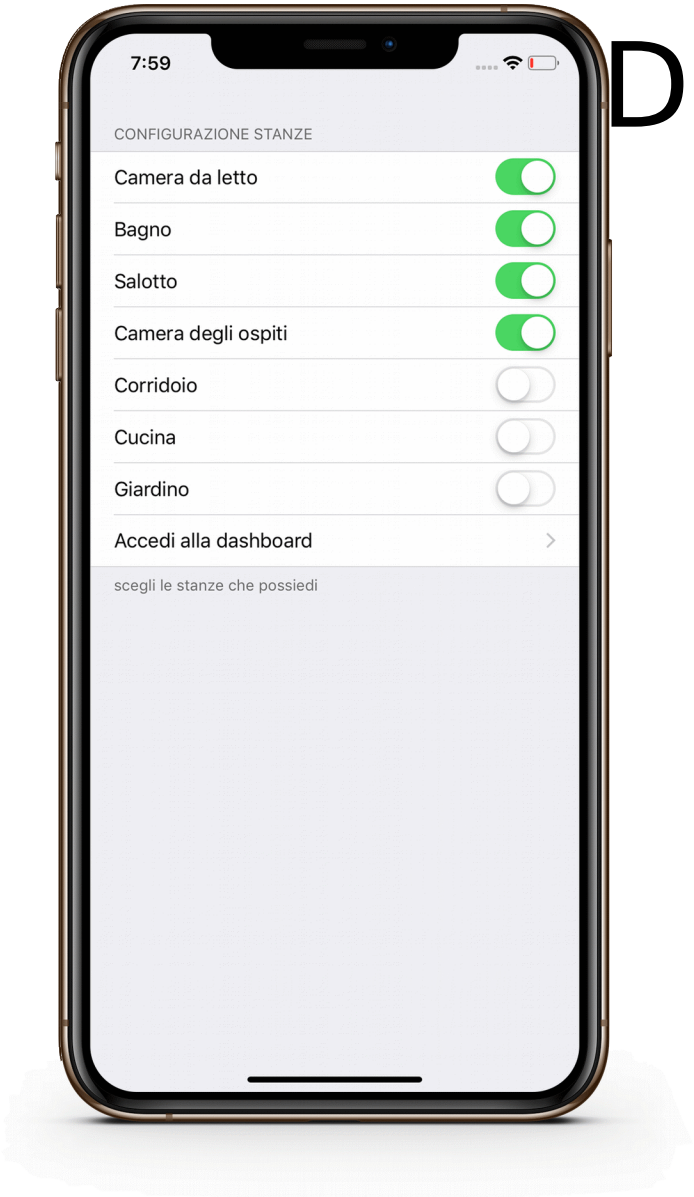
Aut0mated_h0me

Kevin Ngadiuba 5A INF

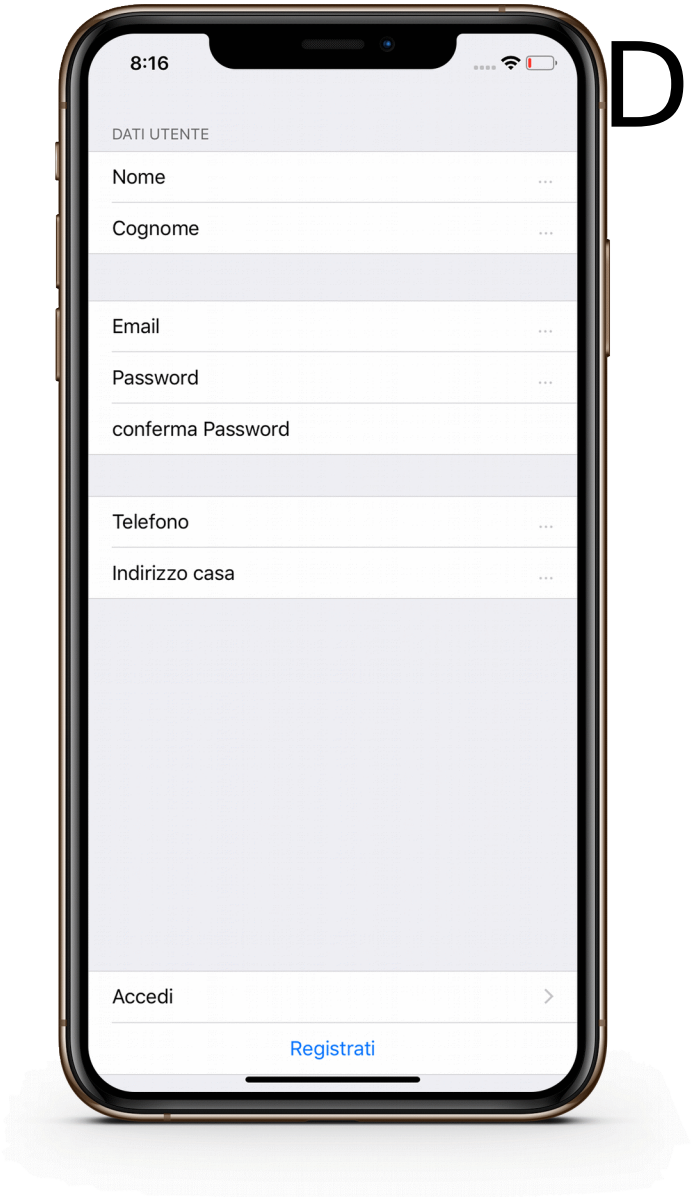
Di cosa si tratta?

- Aut0mated_h0me è un’app per la gestione di una casa domotica sincronizzabile con qualsiasi abitazione avente sensori per rilevamenti di temperatura o di stato (Acceso/Spento).
- L’implementazione è stato fatto in iOS ma grazie ai web API creati , è interamente integrabile con qualsiasi sistema operativo (es. Android/ windows phone).



Di cosa si tratta?

- È completamente personalizzabile e adattabile alle proprie esigenze , si possono aggiungere o togliere le stanze necessarie.
-



Di cosa si tratta?

- Per utilizzare questa app e per preservare la sicurezza dei propri dispositivi domotici , è necessario un account.
- Ogni account è univoco a se per la gestione e personalizzazione della propria casa , quindi è necessaria.
- In caso ci si dimenticasse dei dati del proprio account , è possibile ripristinare la password.
-

Di cosa si tratta?

- Una volta impostato le stanze richieste , si possono ricevere le informazioni dei propri dispositivi.
-

La realizzazione del progetto

Indice :

- Tecnologie sfruttate
- Comunicazione

Tecnologie utilizzate

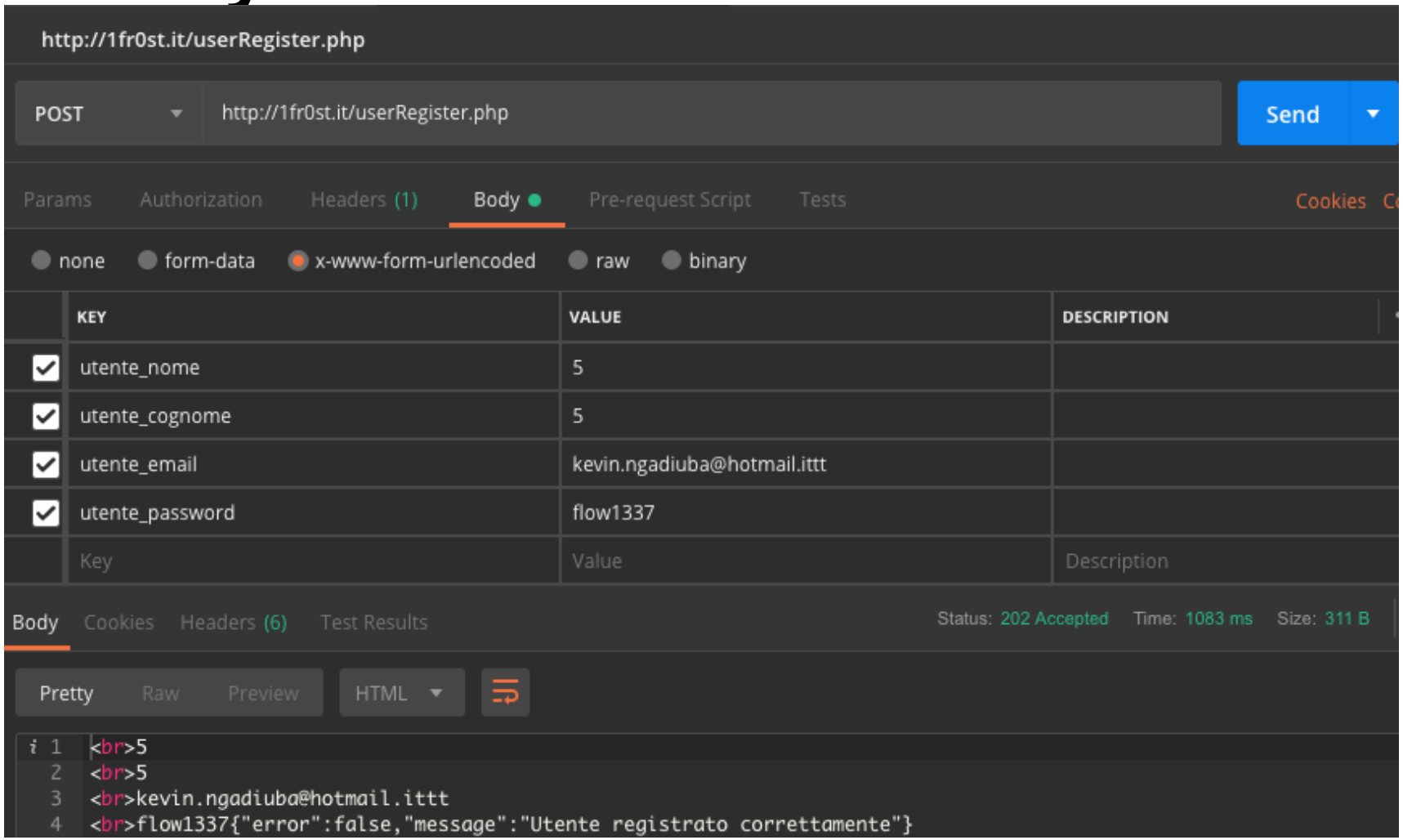
- L’app è stata ideata totalmente su database e interazione tra web API e client , per soddisfare questa richiesta è stato richiesto l’utilizzo di :
-

JSON

- JSON è un semplice formato per lo scambio di dati , supportato dalla maggioranza di linguaggi di programmazione , oltretutto molto facile da leggere per l’umano e altrettanto facile per la macchina analizzare la sintassi , ciò rende il JSON uno standard per la trasmissione di dati.
- Un esempio di dato JSON è {«utente_nome» : «kevin»} , dove abbiamo la chiave a sinistra e a destra il valore.
- Nel mio progetto è stato molto utile per mandare messaggi di riposta dal server al client in caso di successo o fallimento delle richieste da parte del client.

JSON

- Per testare le interrogazioni al server e le corrette risposte , ho utilizzato POSTMAN , analizzo il JSON di risposta con il messaggio annesso.



MySQLi

- Per garantine maggiore sicurezza al sistema , ho preferito utilizzare MySQLi rispetto alle alternative quali PDO o MySQL , etc
- Nonostante il PDO abbia maggiore flessibilità d’uso e supporto di protocolli , il MySQLi ha funzioni aggiornate con applicazioni orientati ad oggetti , una evoluzione alle funzioni deprecated del MySQL. Oltretutto performa più velocemente ed in maniera sicuro , anche se il GAP di differenza di velocità tra PDO e MySQLi è minima , quindi si è puntato di più sulla sicurezza.
- iOS per motivi di protezione , permette più flessibilità a MySQLi.

PHP

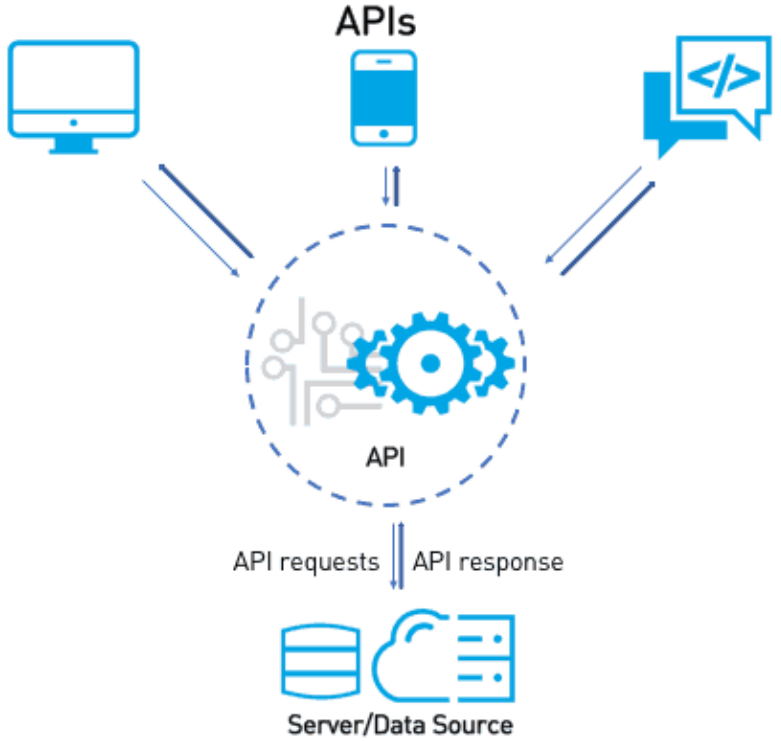
- Linguaggio di scripting interpretato per pagine web dinamiche , in questo progetto utilizzato per creare pagine web non visualizzabili , ma solo al fine di web service per comunicazione client-server

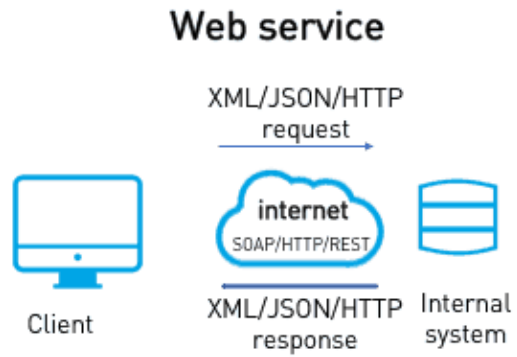
Swift

- Linguaggio di programmazione object-oriented per iOS
- L’unico metodo per programmare in iOS è Swift , essendo linguaggio proprietario Apple

Comunicazione

- La comunicazione client server è di tipo bilaterale , dove possiamo avere comunicazioni sincrone o asincrone verso il server passando attraverso un webAPI che funge da middleman che





Cosa è un webAPI?

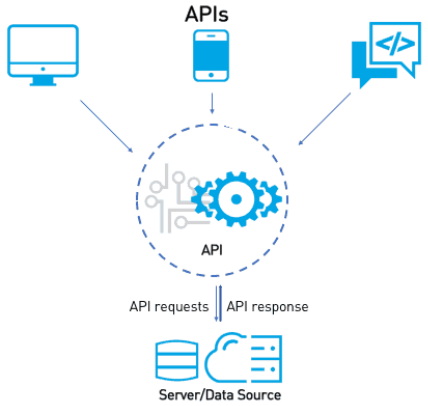
- I Web service consentono di far interagire due applicazioni indipendentemente dal sistema operativo su cui girano e dal linguaggio di programmazione utilizzato.
- L’approccio è sostanzialmente basato su una definizione di funzione richiamabile via Web.
- Come menzionato prima ho creato pagine php , ma non da visualizzare ma solo al fine di essere richiamati perchè contengono tutte le funzioni necessarie a me.

Perchè usare webAPI

- Grazie al web service possiamo gestire in maniera ottimale le funzioni lavorando sopra al protocollo HTTP , e manipolare i dati che riceviamo a nostro piacimento.
- Per garantire la totale fruibilità del sistema e renderlo indipendente da ogni sistema operativo che ci si interfaccia
- Scelta di risposta di ritorno in XML / JSON , come per noi è più comodo a scelta.

Come funziona un webAPI

- Un webAPI riceve la richiesta da parte del client(esempio. Mobile app) con un contenuto pari a chiave e valore in JSON/XML/etc , quindi già formattato in un modo comprensibile. Ovviamente tutto questo passando sopra un protocollo HTTP/HTTPS
- Per eseguire una richiesta ad un server via middleman web service , usiamo dei endpoint , ossia dei link dove sono esposti le funzioni di ricezione input di dati , es: https://www.fr0st.it/registraDati.php



Come funziona un webAPI

- Successivamente alla richiesta da parte del client , il server esegue la richiesta ricevuta , prendendo in carico il contenuto di dati formato da chiave e valore , quindi decodificando dal formato proveniente.
- La richiesta va a fare una interrogazione al database da cui chiediamo informazioni , ovviamente ci sarà una risposta che sia positiva o negativa
- Per la gestione della risposta ci sono due metodi convenzionali trovati per dare una risposta e gestibile :

1) Mandare JSON , modo default
2) Utilizzare HTTP response code :

Come funziona un webAPI (HTTP Response)

- Una risposta HTTP viene effettuata dal server al client richiedente. Lo scopo della risposta è di fornire al client la risorsa richiesta o informare il client che l’azione richiesta è stata eseguita o in caso contrario di un errore nell’elaborazione della richiesta.
- L’http response contiene 3 elementi tra quali : la versione del protocollo HTTP utilizzato , lo status code che generalmente è di 3 cifre indicando il risultato della richiesta , reason phrase ossia lo status text scritto in modo umano tipo OK , NO
- Esempio : HTTP/1.1 200 OK

Come funziona un webAPI (HTTP Response)

- Quindi grazie ai HTTP/HTTPS response , si può gestire il flusso di dati al client , per esempio in caso il server ci desse conferma di avvenuto successo di una richiesta , possiamo proseguire nell’app .
- Esistono già HTTP response code prefissati usati molto dai browser , però nel nostro webAPI possiamo definire un numero a nostro piacimento per identificare gli errori o messaggi.
- Il HTTP Response in confronto al JSON/XML non è un metodo di trasmissione dati , ma solo un metodo di notifica , quindi è consigliato utilizzare entrambi per garantire un miglior flow del sistema. Quindi se dovessimo solo notificare al client l’avvenuto login , basterebbe solo un response con OK , invece per altri casi dove serve uno scambio di dati si codifica in JSON/XML.
-

Come funziona un webAPI

- Infine della trasmissione dati che ritorna la risposta al client , potrà visualizzare ciò che ha richiesto. Finendo il flusso Client-Server