|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Explanation | Returns | Parameters | HTTP Method | Method Name | ID |
| Movies are related to the server, the client isn't aware of it.  Using GET because there is no information to deliver to the server or any wish to change any information. | top 5 trend movies:  [  {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  },  ] | - | GET | Users/Top5TrendMovies | 1 |
| Login needs to pass parameters, thus we'll use POST.  True=success  False=failed  Also, post function doesn't pass the parameters in the url, and it's necessary for login function. | {"login success"}  When success  {"invalid username or password"}  When failed | {"Username":,  "Password":  } | POST | Users/Login | 2 |
| * Register needs to pass parameters, thus we'll use POST.   "Registration success"=success   * "user name is already taken , please pick another name" = when the username is taken. | {'insertion success'}  When success  {'error with user registration'}  When failed | {  "UserName":  "FirstName":  "LastName":  "Address":  "City":  "Phone\_No":  "Cellular":  "CreditCard\_No":  "Password":  "Country":  "Categories":[],  "Questions":[],  "Answers":[]  } | POST | Users/Register | 3 |
| * User sends his username in order to answer the questions he defined in registration, the whole answers validation process happens in the front end. | Returns the questions, answers and password that the user submitted on registration , and the front end is checking the user input and reveals the password  [  {"Question":  "Answer":  "Password":  },  ]  If succeded  Else  {"error with password retrieval  } | {"Username"} | POST | Users/RetrievePassword | 4 |
| * Using get because there's no need to deliever information to the server. | Returns all categories exists in the database  ['Category1','Category2'] | - | GET | Users/Categories | 5 |
| * Using get because there's no need to deliever information to the server. | Returns all directors exists in the database  [Director1, Director2] | - | GET | **/Users/GetAllDirectors** |  |
| Movies are related to the server, the client isn't aware of it.  Using GET because there is no information to deliver to the server or any wish to change any information. | Return the newest movies:  [  {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  },  ] | - | GET | Users/TopNewestMovies | 6 |
| Post method because wer'e passing the category as parameter.  We didn't use this method eventually because we did the filtering by category on the client side. It is much more logical that way because we already get all the movies from the server so its redundant to ask the server again for movies by category. | All movies of the chosen category.  [  {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  },  ] | {"Category":  } | POST | Users/MoviesByCategory | 7 |
| Post method because wer'e passing the username as parameter. | suggested movies for the specific user.  [  {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  },  ] | {"UserName":  } | POST | Users/SuggestedMovies | 8 |
| POST because we pass the movie name to search as parameter.  If movie not fount return movie not found  Didn't use it.  Same as MoviesByCategory | {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  } | {"Name":  } | POST | Users/MovieByName | 9 |
| POST because we pass the director name to search as parameter.  If movie not fount return movie not found  Didn't use it.  Same as MoviesByCategory | {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  } | {"Director":  } | POST | Users/MoviesByDirector | 10 |
| POST because we pass the username as  parameter.  ירד בהקלות -ממומש רק בצד שרת | Returns the user previous purchase history.  [  {"MovieName":  "orderDate":  "totalPrice"  },  ] | {"UserName":} | POST | Users/PurchaseHistory | 11 |
| "purchase success " +orderNumber+  orderProperties  Else  "purchase failed"  ירד בהקלות -ממומש רק בצד שרת | {"cart purchase successful"}  If success  Else {"error with purchasing a cart"} | {  "UserName":  "Movies":[  {"MovieName":  "Price":  "deliveryDate":},  {"MovieName":  Price":  "deliveryDate":  }]  }  הערה :  לשם הכנסת תאריך כמו למשל 11\06\2017  נא להכניס  20170611 | POST | Users/PurchaseCart | 12 |
| "purchase success " +orderNumber+  orderProperties  Else  "purchase failed"  ירד בהקלות -ממומש רק בצד שרת | {"single purchase successful"}  If success  Else {"error with single purchase"} | {  "UserName": MovieName":  TotalPrice":  "deliveryDate":  }  הערה :  לשם הכנסת תאריך כמו4למשל 11\06\2017  נא להכניס  20170611 | POST | Users/PurchaseSingle | 13 |
| Login needs to pass parameters, thus we'll use POST.  True=success  False=failed  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | {"login success "}  If success  Else  {"error with admin login"} | {"UserName":  "Password":  } | POST | Admins/  AdminsLogin | 14 |
| No need for parameters , no change of the data base , thus we'll use GET.  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | All registered customers.  [  {  "UserName":  "FirstName":  "LastName":  "Address":  "City":  "Phone\_No":  "Cellular":  "CreditCard\_No":  "Password":  "Country":  "Categories":[],  "Questions":[],  "Answers":[]  }  ] | - | GET | Admins/  getCustomers | 15 |
| No need for parameters , no change of the data base , thus we'll use GET.  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | All movies exist in the database.  [  {  "Name":  "Director":  "Description":  "Category":  "Year":  "OrdersCounter":  "Image":  "Date":  "Price":  },  ] | - | GET | Admins/  GetAllMovies | 16 |
| No need for parameters , no change of the data base , thus we'll use GET.  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | All reports exist in the database.  [  {UserName:  MovieName:  Time:  deliveryDate:  }  ] | - | GET | Admins/  getReports | 17 |
| method needs to pass parameters, thus we'll use POST.  True=success  False=failed  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | {'insertion succesful'}  In success  Else  {'error with movie insertion'} | {"Name":  "Director":  "Description":  "Category":  "Year":  "Image":  } | POST | Admins/  InsertMovie | 18 |
| We use delete because we need to delete a resource.  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | {'deletion succesful'}  In success  Else  {'error with movie deletion'} | {"MovieName":  } | POST | Admins/  deleteMovie | 19 |
| method needs to pass parameters, thus we'll use POST.  True=success  False=failed  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | {'insertion succesful'}  In success  Else  {'error with user insertion'} | {  "UserName":  "FirstName":  "LastName":  "Address":  "City":  "Phone\_No":  "Cellular":  "CreditCard\_No":  "Password":  "Country":  "Categories":[],  "Questions":[],  "Answers":[]  } | POST | Admins/  InsertUser | 20 |
| We use delete because we need to delete a resource.  לא מימשנו את הבונוס בצד הלקוח. ממומש רק בצד שרת | {'deletion succesful'}  In success  Else  {'error with user deletion'} | {"UserName":  } | POST | Admins/  DeleteUser | 21 |
| We added this method for the cookie-token implementation. post because the client sends his token that stored in the local storage (need to be secured like password) and the server verifies by comparing to the token that saved in the DB | {Token:token in DB} | {Token: cookie} | POST | **/Users/SendCookie** |  |