Data format

User {
  email: String!
  givenName: String
  familyName: String
  age: String
  gender: String
  livesInCountry: String
  hasNationality: String
  hasHealthRestriction: [String!]
  hasCategoryPreference: [String!]
}
On all backend parameters are string
No validations: country, nationalities, genders, healthRestrictions

Status codes: 200, 201, 400, 403, 404.
Obs: everything is application/json as format
Any errors that appear return the payload of:
{
	“status”:”code”
	“message”: “message content”
}

Endpoint user (CRUD): /user-profile/user

R (citire toate datele user)
GET **/user-profile/user/<user_email>**

Input: email
http://127.0.0.1:5000/user-profile/user/ggosnoll3@salon.com
Output:
All user info, but only existent fields (if email exists):
Status code 200


```
{
    "hasCategoryPreference": [
        "dried-products",
        "shelled-almonds"
    ],
    "hasHealthRestriction": [
        "salicylateIntolerance",
        "eggsIntolerance"
    ],
```

```
    "gender": "male",
    "livesInCountry": "Portugal",
    "familyName": "Gosnoll",
    "givenName": "Gasparo",
    "user": "ggosnoll3@salon.com",
    "hasNationality": "British",
    "age": "11"
}
```


Exceptions:
1. If email not existent:
   Status code 404


```
{
    "message": "User ccornei.laura10@gmail.co doesn't exist"
}
```


U (update some info user)
PUT /user-profile/user/<user_email>

Input: user info to be updated, in body, x-www-form-urlencoded, cheile ca la formatul de date de sus

Output: all user info, existent fields (limitations pystardog)
Status code 200


```
{
    "email": "ccornei.laura10@gmail.com",
    "hasHealthRestriction": [“diabetes”],
    "gender": "female",
    "livesInCountry": "India",
    "familyName": "Cornei",
    "givenName": "Laura-Maria",
    "hasNationality": "Indian",
    "age": "100"
}
```


Exceptions:
1. If email not existent:
   Status code 404


```
{
    "message": "User ccornei.laura10@gmail.co doesn't exist"
}
```


2. If existent fields that are either email/firstName/givenName/age is invalid:
   Status code 403

ex:{
"message": "Invalid email"
}

C (create new user)
POST **/user-profile/user/**
Input:
In body, campurile cu cheile “email”, “givenName”, “familyName”, x-www-form-urlencoded

Output : user email
status code 201 (created)

```
{
    "email": "test@acum.com.ro"
}
```

Exceptions:

1. If user email already exists:
   Status code 403
   ```
   {
       "message": "User email test@acum.com already exists"
   }
   ```
2. If email/firstName/givenName is invalid:
   Status code 403
   ex:{
   "message": "Invalid email"
   }
3. If more/less fields are given (ex: age is introduced/ email is deleted):
   Status code: 400
   ```
   {
       "message": "Invalid given user fields"
   }
   ```

D (delete entire user)
DELETE **/user-profile/user/<user_email>**
Input: user email

Output: email of deleted user
Status code 200

```
{
    "message": "User test@acum.com.ro deleted from database"
}
```

Exceptions:
1. If email not existent:
   Status code 404

```
{
    "message": "User ccornei.laura10@gmail.co doesn't exist"
}
```

Endpoint user (R): /user-profile/metadata

R
GET **/user-profile/metadata**
Input: none

Output:
Status code 200

```
{
    "gender": [
        "female",
        "male"
    ],
    "countries": [
        "Afghanistan",
        "Venezuela(BolivarianRepublicof)" ...
    ],
    "nationalities": [
        "Afghan",
        "Albanian",
        "Congolese(DRC)" …
    ],
    "healthRestrictions": [
        "alcoholIntolerance",
        "caffeineIntolerance",
        "cancer",
   ],
    "categoryPreference": [
        "advent-calendars",
        "agave-syrups",
        "agua-mineral",
     ..]
}
```

Toate endpointurile:
Obs ce tine de libraria flask restful: pentru orice request, dc se da altceva in body inafara de campurile din formatul de date =>
Status code 400

```
{
    "message": "Unknown arguments: fdfdf"
}
```


Daca metoda nu e permisa pt url:


```
{
    "message": "The method is not allowed for the requested URL."
}
```