CINI Smart City
University Challenge 2018

Referente progetto: **Luigi De Russis, Davide Cerotti**
{luigi.derussis, davide.cerotti}@uniupo.it
Università: **Università del Piemonte Orientale**
Sede di Vercelli
Referente Nodo: **Stefania Montani** (stefania.montani@uniupo.it)

"LAYW"
Look After Your Weight

Membri del Team

Nome e Cognome	*Numero Matricola*	*MSc o BSc*	*Indirizzo E-mail*
Alessandro Salogni	20013907	BSc	20013907@studenti.uniupo.it
Manuel Peli	20016557	BSc	20016557@studenti.uniupo.it
Riccardo Perotti	20013787	BSc	20013787@studenti.uniupo.it
Simone Coppeta	20016455	BSc	20016455@studenti.uniupo.it

Indice

1. Descrizione generale

Il sistema, relativo ad un ambiente domestico, si pone l'obiettivo di assistere le persone affette da obesità durante il loro percorso di guarigione, aiutandole a perdere peso in modo salutare.

Grazie a **L**ook **A**fter **Y**our **W**eight (**LAYW**) i medici e i pazienti saranno in continuo contatto; i medici (es.: dietologi e cardiologi) potranno seguire i progressi dei propri assistiti in tempo reale fornendo loro obiettivi giornalieri (come ad esempio numero di passi o calorie da bruciare nell'arco della giornata), una scheda di allenamento e una dieta, mettendo loro a disposizione delle alternative quali ad esempio: "a pranzo puoi mangiare 80 grammi di pasta oppure 100 grammi di riso bianco", "fai 10 minuti di bicicletta oppure 20 minuti di corsa a ritmo costante".

Il paziente sarà monitorato durante l'arco della giornata e soprattutto durante la sua attività fisica. I dati raccolti saranno inviati periodicamente al medico affinché rimanga aggiornato e possa vigilare sul comportamento del paziente, ad esempio verificando che segua la scheda di allenamento e la dieta prescritta. Inoltre, in ambito domestico, i dati saranno utilizzati dal sistema per fornire delle indicazioni all'assistito riguardo l'intensità da mantenere durante l'attività fisica, affinché sia utile per perdere peso. Queste indicazioni possono essere di due tipi:

- *Audio*: il sistema farà ascoltare delle canzoni adeguate al ritmo da mantenere durante l'attività fisica, inversamente proporzionale all'intensità della sua attività;
- *Visivi*: in base all'intensità dell'attività, delle lampadine cambieranno colore in modo da stimolare il paziente ad aumentare/diminuire il ritmo.

Inoltre, il sistema, tramite vibrazioni di un braccialetto fitness, si occuperà di spronare il paziente a fare esercizi (nel caso durante il giorno non ne abbia ancora fatti), a pesarsi oppure di avvisarlo di eventuali cambiamenti a dieta e scheda di allenamento.

2. Requisiti

2.1. Requisiti Funzionali

1. Il sistema domestico del paziente deve raccogliere dati riguardo le sue attività e inviarli ai medici associati.
2. Il sistema deve stimolare il paziente a svolgere attività salutare tramite i dispositivi IoT presenti in casa.
3. Il sistema deve permettere al medico di fornire obiettivi giornalieri e/o settimanali, impostare una scheda di allenamento e una dieta al paziente.
4. Il sistema deve notificare al paziente eventuali cambiamenti nella dieta o nella scheda di allenamento tramite allarmi, i quali saranno utilizzati anche per ricordare al paziente di pesarsi e di fare attività fisica.
5. Il sistema deve gestire le luci e la musica in base all'intensità dell'attività in modo da permettere al paziente di capire se sta effettuando attività aerobica o meno (per bruciare grassi).
6. Fornire un'appropriata interfaccia di per la comunicazione tra dottori e pazienti.
7. Il sistema deve permettere al medico di seguire i progressi dei propri assistiti durante la giornata.
8. Il sistema deve garantire l'interattività tra paziente e sistema tramite un'interfaccia vocale e/o testuale.

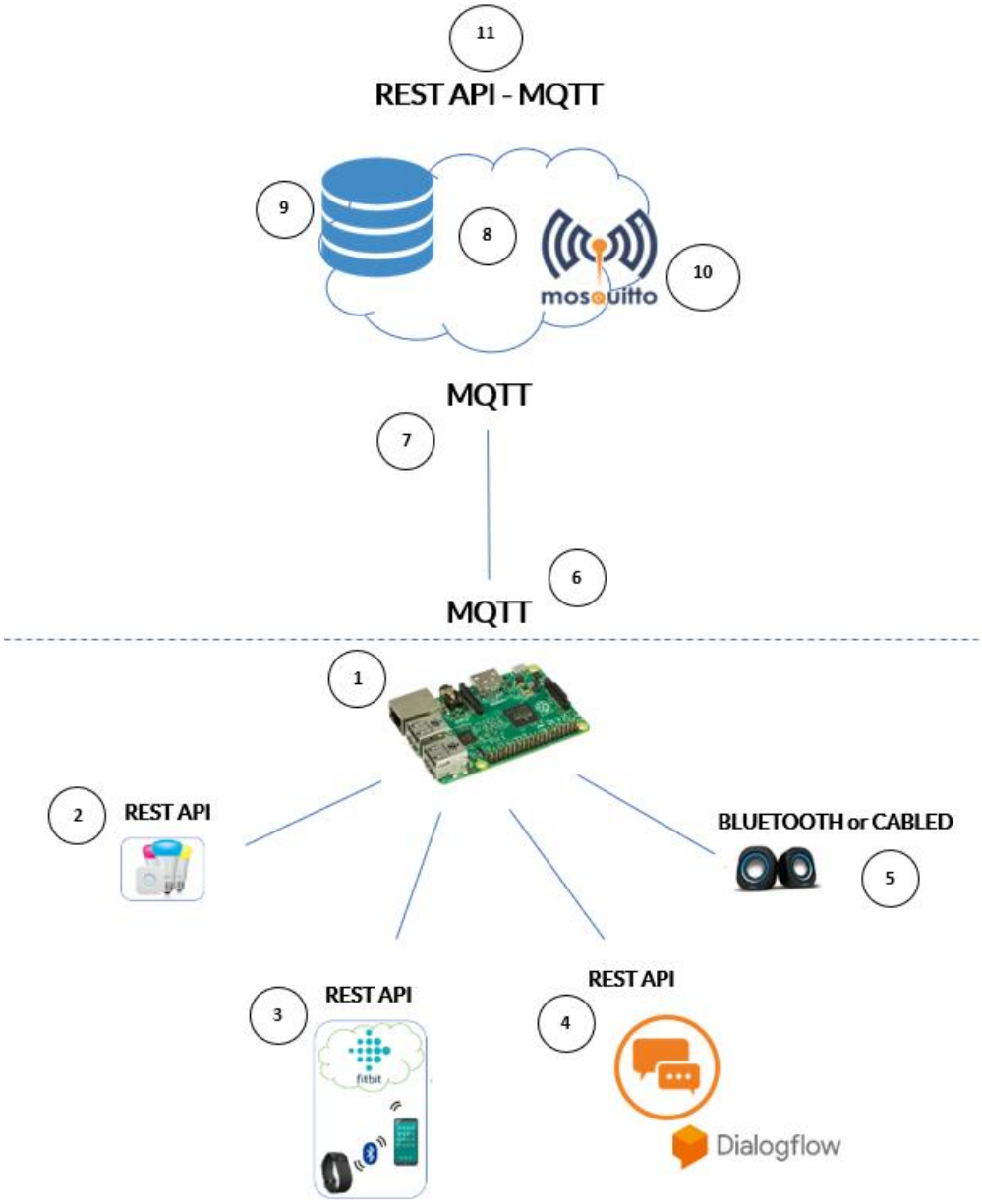
2.2. Requisiti Non-Funzionali

1. Il sistema prevede l'utilizzo di database relazionali.
2. Il sistema dovrebbe interfacciarsi con un braccialetto fitness e comunicare con esso almeno ogni x secondi, dove x dovrebbe essere un tempo significativamente breve, dell'ordine di una decina di secondi massimi.
3. Il sistema necessita di una connessione a Internet sempre attiva.
4. Il sistema fornisce API REST per la comunicazione a livello software tra dottori e pazienti.
5. Il sistema richiede un broker MQTT sempre disponibile localizzato sul server.
6. Le luci interfacciate al sistema permettono il loro controllo remoto e il cambiamento di colore.
7. Una parte del sistema funziona su un Raspberry.
8. Il Raspberry utilizzato dovrà aver installato Java 8 con integrazione di JavaFX per il funzionamento della musica (Java FX non è contenuto in Java 8, per cui è necessario integrarlo).

9. Dovrà essere garantito sul Raspberry un accesso su un indirizzo pubblico con HTTPS.
10. Alcune informazioni del paziente verranno inviate dal sistema ogni 4 ore per permettere al medico di avere i progressi sempre disponibili, mentre altre informazioni verranno inviate al termine della giornata in quanto saranno dati riassuntivi.

3. Architettura

Nel seguente paragrafo si descriveranno le interazioni fra le diverse componenti previste nell'implementazione del sistema.



Una precisa stanza di un paziente, adibita per l'attività fisica, è controllata da un Raspberry Pi, ovvero la **homestation** (1). Esso ha il compito di gestire la

comunicazione da e verso il server cloud al quale è connesso. La homestation inoltre comunica con i dispositivi IoT della stanza in diversi modi: per quanto riguarda le lampadine connesse (2) e il braccialetto fitness (3) tramite API REST, mentre per le casse audio, utili per la musica e i comandi vocali, via cavo (es.: USB, AUX) o Bluetooth (5).

La homestation offre inoltre uno strato di API REST per la richiesta della dieta e della scheda di allenamento e per la gestione della musica del paziente. Inoltre, integra l'interazione con un'assistente conversazionale, costruito con l'utilizzo di DialogFlow (4), utilizzabile attraverso una chat fornita come risorsa statica accessibile tramite l'URL *"<ngrok>/chat.html"*.

Il protocollo MQTT segue un paradigma di pubblicazione e sottoscrizione di messaggi scambiati attraverso un broker, nel nostro caso Mosquitto, che gira sul server Cloud (10).
La homestation esegue un publisher per comunicare al server Cloud i dati raccolti e un subscriber per ricevere le direttive del medico indirizzate al paziente (6). Simmetricamente sul server cloud (8) un subscriber riceve i dati provenienti dalla homestation, mentre un publisher comunica alle rispettive homestation le direttive dei dottori ai pazienti, e pubblica i dati raccolti dalle homestation affinchè i medici possano esaminarli (7).
L'invio dei dati da homestation al server tramite MQTT viene gestito da uno scheduler.
Viceversa, i dati da server a homestation saranno inviati ogni qualvolta il medico effettuerà una modifica.

Oltre al broker MQTT, sul server cloud è presente anche un database relazionale per la memorizzazione dei dati fondamentali che devono essere trasferiti dal medico al paziente e viceversa (9).

Inoltre, è presente uno strato di API REST sul server, che permette ai dispositivi dei medici di richiedere i dati dei loro pazienti e di inviare loro una dieta ed una scheda di allenamento, oltre che fissare gli obiettivi che devono conseguire (11).

I dispositivi utilizzati nell'architettura implementata sono:

- ***Raspberry Pi***: utilizzato come homestation, serve per raccogliere i dati dai diversi dispositivi IoT, effettuare elaborazioni e comandare questi ultimi di conseguenza.

- ***Fitbit Charge 2***: dispositivo utilizzato come braccialetto fitness e dispositivo di interazione con il paziente, che comunica alla homestation i battiti cardiaci in tempo reale del paziente durante la sua attività fisica: questi battiti cardiaci risultano essere i principali protagonisti del funzionamento del sistema, in quanto determinano gli incentivi audio/visivi atti a spronare o rilassare il paziente.
  Inoltre, il Fitbit permette alla homestation di tenere traccia ad esempio di:
  - cronologia dei battiti cardiaci di un'attività fisica
  - lista completa delle attività svolte per ogni giorno
  - calorie bruciate durante le varie attività
  - dei passi totali giornalieri
  - del cambiamento del peso del paziente nel corso del tempo.

  Infine, il medico tramite il cloud può impostare su di esso gli obiettivi giornalieri e settimanali che il paziente deve conseguire, quali il raggiungimento di un determinato peso, il consumo di un determinato numero di calorie e il totalizzare un certo numero di passi.
  Il bracciale vibrerà per notificare all'utente eventuali variazioni alla dieta o alla scheda di allenamento, oppure per ricordargli di pesarsi e fare attività fisica, nel caso non l'abbia ancora fatto.
  Il Fitbit ricopre un ruolo da protagonista nel sistema, infatti grazie ad esso il medico può seguire i progressi del proprio assistito, con un monitoraggio dell'attività fisica completo.
  La homestation comunica con il Fitbit tramite uno strato di API REST definite, che permettono di collegarsi al suo Cloud tramite le sue API Web.

- ***Philips Hue***: dispositivo utilizzato come device di output che riceve comandi dalla homestation conseguenti al battito cardiaco del paziente rilevato. Esse si illuminano di colore diverso a seconda del loro scopo. Incentivano il paziente ad aumentare l'intensità dell'attività fisica tramite una colorazione rossa, rassicurano il paziente che sta svolgendo l'attività fisica in maniera adeguata* tramite una colorazione verde ed infine segnalano al paziente che l'attività fisica che sta svolgendo è troppo intensa, quindi lo rilassano tramite una colorazione blu.
  Per comunicare con le Philips Hue sono state definite API REST necessarie per recuperare i dati dal loro Bridge - il listening device nella tecnologia Z-Wave.

- ***Casse audio***: dispositivo di output che riceve comandi dalla homestation con funzioni simili alle Philips Hue. In caso di attività blanda (colorazione rossa delle Philips Hue) permetteranno l'ascolto di musica molto ritmata

per spronare il paziente ad intensificare l'attività fisica in corso; in caso di attività regolare (colorazione verde) si premia il paziente permettendogli di ascoltare la propria playlist preferita; infine in caso di attività troppo intensa (colorazione blu) le casse permetteranno l'ascolto di musica a ritmo lento per rilassare il paziente.

- ***Chat e DialogFlow***: il paziente può impartire comandi vocali e testuali alla homestation tramite l'utilizzo di una chat utilizzabile su un browser. I comandi permettono di:

    - Iniziare e concludere una nuova attività, gestendo l'accensione/spegnimento dei dispositivi;
    - richiedere aggiornamenti giornalieri sulla dieta e allenamenti impostati dal medico.

**Attività fisica adeguata*: L'adeguatezza dell'attività fisica è misurata secondo il rilevamento dell'aerobia, ovvero situazione nella quale il paziente brucia grassi e perde peso facilmente. L'aerobia è conseguenza di un'attività svolta in un certo range di valori di battiti cardiaci, calcolabile con le formule di Cooper. Esse si basano sul genere e sull'età del paziente (Attività aerobica, my-personaltrainer.it).

3.1. Limitazioni

La parte delle API REST della homestation che riguarda l'interfacciamento con Fitbit al momento è praticamente inutilizzata: quando verranno sviluppate le applicazioni client (mobili e web) saranno utili per effettuare operazioni dirette.

4. Documentazione REST API

URL di base per le REST API: **http://.../api/v1.0/users/user-id**
Il parametro *date*, quando richiesto, avrà il seguente formato: *dd-mm-yyyy*.
Il parametro *period*, quando richiesto, avrà il seguente formato: *1w* oppure *1m*.
Se le richieste che seguono andranno a buon fine, verrà restituito il codice 200.

DATI PAZIENTE

URL: url-base

GET: Ritorna le informazioni di uno specifico paziente. Ad esempio:


```
{
	"user":
	{
		"country":"IT",
		"gender":"MALE",
		"strideLengthRunning":127.4,
		"name":"manuel peli",
		"dateOfBirth":"24-05-1996",
		"averageDailySteps":0,
		"age":22,
		"height":180,
		"strideLengthWalking":74.7
	}
}
```


URL: url-base/**users**

GET: Ritorna la lista di tutti gli utenti. Il JSON di output è uguale a quello precedente, ma al posto di un singolo paziente c'è la lista di tutti i pazienti. Ad esempio:


```
{
	"user": [
			{
				"country":"IT",
				"gender":"MALE",
				"strideLengthRunning":127.4,
				"name":"manuel peli",
				"dateOfBirth":"24-05-1996",
```

```
                    "averageDailySteps":0,
                    "age":22,
                    "height":180,
                    "strideLengthWalking":74.7
                },
                … seguono i restanti pazienti
            ]
}
```


URL: url-base/**users?doctor-id=E-MAIL**

GET: Ritorna la lista dei pazienti associati ad un determinato medico. Il JSON di output ha lo stesso formato della chiamata GET precedentemente descritta.

PESO

URL: url-base/**weights?date=DATE&period=PERIOD**

GET: Ritorna l'andamento del peso di un paziente in un determinato periodo partendo da un determinato giorno, entrambi richiesti come parametri obbligatori. Se in un giorno compreso nel periodo richiesto non c'è un peso registrato, nel JSON di output non sarà presente quel giorno. I parametri sono di nome *date* e *period.* Ad esempio:


```
{"weights":
        [
            {
                "date":"14-08-2018",
                "weight":69,
                "bmi":23.88
            },
            {
                "date":"16-08-2018",
                "weight":68,
                "bmi":23.88
            }
        ]
}
```

URL: url-base/**weights/current**

GET: Restituisce il peso del paziente relativo alla data corrente. Se non è presente alcun peso alla data corrente, viene restituito il più recente. Il JSON di ritorno è strutturato come il precedente.

OBIETTIVI

Calorie da bruciare

URL: url-base/**goals-calories-out?date=DATE**

GET: Richiede un parametro che indica la data di nome *date*, restituisce l'obiettivo relativo alle calorie da bruciare del paziente nella data specificata. Se nella data richiesta non ci sono obiettivi specificati, viene restituito l'obiettivo più recente alla data richiesta. Ad esempio:
{"goals-calories-out":{"date":"16-07-2018","goal":4321}}

POST: Imposta un nuovo obiettivo riguardo le calorie da bruciare che il paziente deve raggiungere. Deve essere aggiunto un JSON al corpo della richiesta, strutturato come il seguente:
{"goals-calories-out":{"goal":4321}}

URL: url-base/**goals-calories-out/current**

GET: Restituisce l'obiettivo relativo alle calorie da bruciare del paziente nella data odierna. Se per la data odierna non ci sono obiettivi specificati, viene restituito il più recente.

Peso da raggiungere

URL: url-base/**goals-weight?date=DATE**

GET: Richiede un parametro che indica la data di nome *date*, restituisce l'obiettivo relativo al peso da raggiungere del paziente nella data specificata. Se nella data richiesta non ci sono obiettivi specificati, viene restituito l'obiettivo più

recente alla data richiesta. Ad esempio:
{"goals-weights":{"date":"29-06-2018","goal":91,"startWeight":132}}

POST: Imposta un nuovo obiettivo riguardo il peso che il paziente deve raggiungere. Deve essere aggiunto un JSON al corpo della richiesta, strutturato come il seguente:
{"goals-weight":{date: 24-07-2018, startWeight: 130,"goal": 90}}

URL: url-base/**goals-weight/current**

GET: Restituisce l'obiettivo relativo al peso da raggiungere del paziente nella data odierna. Se per la data odierna non ci sono obiettivi specificati, viene restituito il più recente.

Passi

URL: url-base/**goals-steps?period=PERIOD**

POST: Imposta al paziente l'obiettivo giornaliero/settimanale relativo ai passi. Deve essere fornito al corpo della richiesta un JSON come il seguente:
{"goals-steps":{"period": "daily", "goal": 10000}}
Il campo *period* può assumere i seguenti valori: daily o weekly.

URL: url-base/**goals-steps-daily?date=DATE**

GET: Richiede un parametro *date*, restituisce l'obiettivo *giornaliero* relativo ai passi del paziente nella data specificata (o nella data più recente se la data specificata non presenta dati). Esempio:
{"goals-steps-daily":{"date":"16-07-2018","goal":10000}}

URL: url-base/**goals-steps-weekly?date=DATE**

GET: Richiede un parametro *date*, restituisce l'obiettivo *settimanale* relativo ai passi del paziente nella data specificata (o nella data più recente se la data specificata non presenta dati). Esempio:
{"goals-steps-weekly":{"date":"16-07-2018","goal":80000}}

URL: url-base/**goals-steps-daily/current**

GET: Restituisce l'obiettivo *giornaliero* relativo ai passi del paziente nella data corrente (o nella data più recente se la data corrente non presenta dati).

URL: url-base/**goals-steps-weekly/current**

GET: Restituisce l'obiettivo *settimanale* relativo ai passi del paziente nella data corrente (o nella data più recente se la data corrente non presenta dati).

Lista di attività di un determinato giorno

URL: url-base/**activities?date=DATE**

GET: Ritorna una lista di tutte le attività svolte dal paziente in un determinato giorno. Il parametro obbligatorio è *date,* che indica la data per la quale si vuole la lista di attività. Ogni attività porta con sé un insieme di dati, come la cronologia dei battiti del paziente (campionati ogni circa 30 secondi) durante lo svolgimento, calorie bruciate, durata e così via. Ad esempio:


```
{
  "activities":[
    {
        "duration":1893000,
        "date":"14-06-2018",
        "activityLevelCategory":{
                "lightlyActiveMinutes":6,
                "fairlyActiveMinutes":0,
                "sedentaryActiveMinutes":25,
                "veryActiveMinutes":0
        },
        "distance":0,
        "heartBeats":[
        {
                 "heartRateTime":"10:44:01 AM",
                "value":56
        },
        {
                "heartRateTime":"10:44:33 AM",
                "value":62
```

```
        },
        {
                "heartRateTime":"10:45:03 AM",
                "value":62
        },
        {
                "heartRateTime":"10:45:33 AM",
                "value":70
        },
        … seguono i restanti battiti
        ],
        "activityName":"Workout",
        "elevationGain":3.353,
        "heartRateZonesList":[
        {
                "min":30,
                "max":99,
                "minutes":31,
                "name":"Out of Range"
        },
        {
                "min":99,
                "max":138,
                "minutes":0,
                "name":"Fat Burn"
        },
        {
                "min":138,
                "max":168,
                "minutes":0,
                "name":"Cardio"
        },
        {
                "min":168,
                "max":220,
                "minutes":0,
                "name":"Peak"
        }
        ],
        "calories":50,
        "averageHeartRate":60,
        "steps":74,
        "speed":0
},
… seguono altre attività svolte ]}
```

URL: url-base/**activities/current**

GET: Ritorna la lista di attività svolte nella giornata corrente. Il JSON di ritorno è strutturato come il precedente.

Sommario delle attività di un determinato giorno

URL: url-base/**activity-summary?date=DATE&period=PERIOD**

GET: Ritorna il sommario di tutte le attività, in un determinato periodo partendo da un determinato giorno in input alla richiesta, come parametri. I parametri obbligatori sono *date* e *period*, ovvero il periodo per cui si vuole il sommario delle attività e la data dalla quale si vuole il riassunto di tutte le attività svolte. Ad esempio:


```
{
  "activity-summary":[{
        "elevation":3.05,
        "restingHeartRate":55,
        "date":"16-07-2018",
        "distanceCategory":{
              "lightlyActiveDistance":0.12,
              "sedentaryActiveDistance":0,
              "veryActiveDistance":0,
              "fairlyActiveDistance":0
        },
        "floors":1,
        "heartRateZonesList":[
        {
              "min":30,
              "max":99,
              "minutes":96,
              "caloriesOut":150.1038,
              "name":"Out of Range"
        },
        {
              "min":99,
              "max":138,
              "minutes":0,
              "caloriesOut":0,
              "name":"Fat Burn"
        },
```

```
            {
                        "min":138,
                        "max":168,
                        "minutes":0,
                        "caloriesOut":0,
                        "name":"Cardio"
            },
            {
                        "min":168,
                        "max":220,
                        "minutes":0,
                        "caloriesOut":0,
                        "name":"Peak"
            }
            ],
            "minutesCategory":{
                        "fairlyActiveMinutes":0,
                        "lightlyActiveMinutes":10,
                        "sedentaryActiveMinutes":1022,
                        "veryActiveMinutes":0
            },
            "steps":807,
            "caloriesCategory":{
                        "outCalories":1265,
                        "activityCalories":33,
                        "marginalCalories":13,
                        "bmrCalories":1229
            }
}]}
```


URL: *url-base*/**activity-summary/current**

GET: Ritorna il sommario dell'attività del giorno corrente. Il JSON di ritorno è strutturato come il precedente.

Sommario del battito cardiaco di un determinato giorno

URL: url-base/**heart-rate-day-summary?date=DATE&period=PERIOD**

GET: Ritorna il sommario di un particolare periodo partendo da un determinato giorno in input alla richiesta come parametri. I parametri obbligatori sono *date* e *period*, ovvero il periodo e la data per i quali si vuole il sommario. Ad esempio:

```
{"heart-rate-day-summary":
        [{
                "min":30,
                "max":99,
                "minutes":96,
                "caloriesOut":150.1038,
                "name":"Out of Range"
        },
        {
                "min":99,
                "max":138,
                "minutes":0,
                "caloriesOut":0,
                "name":"Fat Burn"
        },
        {
                "min":138,
                "max":168,
                "minutes":0,
                "caloriesOut":0,
                "name":"Cardio"
        },
        {
                "min":168,
                "max":220,
                "minutes":0,
                "caloriesOut":0,
                "name":"Peak"
        }]
}
```


URL: *url-base*/**heart-rate-day-summary/current**

GET: Ritorna il sommario del giorno corrente. Il JSON di ritorno è strutturato come il precedente.

Dieta

URL: url-base/**diets**

GET: Ritorna la lista dei giorni della settimana, contenente in ogni giorno una

lista di pasti predefiniti, ciascuno dei quali contiene una lista di alimenti che il paziente può scegliere di mangiare.


```
{
diet-days: [
    {
      "dayName":"lunedi",
      "meals":[
        {
          "mealName":"cena",
          "diets":[{ "description":"carne bianca ai ferri" }]
        },
        {
          "mealName":"colazione",
          "diets":[{"description":"latte con fetta biscottata" }]
        },
        {
          "mealName":"pranzo",
          "diets":[ {"description":"riso"}, {"description":"pasta"}]
        },
        {
          "mealName":"spuntino mattino",
          "diets":[{"description":"mela" }, {"description":"pesca" }]
        },
        {
          "mealName":"spuntino pomeriggio",
          "diets":[{"description":"Macedonia" }]
        },
        {
          "mealName":"spuntino sera",
          "diets":[{"description":"grissini integrali" }]
        }
      ]
    },
    ... Altri giorni della settimana
  ] }
```


POST: Inserisce o sovrascrive una settimana di dieta di un certo paziente.

Questa POST accetta come argomento un JSON, che conterrà una lista di giorni (anche non completa) con all'interno di ciascuno una lista di pasti predefiniti in ciascuno dei quali sono contenuti una lista di alimenti che il paziente può mangiare. Il JSON di input dovrà essere come quello riportato sopra nella GET, tenendo presente che non sono necessari tutti i giorni della settimana e tutti i pasti.

Allenamento

URL: url-base/**trainings**

GET: Ritorna la lista dei giorni della settimana, contenente in ogni giorno una lista di allenamenti tra i quali il paziente deve scegliere di svolgere. Esempio:


```
{
	"training-days":
	[
		{
			"dayName":"domenica",
			"trainings":[]
		},
		{
			"dayName":"giovedi",
			"trainings":
			[{"description":"1 ora Camminata"},{"description":"20 minuti di  ellittica"}]
		},
		{
			"dayName":"lunedi",
			"trainings":
			[{"description":"20 minuti di corsa"},{"description":"30 minuti di bici"}]
		},
		{
			"dayName":"martedi",
			"trainings":
			[{"description":"1 ora camminata"}]
		},
		{
			"dayName":"mercoledi",
			"trainings":[]
		},
		{
			"dayName":"sabato",
			"trainings":[]
		},
		{
			"dayName":"venerdi",
			"trainings":
			[{"description":"50 addominali"},{"description":"20 flessioni"}]
		}
	]
}
```


POST: Inserisce o sovrascrive una settimana di allenamenti di un certo paziente. Questa POST accetta come argomento un JSON, che conterrà una lista di giorni

(anche non completa) con all'interno di ciascuno una lista di allenamenti da cui scegliere. Il JSON di input dovrà essere come quello riportato sopra nella GET, tenendo presente che non sono necessari tutti i giorni della settimana.

Medici

URL: url-base/**doctors**

GET: Ritorna la lista dei medici, contenente per ogni dottore il nome e l'e-mail. Esempio:


```
{
        "doctors":
        [
                {
                        "name":"Simone Coppeta",
                        "email":"s.coppeta11@gmail.com"
                },
                {
                        "name":"Alessandro Salogni",
                        "email":"salogni.alessandro@libero.it"
                },
                {
                        "name":"Riccardo Perotti",
                        "email":"20013787@studenti.uniupo.it"
                }
        ]
}
```


POST: Inserisce un nuovo medico nella lista.
Questa POST accetta come argomento un JSON, che conterrà il nome e l'e-mail del nuovo medico, simile al JSON che si riceve dalla chiamata GET precedentemente descritta. Esempio:

{"doctor":{"name":"Giorgia Bianchi","email":"giorgiabianchi@gmail.com"}}

URL: url-base/**doctors/:id**
POST: Inserisce una nuova associazione medico-paziente.
Questa POST accetta come argomento un JSON, che conterrà gli id dei pazienti

(anche id dei pazienti già presenti) da associare ad un determinato medico. Esempio:
{"users":{[1,2,3,4,-5,5,98]}}

4.1. Messaggi di errore

GET: nel caso non fossero presenti i dati richiesti, viene restituito il seguente messaggio di errore: {"success": false, "message": "No data"}, con associato il codice di errore 404. Lo stesso codice di errore viene restituito anche in assenza dei parametri necessari alla richiesta.

Se tali parametri assumono valori inconsistenti con il loro tipo o la loro struttura, viene restituito il codice di errore 400.

POST: nel caso ci fossero errori nel JSON d'ingresso alla richiesta, possono essere restituiti i seguenti messaggi di errore:

1) Nel caso la richiesta non contenga un JSON verrà restituito {"success": false, "message": "JSON not found"}
2) Nel caso la richiesta contenga un JSON strutturato erroneamente verrà restituito {"success": false, "message": "Malformed JSON"}
3) Nel caso la richiesta contenga un JSON con campi che hanno valori non previsti verrà restituito {"success": false, "message": "Incorrect values"}
4) Nel caso la richiesta non abbia specificati i parametri richiesti verrà restituito {"success": false, "message": "Required parameter not set"}

Nel caso vengano restituiti i precedenti errori, verrà associato il codice di errore 403, mentre se si cerca di eseguire una POST su un utente inesistente, verrà restituito il codice di errore 404.