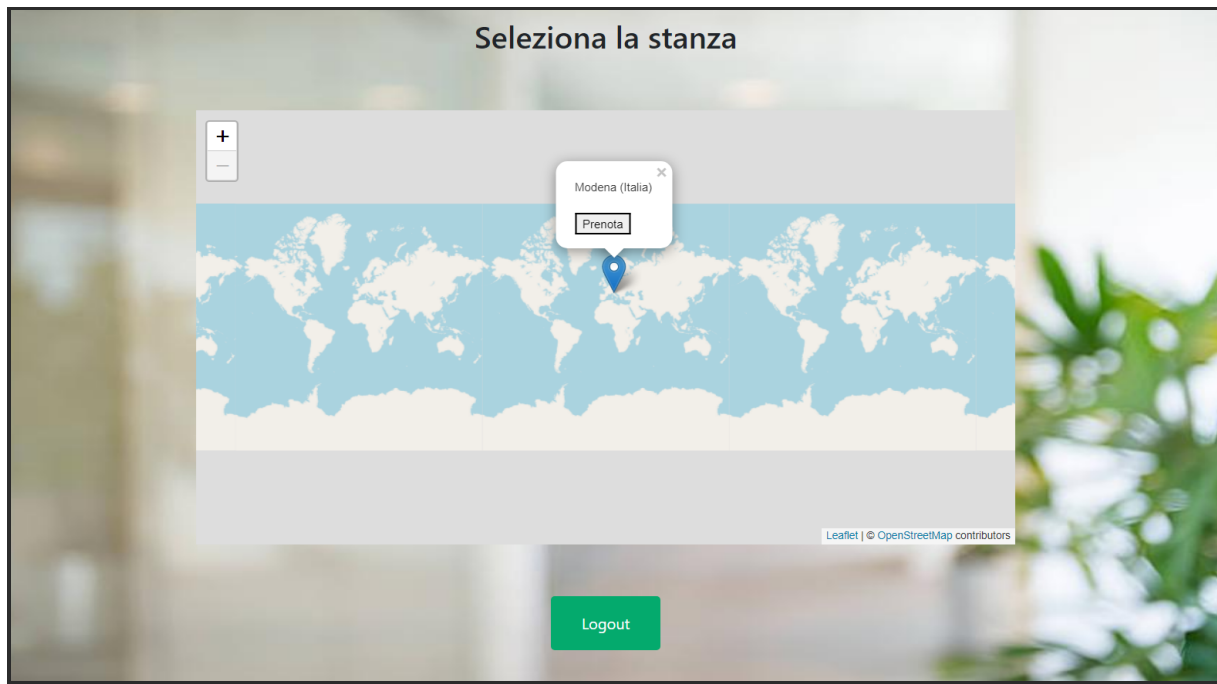
Login con successo e sessione attiva:

```
{ logged_in: True,
  outcome: "Active",
  digitalTwin:{
    room_color:String,
    room_brightness:Integer,
    room_temperature:Integer
  },
  id_edificio:Integer,
  id_room:Integer,
  username:String,
  weather: {temperature:Integer,humidity:Integer}
  token:String
}
```

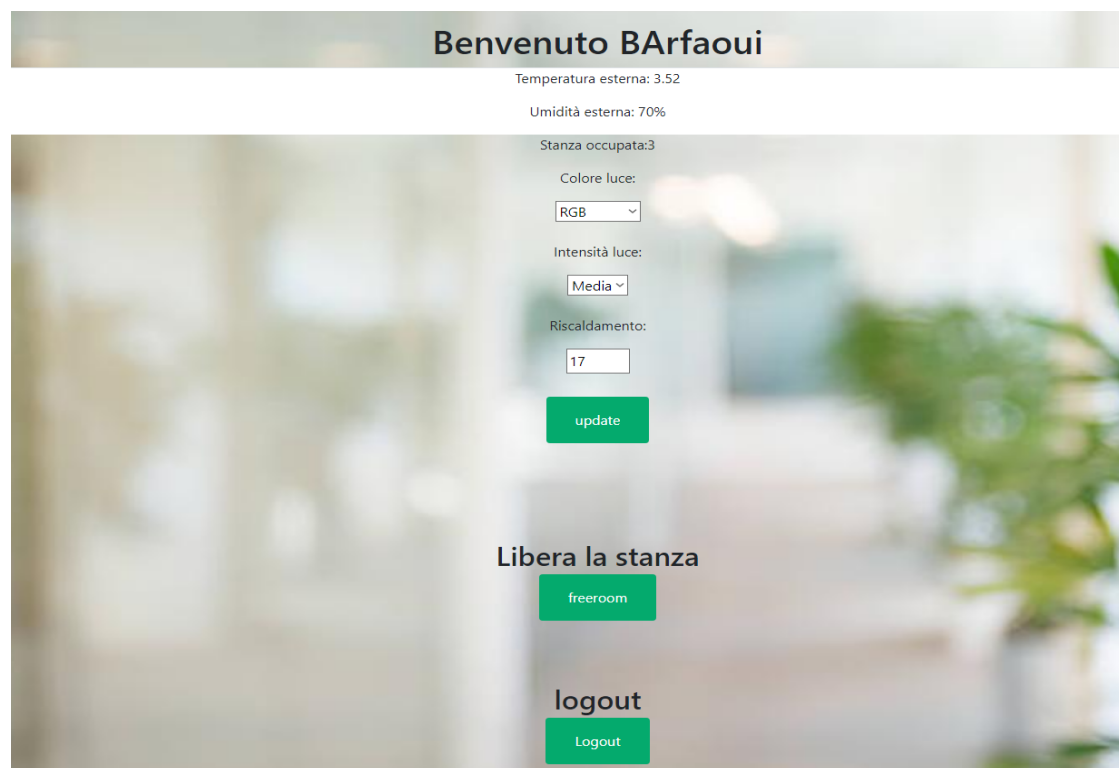


View client WEB

Selezione Stanza



Index (Sessione Attiva)



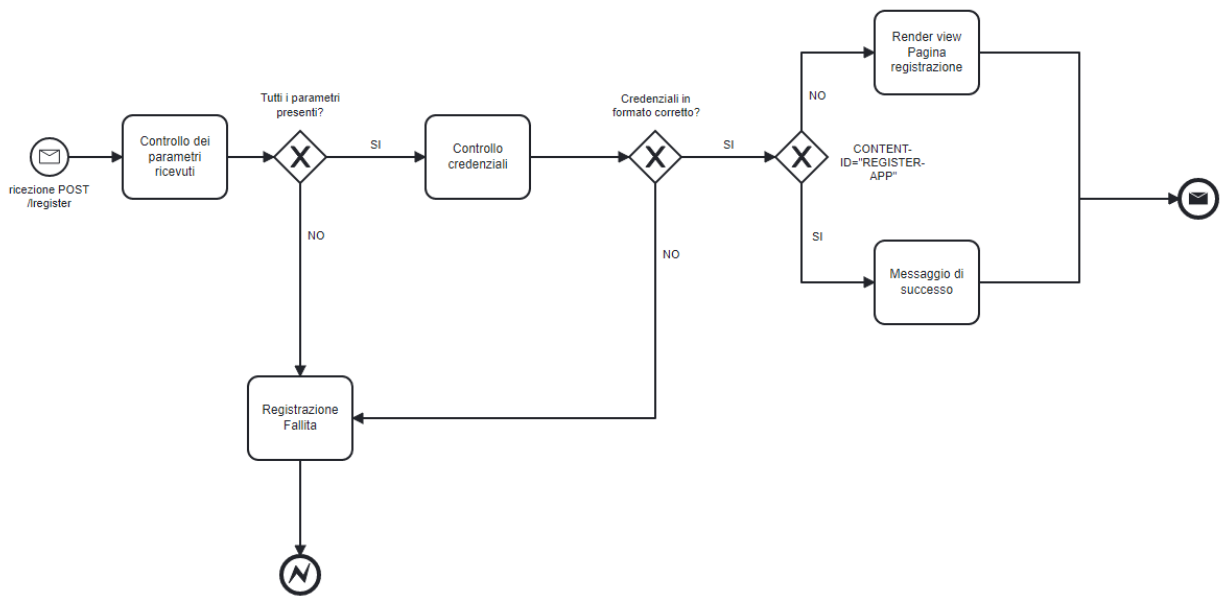
/register [POST]

Descrizione: Route che gestisce la registrazione degli utenti. Il server prevede come Input i seguenti parametri:

Headers:{'Content-ID:Stringa'}

Parametri(Input): **Form** {'username':Integer, 'password':String}

Una volta ricevuti, controllerà che i dati sono nel formato giusto, se si la registrazione verrà eseguita con successo, altrimenti l'utente riceverà un messaggio d'errore.



/logout [POST]@login_required

Descrizione: Route che gestisce il logout. Quando riceve la richiesta, ricava la sessione dal token, una volta fatto ciò rimuove le variabili di sessione per poi chiuderla.

Headers:{'Content-ID: Stringa', 'Token': Stringa}

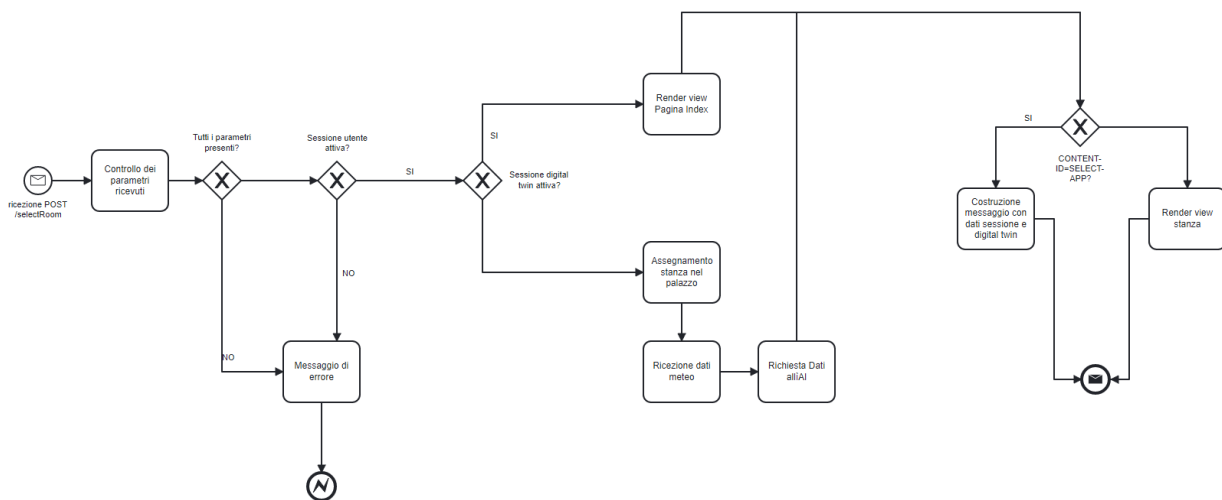
/selectRoom [POST] @login_required

Descrizione: Route che gestisce le richieste per l'occupazione di una stanza.

Headers: {'Content-ID:String', token:String}

Parametri(Input): Form {building:Integer}

Parametri(Input): JSON {building_id:Integer}



Risposta per APP Android {Content-ID:"LOGIN-APP"}

Occupazione fallita : {outcome: "Full", buildings=List}

Occupazione con successo o sessione attiva:

```

{ logged_in: True,
  outcome: "Active",
  digitalTwin:{
    room_color:String,
    room_brightness:Integer,
    room_temperature:Integer
  },
  id_edificio:Integer,
  id_room:Integer,
  username:String,
  weather: {temperature:Integer,humidity:Integer}}

```

/dashboardroom [POST] @login_required

Descrizione: Route che reindirizza verso la pagina di dashboard, che contiene informazioni sullo storico e valori attuali dei sensori e attuatori, meteo e consumi.

Parametri(Input): Form {id_room':Integer}

/dashboardbuilding [POST] @login_required

Descrizione: Route che reindirizza verso la pagina di dashboard, che contiene informazioni sullo storico e valori attuali dei sensori e attuatori, meteo e consumi.

Parametri(Input): Form {id_building:Integer}

/dashboardzone [POST] @login_required

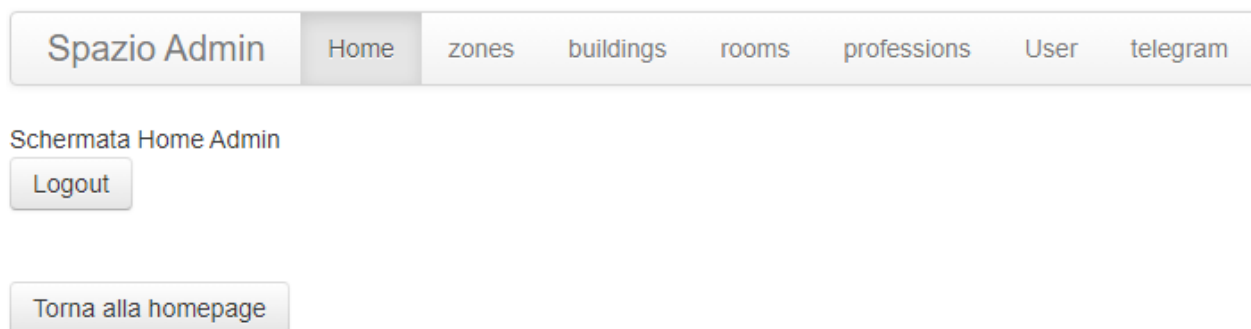
Descrizione: Route che reindirizza verso la pagina di dashboard, che contiene informazioni sullo storico e valori attuali dei sensori e attuatori, meteo e consumi.

Parametri(Input): Form {id_zone:Integer}

/admin [GET] @login_required

Descrizione: Schermata di gestione per gli utenti Admin.

view



/admin/zones [GET] @login_required

Descrizione: Schermata di gestione zone. E' presente una colonna in più che reindirizza alla dashboard /dashboardzone.

view

List (1)	Create	With selected ▼		
<input type="checkbox"/>		City	State	Dashboard
<input type="checkbox"/>		Modena	Italia	Dashboard

/admin/buildings [GET] @login_required

Descrizione: Schermata di gestione palazzi. E' presente una colonna in più che reindirizza alla dashboard /dashboardbuilding.

view

List (1)	Create	With selected ▼			
<input type="checkbox"/>		City	Address	Available	Dashboard
<input type="checkbox"/>		Modena	(Italia)		Dashboard

/admin/User [GET] @login_required

Descrizione: Schermata di gestione utenti. Solo i super admin possono vedere gli altri utenti e assegnare ruoli.







view

Spazio Admin	Home	zones	buildings	rooms	professions	User	telegram
List (2)							
	Username	Profession	Admin	Super User			
	Admin	Administrator					
	BArfaoui	Administrator					

/admin/rooms [GET] @login_required

Descrizione: Schermata di gestione stanze. E' presente una colonna in più che reindirizza alla dashboard /dashboardroom.

view

List (3)	Create	With selected ▾		
<input type="checkbox"/>		Id Room	Description	Available
<input type="checkbox"/>	 	1	Room of building id:1 stationed at Modena (Italia)	●
<input type="checkbox"/>	 	2	Room of building id:1 stationed at Modena (Italia)	⊙
<input type="checkbox"/>	 	3	Room of building id:1 stationed at Modena (Italia)	⊙

Dashboard



































Dashboard

Dashboard

/admin/zones [GET] @login_required

Descrizione: Schermata di gestione delle professioni. Solo i super admin hanno i permessi di creare, eliminare e modificare la tabella.

view

List (17)	Create	With selected ▾	
<input type="checkbox"/>		Name	Category
<input type="checkbox"/>	 	Administrator	5
<input type="checkbox"/>	 	Streamer	0
<input type="checkbox"/>	 	Blogger	0
<input type="checkbox"/>	 	Televendite	0
<input type="checkbox"/>	 	Professore/Istruttore	1
<input type="checkbox"/>	 	Seminarista	1
<input type="checkbox"/>	 	Snake oil seller	1
<input type="checkbox"/>	 	Assistenza telefonica	2
<input type="checkbox"/>	 	Programmatore	2
<input type="checkbox"/>	 	Contabile	2
<input type="checkbox"/>	 	Manager	2
<input type="checkbox"/>	 	Elettricista	3
<input type="checkbox"/>	 	Sistemista	3
<input type="checkbox"/>	 	Colf/Badante	4
<input type="checkbox"/>	 	Babysitter	4
<input type="checkbox"/>	 	Operatore CAF/CISL	5
<input type="checkbox"/>	 	Operatore NASPI	5

Decorator principali

@app.errorhandler(HTTPException)

Descrizione: Decorator generico che gestisce gli errori delle richieste HTTP ricevute, i client WEB verranno reindirizzati sulla schermata home, mentre l'app riceverà un messaggio di errore 400.

@app.route

Descrizione: Decorator che espone una route del server Flask, permette di specificare quali metodi sono autorizzati.

@mqtt.on_message()

Descrizione: Decorator che permette al client Flask di rimanere in ascolto ai canali MQTT, ricevere messaggi e fare operazioni in base a ciò che è stato ricevuto.

@login_required

Descrizione: Decorator che rende una route accessibile solo agli utenti autenticati, quando una route protetta viene chiamata, il decorator farà un controllo tramite @login_manager.request_loader, in caso di esito positivo allora l'accesso è permesso, nel caso contrario la gestione della richiesta verrà passata a @login_manager.unauthorized_handler.

@login_manager.request_loader

Descrizione: Ogni volta che questo decorator viene richiamato, il server ricaverà il token inviato dalla richiesta utente. Il server controlla la validità del token se è valido sarà permesso continuare la richiesta, altrimenti delegherà la gestione a @login_manager.unauthorized_handler.

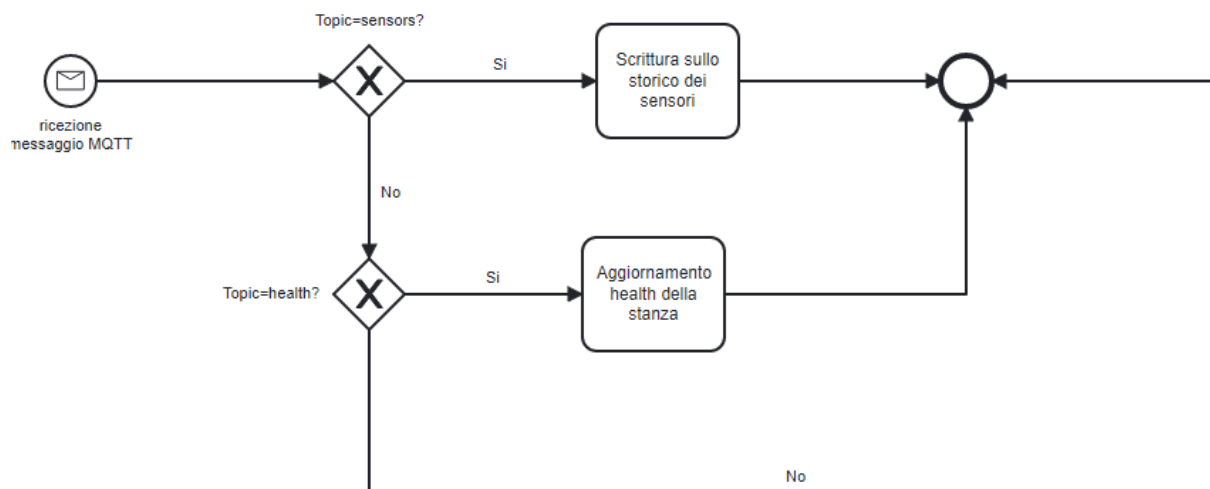
@login_manager.unauthorized_handler

Descrizione: Questo decorator ha il compito di gestire le chiamate non valide alle route non protette. Se il client è WEB ritornerà la pagina home, invece per l'app manderà una risposta di errore "401 not logged in".

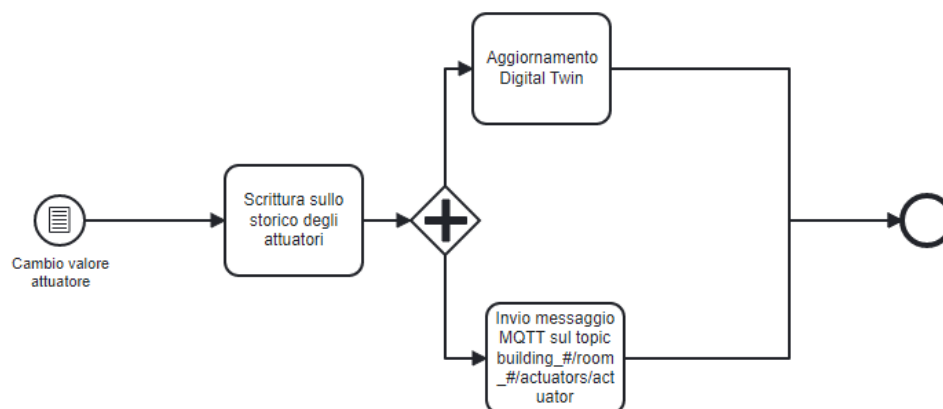
TOPIC MQTT

All'avvio o dopo l'aggiunta di una stanza, il server s'iscrive ai seguenti topic MQTT, grazie al decorator on_message potrà rimanere in ascolto ai topic:

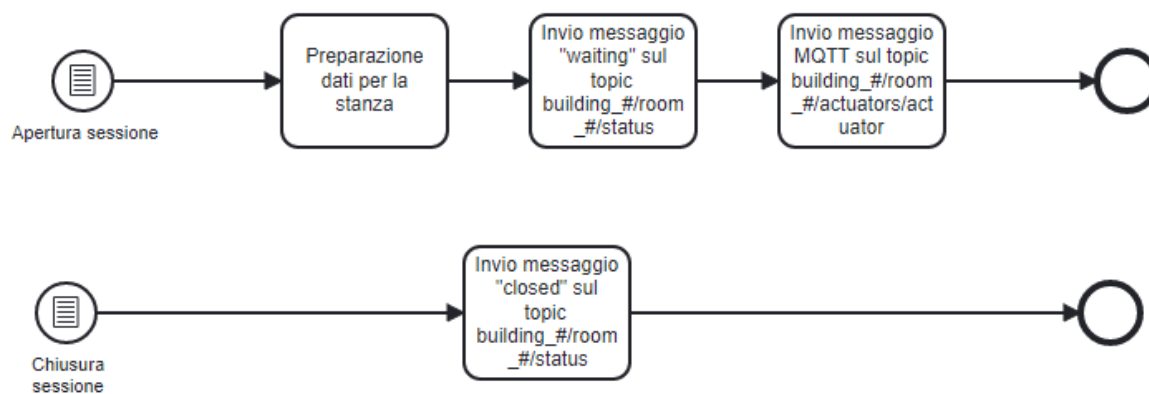
- **building_#/room_#/sensors (read only):** Topic dei sensori
- **building_#/room_#/health (read only):** Topic sulla salute di una stanza



- **building_#/room_#/actuators(write only):** Topic degli attuatori



- **building_#/room_#/status (write only):** Topic per sincronizzare lo stato di una sessione. Questi messaggi serviranno alla sincronizzazione della stanza con il digital twin.



Modelli DB

User

Modello che rappresenta ogni utente che interagisce con il sistema.

id integer

Unique identifier for the given user.

username string

password string

dateOfBirth string <date>

Example: 1997-10-31

sex integer

Set to true if the user's email has been verified.

>= 0 <= 2

admin boolean

The date that the user was created.

super_user boolean

DigitalTwinFeed

Rappresenta una stanza resa disponibile dal sistema per l'occupazione. Contiene i valori più recenti dei sensori e attuatori.

id_room integer

light_sensor integer

noise_sensor integer

led_actuator integer

temperature_actuator integer

led_brightness integer

sensorFeeds

Questo modello serve per costruire lo storico dei sensori di un DigitalTwin.

`id_room` integer

`type_of_sensor` string

`value` string

`timestamp` string

actuatorFeeds

Questo modello serve per costruire lo storico degli attuatori di un DigitalTwin. Permette di tracciare le tendenze e i consumi.

`id_session` integer

`type_of_value` string

`value` string

`timestamp` string

rooms

Questo modello contiene tutte le informazioni aggiuntive di una stanza.

id_room integer

id_building integer

description string

available boolean

dashboard object

Colonna che verrà mostrata come bottone che reindirizzerà verso la Dashboar

buildings

Rappresenta un palazzo registrato nel sistema, contiene informazioni sulla posizione geografica e disponibilità.

id_building integer

city string

address string

lat string

lon string

available string

dashboard object

Colonna che verrà mostrata come bottone che reindirizzerà verso la Dashboard.

Zones

Rappresenta un'area geografica gestita da un Utente Admin.

id_zone integer

id_admin integer

city string

lat string

lon string

dashboard string

Colonna che verrà mostrata come bottone che reindirizzerà verso la Dashboar

dailyBuildingConsumptionReport

Contiene i consumi giornalieri di un edificio, suddiviso per consumi LED e termostato.

id_building integer

temperature integer

Consumi termostato.

light integer

Consumi intensità LED.

timestamp string<date>

Example



```
1  {  
2    "id_building": 2,  
3    "temperature": 30000,  
4    "light": 1000,  
5    "timestamp": "2019-08-24"  
6  }
```

dailyRoomConsumptionReport

Contiene i consumi giornalieri di una stanza, suddiviso per consumi LED e termostato.

id_room integer

temperature integer

Consumi termostato.

light integer

Consumi intensità LED.

timestamp string<date>

Example



```
1  {  
2    "id_room": 3,  
3    "temperature": 1000,  
4    "light": 50,  
5    "timestamp": "2019-08-24"  
6  }
```

monthlyBuildingConsumptionReport

Contiene i consumi mensili di un edificio, suddiviso per consumi LED e termostato.

id_building integer

temperature integer

light integer

timestamp string<date>

Example



```
1  {  
2    "id_building": 0,  
3    "temperature": 10000000,  
4    "light": 40000,  
5    "timestamp": "2019-08"  
6  }
```

profession

Rappresenta le professioni presenti nel DB

id_profession integer

name string

category integer

monthlyRoomConsumptionReport

Contiene i consumi mensili di una stanza, suddiviso per consumi LED e termostato.

id_room integer

temperature integer

Consumi termostato.

light integer

Consumi intensità LED.

timestamp string<date>

Example



```
1  {  
2    "id_room": 2,  
3    "temperature": 10000,  
4    "light": 450,  
5    "timestamp": "2019-08-24"  
6  }
```

sessions

Tabella che contiene le informazioni sulla durata della sessione.

id_session integer

timestamp_begin string<date-time>

timestamp_end string<date-time>

sessionStates

Contiene tutte le informazioni riguardanti una sessione del DigitalTwin.

id_session integer

id_user integer

id_room integer

active boolean

telegram

Tabella che contiene le credenziali per l'autenticazione al BOT Telegram.

id_user integer

telegram_key string

Codice univoco a 6 cifre.

weatherReport

Storico del meteo associato a una zona.

id_zone integer

temperature integer

humidity integer

timestamp string<date-time>

AI Locale

Il server Flask AI locale funge come failsafe nel caso, il server AI remoto non sia remoto.

Route

/AI [POST]

Questa route prende in input la temperatura meteo, sesso ed età dell'utente che vuole occupare la stanza, il server elabora i valori degli attuatori in base ai parametri ricevuti.

Parametri(Input): **JSON** {'user_sex':Integer, 'ext_temp':Integer,'user_age':Integer}

Risposta(Output): {'user_temp':Integer, 'user_light':Integer, 'user_color':Integer}