SAKARYA
ÜNİVERSİTESİ

Bilgisayar Ve Bilişim Bilimleri Fakültesi

Yazılım Mühendisliği

WEB PROGRAMMING

Dr. Öğr. Deniz BALTA

DESIGNERS

Hazal Tuğrul	B201202048
Sinem Taşdemir	B201202006
Ayşenur Yılmaz	B201202002

hazal.tugrul@ogr.sakarya.edu.tr
sinem.tasdemir1@ogr.sakarya.edu.tr
ayse.yilmaz28@ogr.sakarya.edu.tr

REPORT

We wanted to make a site that guides people who want to go to Amsterdam to complete their work before starting the journey and to do activities during their trip.

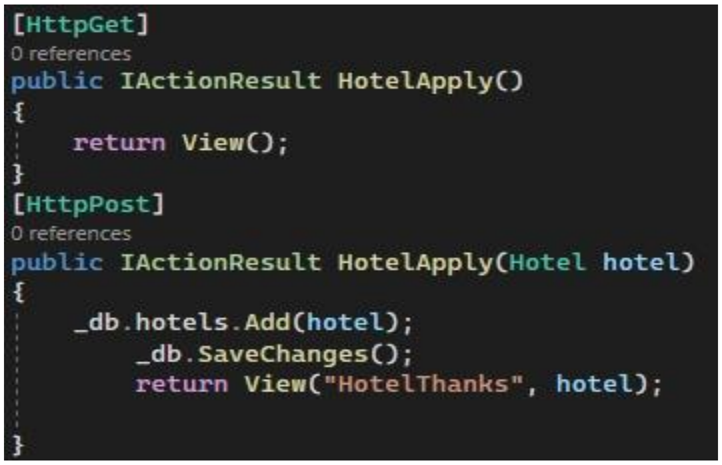
Bon Voyage means have a good journey. Like the name of our page, we aimed to ensure that our users can get information about the city before they go and make their plans according to the city by seeing the places to visit so that they can have a good trip. If they want to start the process to go and do not know where to start, they will find their answers on our site.

In addition to easily accessing all information about Amsterdam through our site, we also save our users who decide to go from work that requires effort such as hotel and flight tickets. Thanks to the information on our page, thanks to the valuable comments and ratings of our users who have traveled to Amsterdam, we help our users who decide to travel, to have information about the places they need to go and to draw their road maps accordingly. Thanks to the new functions we added, our site has become dynamic and the factors that make it dynamic are: routing, razor, view component, view model, partial view, layout, section and entity framework.

Some features we added:

Routing

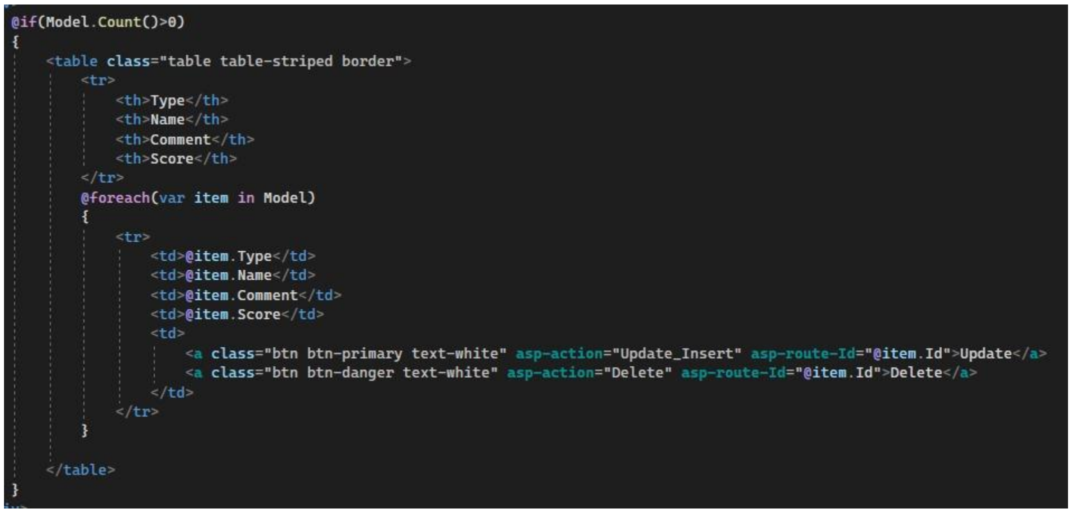
We direct the views through the controller.



```
[HttpGet]
0 references
public IActionResult HotelApply()
{
    return View();
}
[HttpPost]
0 references
public IActionResult HotelApply(Hotel hotel)
{
    _db.hotels.Add(hotel);
        _db.SaveChanges();
        return View("HotelThanks", hotel);

}
```

Razor

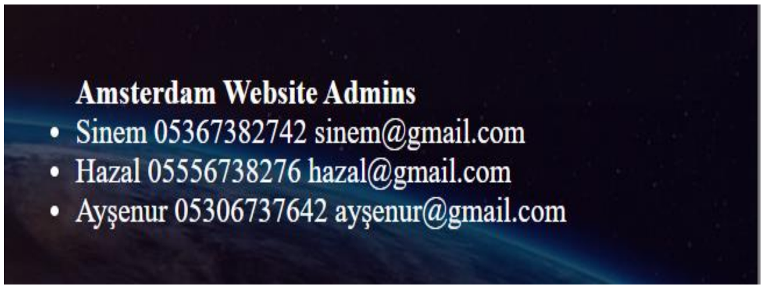
Thanks to Razor, we can use c# and html codes together.



```
@if(Model.Count()>0)
{
    <table class="table table-striped border">
        <tr>
            <th>Type</th>
            <th>Name</th>
            <th>Comment</th>
            <th>Score</th>
        </tr>
        @foreach(var item in Model)
        {
            <tr>
                <td>@item.Type</td>
                <td>@item.Name</td>
                <td>@item.Comment</td>
                <td>@item.Score</td>
                <td>
                    <a class="btn btn-primary text-white" asp-action="Update_Insert" asp-route-Id="@item.Id">Update</a>
                    <a class="btn btn-danger text-white" asp-action="Delete" asp-route-Id="@item.Id">Delete</a>
                </td>
            </tr>
        }

    </table>
}
```

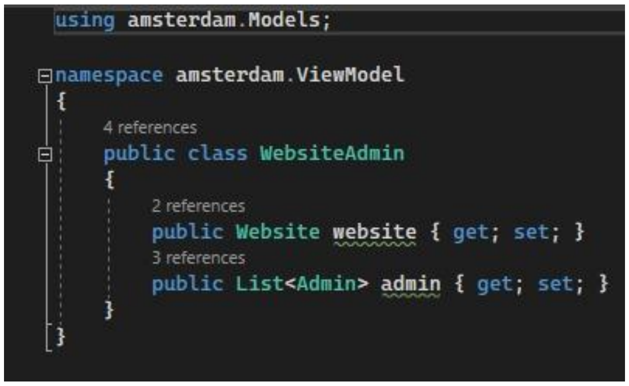
ViewModel

We are taking the admin information on our Contact Us page using viewmodel.

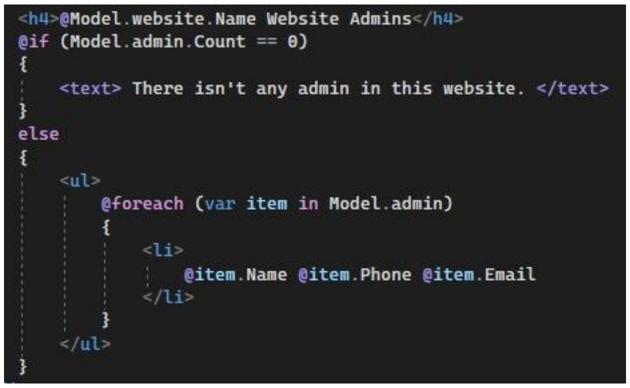


Amsterdam Website Admins

- Sinem 05367382742 sinem@gmail.com
- Hazal 05556738276 hazal@gmail.com
- Ayşenur 05306737642 ayşenur@gmail.com



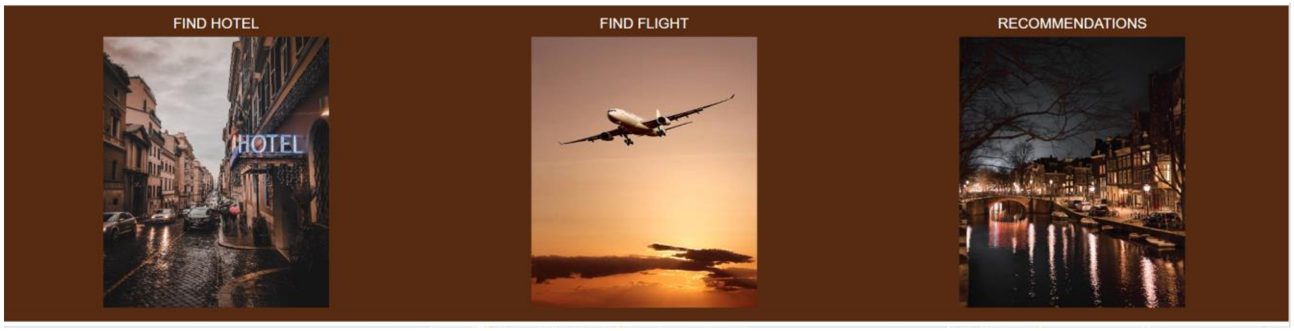
```
using amsterdam.Models;

namespace amsterdam.ViewModel
{
    4 references
    public class WebsiteAdmin
    {
        2 references
        public Website website { get; set; }
        3 references
        public List<Admin> admin { get; set; }
    }
}
```

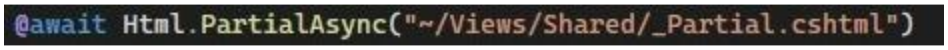


```
<h4>@Model.website.Name Website Admins</h4>
@if (Model.admin.Count == 0)
{
    <text> There isn't any admin in this website. </text>
}
else
{
    <ul>
        @foreach (var item in Model.admin)
        {
            <li>
                @item.Name @item.Phone @item.Email
            </li>
        }
    </ul>
}
```

Partial View

We used the partial view part to make our website dynamic and to adapt it later if we want to make changes to our reservation page.

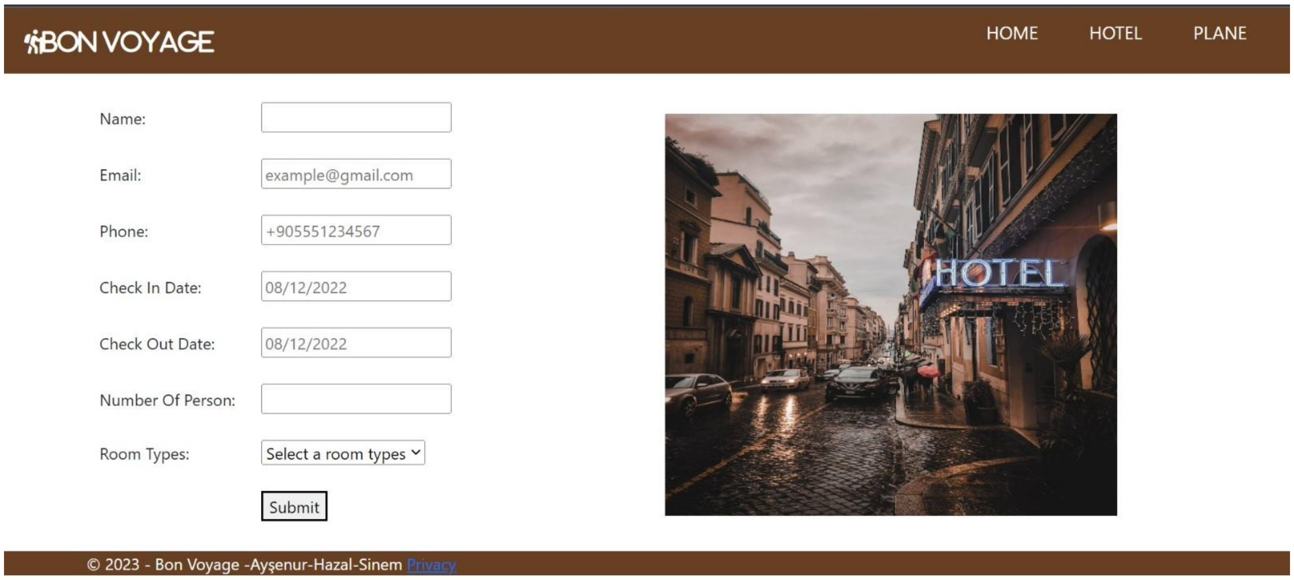




```
@await Html.PartialAsync("~/Views/Shared/_Partial.cshtml")
```

Layout

We get the header and footer parts of all our dynamic pages using layout.

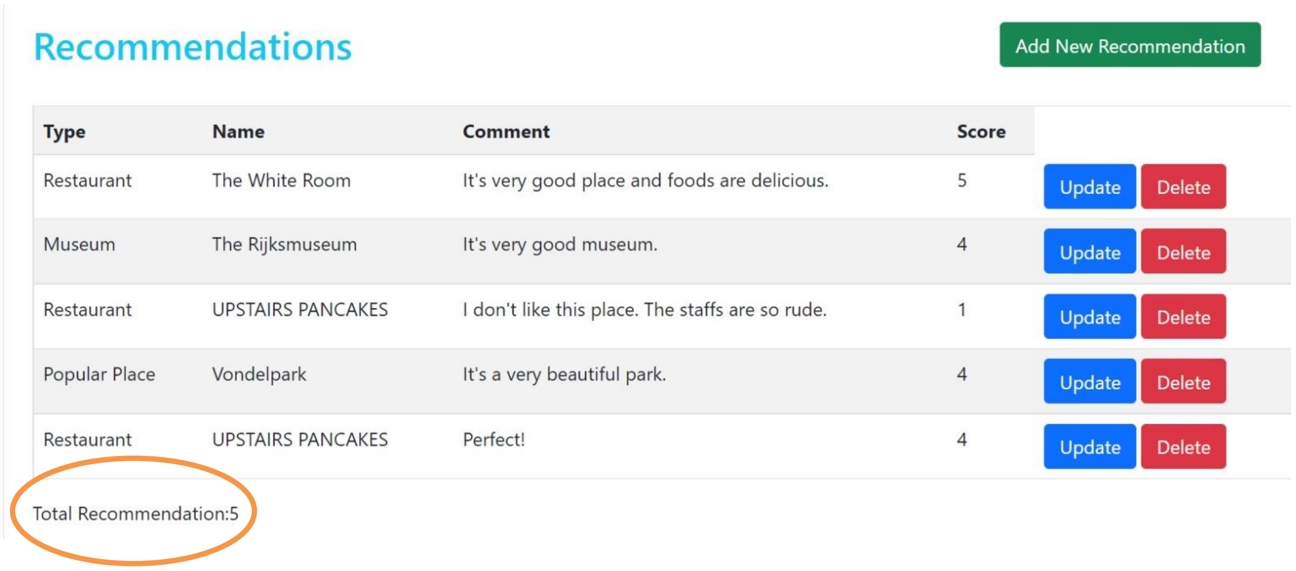


Name:	
Email:	example@gmail.com
Phone:	+905551234567
Check In Date:	08/12/2022
Check Out Date:	08/12/2022
Number Of Person:	
Room Types:	Select a room types

© 2023 - Bon Voyage -Ayşenur-Hazal-Sinem Privacy

ViewComponent

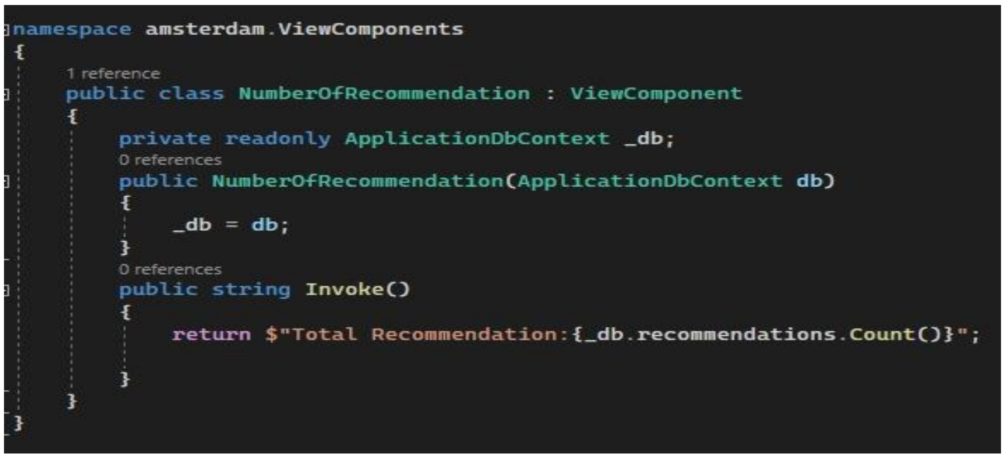
The total recommendation part on our recommendation page was made with the viewcomponent. The “total recommendation” value is updated after each CRUD operation.



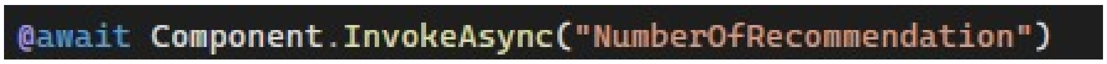
Recommendations

Type	Name	Comment	Score	
Restaurant	The White Room	It's very good place and foods are delicious.	5	Update Delete
Museum	The Rijksmuseum	It's very good museum.	4	Update Delete
Restaurant	UPSTAIRS PANCAKES	I don't like this place. The staffs are so rude.	1	Update Delete
Popular Place	Vondelpark	It's a very beautiful park.	4	Update Delete
Restaurant	UPSTAIRS PANCAKES	Perfect!	4	Update Delete

Total Recommendation:5



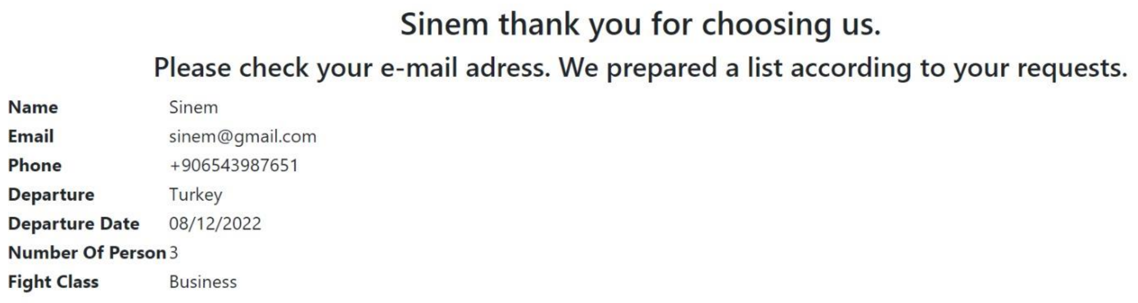
```
namespace amsterdam.ViewComponents
{
    1 reference
    public class NumberOfRecommendation : ViewComponent
    {
        private readonly ApplicationDbContext _db;
        0 references
        public NumberOfRecommendation(ApplicationDbContext db)
        {
            _db = db;
        }
        0 references
        public string Invoke()
        {
            return $"Total Recommendation:{_db.recommendations.Count()}";

        }
    }
}
```



```
@await Component.InvokeAsync("NumberOfRecommendation")
```

Section

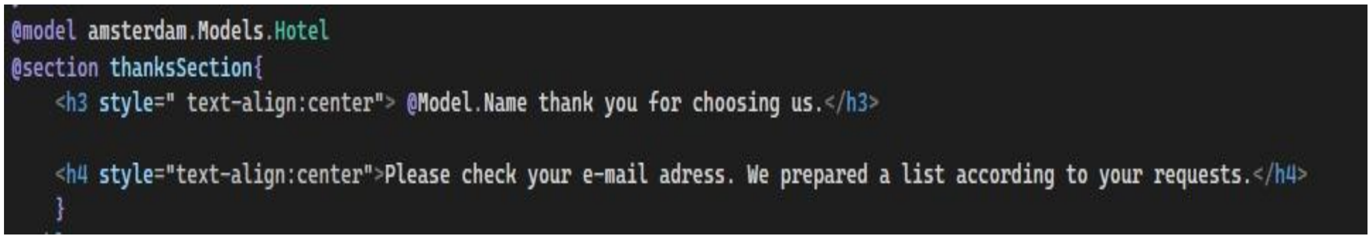
On our thanks page, which we directed to our users after purchasing their tickets, the above text "Thank you for choosing us. Please check your e-mail addresses. We prepared a list according to your requests." part is section.



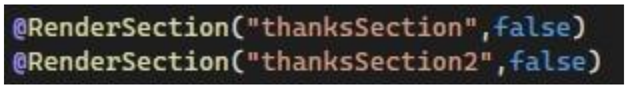
Sinem thank you for choosing us.

Please check your e-mail adress. We prepared a list according to your requests.

Name	Sinem
Email	sinem@gmail.com
Phone	+906543987651
Departure	Turkey
Departure Date	08/12/2022
Number Of Person	3
Fight Class	Business



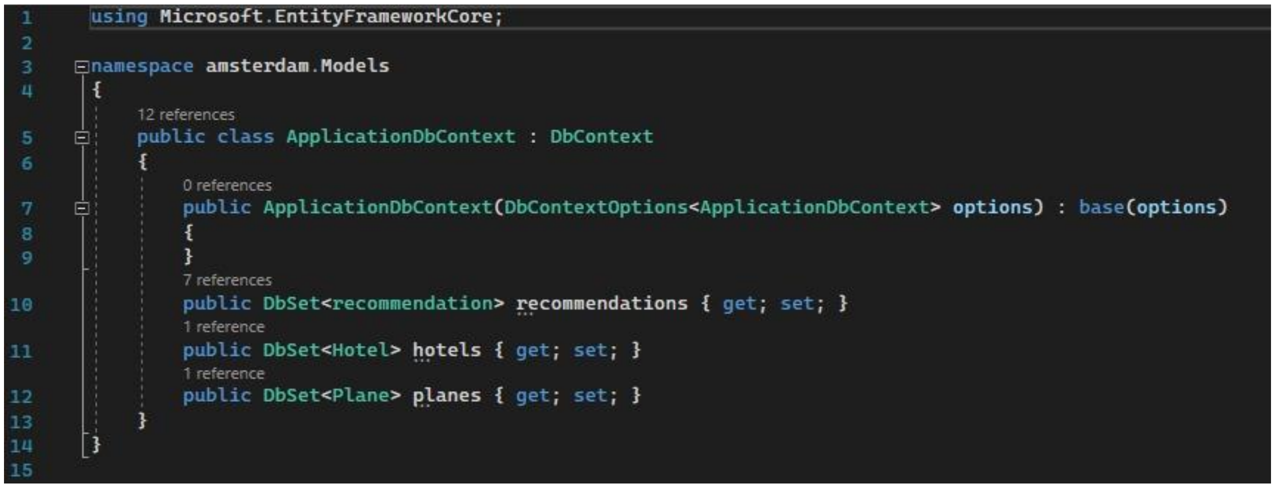
```
@model amsterdam.Models.Hotel
@section thanksSection{
    <h3 style=" text-align:center"> @Model.Name thank you for choosing us.</h3>

    <h4 style="text-align:center">Please check your e-mail adress. We prepared a list according to your requests.</h4>
    }
```



```
@RenderSection("thanksSection",false)
@RenderSection("thanksSection2",false)
```

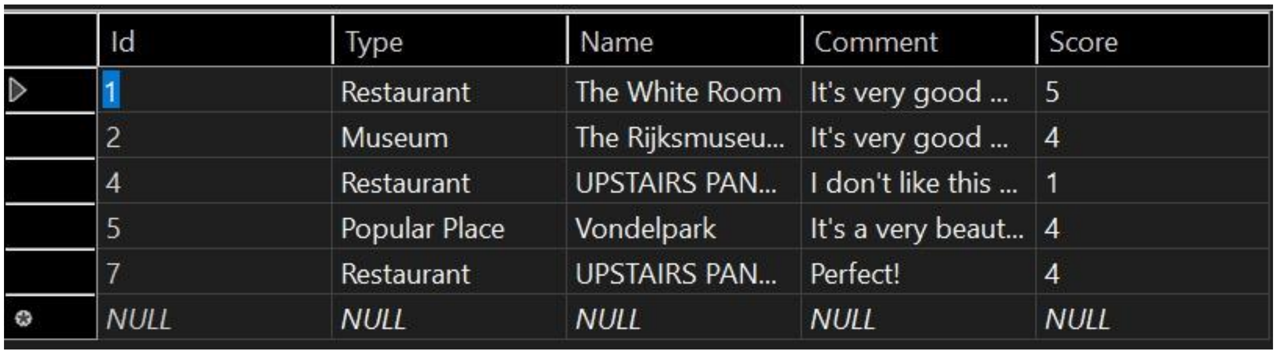
Entity Framework

We add the information and comments of our users to the database we created using the dbcontext class together with the Entity framework.



```
using Microsoft.EntityFrameworkCore;

namespace amsterdam.Models
{
    12 references
    public class ApplicationDbContext : DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
        {
        }
        7 references
        public DbSet<recommendation> recommendations { get; set; }
        1 reference
        public DbSet<Hotel> hotels { get; set; }
        1 reference
        public DbSet<Plane> planes { get; set; }
    }
}
```

Datas of Recommendations



Id	Type	Name	Comment	Score
1	Restaurant	The White Room	It's very good ...	5
2	Museum	The Rijksmuseu...	It's very good ...	4
4	Restaurant	UPSTAIRS PAN...	I don't like this ...	1
5	Popular Place	Vondelpark	It's a very beaut...	4
7	Restaurant	UPSTAIRS PAN...	Perfect!	4
NULL	*NULL*	*NULL*	*NULL*	*NULL*

- ✓ This assignment was done entirely by group work. Each page was made together.