Documentation for BE API

```
**Must-have fields

======================================== User
========================================

Without Id ======================
.post(localhost-port../register):
Passport - registration: returns token of authentication/Unauthorized
if fail
  {
    **"username": string,                                    # can
contain special characters
    **"password": string,                                    # can
contain special characters
    **"email": string,                                         # in
email format
    **"firstName": string,                                    # cannot
contain special characters
    **"lastName": string,                                    # cannot
contain special characters
    **"dateOfBirth": "YYYY-MM-DD"/"YYYY-DD-MM",        # must be a
valid date
      "biography": string,
  }
.post(localhost-port../login):
Passport - login: returns token of authentication/Unauthorized if fail
  {
    **"username": string,                                    # can
contain special characters
    **"password": string,                                    # can
contain special characters
  }
.get(): gets all users -> returns array of users in the
form:
.get(/search): search with given query, return arrays of matching
  {
       **"query":
string                                                       # can
contain special characters
  }
.put(): updates user with given Id -> returns updated user
Should not include username(unchangeable), password and email (must be
updated under protection).
Other fields if have no updates can be ignored.
  {
     "firstName": string,                                    # cannot
contain special characters
```

```
      "lastName": string,                                    # cannot
contain special characters
      "dateOfBirth": "YYYY-MM-DD"/"YYYY-DD-MM",        # must be a
valid date
      "biography": string,
  }
.delete(): deletes user with give Id -> returns status 200 if successful

With Id ========================
.get(): returns user with given Id -> returns user if found/status 404
otherwise

======================================== Contact
========================================

Without Id ======================
.post(): creates new contact normally -> returns data if successful
  {
    **"firstName": string,                                    # cannot
contain special characters
    **"lastName": string,                                    # cannot
contain special characters
      "email": string,                                              #
(if exists) in email format
      "phoneNumber": string,                                 # (if
exists) contain numbers only
      "dateOfBirth": "YYYY-MM-DD"/"YYYY-DD-MM",        # (if exists)
must be a valid date
      "biography": string,
      "jobTitle": [string],
  }
.get(): gets all contacts -> returns array of contacts
.get(/search): search with given query and owner userId, return arrays
of matching, same as 'user'
.get(/getall): returns all contacts that belongs to this userId ->
status 404 otherwise if none

With Id ========================
.post(/add/:userId): creates new contact with a given user Id ->
returns data if successful
.put(/:id): updates contact with given Id -> returns updated contact
  same as 'create', except for:
  - belongsTo should not be included since this field is fixed and
can't be updated
  - firstName & lastName are not required if no updates on these fields
.delete(): deletes contact with give Id -> return status 200 if
successful
.get(): returns contacts with given Id -> returns contact if found
/status 404 otherwise
```

```
======================================== Event
========================================

Without Id ======================
.post(): creates new event -> returns data successful
  {
    **"name": string,                                                  #
cannot contain special characters
    **"dateTime": "YYYY-MM-DD"/"YYYY-DD-MM",          # must be a valid
date
    **"completed": boolean,
      "participants": [string],                   # Can be names
/contactIds (which will be converted to names in BE)
      "description": string,
  }
.get(): gets all events -> returns array of events
.get(/getall): returns all events that belongs to this userId -> status
404 otherwise if none
.get(/search): search with given query and owner userId, return arrays
of matching, same as 'user'
  {
        **"query": string
          "completed":
boolean                                                 # Get events
that match this status
          "from": "YYYY-MM-DD"/"YYYY-DD-MM",                  # must be
a valid date
          "to": "YYYY-MM-DD"/"YYYY-DD-MM",                      #
must be a valid date
          "participants": [contactIds]                            #
looks for events that contain these contacts as participants
  }

With Id ========================
.put(): updates event with given Id -> returns updated event
  same as 'create', except for name & dateTime & completed are not
required if no updates on these fields
.delete(): deletes event with give Id -> returns status 200 if
successful
.get(): returns event with given Id -> returns event if found/status
404 otherwise

==================================== Relationship
====================================

Without Id ======================
.post(): creates new relationship -> returns data if successful
  {
    **"people": [string],                                              #
array of userIds, exactly 2 elements
```

```
    **"startedDatetime": "YYYY-MM-DD"/"YYYY-DD-MM", # must be a valid
date
      "tag": [string],
      "description": string
  }
.get(): gets all relationships -> returns array of relationships
.get(/search): search with given query and owner userId, return arrays
of matching, same as 'user'
.get(/getall): returns all contacts that belongs to this userId ->
status 404 otherwise if none

With Id ========================
.put(): updates relationship with given Id -> returns updated
relationship
  people and startedDatetime are fixed and can't be updated
  {
    "tag": [string],
    "description": string
  }
.delete(): deletes relationship with give Id -> returns status 200 if
successful
.get(): returns relationship with given Id -> returns relationship if
found/status 404 otherwise

======================================== Conversation
========================================

Without Id ======================
.post(): creates new conversation -> returns data if successful
  {
    **"people": [string],                                # array of
userIds, must greater than 1
      "messages": [
        {
          "sender": string,                             # userId
          "content": string,                          # must include
content
        },
        {
          "sender": string,                             # userId
          "content": string,                          # must include
content
        }
      ]
  }
.get(): gets all conversations -> returns array of conversations
.get(/getall): gets all conversations that belong to this user

With Id ========================
.put(): updates conversation with given Id -> returns updated user
```

```
  Update messages only (push new messages to the array)
  {
    "messages": [
      {
        **"sender": string,                              # userId
        **"content": string,                     # must include content
      }
    ]
  }
.delete(): deletes conversation with give Id -> returns status 200 if
successful
.get(): returns conversation with given Id -> returns conversation if
found/status 404 otherwise
.get(/search/:id): search with given query, and id of the convo, query
json format is same as 'user'

============================ Fields of searching in search engine
============================

Event:                       name, description, dateTime
Contact:               firstName, lastName, email, jobTitle
Relationship:           tag
Conversation:           message content
User:                       username, firstName, lastName, email
```