

Dipartimento di Ingegneria e Scienza dell’Informazione

CORSO DI INGEGNERIA DEL SOFTWARE

****

Documento di progetto: Documento di sviluppo dell'applicazione

Gruppo: T33

**Indice**

[1. Scopo del documento 3](#_Toc118923077)

[2. Requisiti Funzionali 3](#_Toc118923078)

[2.1 Utente anonimo 3](#_Toc118923079)

[2.1.1 UML generale utente anonimo 6](#_Toc118923080)

[2.2 Utente studente 7](#_Toc118923081)

[2.2.1 UML generale utente studente 18](#_Toc118923082)

[2.3 Utente offerente 19](#_Toc118923083)

[2.3.1 UML generale utente offerente 30](#_Toc118923084)

[3. Requisiti Non Funzionali 31](#_Toc118923085)

[4. Analisi del Contesto 36](#_Toc118923088)

[4.1 Utenti e sistemi esterni 36](#_Toc118923089)

[4.2 Diagramma di contesto 37](#_Toc118923090)

[5. Analisi dei Componenti 39](#_Toc118923091)

[5.1 Definizione dei componenti 39](#_Toc118923092)

[5.2 Diagramma dei componenti 43](#_Toc118923093)

[6. Note ed eventuali 51](#_Toc118923094)

# Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte dell’applicazione *StayBusy*. Nello specifico, di seguito vengono presentati tutti gli artefatti necessari per la realizzazione dei servizi di gestione delle funzionalità fornite specificatamente solo alla tipologia di utente offerente dell’applicazione *StayBusy* e non all'utente studente. Questa scelta è dovuta data dal fatto che si è deciso di avere una programmazione più semplice e comprensibile sviluppando solo il lato utente offerente. Infatti, effettuare lo sviluppo dell'applicazione del lato studente avrebbe comportato una maggiore complessità nella creazione degli *User Flows* e delle *API* in quanto il login dello studente deve interfacciarsi con il sistema universitario dell'Ateneo a cui è iscritto. Avendo sviluppato un sito per gli studenti provenienti da tutta Italia sarebbe quindi stato difficile interfacciarsi ai siti di tutte le Università. Si è dunque deciso di concentrare l'attenzione sulle funzionalità associate a questo tipo di utente trattandole e sviluppandole nel modo più dettagliato possibile.

In particolare, nella parte iniziale del documento viene presentata la descrizione degli *User Flows* per il ruolo dell'utente offerente.

Gli **User Flows** si riferiscono ai diversi percorsi che gli utenti possono intraprendere mentre interagiscono con il nostro sito. Questi percorsi sono in genere visualizzati attraverso il diagramma di flusso da noi creato e mostrano i vari passaggi che un utente potrebbe eseguire mentre completa un'attività o raggiunge un obiettivo. I flussi utente vengono utilizzati nel nostro processo di progettazione e sviluppo per aiutare a identificare potenziali aree di miglioramento e per garantire che l'esperienza utente sia il più agevole e intuitiva possibile - come indicato nei requisiti non funzionali del documento *D1 - Analisi dei requisiti*.

Successivamente si prosegue con la **presentazione delle API** - attraverso *l'API Model* e il *Modello delle risorse*) necessarie per il funzionamento del sito. Le API principali sono state oggetto di una **fase di testing** con input corretti ed errati per dimostrare il loro funzionamento.

Infine, sono presenti due sessioni, una dedicata alle informazioni del Git Repository e il deployment dell’applicazione stessa e un'altra destinata alla realizzazione del front end e del back end per poter permettere la visualizzazione delle API in opera in una riproduzione il più simile possibile al front end fornito nel documento *D1 - Analisi dei requisiti.*

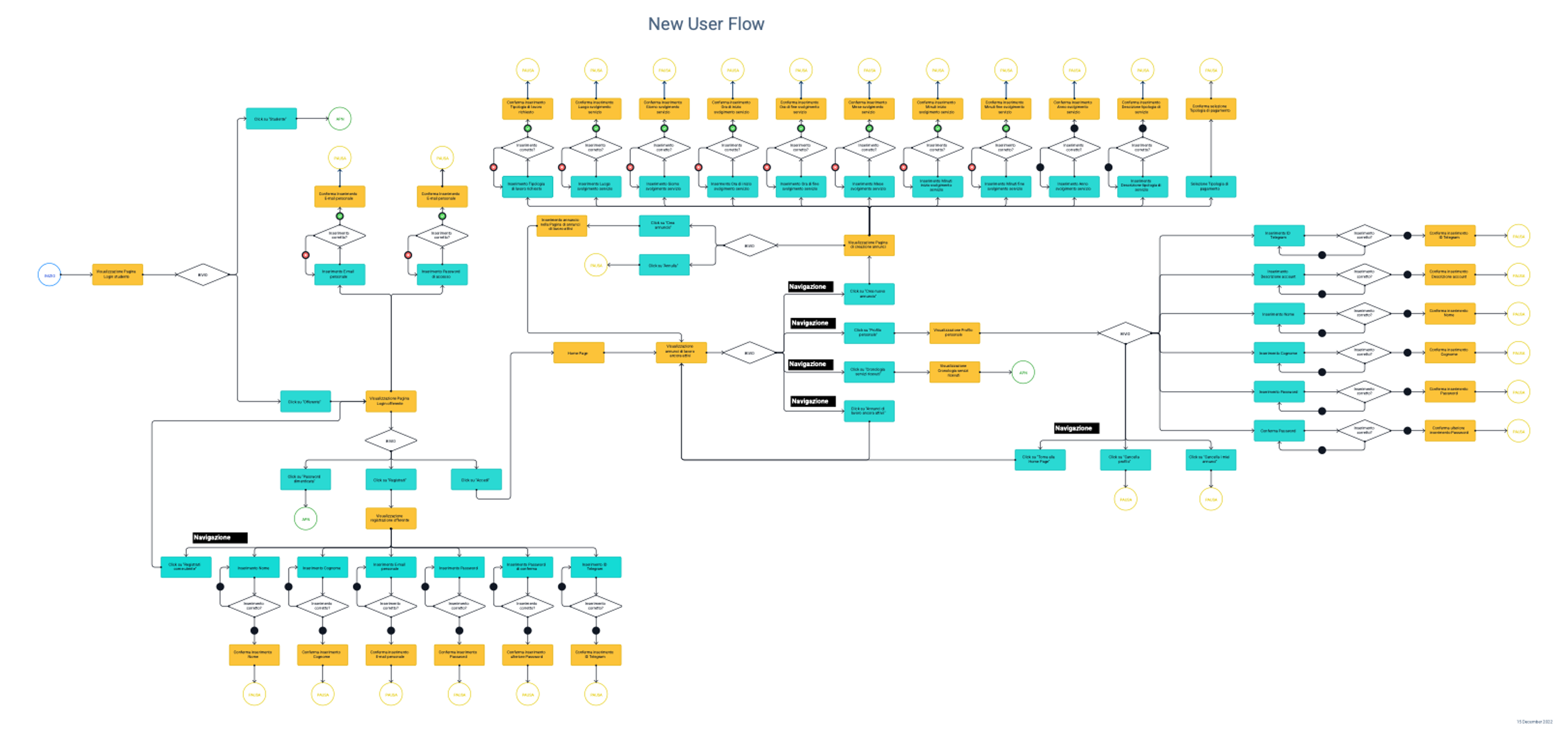
# User Flows

In questa sezione del documento di sviluppo vengono riportati gli “*User Flows*” per il ruolo dell’utente offerente.

L’immagine che segue descrive lo *User Flow* relativo alla gestione delle informazioni riguardanti le funzionalità proposte all'utente offerente. Nello specifico egli ha in ogni momento la possibilità di consultare la lista dei propri annunci attivi e dei corrispettivi studenti candidati, la possibilità di modificare le impostazioni del profilo personale, di visualizzare la lista degli annunci non più attivi - ovvero degli annunci la cui data di inizio è precedente al momento della visualizzazione - ed infine di creare un nuovo annuncio.

Attraverso questo diagramma presentiamo inoltre le relazioni che sussistono tra le varie azioni che l'utente offerente può svolgere e le features descritte nella *sezione 3* del documento. In aggiunta è presente una breve legenda che descrive i simboli utilizzati.

**2.1 Diagramma User Flow**



**LEGENDA**

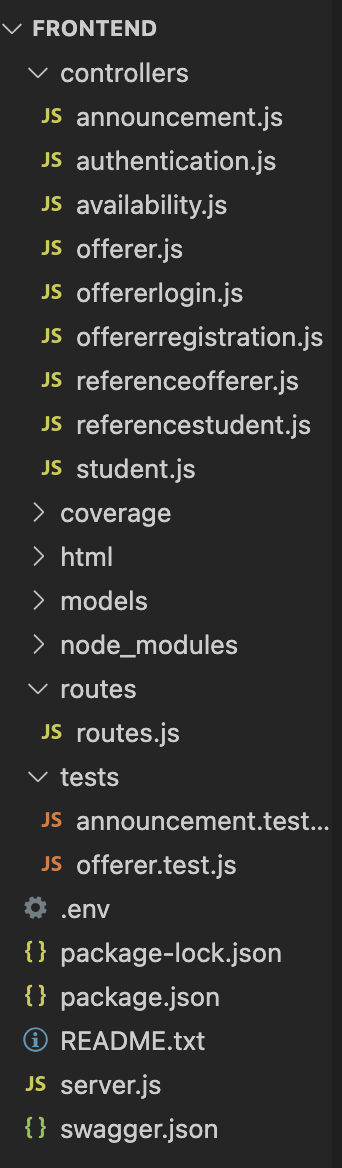
**3. Application Implementation and Documentation**

Nelle sezioni precedenti sono state da noi identificate le features che devono essere implementate per l'applicazione con un’idea di come il nostro utente finale può utilizzarle nel suo flusso applicativo. L’applicazione è stata sviluppata utilizzando *NodeJS* e *VueJS* per uno sviluppo efficiente ed efficace, *HTML* e *CSS* per quanto riguarda il front end e *JavaScript* per il back end e per le API. Infine, l’uso di *MongoDB* ha permesso il salvataggio che tutti i dati vengano archiviati e gestiti in modo scalabile e flessibile. Usando queste tecnologie assieme siamo stati in grado di creare in parte un'applicazione potente e facile da usare che soddisfi le esigenze del nostro utente finale.

**3.1. Project Structure**

Nella figura sottostante è resa visibile la struttura del progetto. Questo è composto dalle seguenti cartelle:

* *Controllers*:
* *Coverage*:
* *Html*:
* *Models*:
* *Node\_modules*:
* *Routes*:
* *Test*:



**3.2 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:

da fare

**3.3 Project Data or DB**

Per la gestione dei dati utili all’applicazione sono state definite una serie di strutture dati presentanti attributi diversi. Di seguito è presente una breve descrizione e un esempio di elemento presente nel database per ognuna tipologia di strutture.

* OFFERER

E' tipo di dato rappresentante l'utente che si registra all'applicazione per richiedere lo svolgimento di un servizio a pagamento creando annunci di lavoro. I dati salvati per ogni elemento Offerer sono: *name*, *surname*, *e-mail*, *idTelegram*, *confirmedAccount*, *password*, *relatedStudentsEmail*, *averageVotes*, e *description*.

* STUDENT

E' tipo di dato rappresentante l'utente studente che si registra all'applicazione ed è disposto a impegnare il proprio tempo libero in attività retribuite. I dati salvati per ogni elemento Student sono: *name*, *surname*, *e-mail*, *city*, *idTelegram*, *confirmedAccount*, *university, countDoneServices*, *countDeclinedServices, countDeclinedServicesBefore24h*, *averageVotes*, *serviceToDo*, *serviceCandidate*, *serviceCronology*, e *description*.

* ANNOUNCEMENT

E' il tipo di dato rappresentante l'annuncio

* REFERENCEOFFERER
* REFERENCESTUDENT
* AVAILABILITY

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 con- sumazione. 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 e` stata presa insieme al gruppo G11, dato che nelle specifiche del pro- getto non era stato dichiarato tale dato. Si riporta un esempio di elemento presente nel database:

**3.4 Project APIs**

Di seguito sono presenti una lista con una breve descrizione delle API che sono state sviluppate per questo progetto. Nella *sezione 4 - API Documentation* può essere trovata la descrizione completa per ogni API mentre il codice può essere trovato al il link alla repository di GitHub in una sezione successiva.

* OFFERER

**GET APIs**

Segue la lista delle API di tipo GET che sono state sviluppate per il lato utente offerente dell'applicazione.

* *E-mail*: recupera, in formato JSON, le informazioni riguardanti un utente di tipo offerente fornendo la sua e-mail personale.

**POST APIs** *IDTelegram*: permette la modifica dell'ID personale pre

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

* *OffererCreation*: si occupa della creazione dell'account di un nuovo utente offerente controllando e inserendo tutti i dati forniti tramite formato JSON nel database del sistema.
* *Login*: verifica se nel database del sistema è presente un utente offerente con le credenziali - ovvero con e-mail e password - inserite.
* *IDTelegram*: permette la modifica dell'ID personale di *Telegram* presente nell'area personale dell'utente inserito al momento della registrazione o eventualmente successivamente cambiato.
* *Description*: permette la modifica della descrizione personale presente nell'area personale dell'utente inserita al momento della registrazione o eventualmente successivamente cambiata.
* *Name*: permette la modifica dell'nome dell'utente presente nell'area personale dell'utente inserito al momento della registrazione o eventualmente successivamente cambiato.
* *Surname*: permette la modifica del cognome dell'utente presente nell'area personale dell'utente inserito al momento della registrazione o eventualmente successivamente cambiato.
* *Password*: permette la modifica della password di accesso presente nell'area personale dell'utente inserita al momento della registrazione o eventualmente successivamente cambiato effettuando anche un controllo di soddisfacimento dei requisiti di *Strong Password* definiti nel documento *D1 - Analisi dei requisiti* e di coincidenza tra password e password di consegna.

**DELETE APIs**

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

* *E-mail*: elimina l'account dell'utente offerente noto l'indirizzo e-mail e si occupa dell'eliminazione degli annunci attivi, se presenti.
* ANNOUNCEMENT

**GET APIs**

* *E-mail*: recupera, in formato JSON, le informazioni riguardanti il/gli annuncio/i creati da un utente di tipo offerente fornendo la sua e-mail personale.

**POST APIs**

* *AnnouncementCreation*: si occupa della creazione di un nuovo annuncio di un utente offerente controllando e inserendo tutti i dati forniti tramite formato JSON nel database del sistema.

**4. API documentation**

Le API locali fornite dall’applicazione *StayBusy* descritte nella sezione precedente sono state documentate utilizzando il modulo *NodeJS* chiamato ***Swagger UI Express***. In questo modo la documentazione relativa alle API è direttamente disponibile a chiunque veda il codice sorgente.

Per poter generare l’endpoint dedicato alla presentazione delle API abbiamo utilizzato *Swagger UI* attraverso cui è stato possibile creare una pagina web dalle definizioni delle specifiche *OpenAPI*.

In particolare, di seguito mostriamo la pagina web relativa alla documentazione che presenta le API per la gestione dei dati della nostra applicazione. Nello sviluppo delle API vengono usate le funzioni *GET*, *POST* e *DELETE* per recuperare, creare ed eliminare dati, rispettivamente. In particolare, la funzione *GET* viene utilizzata per recuperare i dati da un server, la funzione *POST* per creare una nuova risorsa sul server mentre infine la funzione DELETE viene utilizzata per eliminare una risorsa dal serve -ovvero per eliminare un utente, un prodotto o un altro tipo di risorsa che non è più necessario. In generale, queste funzioni vengono utilizzate in modo da consentire a diverse applicazioni e servizi di interagire tra loro. Queste ci ha consentito la creazione di sistemi più potenti e flessibili e di servizi che possono sfruttare i dati e le funzionalità fornite da altri sistemi.

L’endpoint da invocare per raggiungere la seguente documentazione è: **http://localhost:8080/api-docs.**

Si noti che non sono state implementate tutte le API ma solamente quelle necessarie per il funzionamento del sito dal lato dell'utente offerente per le motivazioni precedentemente citate.

Sotto viene riportata l’immagine della pagina web relativa alla documentazione.

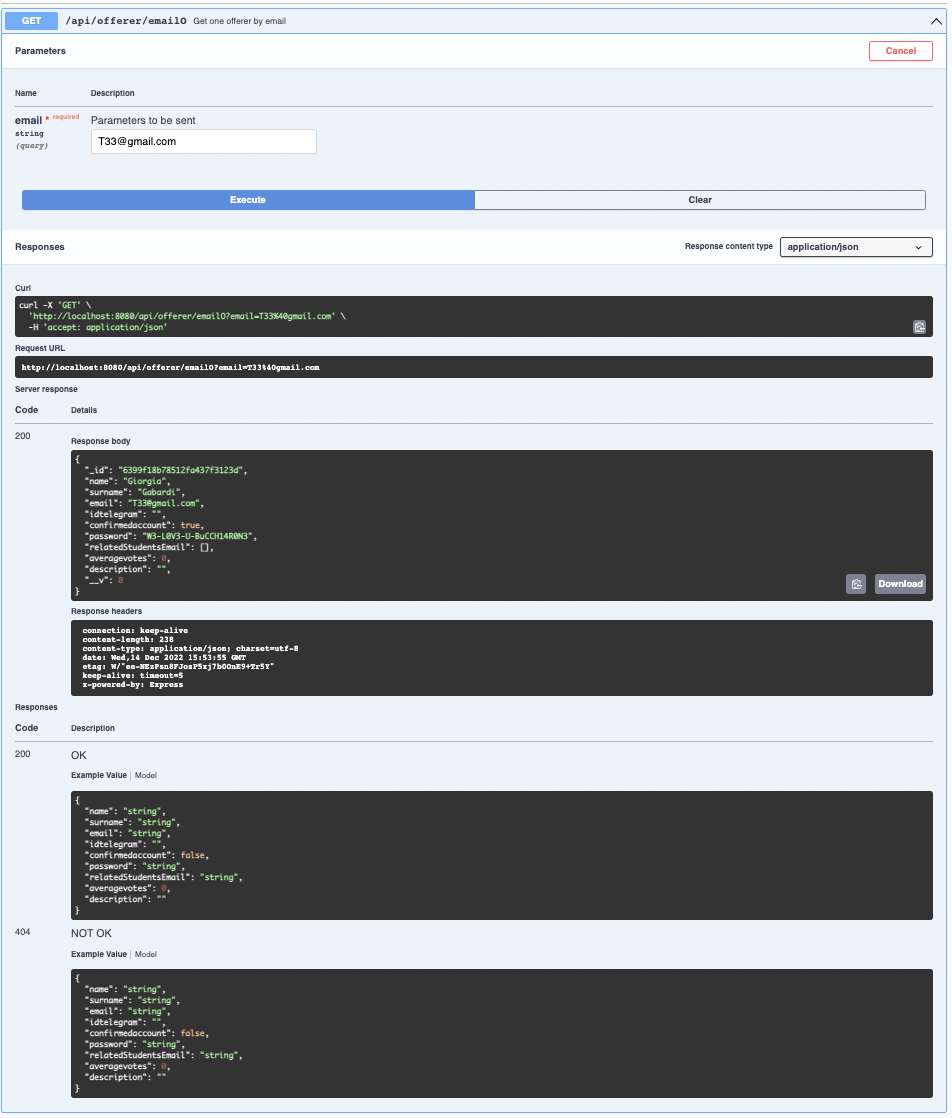


* OFFERER

Immagine che contiene testo

Descrizione generata automaticamente

Di seguito viene descritto l'utilizzo e il funzionamento delle API più importanti con l’aiuto della documentazione.

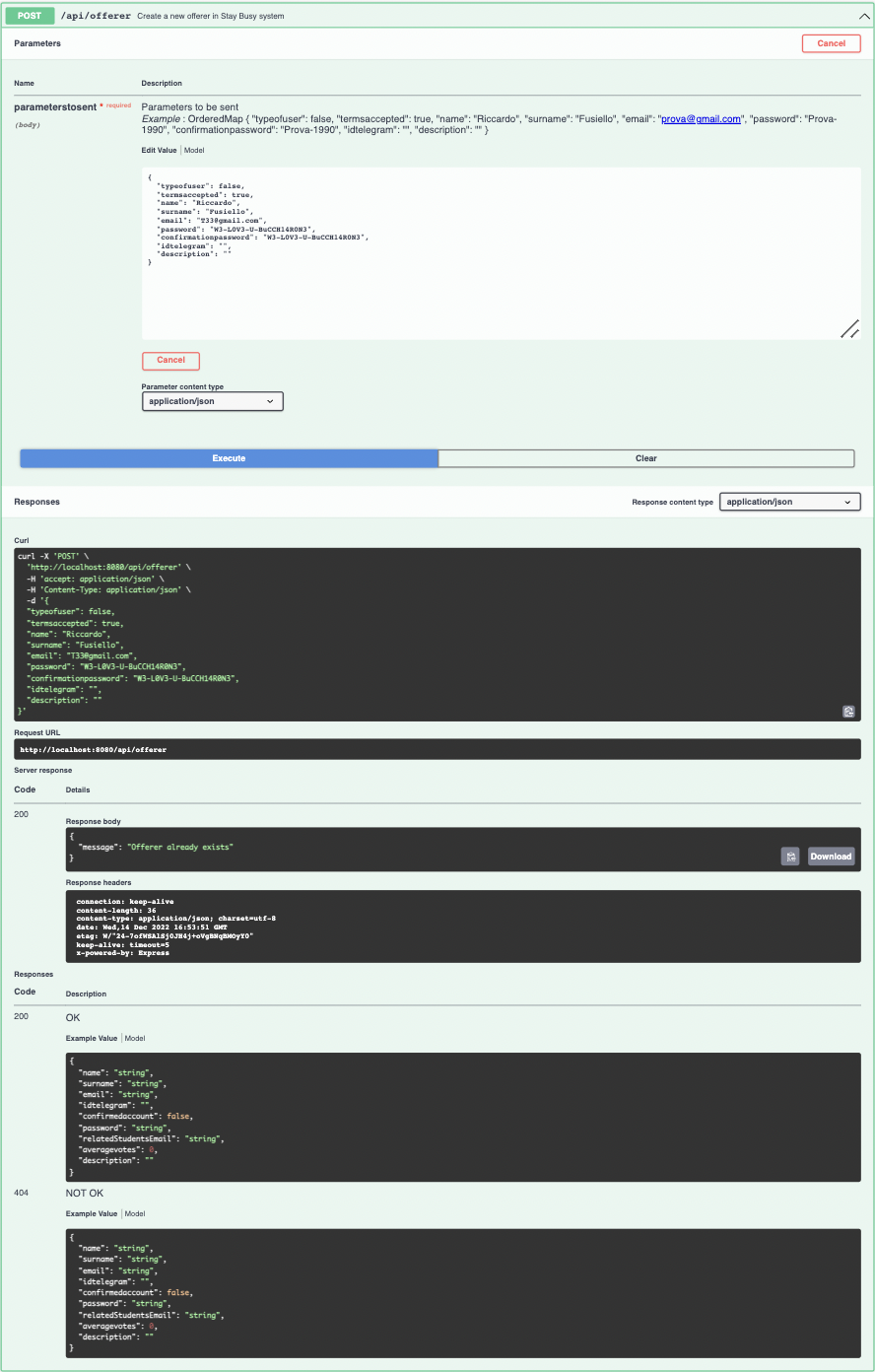


Questa API di tipo *GET* si occupa di effettuare la connessione con il database *MongoDB* eseguendo la funzione *findOne()* con l'e-mail che l’utente passa come *path parameter*, ovvero parametro di ingresso. Questa funzione restituisce i dati di registrazione dell'utente nel caso in cui egli si sia precedentemente registrato mentre, in caso contrario, viene restituito il messaggio "*Offerer doesn't exist*".

Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente

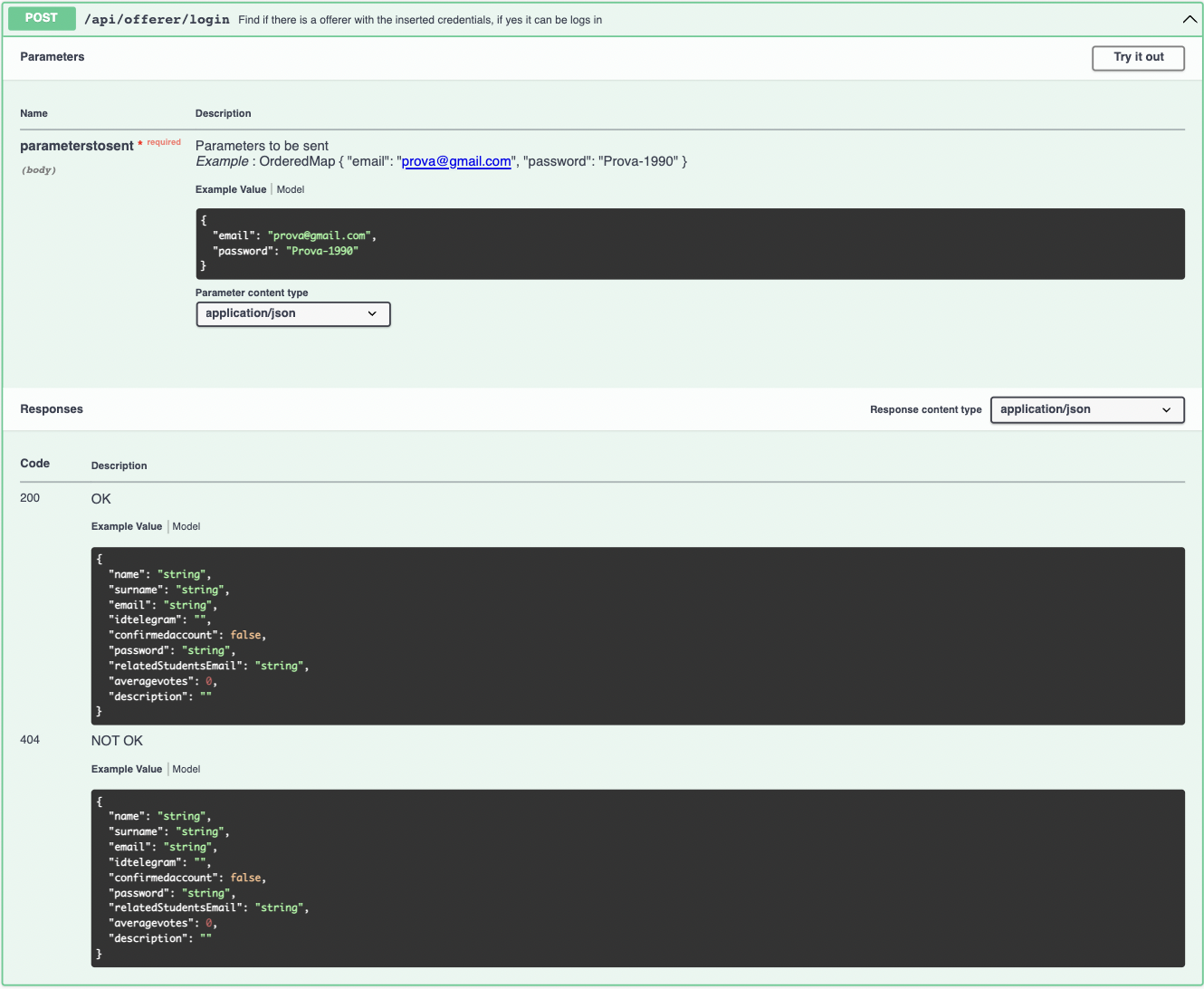


Questa API di tipo *POST* gestisce la creazione dell'utente offerente. In particolare, si occupa di verificare che i dati inseriti dall'utente rispettino le regole implementate dall'applicazione e, attraverso il metodo *findOne()* - a cui viene passata come parametro di ingresso l'e-mail - di verificare che nel database non sia già presente un utente registrato con la stessa e-mail. In questo caso viene restituito il messaggio "*Offerer already exists*".

Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente



Questa API di tipo *POST* è quella che gestisce il controllo dell'esistenza dell'account di un utente - ovvero il controllo di e-mail e password inseriti - restituendo il messaggio "*Wrong credential*" nel caso in cui queste non coincidano, siano inserite in modo errato o in generale non siano presenti all'interno del database.

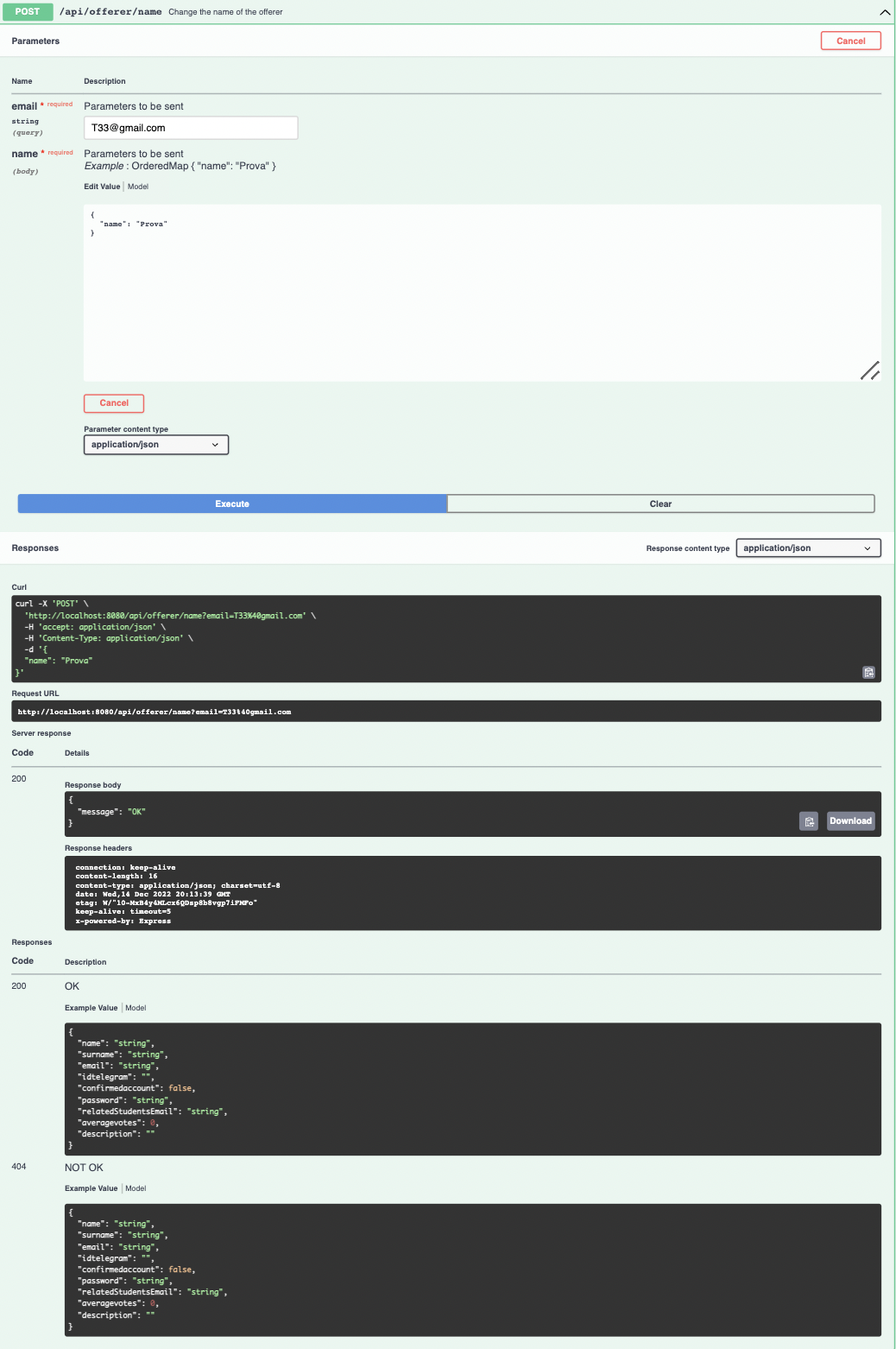
Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente

Le successive API di tipo *POST* sono praticamente uguali, solo che ritornano file *JSON* diversi. Il loro funzionamento è molto simile - ovvero permettono tutte la modifica di uno specifico dato del proprio profilo personale inserito dall'offerente al momento della registrazione ed eventualmente successivamente modificato. Ciò che quindi varia sono solamente i parametri in ingresso passati alla funzione *findOneAndUpdate()* - cioè i parametri che verranno poi modificati attraverso l'utilizzo di quest'ultima.

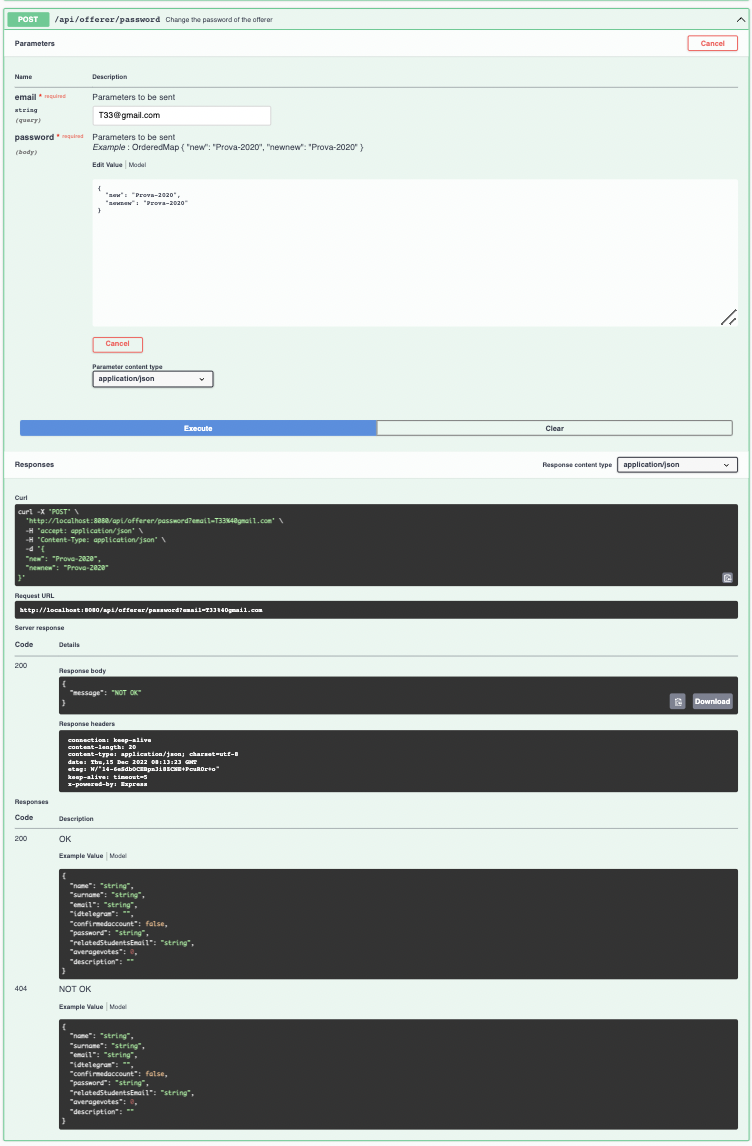
Per non creare un documento troppo lungo, essendoci il codice disponibile su *GitHub* e dato che come detto le successive API *POST* sono tutte abbastanza simili, si riporta l’esempio dell'API di modifica del nome utente:



Di seguito il codice della funzione utilizzata:

Immagine che contiene testo, screenshot, schermo, vicino

Descrizione generata automaticamente

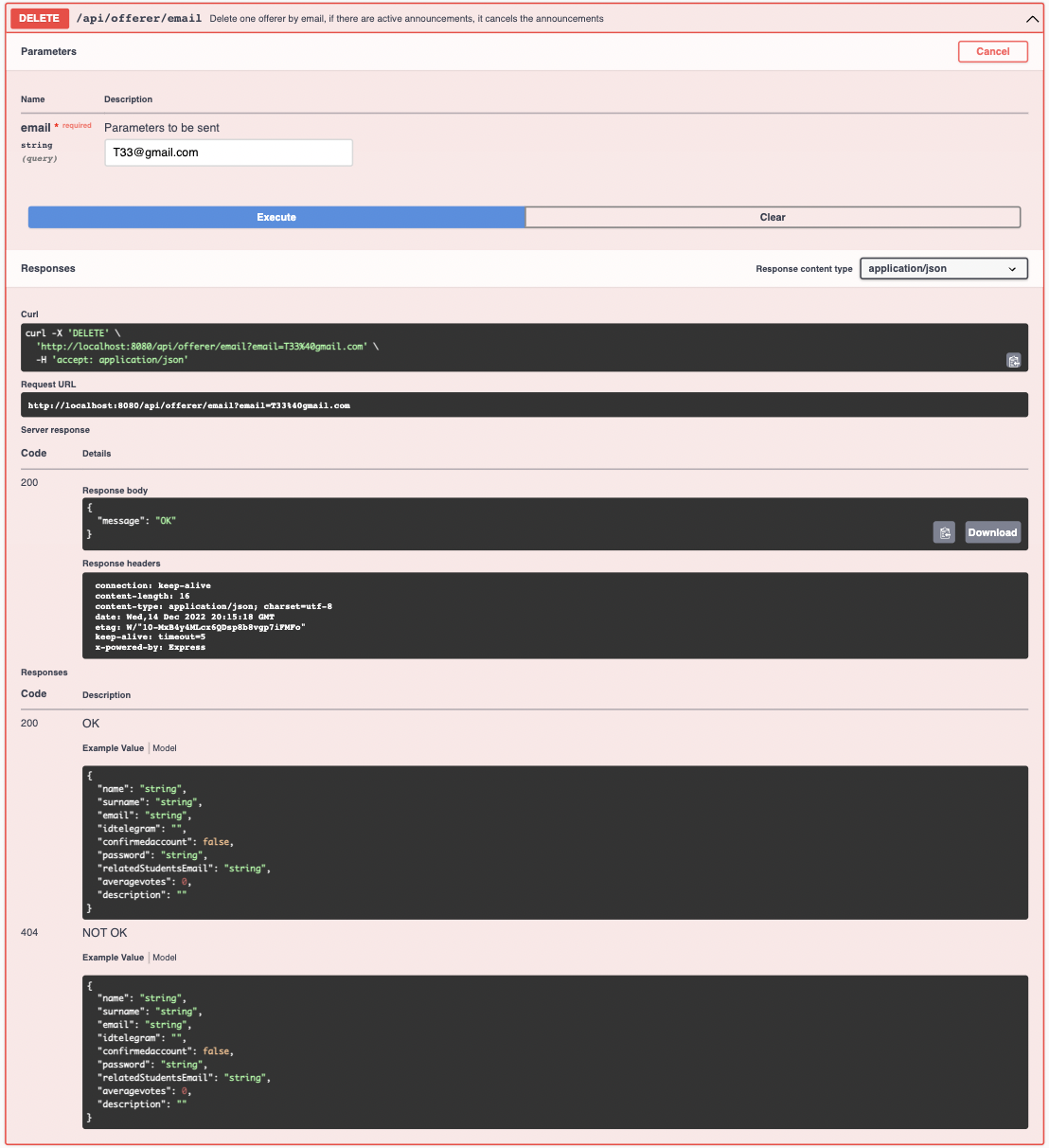


L'API di tipo *POST* che permette di modificare la password di accesso al sistema passando come parametro in ingresso l'e-mail dell'offerente è leggermente differente rispetto a quelle precedenti in quanto in questa è necessario anche implementare il controllo che la password sia inserita correttamente due volte consecutive.

Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente



Questa API di tipo *DELETE* si occupa dell'eliminazione dell'account dell'offerente attraverso l'utilizzo della funzione *deleteOne()* passando come parametro in ingresso la propria e-mail personale.

Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente

* ANNOUNCEMENT

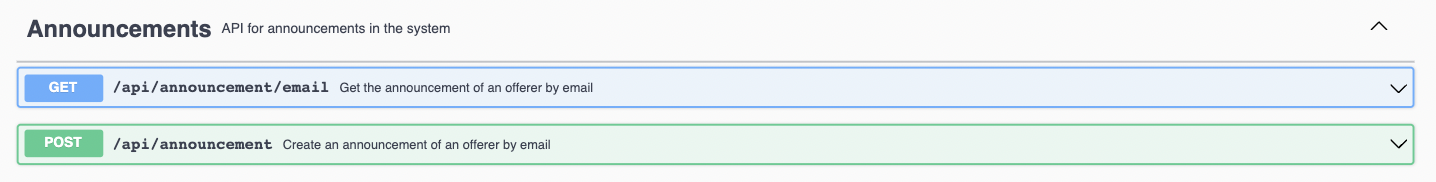


Immagine che contiene testo

Descrizione generata automaticamente

Questa API di tipo *GET* si occupa di effettuare la connessione con il database *MongoDB* eseguendo la funzione *findOne()* con il *typeOfWork* che l’utente passa come *path parameter*, ovvero parametro di ingresso. Questa funzione restituisce i dati relativi all'annuncio creato dall'offerente nel caso in cui egli abbia effettivamente precedentemente creato tale annuncio mentre, in caso contrario, viene restituito il messaggio "*Announcement doesn't exist*".

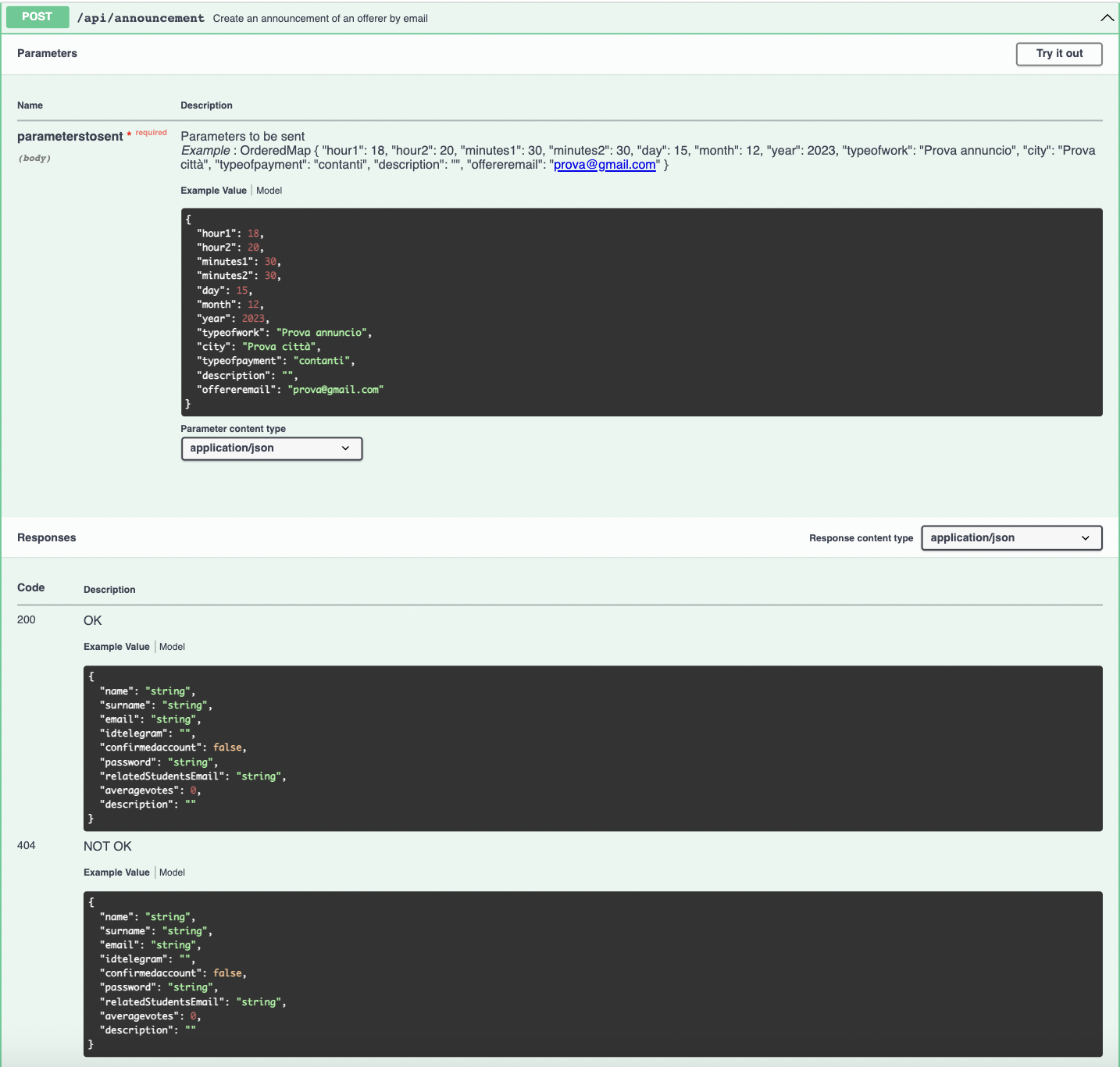
Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente



Questa API di tipo *POST* permette la creazione di un annuncio. In particolare, sono stati implementati tutti i vincoli imposti sull'orario di inizio e fine de servizio richiesto definiti nel documento *D3 - Documento di Architettura.*

Di seguito il codice della funzione utilizzata:

Immagine che contiene testo

Descrizione generata automaticamente

# 5. FrontEnd Implementation

Questa sezione del documento fornisce le funzionalità di visualizzazione, inserimento e cancellazione dei dati fornite nell'applicazione *StayBusy* all'utente offerente attraverso l'utilizzo di tutte le API sviluppate date dalle varie schede create.

La prima scheda mostrata dall'applicazione è la pagina di **Login dell'account studente** da noi non implementata per le ragioni precedentemente scritte. Qui è possibile selezionare l'opzione offerente e visualizzare la pagina **Login dell'account offerente** da cui è possibile accedere all'applicazione come utente offerente di servizi.

FOTO FRONTEND INDEX E LOGIN OFFERENTE

Nella parte centrale viene richiesto l'inserimento dell'e-mail di accesso e della password impostate dall'utente al momento della registrazione. Per semplificare questa procedura è fornita all'utente la possibilità di poter salvare i dati di accesso in modo da non dover ricordare la password e doverla inserire ogni qualvolta che desidera accedere al sito. Inoltre, un'ulteriore funzionalità data consiste nel suggerimento di una password sicura che rispetti i *requisiti di Strong Password* definiti nel documento *D1 - Analisi dei requisiti*. Cliccando su *Password dimenticata?* l'utente può inoltre recuperare la password di accesso nel caso in cui se la sia dimenticata.

In basso è presente il pulsante *Accedi* che, se premuto, consente di effettuare l'accesso con i dati precedentemente inseriti mentre se non si ha mai effettuato l’accesso è possibile effettuare la registrazione cliccando l’apposito bottone *Registrati* che invece manda l'utente alla schermata di registrazione.

FOTO FRONTEND REGISTRAZIONE

Nella pagina di **Registrazione** dell'utente l’offerente dovrà fornire alcune informazioni, quali Nome, Cognome, E-mail con cui si desidera effettuare la registrazione, Password che rispetti i *requisiti di Strong Password* definiti nel documento *D1 - Analisi dei requisiti*, conferma della password precedentemente inserita e ID Telegram. Per completare la registrazione è necessario accettare i *Termini e Condizioni e Privacy Policy* spuntando l’apposita casella.

Effettuati correttamente login o registrazione, l'utente offerente visualizza l'Home Page del sito. Questa propone in primo piano la lista di annunci creati dall’offerente stesso, mostrando per ognuno i dati inseriti dall’offerente al momento della creazione dell’annuncio, il numero di persone da cui è stato visualizzato, gli studenti che effettivamente si sono proposti per svolgere il lavoro e lo stato dell’annuncio (assegnato, non assegnato). Cliccando sull’anteprima dell’annuncio, l’offerente viene rimandato all’annuncio vero e proprio.

FOTO HOME PAGE (ANNUNCI DI LAVORO ATTIVI)

La barra superiore della **Home page** mette a disposizione all’offerente diverse funzionalità quali, rispettivamente da sinistra verso destra:

visualizzare la cronologia dei servizi ricevuti;

visualizzare il proprio profilo, eventualmente modificandolo.

La schermata di **Cronologia dei servizi ricevuti** mostra all'offerente la lista di tutti i servizi non più attivi al momento della visualizzazione - ovvero dei servizi eliminati o che sono stati svolti in passato da uno studente.

FOTO CRONOLOGIA SERVIZI

La pagina di **Profilo personale** permette la modifica dei dati personali dell’offerente e della password di accesso al sito, la possibilità di tornare alla pagina Home page, di cancellare il profilo e di eliminare tutti gli annunci ancora attivi. In entrambi i casi è possibile effettuare queste operazioni solo se vengono rispettate le imposizioni descritte nel documento *D2 - Specifica dei requisiti.*

FOTO PROFILO PERSONALE.

Inoltre, tutte le schermate del profilo forniscono una barra laterale per navigare all’interno del proprio profilo: **Annunci di lavoro attivi** permette la visualizzazione della **Home page** mentre **Crea un nuovo annuncio** permette la creazione di un nuovo annuncio.

Tramite un’apposita etichetta sul menù di sinistra è infatti possibile accedere nella pagina di **Creazione nuovo annuncio***.* Questa permette all’offerente di creare un nuovo annuncio inserendo dei dati specifici, quali il tipo di lavoro – ovvero un insieme di parole chiave per descrivere il servizio richiesto –, (eventuale) luogo dove verrà svolto il lavoro, la data e l’ora in cui deve essere svolto il lavoro, la tipologia di pagamento prevista e una breve descrizione del lavoro che lo studente dovrà effettuare.

L’annuncio verrà creato ed inserito a sistema premendo l’apposito pulsante *Crea annuncio*. Cliccando questo tasto verrà dunque modificata la pagina **Home Page** alla quale verrà aggiunto l'annuncio appena creato.

FOTO CREAZIONE ANNUNCIO.

# GitHub Repository and Deployment Info

# Testing

# Note ed eventuali

Il diagramma *User Flow* presente in questo sito è stato realizzato interamente dal nostro gruppo tramite il sito [www.flowmapp.com](https://www.flowmapp.com/).

MANCA PARTE DOPO.