**סביבות פיתוח באינטרנט – עבודת בית 3 – חלק א'**

**הסבר:**

השתמשנו בפורט 3100

Note: When returning a movie we mean:

Movie

{

MovieName: string,

Year: int,

Description: string,

Director: string,

Production: string,

PublishDate: date

Categories[]: string,

PicturePath: string,

Price: int

}

MovieAdmin

{

MovieName: string,

Year: int,

Description: string,

Director: string,

Production: string,

PublishDate: date

Categories[]: string,

PicturePath: string,

Price: int,

Amount: int

}

Order{

OrderID,

UserName,

OrderDate,

ShipmentDate,

Currency,

TotalAmount,

Movies[{MovieName: string, Amount: int}]

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Method Name | HTTP Method | Parameters | Returns | Explanation |
| 1 | /home | GET | - | Returns 5 hot movies:  Movies[] | Movies are related to the seller, not to the client. Using GET because there is no information to deliver to the server or any wish to change any information. |
| 2 | /register | POST | {  "FirstName":"A", "LastName":"A",  "Address":"A",  "City":"A",  "Country":"A", "Phone":"A", "Cellular":"A", "Mail":"A", "CreditCardNumber":"123", "GrandMother":"a",  "Teacher":"a", "UserName":"A", "Password":"A",  "Categories":[ "Action","Comedy" ]  } | Bool | Register cannot pass in GET. TRUE=success and client was added, FALSE= fail |
| 3 | /retrivePassword | POST | {  "UserName":"a", "Teacher":"a", "GrandMother":"a", "Address":"a”  } | {Password: string} | retrievePassword cannot pass in GET. Returns the password of the user if all his answers are correct, otherwise empty String. |
| 4 | /login | POST | {  "UserName":"a", "Password":"a"  } | Bool | Login cannot pass in GET  True= success  False=fail |
| 5 | /logged | GET | - | Return all new movies (from last month only): Movies[] | Movies are related to the seller, not to the client. Using GET because there is no information to deliver to the server or any wish to change any information. |
| 6 | /showAllMovies | GET | - | Returns all movies:  Movies[] | Movies are related to the seller, not to the client. Using GET because there is no information to deliver to the server or any wish to change any information. If client wants to filter the movies by category, it will be on Client side. If client wants to get information about a specific movie from the list, it will be on Client side. |
| 7 | /recommendedMovies | GET | /recommendedMovies/a | Returns top five recommended movies to the specific user. | Using GET because only one parameter is being transferred.  Method happens behind the scenes. Server needed in order to get recommendations for the specific user. |
| 8 | /search | GET | /search?category=MovieName&value=a  /search?category=Year&value=a  /search?category=Production&value=a  /search?category=Price&value=a | Returns all movies that answer the search:  Movies[] | Movies are related to the seller, not to the client. Using GET because parameters are not important enough to not be presented on URI. There is no any wish to change any information on Server. If client wants to get information about a specific movie from the list, it will be on Client side. |
| 9 | /showMyLastOrders | GET | /showMyLastOrders /a | Returns all last orders of the specific orders. | Using GET because only one parameter is being transferred.  Server needed in order to get orders of the specific user. |
| 10 | /getExchange | GET | /getExchange?Price=1&From=ILS&To=USD | {Price: float}  **הערה חשובה:**  **מוגבל ל-900 שימושים, דרוש אינטרנט!** | Server gets current exchange rate and calculates the new price accordingly. Using GET because parameters are not important enough to not be presented on URI. There is no any wish to change any information on Server. |
| 11 | /buyCart | PUT | {  "UserName":"a",  "Currency":"USD", "ShipmentDate":"2017-06-11", "TotalAmount":"120", "MoviesAndAmounts":[ {"MovieName":"A", "MovieAmount":"1"} , {"MovieName":"B","MovieAmount":"1"}]  } | |  | | --- | | {  "OrderID": 25,  "ClientID": 3,  "OrderDate": "2017-06-04",  "ShipmentDate": "2017-06-11",  "Currency": "USD",  "TotalAmount": "600",  "Movies": [ {  "MovieName":  "Wonder Women",  "Amount": "4" }, {  "MovieName": "saw",  "Amount": "1" } ] } | | If all movies in cart are available in stock, the server updates movies table, updates orders table, updates moviesInOrder table and returns the Order details.  Otherwise, update nothing and return empty order with error in its content. |
| 12 | /showAllOrders | GET | - | List of all orders in system:  Orders[] | Orders are related to the seller, not to the client. Using GET because there is no information to deliver to the server or any wish to change any information. If admin wants to filter the orders by category, it will be on Client side. |
| 13 | /addMovie | POST | |  | | --- | | { "MovieName":"Aladin", "DirectorFirstName":  "moshe", "DirectorLastName":  "moshe", "Year":"1997", "ProductionName":  "prod2", "Description":"a", "PicturePath":"a", "PublishDate":  "2000-05-30", "Price":"123", "Amount":"42", "Categories":[  "Comedy","Animation"] } | | Bool | Using POST because method creates new resource.  TRUE= success and movie was added  FALSE= fail |
| 14 | /deleteMovie | DELETE | {  "MovieName":"A"  } | Bool | Using DELETE because method deletes a resource.  TRUE= success and movie was deleted  FALSE= fail |
| 15 | /addClient | POST | {  "FirstName":"A", "LastName":"A",  "Address":"A",  "City":"A",  "Country":"A", "Phone":"A", "Cellular":"A", "Mail":"A", "CreditCardNumber":"123", "GrandMother":"a",  "Teacher":"a", "UserName":"A", "Password":"A",  "Categories":[ "Action","Comedy" ],  “IsAdmin”:”1”  } | Bool | addClient cannot pass in GET. TRUE=success and client was added, FALSE= fail |
| 16 | /deleteClient | DELETE | {  "UserName":"A"  } | Bool | Using DELETE because method deletes a resource.  TRUE= success and client was deleted  FALSE= fail |
| 17 | /listMoviesAdmin | GET | - | MovieAdmin[] | amounts are related to the seller, not to the client. Using GET because there is no information to deliver to the server or any wish to change any information. If admin wants to filter the stock amounts by amount, it will be on Client side. |
| 18 | /updateAmount | PUT | {  “MovieName”:”a”,  “StokAmount”:1  } | Bool | Using PUT because method updates a resource.  TRUE= success and amount was updated,  FALSE= fail |