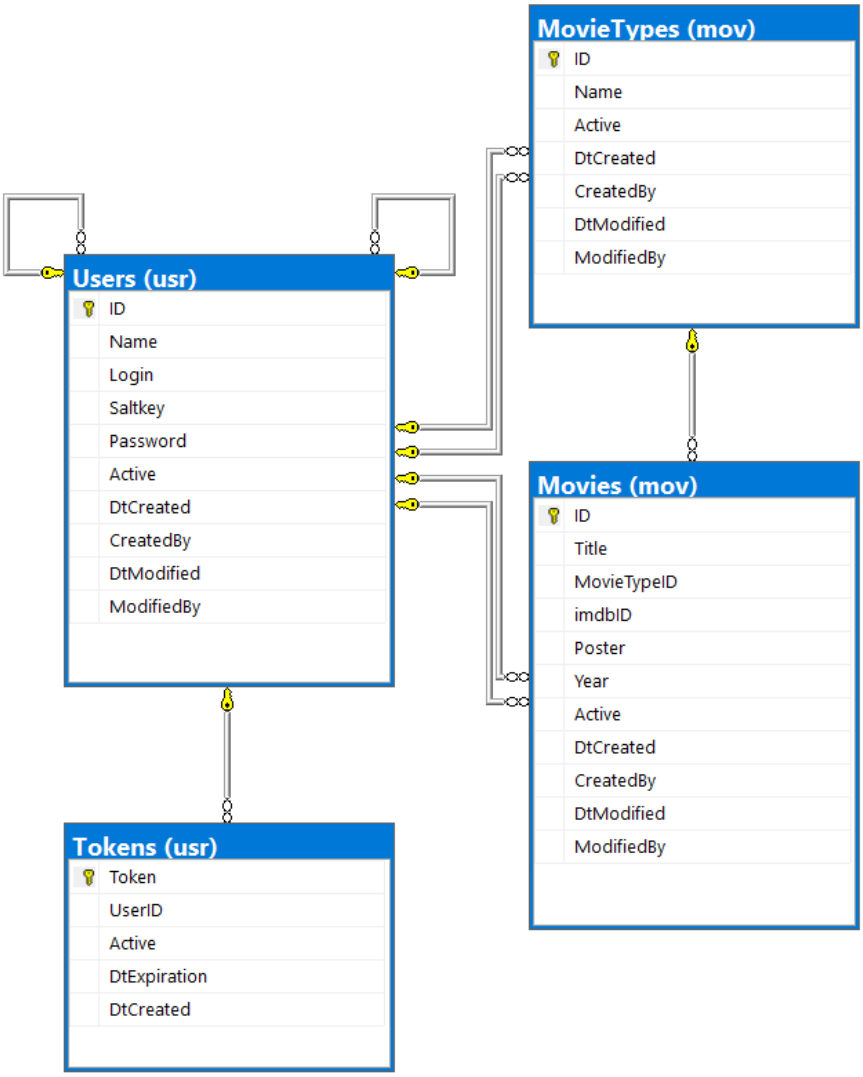
Renato Matos
renato.matos79@gmail.com
+55 11 95494 0703
Skype: renato.matos.79

iCreate Test
All answers for this test are available at Github
https://github.com/renatomatos79/movies.git

Question 1:
For this question I've done:
- Designed the Database Model using SQL Server;
- Imported some sample data to the database (movie list);
- Created a backend API using .NET Core with some endpoints to:
Authenticate and Authorize the user persisting an API Token which must be sent on header
- I also organized the backend into these simlple layers, such as: Domain, Service and Api. But I also added Log Control, IoC and Api Token Validation
- As a result I could test my endpoints using Postman

Database Model



Samples for my test

SQLQuery4.sql - loc...vieapi (movie (54))*

```sql
select mt.Name as MovieTypeName, mv.*
  from mov.Movies mv
  inner join mov.MovieTypes mt
  on mt.ID = mv.MovieTypeID
```

233 %

Results | Messages

	MovieTypeName	ID	Title	MovieTypeID	imdbID	Poster	Year	Active	DtCreated	CreatedBy
1	Not Classified	1	The Godfather: Part II (1974	4	E05D9782-A2E3-4D17-A4BE-4B94B4F95C22	ni	2014	1	2019-08-07 00:54:44.507	1
2	Not Classified	2	Pulp Fiction (1994	4	E1C9B3F7-9E98-42C4-AE6B-907A96CD0AF3	ni	2014	1	2019-08-07 00:54:44.507	1
3	Not Classified	3	The Good the Bad and the Ugly (1966	4	55AFCE10-8E08-4975-A388-6CCA605D4B3C	ni	2014	1	2019-08-07 00:54:44.507	1
4	Not Classified	4	The Dark Knight (2008	4	C347D238-FF9E-4BF7-B11E-F0E16E85C020	ni	2014	1	2019-08-07 00:54:44.507	1
5	Not Classified	5	12 Angry Men (1957	4	F8567F3E-0941-4920-9155-B7F7A3E602E4	ni	2014	1	2019-08-07 00:54:44.507	1
6	Not Classified	6	Schindlers List (1993	4	EB400BC6-E74C-40E2-A705-C399C7679761	ni	2014	1	2019-08-07 00:54:44.507	1
7	Not Classified	7	The Lord of the Rings: The Return of the King (20...	4	CA7415E5-46BD-4AB7-BB0A-95ABE803E78E	ni	2014	1	2019-08-07 00:54:44.507	1
8	Not Classified	8	Fight Club (1999	4	9BA9D32B-E326-491A-8758-F28FE54DADD5	ni	2014	1	2019-08-07 00:54:44.507	1
9	Not Classified	9	Star Wars: Episode V - The Empire Strikes Back (...	4	831F5F1E-ABCE-48D1-973F-625C3FB9972D	ni	2014	1	2019-08-07 00:54:44.507	1
10	Not Classified	10	The Lord of the Rings: The Fellowship of the Ring...	4	FEB8677F-99B3-4F1C-B30D-AA42AE2B23E8	ni	2014	1	2019-08-07 00:54:44.507	1
11	Not Classified	11	One Flew Over the Cuckoos Nest (1975	4	242D0F29-71DE-451A-9CF9-92FBC130E618	ni	2014	1	2019-08-07 00:54:44.507	1
12	Not Classified	12	Goodfellas (1990	4	87F3F580-3F96-4D40-94A2-E748C92D3850	ni	2014	1	2019-08-07 00:54:44.507	1
13	Not Classified	13	Seven Samurai (1954	4	73B0B4A0-F54B-400F-B596-30BA083F62E3	ni	2014	1	2019-08-07 00:54:44.507	1
14	Not Classified	14	Inception (2010	4	939A0CB4-C336-4893-8C57-5721F6F4C27E	ni	2014	1	2019-08-07 00:54:44.507	1
15	Not Classified	15	Star Wars (1977	4	6AF277E0-96BE-4F62-96A0-A8530BB7FA03	ni	2014	1	2019-08-07 00:54:44.507	1
16	Not Classified	16	Forrest Gump (1994	4	ED488808-B9CB-436B-AFE4-F79BA0B0B764	ni	2014	1	2019-08-07 00:54:44.507	1
17	Not Classified	17	The Matrix (1999	4	90993781-0C9C-4E58-9B84-C96C50EE172A	ni	2014	1	2019-08-07 00:54:44.507	1
18	Not Classified	18	The Lord of the Rings: The Two Towers (2002	4	38DFDC36-502E-48BA-BCFE-7EC07CDC34C3	ni	2014	1	2019-08-07 00:54:44.507	1
19	Not Classified	19	City of God (2002	4	422A3FE8-35BD-4EC8-A192-0019A64E4968	ni	2014	1	2019-08-07 00:54:44.507	1
20	Not Classified	20	The Silence of the Lambs (1991	4	99302F2C-2083-4DA5-A680-31AC210F147E	ni	2014	1	2019-08-07 00:54:44.507	1
21	Not Classified	21	Se7en (1995	4	F39340F1-D56E-4FAE-BF93-D5A9475E4E4D	ni	2014	1	2019-08-07 00:54:44.507	1
22	Not Classified	22	Once Upon a Time in the West (1968	4	AE5E056E-2E90-4239-9422-70F407A1A347	ni	2014	1	2019-08-07 00:54:44.507	1
23	Not Classified	23	Casablanca (1942	4	4DA206DF-4FDE-47C0-B7EE-BACEC17E0A...	ni	2014	1	2019-08-07 00:54:44.507	1
24	Not Classified	24	The Usual Suspects (1995	4	9FF6E39A-AFE4-45B4-97C6-0E93E67CDE97	ni	2014	1	2019-08-07 00:54:44.507	1
25	Not Classified	25	Raiders of the Lost Ark (1981	4	FDB16EFB-CB8D-4B1B-96F1-078B503A6430	ni	2014	1	2019-08-07 00:54:44.507	1
26	Not Classified	26	Rear Window (1954	4	5807C255-7E8E-43F8-93C1-240A73FECB9D	ni	2014	1	2019-08-07 00:54:44.507	1
27	Not Classified	27	Its a Wonderful Life (1946	4	8B3D91C6-B364-4769-9903-92F23915D36A	ni	2014	1	2019-08-07 00:54:44.507	1

Authenticating the User

POST https://localhost:44385/api/token

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "Login": "renato.matos79@gmail.com",
    "Password": "P@ssw0rd"
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
{
    "token": "3971be17-690e-49c1-b632-354e3248fa2f",
    "userName": "Renato Matos",
    "dtExpiration": "2019-08-08 05:50:55 AM",
    "message": "",
    "success": true
}
```

Simulating a wrong password

POST https://localhost:44385/api/token Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "Login": "renato.matos79@gmail.com",
    "Password": "P@ssw0rd3"
}
```

Body Cookies Headers (5) Test Results Status: 400 Bad Request Time: 277 ms

Pretty Raw Preview JSON

```
{
    "token": null,
    "userName": null,
    "dtExpiration": null,
    "message": "Invalid Login or Password!",
    "success": false
}
```

Getting the movie types

GET https://localhost:44385/api/movie-types

Authorization | Headers (1) | Body | Pre-request Script | Tests

Key	Value
AccessToken	522f8804-174f-4cd6-a4af-7cde1a860c15
New key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | JSON


```
[
    {
        "id": 1,
        "name": "Romane",
        "active": true,
        "message": "",
        "success": true
    },
    {
        "id": 2,
        "name": "Action",
        "active": true,
        "message": "",
        "success": true
    },
    {
        "id": 3,
        "name": "Adventure",
        "active": true,
        "message": "",
        "success": true
    },
    {
        "id": 4,
        "name": "Not Classified",
        "active": true,
        "message": "",
        "success": true
    }
]
```


Trying to get the list using an invalid Token

GET https://localhost:44385/api/movie-types

Authorization | Headers (1) | Body | Pre-request Script | Tests

Key	Value
AccessToken	522f8804-174f-4cd6-a4af-7cde1a860c15 XPTO
New key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | JSON


```
{
    "ClassName": "System.UnauthorizedAccessException",
    "Message": "Attempted to perform an unauthorized operation.",
    "Data": null,
    "InnerException": null,
    "HelpURL": null,
    "StackTraceString": null,
    "RemoteStackTraceString": null,
    "RemoteStackIndex": 0,
    "ExceptionMethod": null,
    "HResult": -2147024891,
    "Source": null,
    "WatsonBuckets": null
}
```


Searching movies on page 2

GET https://localhost:44385/api/movies/search/back/page/1

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

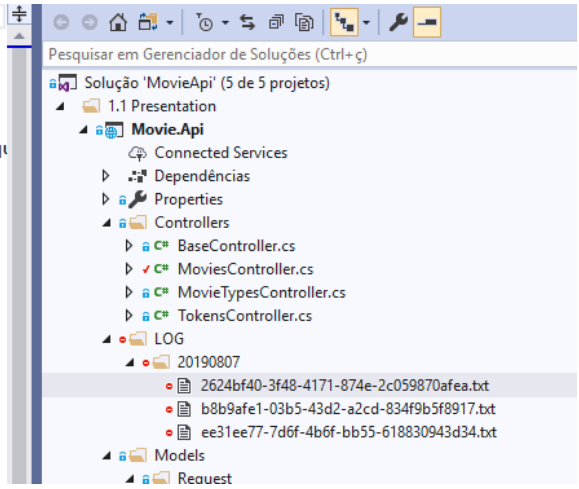
Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON


```
{
    "page": 1,
    "perPage": 10,
    "total": 2,
    "totalPages": 1,
    "data": [
        {
            "poster": "ni",
            "title": "Back to the Future (1985",
            "type": "Not Classified",
            "year": 2014,
            "imdbID": "EEDE5063-EBA9-46A8-8BEC-010DC761C7E7"
        },
        {
            "poster": "ni",
            "title": "Star Wars: Episode V - The Empire Strikes Back (1980",
            "type": "Not Classified",
            "year": 2014,
            "imdbID": "831F5F1E-ABCE-48D1-973F-625C3FB9972D"
        }
    ],
    "message": "",
    "success": true
}
```


Checking the LOG file

```
=============================================
=============================================
Data: 07/08/2019 03:08:48
Message: TokensController.Authenticate
Exception: System.ApplicationException: Invalid Login or Password!
   at Movie.Api.Controllers.TokensController.Authenticate(LoginRequ
=============================================
=============================================
```



Question 2:

For this question I decided to add another class named **BaseCar** just to implement **toString** and **getMileage** as a way to make codeless the children classes:

```
4 referências
public class BaseCar : Car
{
    private string _mileage;

    3 referências
    public BaseCar(string mileage, bool isSedan, string seats) : base(isSedan, seats)
    {
        _mileage = mileage;
    }

    2 referências
    public override string getMileage()
    {
        return $"{_mileage} kmpl";
    }

    0 referências
    public override string ToString()
    {
        var name = this.GetType().Name;
        var prefixSedan = this.getIsSedan() ? "is" : "is not";

        return $"A {name} {prefixSedan} Sedan, is {getSeats()} - seater, and has a mileage of around {getMileage()}";
    }
}
```

My Children classes:

```
public class HondaCity : BaseCar
{
    1 referência
    public HondaCity(string mileage) : base(mileage, true, "4") { }

    2 referências
    public override int GetHashCode()
    {
        return 1;
    }
}
```

```
2 referências
public class InnovaCrysta : BaseCar
{
    1 referência
    public InnovaCrysta(string mileage) : base(mileage, false, "6") { }

    2 referências
    public override int GetHashCode()
    {
        return 2;
    }
}
```

```
0 referências
public class WagonR : BaseCar
{
    1 referência
    public WagonR(string mileage) : base(mileage, false, "4") { }

    2 referências
    public override int GetHashCode()
    {
        return 0;
    }
}
```

The result after two executions:

```
C:\Projetos\i-create\Question2\AbstractClass\bin\Debug\AbstractClass.exe
C:\Projetos\i-create\Question2\AbstractClass\bin\Debug\AbstractClass.exe
0
22
A WagonR is not Sedan, is 4 - seater, and has a mileage of around 22 kmpl


Press Enter to Finish
```

Question 3:
Thats the way I fixed the source codeless

```
namespace BugFix
{
    public class Person
    {
        public Person(string name)
        {
            this.Name = name;
        }
        public Person(string name, DateTime birthdate) : this(name)
        {
            this.CalcAge(birthdate);
        }

        public string Name { get; set; }
        public int? Age { get; set; }
        public void CalcAge(DateTime birthdate)
        {
            var today = DateTime.Today;
            var age = today.Year - birthdate.Year;
            if (birthdate.Date > today.AddYears(-age))
            {
                age--;
            }
            this.Age = age;
        }
    }

    public static class PersonManager
    {
        private static readonly List<Person> DB = new List<Person>();

        public static void AddPerson(string name, DateTime birthDate)
        {
            DB.Add(new Person(name, birthDate));
        }
        public static void DeletePerson(int id)
        {
            if (DB.Count >= id)
            {
                DB.RemoveAt(id);
            }
        }
        public static List<Person> People { get { return DB; } }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Adding people...");
            PersonManager.AddPerson("Renato", new DateTime(1979, 6, 23));
            PersonManager.AddPerson("Vanessa", new DateTime(1978, 5, 5));
            PersonManager.People.ToList().ForEach(p =>
            {
                Console.WriteLine($"{p.Name}, Age {p.Age}");
            });
```

```
            Console.WriteLine("Removing person at index 0...");
            PersonManager.DeletePerson(0);
            PersonManager.People.ToList().ForEach(p =>
            {
                Console.WriteLine($"{p.Name}, Age {p.Age}");
            });

            Console.ReadLine();
        }
    }
}
```

The result window after programming execution:

```
0 referências
static void Main(string[] args)
{
    Console.WriteLine("Adding people...");
    PersonManager.AddPerson("Renato", new DateTime(1979, 6, 23));
    PersonManager.AddPerson("Vanessa", new DateTime(1978, 5, 5));
    PersonManager.People.ToList().ForEach(p =>
    {
        Console.WriteLine($"{p.Name}, Age {p.Age}");
    });

    Console.WriteLine("Removing person at index 0...");
    PersonManager.DeletePerson(0);
    PersonManager.People.ToList().ForEach(p =>
    {
        Console.WriteLine($"{p.Name}, Age {p.Age}");
    });

    Console.ReadLine();
}
```

```
C:\Projetos\i-create\Question3\BugFix\bin\Debug\Bug

Adding people...
Renato, Age 40
Vanessa, Age 41
Removing person at index 0...
Vanessa, Age 41
```