

Documento di sviluppo dell'applicazione

G12

Contents

1	Scopo del documento	2
2	User stories	2
3	Features	3
4	User Flow	4
5	Application Implementation and Documentation	4
5.1	Project Structure	4
5.2	Project Dependencies	5
5.3	Project Data or DB	5
5.4	Project APIs	6
6	API documentation	7
7	FrontEnd Implementation	15
8	GitHub Repository Info	17
8.1	Stringa di connessione a MongoDB	17
9	API Testing	17

1 Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di un aspetto dell'applicazione SmartFit. In particolare, presenta tutti gli artefatti necessari per realizzare i servizi di gestione delle funzionalità di un utente non allenatore (da ora semplicemente "utente") dell'applicazione SmartFit. Purtroppo non abbiamo potuto svolgere questo lavoro per l'utente di tipo allenatore poiché le funzionalità in più per quel tipo di utente riguardavano l'interazione tra più utenti e la creazione, eliminazione e condivisione di file plain text e non. Si è deciso di svolgere quindi il più possibile per quanto riguarda le funzionalità associate a un utente trattandole e sviluppandole in modo molto dettagliato.

Sono state quindi descritte tutte le user stories per l'utente dell'applicazione, comprendendo tutte le funzionalità che erano descritte nel documento di specifiche del progetto nel modo più completo possibile. Partendo da questo, si sono individuate le varie features da implementare. Da queste ultime sono state sviluppate le varie API necessarie, di cui è stata svolta la completa ed esaustiva documentazione. Delle principali API è stata svolta anche una fase di testing con input corretti ed errati per dimostrare il loro funzionamento. Infine, sono stati realizzati front end e back end per poter visualizzare le varie API in opera in una riproduzione del front end fornito nel primo documento dal gruppo G11.

2 User stories

Una "User Story" è una spiegazione informale e generale di una funzionalità o caratteristica del software che viene descritta dall'utente finale. In pratica serve per dire quali funzionalità servono all'utente, e il motivo per cui gli servono. In questo documento le varie User Stories vanno a descrivere le necessità dell'utente nell'applicazione SmartFit. Le User Stories non sono scritte in modo formale o con un linguaggio tecnico. Sono bensì in linguaggio naturale, servono solo per rispondere alle domande: "A chi serve qualcosa?", "Cosa gli serve?", e, necessariamente, "Perché gli serve?". Questo processo verrà eseguito per ogni singola funzionalità che si avrà intenzione di implementare, ovvero tutte quelle associate ad un utente di SmartFit. Si riporta ora la tabella con tutte le User Stories legate all'utente dell'applicazione SmartFit.

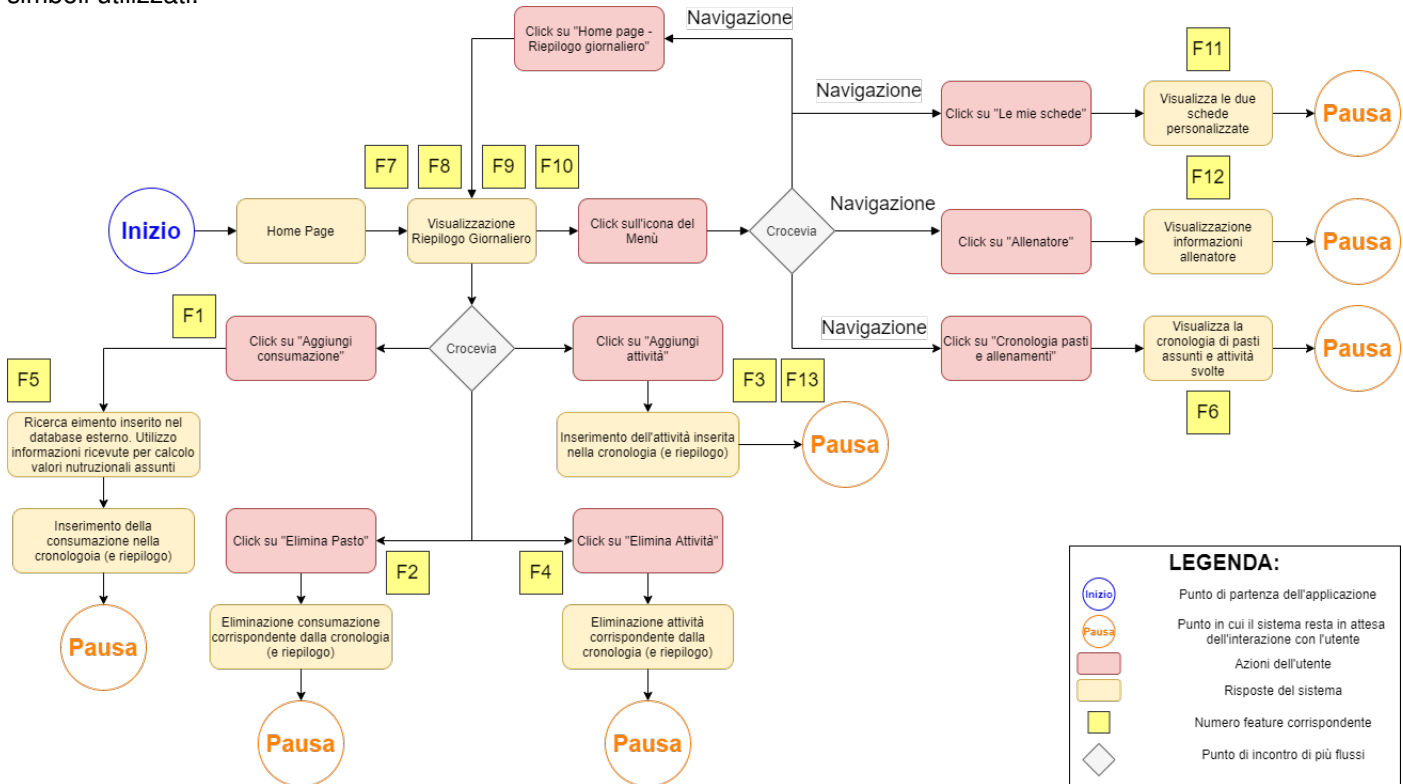
#	Descrizione user story	Requisiti funzionali	Requisiti non funzionali
US1	Come utente, voglio inserire i pasti che assumo, così posso tenere traccia. Voglio anche avere la possibilità di eliminare l'ultimo pasto inserito nel caso in cui compissi un errore di inserimento.	RF7	
US2	Come utente, voglio inserire i miei allenamenti così posso tenere traccia. Voglio anche avere la possibilità di eliminare l'ultima attività inserita nel caso in cui compissi un errore di inserimento.	RF7	
US3	Come utente, voglio visualizzare i valori nutrizionali di ciò che assumo, in modo da poter gestire la mia dieta in modo efficace	RF6	
US4	Come utente, voglio visualizzare la mia cronologia dei pasti assunti e attività svolte, così posso tenere traccia dei miei progressi nel tempo	RF6	
US5	Come utente, voglio visualizzare il mio riepilogo giornaliero sia di dieta che di allenamento così ho una panoramica immediata sul mio andamento giornaliero	RF5	
US6	Come utente, voglio visualizzare, nel riepilogo giornaliero, il bilancio energetico, e un semplice grafico che mi permetta di mettere a confronto i macro nutrienti che assumo, così posso rendermi conto se sto sbagliando qualcosa nella mia alimentazione	RF5	
US7	Come utente, voglio visualizzare, nel riepilogo giornaliero, i principali micronutrienti che ho assunto durante la giornata così posso rendermi conto se la mia dieta è bilanciata	RF5	
US8	Come utente, voglio poter visualizzare le schede personalizzate che mi ha inviato il mio allenatore, così posso averle a portata di mano e poterle utilizzare	RF9	RNF9
US9	Come utente, voglio poter visualizzare delle informazioni sul mio allenatore, così posso verificare la sua identità	RF10	
US10	Come utente, voglio poter inserire le attività fisiche svolte non solo manualmente ma anche da un dispositivo esterno al sistema	RF8	

3 Features

User Story	Feature Index	Feature
US1	F1	Delle caselle di testo per inserire le informazioni e un pulsante che permette di aggiungere un nuovo pasto
US1	F2	Per permettere l'eliminazione di un pasto inserito sarà presente un pulsante a fianco ad ogni voce del riepilogo giornaliero, descritto successivamente
US2	F3	Delle caselle di testo per inserire le informazioni e un pulsante che permette di aggiungere una nuova attività
US2	F4	Per permettere l'eliminazione di un'attività inserita sarà presente un pulsante a fianco ad ogni voce del riepilogo giornaliero, descritto successivamente
US3	F5	Utilizzo di un database esterno per recuperare le informazioni nutrizionali sugli alimenti consumati
US4	F6	Mostra le liste di pasti assunti e attività svolte. Le cronologie di dieta e allenamento saranno salvate su due file JSON locali
US5	F7	Mostra la lista giornaliera di pasti assunti e attività svolte nella giornata corrente
US6	F8	Nella schermata di riepilogo giornaliero (pagina principale dell'app), verrà mostrato un conteggio per il bilancio energetico
US6	F9	Nella schermata di riepilogo giornaliero verrà mostrato un grafico, con delle progress bar, che metterà a confronto le quantità di macronutrienti assunti
US7	F10	Nella schermata di riepilogo giornaliero, verrà mostrata una lista di micronutrienti con i rispettivi quantitativi assunti durante la giornata
US8	F11	In una schermata, verranno mostrate le due schede personalizzate, di allenamento e dieta, le quali verranno importate tramite file plain text locali
US9	F12	Mostra le generalità dell'allenatore in una schermata adibita, recuperate da un file JSON locale
US10	F13	La API di inserimento di un'attività fisica prende come input un file JSON contenente le informazioni sull'allenamento dell'utente in modo da essere molto generale e poter essere adattata in caso la si voglia collegare all'API esterna per un dispositivo smart

4 User Flow

In questa sezione del documento si riportano gli “User Flows” per il ruolo dell’utente. L’immagine che segue descrive lo user flow relativo alle varie funzioni trattate in questa fase di sviluppo. Sarà anche presente una breve legenda che descrive i simboli utilizzati.



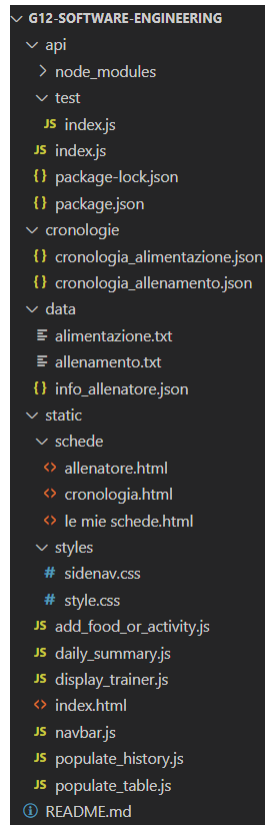
5 Application Implementation and Documentation

Nelle sezioni precedenti del presente documento, abbiamo identificato le varie features individuate partendo dalle varie User Stories. Queste features individuate devono essere ora implementate. L'applicazione è stata sviluppata con HTML e CSS per quanto riguarda il front end e JavaScript per il back end e per le API. Si è inoltre fatto uso di MongoDB per la gestione dei dati dei valori nutrizionali degli alimenti.

5.1 Project Structure

La struttura del codice sorgente del progetto è presentata nella figura sottostante. Il progetto è composto dalle cartelle:

- **api** per la gestione delle api locali e in cui è contenuto anche lo script per il testing, nella sottocartella **test**
- **cronologie** in cui sono presenti le cronologie di alimentazione e di allenamento dell'utente
- **data** in cui sono presenti le schede personalizzate dell'utente e le informazioni sull'allenatore
- **static** in cui è presente tutto il codice che riguarda la User Interface, quindi front end e back end con HTML, CSS e JavaScript.



5.2 Project Dependencies

Per la realizzazione dell'applicazione vari moduli sono stati utilizzati e aggiunti al Package.json. Si riporta lo screenshot della sezione nel package.json:

```
"dependencies": {
  "body-parser": "^1.19.0",
  "cors": "^2.8.5",
  "express": "^4.17.1",
  "mongodb": "^4.2.1",
  "swagger-jsdoc": "^6.1.0",
  "swagger-ui-express": "^4.2.0",
  "xhr2": "^0.2.1"
},
"devDependencies": {
  "supertest": "^6.1.6",
  "tap-spec": "^2.2.2",
  "tape": "^5.3.2"
}
```

5.3 Project Data or DB

Per la gestione dei dati utili all'applicazione abbiamo definito una struttura dati "Food" con MongoDB in cui abbiamo inserito alcuni alimenti/pasti per poter provare le APIs sviluppate. Gli alimenti inseriti sono: Mela, Pera, Banana, Kiwi, Lattuga, Cipolla, Pollo, Manzo, Risotto, Pasta al pomodoro, ecc.

Per ogni elemento presente nel database abbiamo inserito dei valori nutrizionali di esempio, su 100 grammi di consumazione. I macronutrienti sono espressi in grammi, mentre i micronutrienti sono espressi in milligrammi.

I dati salvati per ogni elemento sono: nome, energia, grassi, carboidrati, proteine, fibre, ferro, iodio, magnesio. La decisione su quali macro e micronutrienti dovessimo inserire è stata presa insieme al gruppo G11, dato che nelle specifiche del progetto non era stato dichiarato tale dato. Si riporta un esempio di elemento presente nel database:

```
_id: ObjectId("61bc955eb2e3e6dd5c80cb89")
nome: "Pollo"
energia: 70
grassi: 9
carboidrati: 13
proteine: 80
fibre: 34
ferro: 2.8
iodio: 0.1
magnesio: 0.6
```

La stringa di connessione al database verrà fornita in una sezione successiva.

5.4 Project APIs

Seguono la lista e una breve descrizione delle API che sono state sviluppate per questo progetto. La descrizione completa per ogni API può essere trovata nella sezione che riguarda la documentazione. Per quanto riguarda il codice, si può trovare il link alla repository di GitHub in una sezione successiva.

GET APIs

Segue la lista di API di tipo GET che sono state sviluppate.

- Dati allenatore: recupera le informazioni riguardanti l'allenatore, in formato JSON.
- Cronologia alimentazione: recupera la cronologia della dieta dell'utente, in formato JSON.
- Cronologia allenamento: recupera la cronologia dell'allenamento dell'utente, in formato JSON.
- Riepilogo alimentazione: recupera il riepilogo giornaliero della cronologia della dieta dell'utente (consumazioni nella giornata corrente), in formato JSON.
- Riepilogo allenamento: recupera il riepilogo giornaliero dalla cronologia dell'allenamento dell'utente (attività svolte nella giornata corrente), in formato JSON.

POST APIs

Segue la lista di API di tipo POST che sono state sviluppate.

- Cronologia allenamento: inserisce l'attività fornita tramite formato JSON nella cronologia dell'allenamento dell'utente.
- Cronologia alimentazione: inserisce la consumazione fornita tramite parametro nella cronologia della dieta dell'utente.

DELETE APIs

Segue la lista di API di tipo DELETE che sono state sviluppate.

- Cronologia allenamento: elimina l'attività fornita tramite parametro dalla cronologia della dieta dell'utente.
- Cronologia alimentazione: elimina la consumazione fornita tramite parametro dalla cronologia della dieta dell'utente.

Utility APIs

Segue la lista di API che sono state sviluppate e che vengono utilizzate o come supporto di altre API (la prima) o per recuperare dei dati per poi mostrarli all'utente come richiesto nella lista delle features.

- GET:
 - Valori nutrizionali: recupera le informazioni nutrizionali di un alimento passato come parametro contattando il database remoto.
 - Calorie assunte: restituisce l'assunzione energetica giornaliera da parte dell'utente, in kcal.
 - Calorie bruciate: restituisce l'energia bruciata dall'utente nella giornata corrente, in kcal.
 - Assunzione giornaliera: restituisce la quantità assunta nella giornata corrente del valore nutrizionale passato come parametro. La quantità è espressa in grammi per i macronutrienti e in milligrammi per i micronutrienti.

6 API documentation

Le API locali fornite dall'applicazione SmartFit elencate nella sezione precedente sono state interamente documentate utilizzando il modulo NodeJS chiamato Swagger UI Express. In particolare, le API sono state annotate utilizzando JSDoc: uno strumento la cui funzione è di generare una documentazione partendo da commenti inseriti nel codice sorgente. Swagger UI invece è stato utilizzato per creare un endpoint in cui si può visualizzare una pagina web con l'intera documentazione. L'endpoint si trova all'indirizzo: `localhost:5000/api-docs`
Di seguito viene riportata l'immagine della pagina web relativa alla documentazione.

Documentazione API sviluppate SmartFit gruppo G12 1.0.0 OpenAPI 3.0

Questa è un'applicazione REST API sviluppata con Express.

[Gruppo G12 - Website](#)
[Licensed Under MIT](#)

Servers	
http://localhost:5000/ - Development server	
GET APIs	
GET	/api/datiAllenatore Recupera le informazioni riguardanti l'allenatore.
GET	/api/cronologia_alimentazione Recupera la cronologia della dieta dell'utente.
GET	/api/cronologia_allenamento Recupera la cronologia dell'allenamento dell'utente.
GET	/api/riepilogo_allenamento Recupera il riepilogo giornaliero dell'allenamento dell'utente.
GET	/api/riepilogo_alimentazione Recupera il riepilogo giornaliero della dieta dell'utente.
POST APIs	
POST	/api/cronologia_allenamento Inserisce l'attività fornita nella cronologia dell'allenamento dell'utente.
POST	/api/cronologia_alimentazione/{nome}/{quantità} Inserisce la consumazione fornita nella cronologia della dieta dell'utente.
DELETE APIs	
DELETE	/api/cronologia_alimentazione/{nome}/{data} Elimina la consumazione fornita dalla cronologia della dieta dell'utente.
DELETE	/api/cronologia_allenamento/{nome}/{data} Elimina l'attività fornita dalla cronologia dell'allenamento dell'utente.
Utility APIs	
GET	/api/valori_nutrizionali/{nome}/{quantità} Recupera le informazioni nutrizionali di un alimento fornito.
GET	/api/calorie_assunte Restituisce l'assunzione energetica giornaliera da parte dell'utente in kcal.
GET	/api/calorie_bruciate Restituisce l'energia bruciata dall'utente nella giornata corrente in kcal.
GET	/api/assunzione_giornaliera/{valore_nutrizionale} Recupera il totale assunto nella giornata corrente del valore nutrizionale inserito.

Si descrive ora l'utilizzo delle API più importanti con l'aiuto della documentazione. Per non creare un documento enorme, non ci si dilungherà molto nella spiegazione del funzionamento, essendoci il codice disponibile su GitHub e molti dettagli sono già scritti negli screenshot della documentazione che verranno riportati. Le GET sono tutte abbastanza simili, si porta l'esempio della cronologia allenamento:

GET `/api/cronologia_allenamento` Recupera la cronologia dell'allenamento dell'utente.

Recupera la cronologia dell'allenamento, ovvero la lista di attività svolte dall'utente, dal server.

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Una lista di attività. L'energia bruciata è espressa in kcal, il tempo in minuti.	No links

Media type

Controls Accept header.

Example Value | Schema

```
{
  "data": [
    {
      "nome": "Salti",
      "tempo": 85,
      "energia_bruciata": 654.1,
      "data": "12_12_2021:16.44"
    }
  ]
}
```

Quando viene chiamata, l'API ritorna tutta la cronologia dell'allenamento dell'utente. In pratica va a prendere il JSON contenente tale cronologia e lo invia:

Curl

```
curl -X 'GET' \
  'http://localhost:5000/api/cronologia_allenamento' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:5000/api/cronologia_allenamento
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "nome": "Corsa", "tempo": 30, "energia_bruciata": 102.1, "data": "17_12_2021:17.44" }, { "nome": "Salti", "tempo": 85, "energia_bruciata": 654.1, "data": "12_12_2021:15.44" }, { "nome": "Corda", "tempo": 65, "energia_bruciata": 65.1, "data": "12_12_2021:16.44" }, { "nome": "Trazioni", "tempo": 20, "energia_bruciata": 400.2, "data": "16_12_2021:13.25" }, { "nome": "Trazioni", "tempo": 20, "energia_bruciata": 400.2, "data": "16_12_2021:13.25" }]</pre> <p>Response headers</p> <pre>access-control-allow-origin: * content-length: 648 content-type: application/json; charset=utf-8 date: Sun, 19 Dec 2021 13:49:09 GMT etag: W/"288-CHSKd4RRnmV0uuxfCSd1qCLoiqU" x-powered-by: Express</pre>

Le API GET “dati allenatore” e “cronologia alimentazione” sono praticamente uguali, solo che ritornano file JSON diversi, ovviamente. Per quanto riguarda le API GET “riepilogo alimentazione” e “riepilogo allenamento”, il funzionamento è molto simile solo che svolgono una fase di filtraggio del JSON della cronologia, ovvero costruiscono un JSON a partire da quello della cronologia inserendo solo gli elementi la cui data corrisponde con quella della giornata corrente, per poi inviarlo come risposta.

Le due POST API sono leggermente diverse, anche se la funzione è simile. La API POST “cronologia alimentazione” prende come parametri (path parameters) il nome e la quantità di un determinato alimento consumato dall'utente:

POST
/api/cronologia_alimentazione/{nome}/{quantità}
Inserisce la consumazione fornita nella cronologia della dieta dell'utente.

Inserisce la consumazione fornita nella cronologia della dieta dell'utente, dopo aver chiamato un'altra API che si occupa di contattare un server esterno per calcolare, in base alla quantità consumata, i valori nutritivi assunti. Non viene richiesto di inserire la data di svolgimento perché prende in automatico dal sistema.

Parameters

Try it out

Name	Description
nome * required string (path)	Il nome della pietanza. <input type="text" value="Pera"/>
quantità * required number(\$float) (path)	La quantità consumata [g]. <input type="text" value="180.5"/>

Responses

Code	Description	Links
201	Conferma dell'inserimento della consumazione fornita dall'utente nella cronologia.	No links
400	Bad request: la quantità inserita non era un numero oppure era negativa o nulla.	No links
404	La pietanza ricercata non è presente nel database.	No links

nome * required
string
(path)

Il nome della pietanza.

quantità * required
number(\$float)
(path)

La quantità consumata [g].

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:5000/api/cronologia_alimentazione/Risotto/125' \
-H 'accept: */*' \
-d ''
```

Request URL

```
http://localhost:5000/api/cronologia_alimentazione/Risotto/125
```

Server response

Code	Details
201	<div>Response body</div> <div>Consumazione aggiunta correttamente alla cronologia.</div> <div> Download </div> <div>Response headers</div> <pre> access-control-allow-origin: * connection: keep-alive content-length: 52 content-type: text/html; charset=utf-8 date: Sun, 19 Dec 2021 13:57:27 GMT etag: W/"34-R07CgC8gLA03RwXzRqHc90wQH" keep-alive: timeout=5 x-powered-by: Express </pre>

Per quanto riguarda la API POST “cronologia allenamento” l’unica cosa che cambia è che l’input non viene passato come path parameter bensì come body della richiesta, in formato JSONF:

POST

/api/cronologia_allenamento

Inserisce l'attività fornita nella cronologia dell'allenamento dell'utente.

^

Inserisce l'attività fornita nella cronologia dell'allenamento dell'utente. È stato deciso di richiedere un JSON in questa API POST al posto di parametri in ingresso perché si è assunto che fosse più facile poi da gestire utilizzandola con dispositivi smart come era descritto nelle specifiche del progetto. Non viene richiesto di inserire la data di svolgimento perché prende in automatico dal sistema.

Parameters

Try it out

No parameters

Request body

required

application/json

▼

Example Value

Schema

```
{
  "nome": "Salti",
  "tempo": 85,
  "energia_bruciata": 654.1
}
```

Responses

Code	Description	Links
201	Conferma inserimento dell'attività fornita dall'utente nella cronologia.	No links
400	Il JSON inserito è malformato o il contenuto non soddisfa i requisiti richiesti. I requisiti sono i controlli sull'input, soprattutto per energia e tempo: non devono essere negativi, nulli, NaN, ecc.	No links

Request body

required

application/json

▼

```
{
  "nome": "Salto con l'asta",
  "tempo": 35,
  "energia_bruciata": 128
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/api/cronologia_allenamento' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "nome": "Salto con l'\'asta",
    "tempo": 35,
    "energia_bruciata": 128
  }'
```

Request URL

```
http://localhost:5000/api/cronologia_allenamento
```

Server response

Code	Details
201	<div>Response body</div> <div>Attività aggiunta correttamente alla cronologia.</div>

Si riporta ora una delle due API DELETE, dato che sono molto simili.

DELETE /api/cronologiaAllenamento/{nome}/{data} Elimina l'attività fornita dalla cronologia dell'allenamento dell'utente.

Elimina l'attività fornita dalla cronologia dell'allenamento dell'utente. La data deve essere del formato [dd_mm_yyyy:hh.mm]. Si faccia attenzione che l'orario è in formato 24h e che per ore e minuti non ci va lo zero davanti quando minori di dieci, ad esempio 09.03 va scritto 9.3.

ParametersTry it out

Name	Description
nome * required string (path)	Il nome dell'attività.
data * required string (path)	La data di svolgimento dd_mm_yyyy:hh.mm .

Responses

Code	Description	Links
200	Conferma eliminazione dell'attività fornita dall'utente dalla cronologia.	No links
304	Nessuna eliminazione effettuata, l'attività cercata non è stata trovata nella cronologia.	No links
400	I dati forniti non rispettano i requisiti. Si controlli che la data inserita soddisfi i requisiti richiesti (formato sopracitato).	No links

ParametersCancel

Name	Description
nome * required string (path)	Il nome dell'attività.
data * required string (path)	La data di svolgimento dd_mm_yyyy:hh.mm .

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:5000/api/cronologiaAllenamento/Salto%20con%201x27asta/19_12_2021x3A15.4' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5000/api/cronologiaAllenamento/Salto%20con%201x27asta/19_12_2021x3A15.4
```

Server response

Code	Details
200	<div>Response body<pre>Attività eliminata correttamente dalla cronologia.</pre><div>Download</div></div> <div>Response headers</div>

Le Utility API GET “calorie assunte” e “calorie bruciate” sono abbastanza banali, si riporta solo lo screenshot della documentazione di una delle due:

GET /api/calorie_assunte Restituisce l'assunzione energetica giornaliera da parte dell'utente in kcal.

Restituisce l'assunzione energetica giornaliera da parte dell'utente in kcal.

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	L'assunzione energetica [kcal]. Media type <div>schema</div> Controls Accept header. Example Value <div>780.6</div>	No links

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/api/calorie_assunte' \
  -H 'accept: schema'
```

Request URL

```
http://localhost:5000/api/calorie_assunte
```

Server response

Code	Details
200	<p>Response body</p> <div>575.7</div> <p>Response headers</p> <pre>access-control-allow-origin: * content-length: 5 content-type: application/json; charset=utf-8 date: Sun, 19 Dec 2021 14:17:56 GMT etag: W/"5-muHksftRTif6+7pHXf046q/FdQ" x-powered-by: Express</pre>

La Utility API GET "valori_nutrizionali" è quella che si occupa della connessione con il database MongoDB eseguendo una findOne() con il nome dell'alimento che l'utente passa come parametro di ingresso (path parameter).

GET

/api/valori_nutrizionali/{nome}/{quantità} Recupera le informazioni nutrizionali di un alimento fornito.

⌵

Recupera le informazioni nutrizionali di un alimento fornito contattando un server esterno. Riscalda i valori in base alla quantità consumata dall'utente.

Parameters

Try it out

Name	Description
nome * required string (path)	Il nome della pietanza.
<input type="text" value="Banana"/>	
quantità * required number(\$float) (path)	La quantità consumata.
<input type="text" value="130.12"/>	

Responses

Code	Description	Links
200	Le informazioni nutrizionali della pietanza inserita, in base alla quantità consumata.	No links
<div>Media type <div>application/json</div><div>Controls Accept header.</div><div>Example Value Schema</div><pre>{ "nome": "Banana", "energia": 78.072, "grassi": 0.13, "carboidrati": 26.024, "proteine": 1.04, "fibre": 13.012, "ferro": 1.17, "iodio": 0.013, "magnesio": 1.171, "data": "12_12_2021:21.43", "quantita": 130.12 }</pre></div>		
400	Bad request: la quantità inserita non era un numero oppure era negativa o nulla.	No links
404	La pietanza ricercata non è presente nel database.	No links

Infine, la Utility GET API "assunzione giornaliera":

GET /api/assunzione_giornaliera/{valore_nutrizionale} Recupera il totale assunto nella giornata corrente del valore nutrizionale inserito.

Recupera il totale assunto nella giornata corrente del valore nutrizionale inserito. Il valore è espresso in grammi nel caso di macronutrienti e di milligrammi in caso di micronutrienti.

Parameters Try it out

Name	Description
valore_nutrizionale * required string (path)	Il nome del valore nutrizionale.

grassi

Responses

Code	Description	Links
200	Viene restituita la quantità assunta.	No links
	<p>Media type</p> <p>schema</p> <p>Controls Accept header.</p> <p>Example Value</p> <p>34.65</p>	
404	Il valore nutrizionale richiesto non è presente.	No links

Parameters Cancel

Name	Description
valore_nutrizionale * required string (path)	Il nome del valore nutrizionale.

magnesio

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5000/api/assunzione_giornaliera/magnesio' \
  -H 'accept: schema'
```

Request URL

```
http://localhost:5000/api/assunzione_giornaliera/magnesio
```

Server response

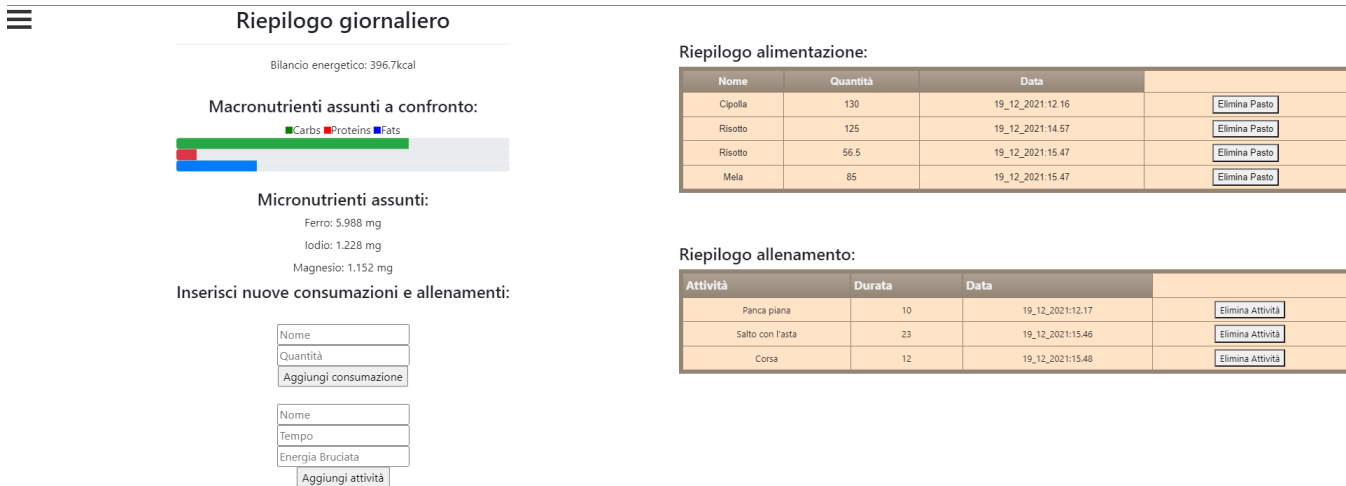
Code	Details
200	<p>Response body</p> <p>2.11</p> <p>Download</p>

Si tiene a precisare che le API che richiedono un input da parte dell'utente, quindi tutte le POST e DELETE, e due API GET di utilità, non generano errori in caso di input errati. Quantità / Tempo / Data possono anche essere nulli, negativi o NaN e l'API semplicemente ritorna un codice di errore come risposta, come illustrato nella documentazione. Lo stesso vale per il database, se ci cerca di inserire una consumazione che non è presente, le due API che ne fanno uso riportano semplicemente l'errore 404. Ugualmente anche per JSON di input malformati o con valori non corretti, o per un tentativo di eliminazione di un'attività o una consumazione non presente nella cronologia.

7 FrontEnd Implementation

Il Front end fornisce le funzionalità di visualizzazione, inserimento e cancellazione dei dati nell'applicazione SmartFit. Permette di utilizzare tutte le API sviluppate, con le varie schede che fornisce. Siamo consapevoli che sia abbastanza grezzo, quasi inguardabile, ma non avevamo mai fatto nulla di simile e avevamo concentrato davvero tutte le nostre forze nello sviluppo delle API e back end nelle settimane antecedenti, trascurando l'aspetto grafico.

La prima scheda dell'applicazione è la scheda di riepilogo giornaliero, che verrà poi anche chiamata "Home - Riepilogo giornaliero" durante la navigazione delle altre schede:



Qui si possono vedere in azione la maggior parte delle API descritte precedentemente, e quindi anche la realizzazione di molte delle features elencate all'inizio del documento. Nella parte in alto a sinistra si possono trovare:

- Il bilancio energetico: differenza tra chilocalorie assunte e bruciate durante la giornata corrente, calcolato con l'ausilio delle due API GET di utilità nel back end, "calorie bruciate" e "calorie assunte".
- Le progress bar per confrontare l'assunzione giornaliera dei principali macronutrienti: il valore delle progress bar viene calcolato con un rapporto tra il risultato del valore ritornato dalla API GET "assunzione giornaliera" e il totale giornaliero assunto tra i vari macronutrienti.
- Le quantità di micronutrienti assunti: anche in questo caso viene utilizzata la API GET "assunzione giornaliera".

Sempre sulla sinistra della scheda sono presenti i form per l'inserimento di nuove consumazioni e allenamenti. Queste utilizzano le API POST descritte in precedenza, ovvero "cronologia alimentazione" e "cronologia allenamento".

A destra invece si possono trovare le tabelle con i riepiloghi giornalieri di alimentazione e allenamento. Le tabelle vengono create dinamicamente in base al JSON ritornato dalle due API GET di riepilogo, "riepilogo alimentazione" e "riepilogo allenamento". Su ogni riga è presente il pulsante per eliminare la consumazione o l'attività presente su quella riga. Questi pulsanti poi chiamano le API DELETE "cronologia alimentazione" e "cronologia allenamento".

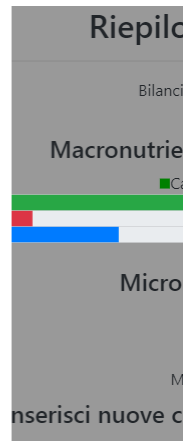
In alto a sinistra si trova il pulsante per aprire il menù che permette di raggiungere le altre schede:



Cronologia pasti e
allenamenti

Le mie schede

Allenatore



Cliccando su “Cronologia pasti e allenamenti” ci si potrà recare alla scheda dove saranno presenti le tabelle con l’intera cronologia di alimentazione e allenamento. Le due tabelle sono create dinamicamente, utilizzando le API GET “cronologia allenamento” e “cronologia alimentazione”.



Alimentazione:

Nome	Quantità	Energia	Grassi	Carboidrati	Proteine	Fibre	Ferro	Iodio	Magnesio	Data
Banana	45.54	60	0.1	20	0.8	10	0.9	0.01	0.9	15_10_2021:12.2
Banana	130.12	78.072	0.13012	26.024	1.04	13.012	1.171	0.013	1.171	12_12_2021:21.43
Cipolla	100	99	9	13	5.3	34	2.8	0.1	0.6	12_12_2021:22.11
Mela	124	70.679	0.124	13.64	0.372	28.52	0.248	0.012	0.012	13_12_2021:16.27
Mela	124	70.68	0.124	13.64	0.372	28.52	0.248	0.012	0.012	13_12_2021:16.30
Mela	150	85.5	0.15	16.5	0.45	34.5	0.3	0.015	0.015	17_12_2021:23.1
Cipolla	120	118.8	10.8	15.6	6.36	40.8	3.36	0.12	0.72	18_12_2021:8.55
Pollo	250	175	22.5	32.5	200	85	7	0.25	1.5	18_12_2021:8.56
Pera	120	84	0.36	18	0.24	36	0.6	0.012	0.012	18_12_2021:22.13
Mela	95	54.15	0.095	10.45	0.285	21.85	0.19	0.01	0.01	18_12_2021:22.14
Lattuga	120	18	0.12	0.036	0.36	12	0.912	0.012	0.672	18_12_2021:22.40
Cipolla	130	128.7	11.7	16.9	6.89	44.2	3.64	0.13	0.78	19_12_2021:12.16
Risotto	125	375	37.5	113.75	6.625	3.75	1.5	0.75	0.25	19_12_2021:14.57
Risotto	56.5	169.5	16.95	51.415	2.994	1.695	0.678	0.339	0.113	19_12_2021:15.47
Mela	85	48.45	0.085	9.35	0.255	19.55	0.17	0.009	0.009	19_12_2021:15.47

Allenamento:

Nome	Tempo	Energia bruciata	Data
Corsa	30	102.1	17_12_2021:17.44
Salto	85	654.1	12_12_2021:15.44
Corda	65	65.1	12_12_2021:16.44
Trazioni	20	400.2	16_12_2021:13.25
Trazioni	20	400.2	16_12_2021:13.26
Trazioni	20	400.2	16_12_2021:14.14
Flessioni	30	125.2	18_12_2021:21.14
Panca piana	10	50	19_12_2021:12.17
Salto con l'asta	23	151	19_12_2021:15.46
Corsa	12	124	19_12_2021:15.48

Sempre cliccando sull'icona a forma di hamburger ci si può spostare su altre schede come quella “Allenatore”:



Dettagli allenatore

Nome: Giovanni

Cognome: Giorno

Età: 21

Certificazione allenatore: FIPE

In questa schermata si possono vedere le informazioni sull'allenatore. Per recuperare tali informazioni si utilizza la API GET “dati allenatore”. Infine, nella schermata “Le mie schede” si potranno vedere le due schede personalizzate:



Scheda di allenamento

Scheda di allenamento personalizzata per
Mario Rossi

Lunedì:
Addominali

Martedì:
Pausa

Mercoledì:
Gambe

Giovedì:
Pausa

Venerdì:
Braccia e Petto

Fine settimana:
Pausa

Scheda di alimentazione

Scheda di alimentazione personalizzata per
Mario Rossi

Colazione:
2 Gallette

Merenda:
1 frutto
tazza di the

Pranzo:
100g primo piatto
120 g secondo piatto

Merenda:
1 frutto

Cena:
100g primo piatto
120g secondo piatto

8 GitHub Repository Info

Il progetto SmartFit è disponibile al seguente link:

<https://github.com/StellaDaniele/G12-software-engineering>

Per poter eseguire il progetto è sufficiente recarsi all'interno della cartella `api` e poi eseguire il comando `npm start`. Il server, come verrà scritto in console, sarà in ascolto all'indirizzo `localhost:5000/`

8.1 Stringa di connessione a MongoDB

Viene di seguito riportata la stringa di connessione al database MongoDB, che non volevamo inserire pubblica su GitHub e che comunque cambieremo appena l'esito verrà pubblicato.

9 API Testing

Per eseguire il testing abbiamo definito nel nostro progetto la cartella “test” nella cartella `api`. Al suo interno, è presente uno script `index.js` che va a definire alcuni test cases. Si è andato a verificare il corretto funzionamento di una API GET, di una API POST provando anche a inserire dati errati e che non rispettavano i requisiti per dimostrarne la robustezza, e infine di una DELETE, anch'essa viene provata con input errati per dimostrarne la robustezza. Per eseguire il test abbiamo modificato il file `package.json` inserendo la riga di codice per l'esecuzione dello script. Per avviare i test basta recarsi nella cartella **api** ed eseguire il comando `npm test`. Si rammenta che dal momento che il server parte in ascolto, il testing non termina, per farlo terminare bisogna bloccare il server dal mettersi in ascolto dal file `index.js` nella cartella `api`. Questo è il risultato del testing:

```
PS C:\Users\Daniele\Desktop\G12-software-engineering\api> npm test

> api@1.0.0 test
> node test | tap-spec

TEST1: Informazioni allenatore
  ✓ Nessun errore
  ✓ Informazioni allenatore recuperate correttamente

TEST2: Aggiunta attività alla cronologia allenamenti
  ✓ Nessun errore
  ✓ Attività aggiunta correttamente
  ✓ API robusta e protetta da JSON malformati e/o con contenuto incorretto

TEST3: Eliminazione attività dalla cronologia allenamenti
  ✓ Nessun errore
  ✓ Attività eliminata correttamente

TEST4: Robustezza API eliminazione attività da cronologia allenamento
  ✓ Nessun errore
  ✓ API robusta e protetta da date che non rispettano il formato specificato
  ✓ La API non genera errori in caso si cerchi di eliminare attività non presenti nella cronologia

total:      10
passing:    10
duration:   725ms

All tests pass!
PS C:\Users\Daniele\Desktop\G12-software-engineering\api> |
```