|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Returns** | **Parameters** | **HTTP Method** | **Method Name** | **ID** |
| Using POST since we need to send the user details inside the body (for user security). If login success, add hello [user] to the top of the window and hide register. If login failed, show error message. | token | username, password | POST | users/login | 1 |
| Send user details to server in order to save user in system database. The server generates automatically username and password and send back to the user. Using POST since we add new user to server database. | username, password | firstName, lastName, city, country, email, catergory1, category2, category3, category4, verificationQues1, verificationQues2, verificationAns1, verificationAns2 | POST | users/register | 2 |
| Get user password in case of the user answered correctly to verification question. Using POST since we want to pass parameters in message body. | password | username,  verificationQues,  verificationAns | POST | users/retrievePassword | 3 |
| Get 3 random locations with rate above 4. Using GET since the method doesn't change details in server and there are no parameters to pass. | Return 3 random and popular locations:  popularLocations[] | - | GET | general/randomPopularLocations | 4 |
| Update favorites list when user press on "save favorites location" button. The array order indicate on the user required order to the results. Using PUT since we need to need to change the favorites list in server. | - | favoritesListIds[]  [{"locationId": id}] | PUT | reg/locations/updateFavoriteList | 5 |
| Get user favorites locations according to the locations he saved in the past. Using GET since he method doesn't change details in server. | Return saved locations:  }[  locationId:id,  index: idx  }] | - | GET | reg/locations/favoriteLocations | 6 |
| add user review to location. Using POST since we want to add new review. | - | review (string),  locationId | POST | reg/locations/addReview | 7 |
| Change location rate according to user rate (number between 1 to 5). Using PUT since we want to change location rate in server. | New location rate | rate(int), locationId | PUT | reg/locations/rateLocation | 8 |
| Get location information according to his unique id. every time the user ask to watch the detail the server add 1 to "numberOfViewers". Using GET since we can pass the parameter in URL. | Return location object with 2 latest reviews (as written below) | locationid | GET | general/locationInfo | 9 |
| Get all possible categories for point of interests (historical sites, museum, etc.). Using GET since the method doesn't change details in the server. | Return all Categories:  Categories[] | - | GET | general/allCategories  צריך להוסיף קטגוריה ריקה | 10 |
| Get all possible countries .Using GET since the method doesn't change details in the server. | Return all countries:  Countries[] | - | GET | general/allCountries | 11 |
| Get all locations in DB.  Using GET since the method doesn't change details in the server. | Return all locations:  Locations[] | - | GET | general/allLocations | 12 |
| Get 2 last saved locations for user.  Using GET since the method doesn't change details in the server. | Return 2 favorite location ids and dates:  Location1, location2 | - | GET | reg/locations/lastSavedLocations | 13 |
| Get most popular location for user according his favorite categories. Using GET since the method doesn't change details in the server. | Return 2 locations from 2 different categories:  Location1, location2 | - | GET | reg/locations/mostPopularLocationsForUser | 14 |

**הערה**: מרגע התחברות בכל אחת מהפונקציות עובר גם ה-token כפרמטר ב-header של ה-request. Token זה מכיל גם את ה-userName.

**להלן מבנה ה-JSON של אובייקט מסוג USER:**

}

"firstName" : "Erez"

"lastName" : "Shalem"

"city" : "Beer-Sheva"

"country" : "Israel"

“email” : “erez@gmail.com”

"category1" : "night life"

"category2": "museums"

"category3":

"category4":

"verificationQues1":

"verificationQues1":

"verificationAns1":

"verificationAns1":

}

**להלן מבנה ה-JSON של אובייקט מסוג Location:**

{

"id" :1234

"name" : "BGU University"

"picture" : pic1

"numberOfViewers": 23

"description" : "good university"

"rate" : 4

"rateCounter" : 6

}

**הערות:**

* שמירת נקודת עניין, הסרת נקודת עניין ומיון נקודות עניין לפי בקשת המשתמש עובר נקודות העיין המועדפות הינן פעולות המתבצעות באופן לוקלי בצד לקוח ולכן אין צורך לבצע בקשה לשרת בכל אחת מהפעולות הללו. רק כאשר המשתמש ילחץ על "שמור נקודות מועדפות" תישלח בקשה לשרת עם רשימת המועדפים העדכנית שתדרוס את הרשימה שקיימת כרגע בשרת, וזאת באמצעות מתודת updateFavoriteList(favoritesList[]).
* כל החיפושים והמיונים מתבצעים בצד לקוח.

**קישור לGIT:** [**https://github.com/assafna/pointsOfInterests**](https://github.com/assafna/pointsOfInterests)