İSTANBUL TECHNICAL UNIVERSITY
Faculty of Computer Science and Informatics

Information Technologies
Data query and Web Api for client server communication.

INTERNSHIP PROGRAM REPORT

Lına Fahad A Alrehaılı
150160930

29 July – 16 August / 2019

İSTANBUL TECHNICAL UNIVERSITY

Faculty of Computer Science and Informatics

INTERNSHIP PROGRAM ACTIVITY REPORT

Academic Year: 2018-2019

Period of Training: [✓ Summer] ☐Spring ☐Autumn

Student Information

Name Surname:	Lına Fahad A Alrehaılı
Student ID:	150160930
Department:	Computer Engineer
Program:	100% English
E-Mail:	Alrehailil16@itu.edu.tr
Mobile Phone:	+90 536 860 37 67
Pursuing a Double Major?	☐Yes (Faculty/Department of DM: ________________________________) [✓ No]
Graduated?	☐ Yes [✓ No]
Taking a class at Summer School?	☐Yes(Number of Classes: 0) [✓ No]

Institution Information

Company Name:	Erlab Technology
Department:	Information Technology Department
Web Address:	http://erlab.tech /
Postal Address:	Resitpasa Mahallesi, Katar Caddesi, ITU Ayazaga Kampusu, Ari 1 Teknokent Binasi, No: 2/5/1 PK: 34467 Sariyer ISTANBUL / TURKEY

Authorized Person Information

Department:	IT Department
Title:	CTO
Name Surname:	Mahmut Şamil Sağıroğlu
orporateE-Mail:	mahmut.sagiroglu@erlab.com.tr
Corporate Phone:	+ 905052645506

Internship Program Information

Location	☐Turkey
	☐Abroad
Starting Date	29.07.2019
End Date	16.08.2019
Number of Days Worked	12 days
During your internship, did you have insurance?	☐Yes, I was insured by İTÜ.
	☐Yes, I was insured by institution.
	☐No, I did my internship abroad.
	☐No.

Table of Contents

The internship report with the above given “Table of Contents” was seen and approved for the student with student ID “150160930” and name “Lına Fahad A Alrehaılı”

Authorized staff of the institution who filled in the form

Name and Surname: Mahmut Şamil Sağıroğlu
Title: Table of Contents Approval
Manager-Signature-Seal:

ERLab TEKNOLOJİ A.Ş.
İTÜ Ayazağa Yerleşkesi, Reşitpaşa Mah.Katar Cad.
Teknokent Arı 1 No:2/5/1 Sarıyer/İSTANBUL
Tel:(0212)286 63 64 Faks:(0212)286 63 65
Sarıyer Vergi Dairesi 364 030 0344
Mersis Numarası: 0364030034400018

[illegible]

(Institutional) E-mail: Mahmut.sagiroglu@erlab.com.tr
(Institutional) Phone Number: +90 505 264 5506

1. INFORMATION ABOUT THE COMPANY

ERLAB TEKNOLOJİ A.Ş. Company contribute to their customers with new technologies and add value to their workflows by innovative solutions.

They focus on media and bioinformatics where they have their core technologies. With their patented products, they provide faster and better quality encoding, transcoding, content management, video streaming and internet TV solutions for their customers with high flow of media content, and with patented Gencrobat product, they deliver highly efficient computing and analysis solutions to their customers in bioinformatics area [1].

They have been listed in Deloitte's prestigious Technology Fast 50 Turkey Program, a ranking of the fastest growing technology companies. Rankings are based on the revenue growth over five years. They also rank 30th in Turkey fastest growing companies list AllWorld Network Turkey 100 [1].

The quality policy of Erlab Technology Company [1]:

- Solution-oriented
- Investment to technology
- Innovation
- Highly skilled Employees
- Focus on costumers
- Cooperation
- Continual improvements

The company's mission is to provide their customers with projects that consist fast, innovative and value-added solutions to their needs using new technologies and integrated problem solving techniques. Their commitment to their founders, employees, associates and customers is to present our products, knowledge, experience and services to the market in the most competing way via developing our corporate and technological infrastructure continuously. They intend to grow while meeting their customers' expectations, using high-technology and the perfect solutions including new products and services [1].

The company's vision is to provide an added-value to their customers; constantly adapting to new technologies and solutions; developing novel tools and technologies to be an important player in the world markets. To be a company, that adds value to it's' customers, continuously producing and generalizing adaptive solutions, using new technologies and selling to international markets [1].

2. INTRODUCTION

During this report, I will review what I have learned during my internship at ERLAB TEKNOLOJİ Company, which lasted for 12 working days (29 July – 16 August 2019), the internship type I applied for is Information Technology, I applied for this internship because I am interested in learning the web API development, and I would like to learn more about it for my future career plans. Throughout the internship, I took the chance to use the knowledge and skills I have gained from my education time at Istanbul Technical University in the real world field, my main duty in the company was to solve and implement many task assignments provided by supervisor. What I have gained from this experience:

- View the working environment of the company.
- Participate and work as a team.
- Find my weak and strength points.
- Find a problem and solve it.
- Find a different solution from the existed solution.

Therefore, I am pleased that I had the chance to gain this knowledge and experience, as I will know what to do later in my life when I start a particular career.

3. INTERNSHIP WORK AND PROJECTS DESCRIPTION

In this section, I will describe the work that I have done during my internship, what I have learned, the company projects which I have worked on. Therefore, I was engaged in solving some issues distributed to multiple of projects, and I was able to solve those assigned issues.

3.1 Query data with web service (C#)

In the first day, I had to learn the steps to query a data with a web service with C# programming language. Therefore, I did some research to understand everything related to data query and web services before starting to work.

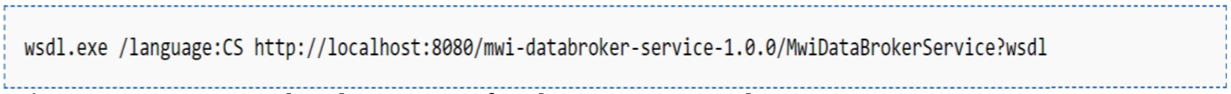
Afterwards, my assignment was related to parsing a query info about a musical work. By taking this info and designing an xml request for a service. The service usage will be researched and then the response will be processed and parsed. The CIS-Net page will show the parsed info's. The 'CIS-Net controlled by FastTrack' gives a worldwide and interconnected system of documentation instruments relating to the distinguishing proof of musicalworks, audio-visual works, invested individuals and understandings data [2].

To solve this assignment I had to read the CIS-Net documentation at first [3]. Therefore, I generated a service-proxy stub with the WSDL. The first generation is the client proxy for the locator service by the following line of code in the terminal of visual studio application as shown in Figure 1.

```
wsdl.exe /language:CS http://localhost:8080/locator-service-1.0.0/LocatorService?wsdl
```

Figure 1: *Generate the client proxy for the locator service*

Next, I have generated the client proxy for the MwiDataBroker Service as shown in Figure 2.



```
wsdl.exe /language:CS http://localhost:8080/mwi-databroker-service-1.0.0/MwiDataBrokerService?wsdl
```

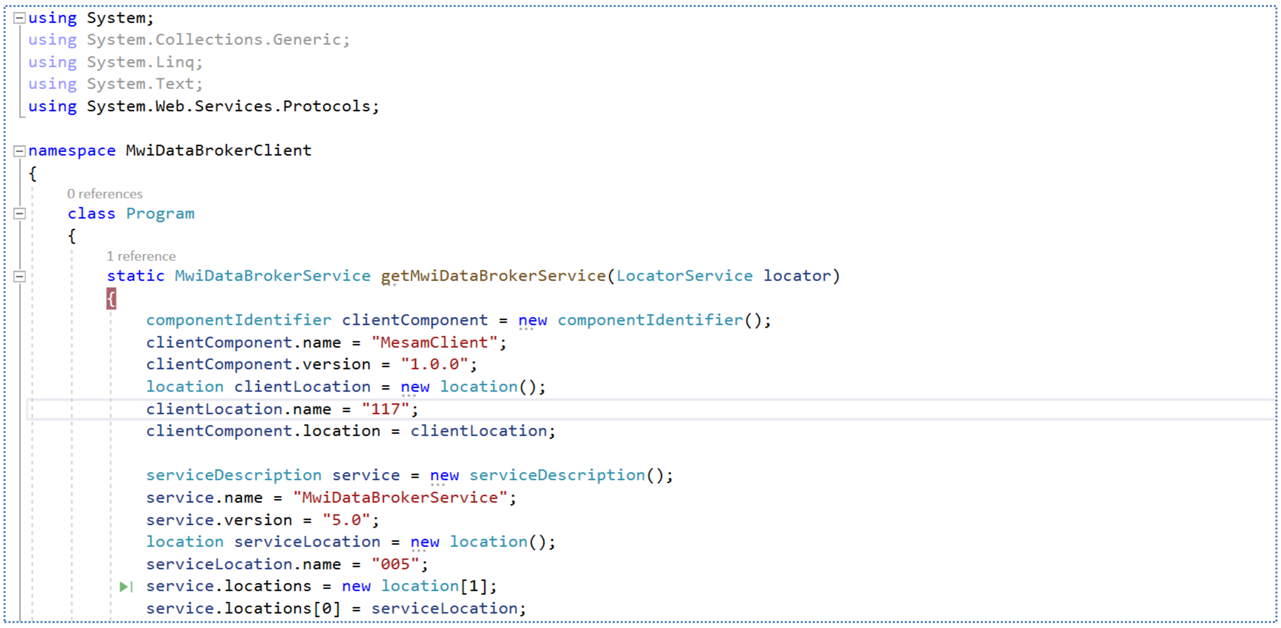
Figure 2: *Generate the client proxy for the MwiDataBroker service*

Later on, I had to add my user information by creating a soap extension in a class called (*ClientIdentifierExtension.cs*) so that I can add the needed information into the soap header when the service is called. Furthermore, we need to declare this extension into the configuration file of the application as well.

The next step is to write the client code. Therefore, I declared the client name, version, location and all the needed information as shown in Figure 3. I declared the service that I need to use which is (*MwiDataBroker*) Service. Therefore, the first step of declaring the application is finished. The code bellow assumes that the client is the MesamClient 1.0.0 deployed at SACEM (117) and it is calling the MwidataBrokerService 5.0 from SACEM. At the end of this method, there are two-line of code as well as written below:

```
serviceDistribution[] services = locator.findServices(clientComponent, service);

return getMwiDataBrokerService(services[0].urls[0]);
```

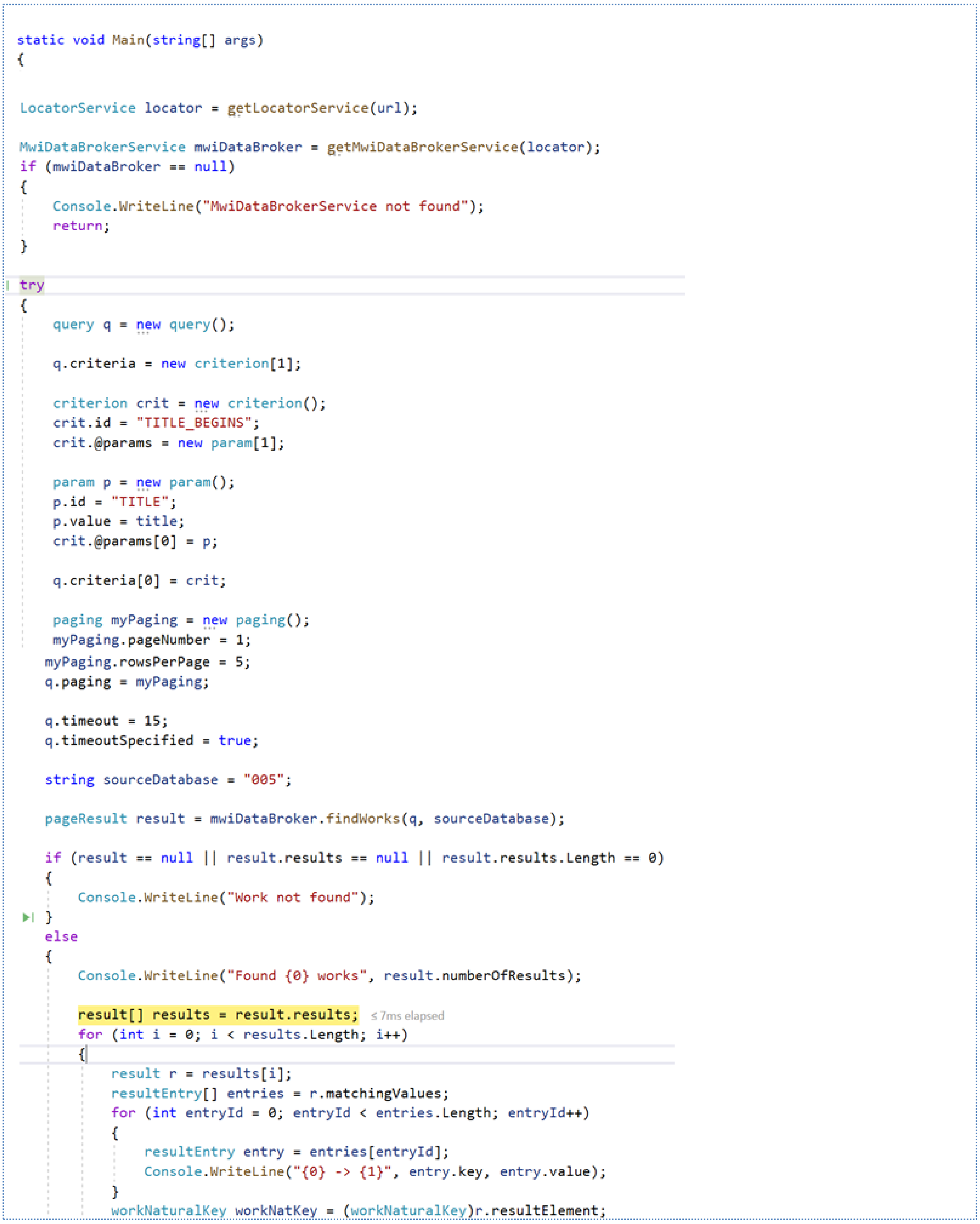
The first line of code will find the service of the given client, and the second line will return this service for the function.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.Services.Protocols;

namespace MwiDataBrokerClient
{
    0 references
    class Program
    {
        1 reference
        static MwiDataBrokerService getMwiDataBrokerService(LocatorService locator)
        {
            componentIdentifier clientComponent = new componentIdentifier();
            clientComponent.name = "MesamClient";
            clientComponent.version = "1.0.0";
            location clientLocation = new location();
            clientLocation.name = "117";
            clientComponent.location = clientLocation;

            serviceDescription service = new serviceDescription();
            service.name = "MwiDataBrokerService";
            service.version = "5.0";
            location serviceLocation = new location();
            serviceLocation.name = "005";
            service.locations = new location[1];
            service.locations[0] = serviceLocation;
```

Figure 3: *Client-server side code*

Later on, in the main class I called the previous method to find and use the service as shown in Figure 4.

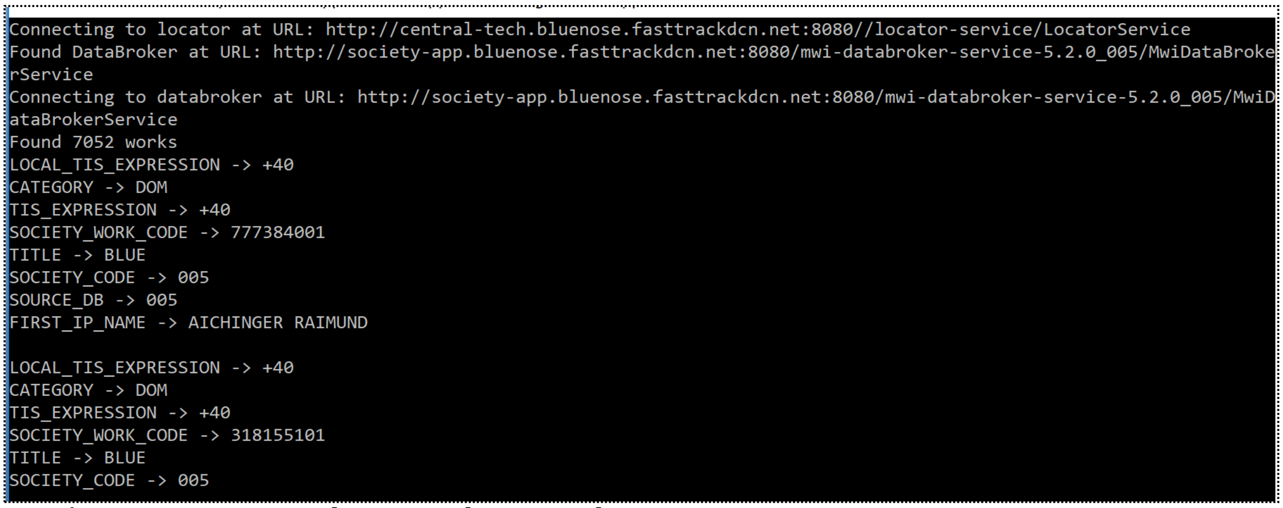


```
static void Main(string[] args)
{

 LocatorService locator = getLocatorService(url);

 MwiDataBrokerService mwiDataBroker = getMwiDataBrokerService(locator);
 if (mwiDataBroker == null)
 {
     Console.WriteLine("MwiDataBrokerService not found");
     return;
 }

 try
 {
     query q = new query();

     q.criteria = new criterion[1];

     criterion crit = new criterion();
     crit.id = "TITLE_BEGINS";
     crit.@params = new param[1];

     param p = new param();
     p.id = "TITLE";
     p.value = title;
     crit.@params[0] = p;

     q.criteria[0] = crit;

     paging myPaging = new paging();
     myPaging.pageNumber = 1;
    myPaging.rowsPerPage = 5;
    q.paging = myPaging;

    q.timeout = 15;
    q.timeoutSpecified = true;

    string sourceDatabase = "005";

    pageResult result = mwiDataBroker.findWorks(q, sourceDatabase);

    if (result == null || result.results == null || result.results.Length == 0)
    {
        Console.WriteLine("Work not found");
    }
    else
    {
        Console.WriteLine("Found {0} works", result.numberOfResults);

        result[] results = result.results;  ≤ 7ms elapsed
        for (int i = 0; i < results.Length; i++)
        {
            result r = results[i];
            resultEntry[] entries = r.matchingValues;
            for (int entryId = 0; entryId < entries.Length; entryId++)
            {
                resultEntry entry = entries[entryId];
                Console.WriteLine("{0} -> {1}", entry.key, entry.value);
            }
            workNaturalKey workNatKey = (workNaturalKey)r.resultElement;
```

Figure 4: *Data query in main-class code*

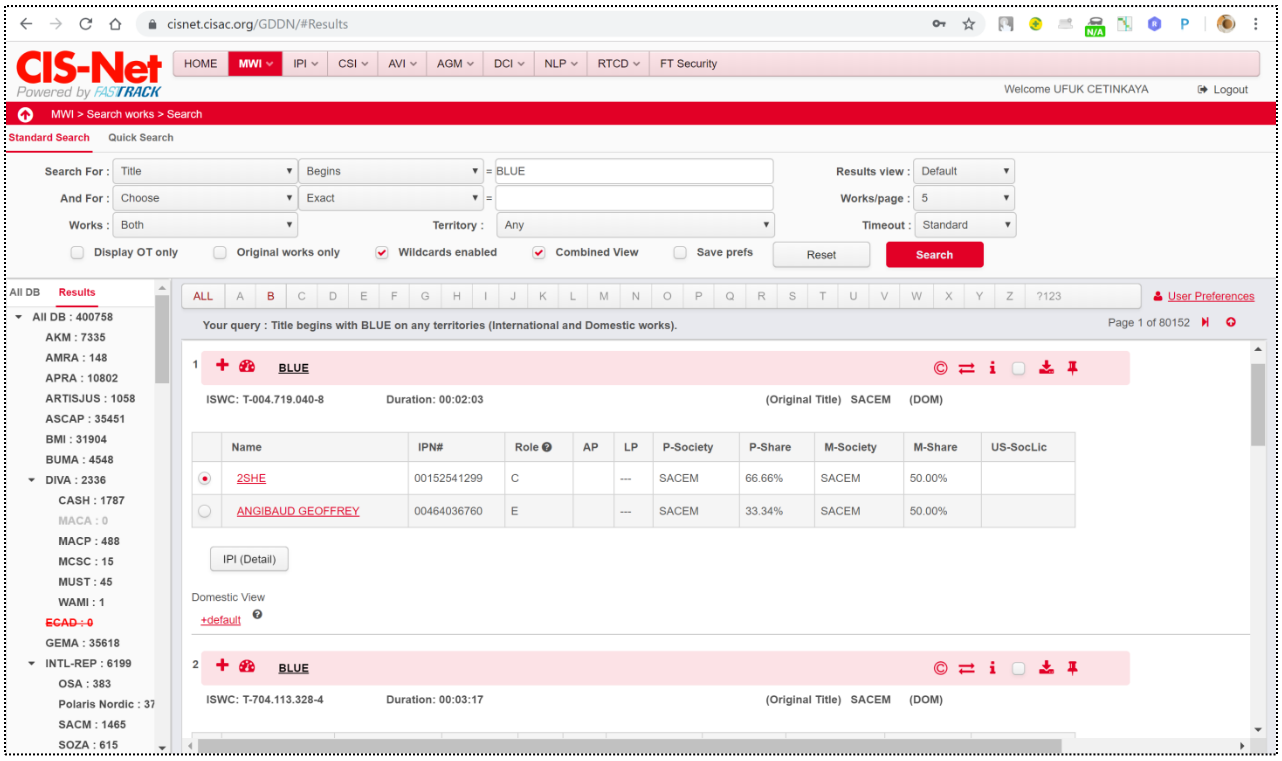
The explanation of the above code shows that I defined the locator of the required service, later on I called the previous function to use this service. I created a query instance to start queering the data we need, and then we created a query criterion from the given instance.

Next, I added the specified parameter to the query, and I did paging throw query result, paging throw query result will give us a query result in a small subset of a selected number of pages. After that, we query to find the musical work by the built-in function "*findWroks*". The result will be searched one by one in the for-loop. The final query result will be printed in the console terminal and posted in CIS-Net page as shown in figure-5 and figure-6.



```
Connecting to locator at URL: http://central-tech.bluenose.fasttrackdcn.net:8080//locator-service/LocatorService
Found DataBroker at URL: http://society-app.bluenose.fasttrackdcn.net:8080/mwi-databroker-service-5.2.0_005/MwiDataBroke
rService
Connecting to databroker at URL: http://society-app.bluenose.fasttrackdcn.net:8080/mwi-databroker-service-5.2.0_005/MwiD
ataBrokerService
Found 7052 works
LOCAL_TIS_EXPRESSION -> +40
CATEGORY -> DOM
TIS_EXPRESSION -> +40
SOCIETY_WORK_CODE -> 777384001
TITLE -> BLUE
SOCIETY_CODE -> 005
SOURCE_DB -> 005
FIRST_IP_NAME -> AICHINGER RAIMUND

LOCAL_TIS_EXPRESSION -> +40
CATEGORY -> DOM
TIS_EXPRESSION -> +40
SOCIETY_WORK_CODE -> 318155101
TITLE -> BLUE
SOCIETY_CODE -> 005
```

Figure 5: *Query result in console terminal*



Search For : Title | Begins | = BLUE
And For : Choose | Exact | =
Works : Both   Territory : Any
Results view : Default
Works/page : 5
Timeout : Standard
Display OT only ☐ Original works only ☐ Wildcards enabled ✓ Combined View ✓ Save prefs ☐

All DB : 400758
AKM : 7335
AMRA : 148
APRA : 10802
ARTISJUS : 1058
ASCAP : 35451
BMI : 31904
BUMA : 4548
DIVA : 2336
CASH : 1787
MACA : 0
MACP : 488
MCSC : 15
MUST : 45
WAMI : 1
~~ECAD : 0~~
GEMA : 35618
INTL-REP : 6199
OSA : 383
Polaris Nordic : 37
SACM : 1465
SOZA : 615

Your query : Title begins with BLUE on any territories (International and Domestic works).

Page 1 of 80152

1 BLUE

ISWC: T-004.719.040-8 Duration: 00:02:03 (Original Title) SACEM (DOM)

	Name	IPN#	Role	AP	LP	P-Society	P-Share	M-Society	M-Share	US-SocLic
●	2SHE	00152541299	C		---	SACEM	66.66%	SACEM	50.00%	
○	ANGIBAUD GEOFFREY	00464036760	E		---	SACEM	33.34%	SACEM	50.00%	

2 BLUE

ISWC: T-704.113.328-4 Duration: 00:03:17 (Original Title) SACEM (DOM)

Figure 6: *Query result in CIS-Net page*

3.2 Localhost client with web API (C#)

The next assignment was to create a web API application in my localhost server, the web application should show the data in an xml format and the data can be filtered by an id. Therefore, in the model of the application I added a class called “Employee.cs”, this class defines all the needed data of the application as shown in Figure-7.

```
namespace Employee.Models
{
    public class Employee
    {
        public int Country
        {
            get;
            set;
        }
        public string Id
        {
            get;
            set;
        }
        public string Job
        {
            get;
            set;
        }
        public string Name
        {
            get;
            set;
        }
    }
}
```

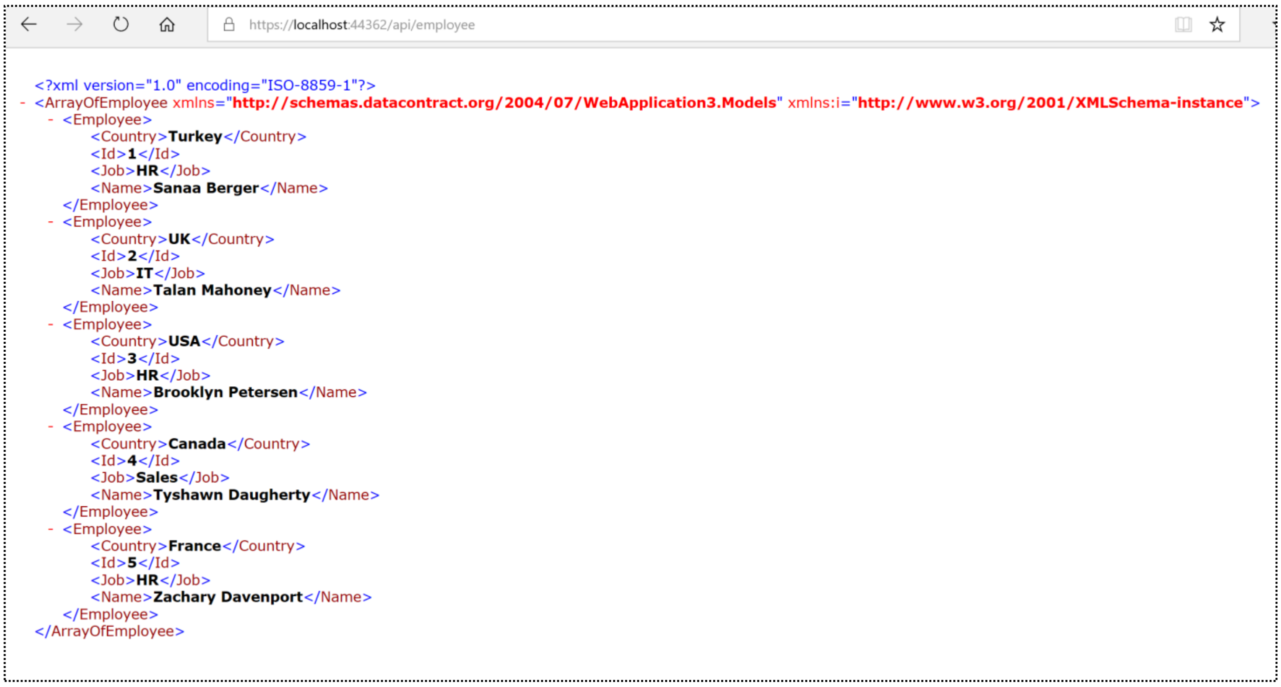
Figure 7: *Model class (Employee.cs)*

Furthermore, in the controller of the application I added a class called “EmployeeController.cs” as shown in Figure-8, this class is derived from the base ApiController class and it handles any URL requests of the application, all the methods within this class act as an action to return the appropriate response.

```
namespace Employee.Controllers
{
    public class EmployeeController: ApiController
    {
        IList < Employee > employees = new List < Employee > ()
        {
            new Employee()
                {
                    Id   = 1, Name   = "Sanaa Berger", Job   = "HR ", Country   = "Turkey"
                },
                new Employee()
                {
                    Id   = 2, Name   = "Talan Mahoney ", Job   = "IT", Country   = "UK"
                },
                ….
                },
        };
        public IList < Employee > GetEmployees()  //Return all employees
        {
            return employees;
        }

        public Employee GetEmployeeById(int id)  //Return employee by id
        {
            var employee = employees.FirstOrDefault(e => e.EmployeeId == id);
            if (employee == null)
            {
                throw new HttpResponseException(Request.CreateResponse(HttpStatusCode.NotFound));
            }
            return employee;
        }
    }
}
```

Figure 8: *Controller class (EmployeeController.cs)*

To show the data in the localhost we write api/employee after it as shown in Figure-9, and we can write api/employee/#id as well to filter and show the specified data with the id number.


https://localhost:44362/api/employee

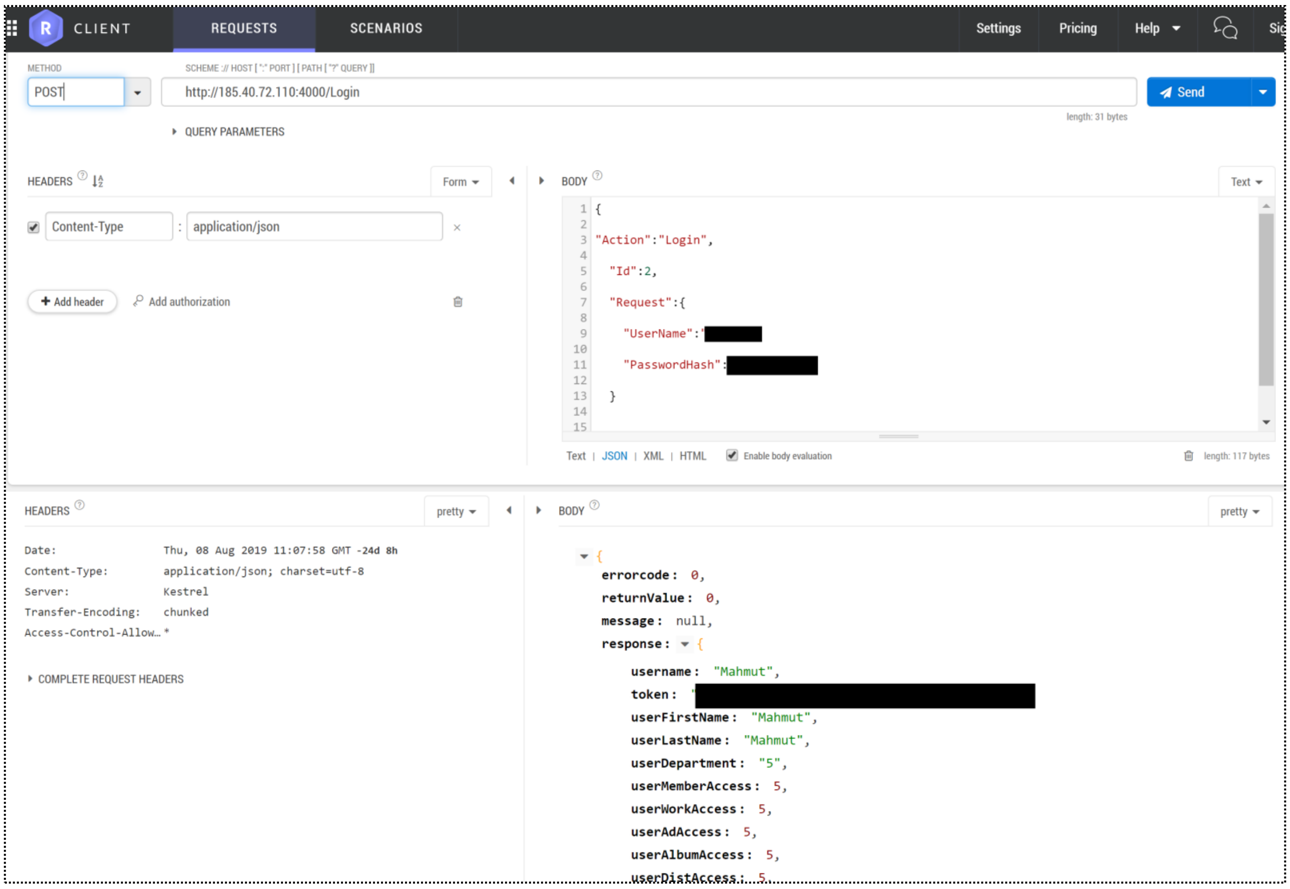
```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <ArrayOfEmployee xmlns="http://schemas.datacontract.org/2004/07/WebApplication3.Models" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  - <Employee>
      <Country>Turkey</Country>
      <Id>1</Id>
      <Job>HR</Job>
      <Name>Sanaa Berger</Name>
    </Employee>
  - <Employee>
      <Country>UK</Country>
      <Id>2</Id>
      <Job>IT</Job>
      <Name>Talan Mahoney</Name>
    </Employee>
  - <Employee>
      <Country>USA</Country>
      <Id>3</Id>
      <Job>HR</Job>
      <Name>Brooklyn Petersen</Name>
    </Employee>
  - <Employee>
      <Country>Canada</Country>
      <Id>4</Id>
      <Job>Sales</Job>
      <Name>Tyshawn Daugherty</Name>
    </Employee>
  - <Employee>
      <Country>France</Country>
      <Id>5</Id>
      <Job>HR</Job>
      <Name>Zachary Davenport</Name>
    </Employee>
  </ArrayOfEmployee>
```

Figure 9: *Run localhost client with web API*

3.3 Web host client with web API (C#)

The last internship assignment was to print the user information from its token track number. I should use the company's provided web host which is http://185.40.72.110:4000.
Therefore, firstly I will post the login information to the host http://185.40.72.110:4000/Login by the given username and password. After we post the correct login credentials, we can get the current token number, so that we can use it later to access the information we need, given that the token number changes every time we login. Later on, we save this token number and add it as a header value to the following web http://185.40.72.110:4000/Work and we post the correct id with it so that at the end we have a list of information that we can print on screen.

Before coding, I used a Rest API tool called Restlet Client [4], to test the post and get methods in order to understand the response results. Therefore, I posted the credential information of the login page as given from the company, after that the response result shows a token number as shown in Figure-10. We take this given token number and use it as a header extension with the required post values as shown in Figure-11, and at the end we should print all the provided information on screen.



POST http://185.40.72.110:4000/Login

length: 31 bytes

Content-Type : application/json

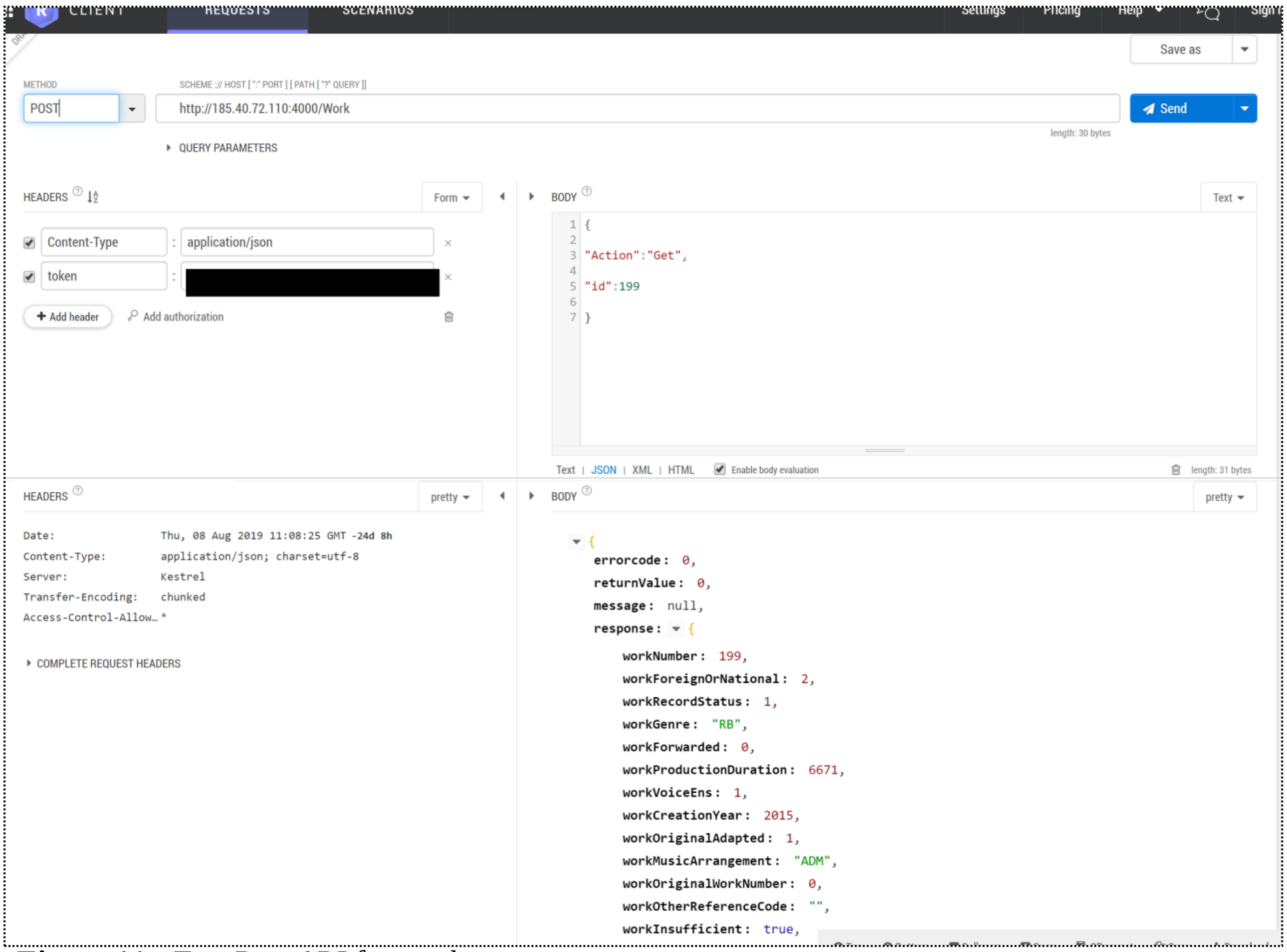
```
{

"Action":"Login",

  "Id":2,

  "Request":{

    "UserName":"[redacted]

    "PasswordHash":[redacted]

  }
```

length: 117 bytes

```
Date:                Thu, 08 Aug 2019 11:07:58 GMT -24d 8h
Content-Type:        application/json; charset=utf-8
Server:              Kestrel
Transfer-Encoding:   chunked
Access-Control-Allow… *
```

```
{
  errorcode :  0,
  returnValue :  0,
  message :  null,
  response :  {
      username :  "Mahmut",
      token :  "[redacted]
      userFirstName :  "Mahmut",
      userLastName :  "Mahmut",
      userDepartment :  "5",
      userMemberAccess :  5,
      userWorkAccess :  5,
      userAdAccess :  5,
      userAlbumAccess :  5,
      userDistAccess :  5
```

Figure 10: *Test Rest API for login page*



POST http://185.40.72.110:4000/Work

length: 30 bytes

Content-Type : application/json
token : [redacted]

```
{

"Action":"Get",

"id":199

}
```

length: 31 bytes

```
Date:                Thu, 08 Aug 2019 11:08:25 GMT -24d 8h
Content-Type:        application/json; charset=utf-8
Server:              Kestrel
Transfer-Encoding:   chunked
Access-Control-Allow… *
```

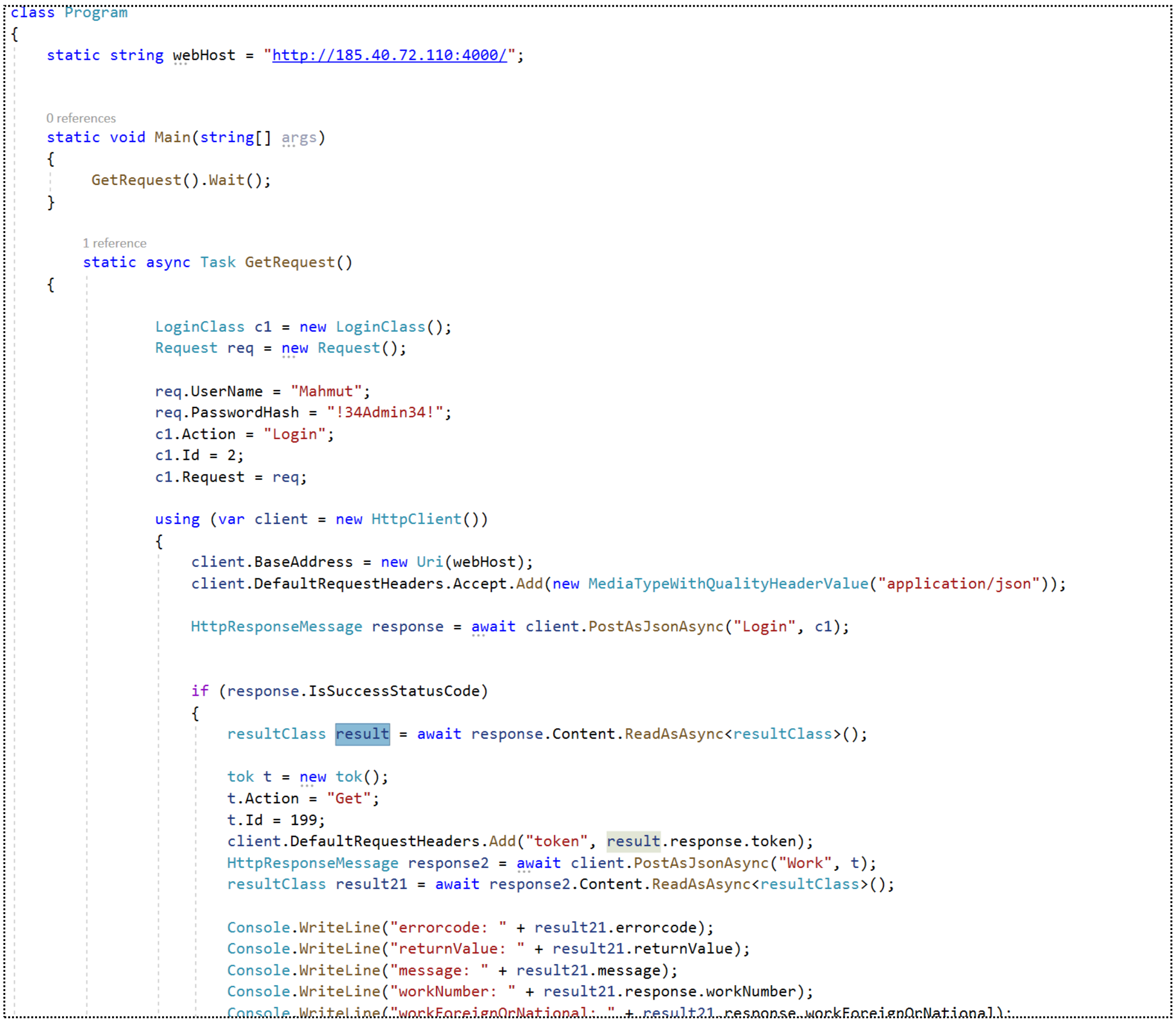
```
{
  errorcode :  0,
  returnValue :  0,
  message :  null,
  response :  {
      workNumber :  199,
      workForeignOrNational :  2,
      workRecordStatus :  1,
      workGenre :  "RB",
      workForwarded :  0,
      workProductionDuration :  6671,
      workVoiceEns :  1,
      workCreationYear :  2015,
      workOriginalAdapted :  1,
      workMusicArrangement :  "ADM",
      workOriginalWorkNumber :  0,
      workOtherReferenceCode :  "",
      workInsufficient :  true,
```

Figure 11: *Test Rest API for work page*

Next, I am going to explain the code implementation as shown below on Figure-12. At start, I made the web host link as a global string, and in the main class I called the method (GetRequest) which will print all the necessary information we need for this assignment. The GetRequest method is an async Task method which means if a process is blocked then the entire system won't be blocked as in sync Task, instead the application can continue to work without depending on the completion of others work.

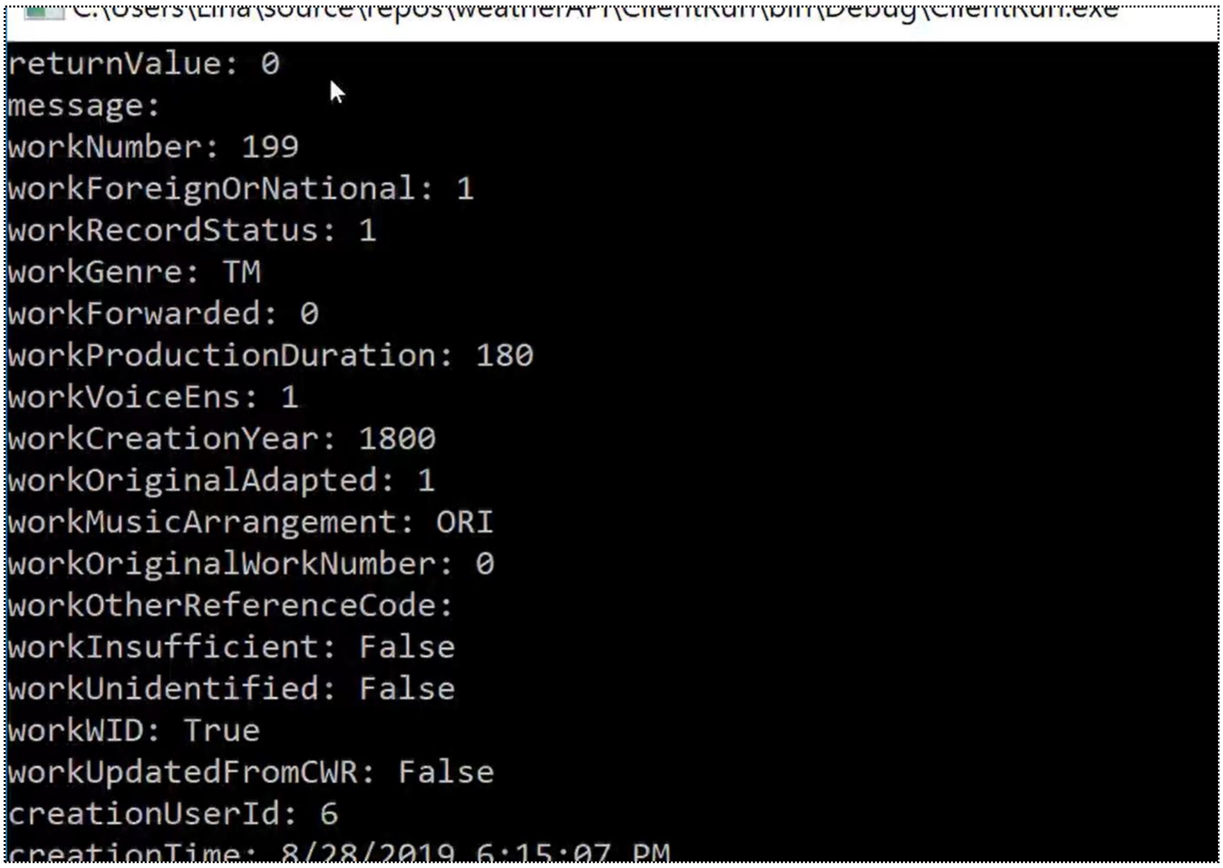
Therefore, on this method I used the LoginClass and Request objects as defined in the objects.cs class, in objects.cs class I have defined all the necessary objects that I need to use, for example LoginClass keeps the Action, id and Request variables, and Request class keeps username and password. After giving values to all the necessary variables, I have defined an http client variable that will be responsible for sending and receiving HTTP response from a given URL resource.

Later on, I have defined the base address of the given host link, and then I have added a header with a value (application/json), and after that I have posted the request values to the (Login) page in order to login successfully. When the response value is successful, we will keep the values in a class called resultClass, and then we defined the values of the tok class to post it later on the next page (Work). Before we post the token values we will add a header values from the previous result to keep it as a header value for (token). Lastly, we will post the required values to the work page and save the result in resultClass object, and at the end we will print the final results in console page as shown in Figure-13.



```
class Program
{
    static string webHost = "http://185.40.72.110:4000/";


    0 references
    static void Main(string[] args)
    {
        GetRequest().Wait();
    }

        1 reference
        static async Task GetRequest()
    {

            LoginClass c1 = new LoginClass();
            Request req = new Request();

            req.UserName = "Mahmut";
            req.PasswordHash = "!34Admin34!";
            c1.Action = "Login";
            c1.Id = 2;
            c1.Request = req;

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri(webHost);
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

                HttpResponseMessage response = await client.PostAsJsonAsync("Login", c1);


                if (response.IsSuccessStatusCode)
                {
                    resultClass result = await response.Content.ReadAsAsync<resultClass>();

                    tok t = new tok();
                    t.Action = "Get";
                    t.Id = 199;
                    client.DefaultRequestHeaders.Add("token", result.response.token);
                    HttpResponseMessage response2 = await client.PostAsJsonAsync("Work", t);
                    resultClass result21 = await response2.Content.ReadAsAsync<resultClass>();

                    Console.WriteLine("errorcode: " + result21.errorcode);
                    Console.WriteLine("returnValue: " + result21.returnValue);
                    Console.WriteLine("message: " + result21.message);
                    Console.WriteLine("workNumber: " + result21.response.workNumber);
                    Console.WriteLine("workForeignOrNational: " + result21.response.workForeignOrNational);
```

Figure 12: *Code of token result from web host*



```
C:\Users\Elma\source\repos\WeatherAPI\ClientRun\bin\Debug\ClientRun.exe
returnValue: 0
message:
workNumber: 199
workForeignOrNational: 1
workRecordStatus: 1
workGenre: TM
workForwarded: 0
workProductionDuration: 180
workVoiceEns: 1
workCreationYear: 1800
workOriginalAdapted: 1
workMusicArrangement: ORI
workOriginalWorkNumber: 0
workOtherReferenceCode:
workInsufficient: False
workUnidentified: False
workWID: True
workUpdatedFromCWR: False
creationUserId: 6
creationTime: 8/28/2019 6:15:07 PM
```

Figure 13: *Result from web host*

4. CONCLUSION

In conclusion, Internship period at ERLAB TEKNOLOJİ company was an excellent experience for me. I gained a lot of knowledge, skills and met new people. The company's staff have helped me whenever I had any issue or a question, and I am very thankful for that. Before taking the internship, my knowledge and experience in the Web API Development was not the best, I have gained a lot of skills and I had the chance to learn more things about myself as a worker, I focused on improving my weakness points, and how to handle pressure at work.

At the end, I believe that each student should have the opportunity to have an internship opprtunity before applying to the real world application. As it will prepare him to the real working environment, and give him the chance to decide which field he wants to work on.

5. REFERENCES

[1] http://erlab.tech/about/

[2] http://www.fasttrackdcn.net/our-products/cis-net/

[3] http://wiki.fasttrackdcn.net/mediawiki/index.php/Main_Page

[4] chrome-extension://aejoelaoggembcahagimdiliamlcdmfm/restlet_client.html