Relazione Tecnica NAO Challenge23

NAONEXUS

PER MAGGIORI INFORMAZIONI:

Visita il nostro sito
Visita la nostra repository
Scrivi una email a: *socialnaonexus@gmail.com*

Contents

Abstract:

Nella *NAO Challenge* di quest'anno è stato scelto di sviluppare due progetti orientati alla sostenibilità ed energia green. La *missione* del team è di avvicinare le persone alla sostenibiltà rendendola più accessibile e facilmente comprensibile. Questi progetti introducono le persone nel mondo dell'energia solare ed il loro scopo è di prevenire sprechi di energia in grandi strutture.

1. METODOLOGIA E TECNICHE USATE

Essendo un gruppo numeroso c'era la necessità di tenere traccia dei *tasks* e tenere ogni componente al passo con l'ultima versione del software sviluppato. Per raggiungere questo scopo il team ha sfruttato due importanti tecnologie: *GitHub* e *Notion*.

1.1. DevOps con Notion e metodologia Agile

Per lavorare al progetto e tenere traccia del progresso il team ha usato *Notion*, una piattaforma collaborativa per la gestione di note. Il termine *DevOps* proviene da *software development* e *technology operations*, ed è costituito dai vari metodi che hanno scopo di migliorare la collaborazione *end-to-end*. La web-app Notion presenta molti strumenti come gestione di task, tracciamento andamento progetto e liste to-do e grazie ad essi ogni componente del team è stato al corrente degli incarichi da portare a termine ad ogni appuntamento. Questa piattaforma di collaborazione integra wiki e database creando uno spazio di lavoro per gestione di dati e task. I progetti sono stati gestiti con la metodologia *Agile*: i compiti principali sono stati divisi in sub-tasks per essere poi distribuiti tra i membri del team; questo ha portato ad una maggiore efficenza nello sviluppo del software.

1.2. GitHub

GitHub è uno strumento importante che ha aiutato il team a salvare e gestire il codice. È una piattaforma cloud-based di hosting di respository Git che aiuta i membri a tenere traccia dei cambiamenti fatti nel codice grazie al controllo versione.
Attraverso il *branching* e *merging* ogni membro è stato capace di lavorare in modo sicuro sul codice evitando di intaccare direttamente la versione principale. Tutte le modifiche apportate sono tracciate e possono essere annullate nel caso portino ad errori.

I DUE PROGETTI

Il primo progetto vede NAO come un *assistente* in un edificio. NAO è connesso ad un telefono grazie ad un server HTTP. Gli utenti possono connettere lo smartphone al server ed inviare dati dell'ambiente scelto rilevati grazie ad un sensore. Quando NAO riceve questi dati controlla la domotica dell'edificio accendendo e spegnendo le *luci*, alzando e abbassando le *tapparelle* o accendendo e spegnendo il *riscaldamento*.

Il secondo progetto prevede la collaborazione con un'azienda esterna e il team ha collaborato con *Amperia*, la quale punta a rendere la transizione green più efficente e semplice. Connettendo NAO al sito *SolarEdge* (una piattaforma che consente di progettare installazioni di pannelli fotovoltaici) attraverso un server HTTP l'utente viene guidato nella compilazione del sito che crea questo progetto.

2. NAO COME AIUTANTE DOMOTICO

Questo progetto comprende che NAO comunichi con un sensore per riuscire a ricavare le condizioni ambientali e sulla base di queste misurazioni controlla il server della domotica. Il progetto si sviluppa su più step: inizialmente i dati vengono misurati con il sensore, il PDF generato viene poi selezionato con l'app sviluppata. Questi dati vengono poi inviati al server insieme al numero di persone presenti nella stanza, che viene inserito direttamente sull'app. Questi dati vengono inviati a NAO, il quale manda i comandi alla domotica se specifici limiti sono sorpassati. Il progetto presenta vari elementi:

- Sensore: dispositivo esterno che misura dati ambientali ed esporta i risultati su un PDF;
- App: una app che legge il PDF e lo invia al server HTTP insieme al numero di persone presenti nella stanza e alla luce interna;
- HTTP Server: un server Rest che consente di visualizzare i dati estratti dal PDF tramite pagine web e comunica con il server della domotica;
- NAO: il robot umanoide che ricava i dati dal server e controlla la domotica sulla base di questi dati, comunicando inoltre i dati ricavati all'utente.

.

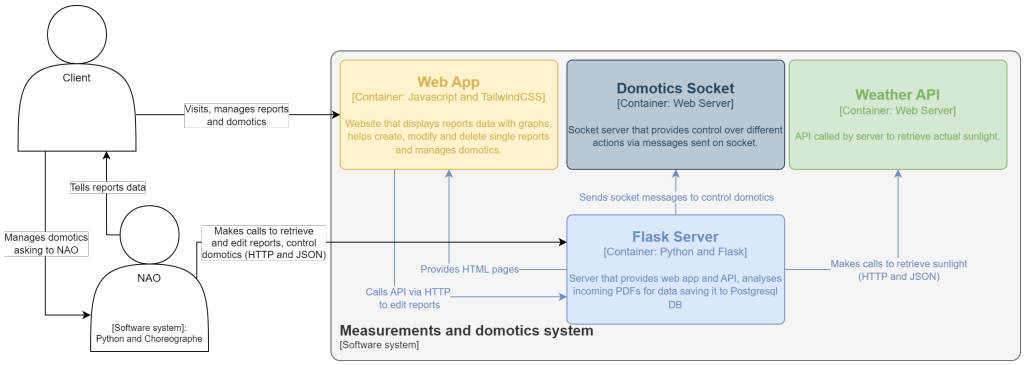

Fig. 1. architettura software NAO domotica

2.1. Il sensore

Una parte significativa del progetto era quella di riuscire a ricavare ed analizzare i dati della concentrazione di CO2, temperatura e umidità dell'ambiente selezionato. In un primo momento si pensò di usare un Arduino da connettere al server HTTP. Durante lo sviluppo del software il team ha incontrato molti problemi ed è stato deciso di comprare un sensore. Questo ha garantito risultati più precisi, i quali però vengono esportati in un file PDF.

Per analizzare i valori c'era il bisogno di avere un dispositivo che ricavasse i dati inviandoli al server. In un primo momento il team decise di connettere il sensore direttamente in una delle USB di NAO ma senza i permessi root gli archivi esterni non possono essere montati sull'umanoide, rendendo impossibile l'accesso a qualsiasi memoria esterna da NAO. E' stato quindi deciso di usare una app per smartphone.

2.2. L' App

Questa app viene usata per caricare i PDF e mandarli al server. E' stata creata in quanto non c'era alcun modo per connettere il sensore direttamente al server. Questo device manda i dati del sensore al server aggiungendo altre misurazioni utili che esso non può effettuare. Questa applicazione è stata scritta in *Flutter* per renderla cross-platform e flessibile. Include un file picker e un input per inserire il *numero di persone* in una stanza. Nel momento in cui viene inviato il file viene aggiunto alla richiesta il valore della *luce* misurato con il sensore del telefono. L'applicazione inoltre elenca tutti i dati conservati nel server.

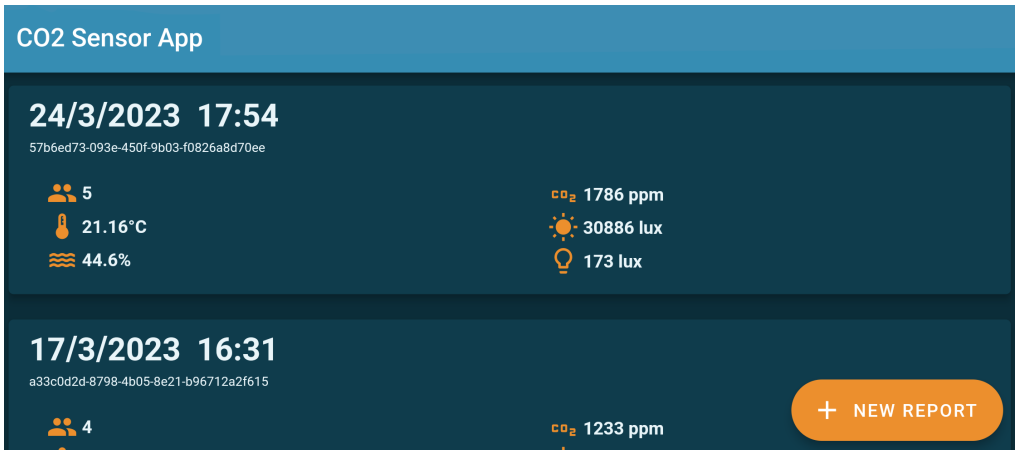

Fig. 2. Applicazione domotica

2.3. NAO

NAO è la parte centrale del progetto. È connesso al server HTTP e riceve e salva i dati dell'ambiente selezionato, i quali sono usati da NAO per capire le chiamate da fare al server. Con il controllo domotico, NAO dice all'utente cosa sta facendo. Il robot ha una funzionalità che permette di farlo diventare uno "smart home speaker" come *"Alexa"*: l'end-user può chiedere a NAO di apportare modifiche alla domotica controllandola direttamente. Quando vengono ricevuti nuovi dati, NAO li comunica all'utente e basandosi su vari *thresholds* può controllare gli smart devices.

2.4. HTTP Server

È il punto centrale in cui tutti i dati del progetto sono gestiti, è una parte del codice molto complessa scritta in Python, la quale analizza PDF, aggiunge informazioni mancanti, presenta una Web App che visualizza e gestisce dati e ha al suo interno la Rest API per connettere tutti i client come l' app e NAO. L'idea iniziale era di eseguire il server su NAO ma le ultime versioni del software hanno rimosso la possibilità di entrare come utente *root* e di installare pacchetti Python. È stato successivamente deciso di eseguire il server su un *Raspberry PI* con un *IP statico* per renderlo accessibile da ogni client nella rete.

2.4.1. Web App

Una parte importante del server è la web app che viene servita da esso: le pagine *HTML* sono popolate utilizzando Jinja e sono inviate usando Flask. Questa libreria è reponsabile del

funzionamento del server Rest che distribuisce i dati anche in formato *JSON*. In questa web app i dati dei report sono visualizzabili sotto forma di *grafico* o *lista*. Ogni report è *modificabile* ed *eliminabile*; è anche possibile *aggiungere nuovi report* direttamente dall'UI.

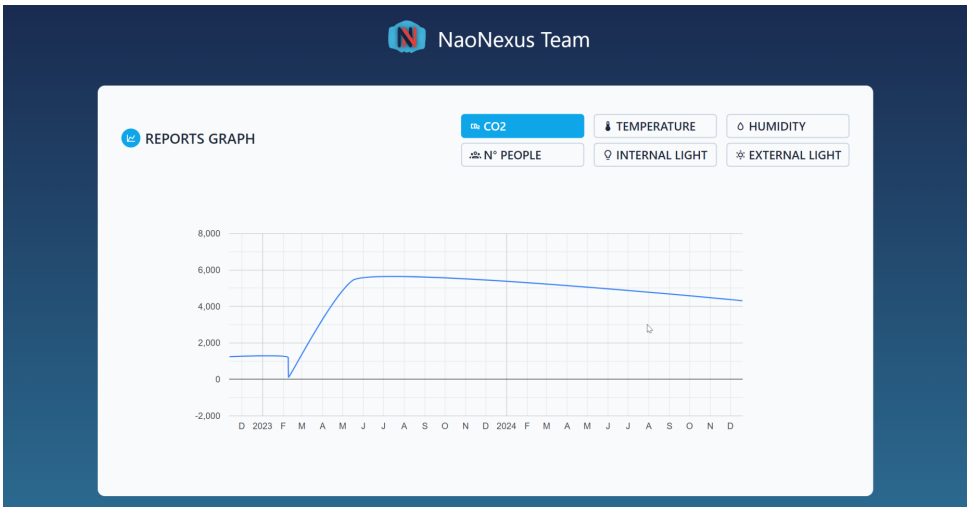

Fig. 3. Schermata con il riassunto dei report

2.4.2. Rest API

La documentazione per l'API può essere trovata qui.

Il servizio Rest facilita lo scambio di dati tra tutti i client connessi, i quali sono poi inviati con *JSON* usando *HTTP* e sono accessibili poi tramite diversi endpoints. Il server Rest è scritto in Flask come la web app.

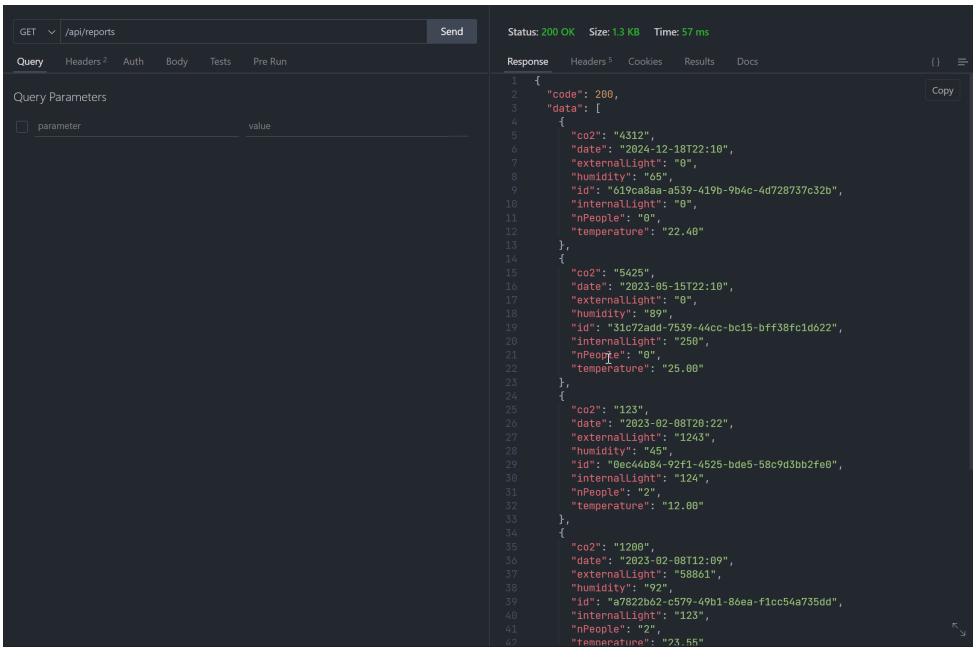

Fig. 4. Esempio risposta servizio Rest

2.4.3. Analisi PDF

Per analizzare il PDF è stato necessario usare la libreria pyPDF2, che ha consentito la conversione del file creato dal sensore di CO2 in una stringa. Il team ha trovato varie difficoltà nell'individuare i dati all'interno della stringa ma dopo aver eliminato *spazi* e *caratteri di fine linea* il problema è stato risolto. Grazie a questa modifica si è riuscito ad individuare i dati cercando delle *keywords* presenti in specifiche parti del PDF. In questo modo diviene possibile estrapolare vari dati dal PDF come data della misurazione, CO2, temperatura e umidità.

2.4.4. Controlli domotici

Per controllare le luci il server è stato connesso con *Telnet* al server domotico dell'edificio. Dopo aver riscontrato alcune difficoltà nel trovare i comandi giusti e grazie alla guida di *Amperia*, è stato possibile capire i giusti codici univoci assegnati a *luci*, *LIM*, *tapparelle* e *termostati* in un determinato ambiente, in maniera da controllarli tramite connessione *Telnet*. Questa connessione ha fatto diventare l' HTTP Server un *client* per il server domotico in modo da controllare i dispositivi connessi.

2.4.5. Analisi luce esterna

La misurazione della luce interna è ricavata dal sensore del telefono tramite un'app ma la luce esterna non può essere ricavata direttamente da quest'ultima. La prima idea fu di aggiungere un *Arduino* esternamente ma questo metodo non poteva essere applicato ad ogni possibile situazione ed edificio. La soluzione trovata è stata quella di fare chiamate ad una API esterna (Open meteo), la quale ritorna l'irradiazione solare di una posizione in $\frac{W}{m^2}$ che è poi convertita in *lux* dal server.

3. NAO COME GUIDA A SOLAREDGE

Il progetto consiste in un interazione del NAO con il client del partner *Amperia*, in cui vengoni richiesti i dettagli della struttura scelta, al fine di generare un progetto per una transizione solare con il sito Solaredge, rendendo la sostenibilità accessibile a tutti. Tutto ciò si suddivide in due fasi: una fase iniziale dove NAO interroga il client e inserisce le risposte su un sito, richiedendo eventualmente un inserimento manuale quando non è in grado di controllare il sito. La seconda fase consiste in una spiegazione da parte del NAO di tutti i risultati ottenibili dal progetto e del suo relativo impatto ambientale. Il progetto è suddiviso in più elementi:

- Server HTTP: un server Rest che comunica con il NAO e controlla il sito inserendo i dati richiesti
- NAO: un robot umanoide che richiede i dettagli e invia le risposte al server che procederà all'analisi. Inoltre presenta a voce i risultati del progetto generato con il sito web Solaredge.

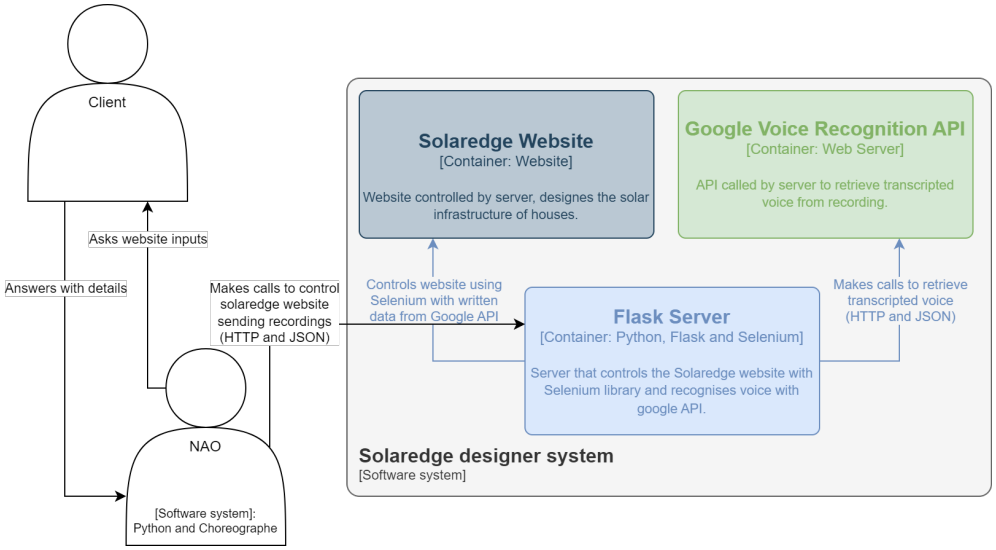

Fig. 5. Solaredge architecture

3.1. NAO

NAO è la parte essenziale di questo progetto in quanto gestisce l'interazione con il cliente che dovrà configurare la sua installazione di pannelli solari. L'idea iniziale consisteva nell'aggiunta delle librerie necessarie per controllare il sito Solaredge direttamente come un *pacchetto* Python del NAO. Tuttavia, vennero riscontrati gli stessi problemi del progetto precedente: le librerie non potevano essere installate. La soluzione, fu quella di utilizzare un computer come *server*, accedendo al sito direttamente da esso e rendendo dunque, la pagina web accessibile anche al cliente. Nell'interpretare i comandi del cliente il team ha incontrato un secondo problema: il riconoscimento vocale di NAO lavora solo con parole già definite in una lista ma, gli input dall'utente, possono essere potenzialmente infiniti. Il team ha quindi deciso di registrare la risposta con NAO ed inviare il file WAV al server affinchè possa essere analizzato. Quando il progetto è completo, il robot espone a voce i risultati del progetto generati dal sito web.

3.2. HTTP Server

È il punto centrale in cui tutti i dati del progetto sono immagazzinati. È scritto in Python e comunica con NAO ricevendo le registrazioni vocali. Esso analizza le registrazioni estraendo la trascrizione che viene poi inserita nel sito Solaredge. Include vari endpoints statici e dinamici per gestire i dati. È stato scritto in Python utilizzando Flask. Similarmente all'altro progetto, il server non è eseguito direttamente su NAO ma su un PC esterno, il che rende il codice flessibile aiutando l'utente ad inserire dati e ad accedere al sito.

3.2.1. Rest API

La documentazione per l'API può essere trovata qui.

Questo servizio Rest gestisce il traffico di dati dal NAO al server e nel senso opposto. Include vari endpoint e usa il protocollo HTTP. Inoltre i file WAV registrati da NAO sono trascritti ed inseriti nel sito web Solaredge.

3.2.2. Riconoscimento vocale

La trascrizione della voce viene fatta usando la libreria Python SpeechRecognition, che utilizza il modello di *Google* per il riconoscimento vocale e ritorna un testo a partire da un file audio. Il file passato a questa libreria è composto da due canali e include le registrazioni del microfono *destro* e *sinistro* del NAO.

3.2.3. Solaredge website Control

Selenium è una libreria Python che contiene numerose istruzioni, le quali permettono un'interazione diretta con un browser locale a scelta. Questo strumento può analizzare il codice HTML del sito e così facendo è in grado di selezionare ed interagire con specifici elementi grafici presenti nel codice stesso (come caselle di testo, liste, links etc.). Esso *automatizza* sia l' *estrazione* che l' *importazione* dei dati, ottimizzando il processo e rendendolo più efficiente. Questa libreria è fondamentale per inserire le trascrizioni dei file audio estratti dalla conversazione tra NAO e il cliente.

4. TEAM E AUTORI

Il team *NaoNexus* è diviso in due sezioni: *coding* e *social*. I membri del team che hanno lavorato alla parte social del progetto si sono impegnati per rappresentare l'esperienza NaoNexus al meglio, creando e postando nuovi contenuti ogni giorno per evidenziare i progressi fatti dal gruppo durante questi mesi. Il sub-team social ha avuto un ruolo chiave nel rappresentare i progetti con video, immagini, reel e TikTok. Essi sono stati postati sui nostri social e sul sito web NaoNexus.

Il gruppo coding si è occupato della parte di robotica, sviluppando il codice e lavorando con le tecnologie e metodologie pecedentemente presentate.

4.1. GRUPPO CODING

Riccardo Antonelli - Team Leader
Francesco Bernardi
Elisa D'Iseppi
Filippo Buso
Edoardo Polfranceschi

4.2. GRUPPO SOCIAL

Alberto Rubini - Team Leader
Davide Masini
Antonio Galati
Arianna Antonelli