CET 322

FINAL PROJECT REPORT

KİTAPHANE

Behzat Zihni

2014100162

Emir Dikmen

2013100051

Contents

Kitaphane

Project Name and Contributors

The name of our project is Kitaphane. Kitaphane will be designed by Behzat Zihni and Emir Dikmen.

Project Description

Reading is one of the pillars of education and essential in a life of a person. For centuries, people used paper as the main tool of writing and reading.

Our project is about creating an online unique book selling site. In Kitaphane, people both purchase and search unique books from all over the world. People do not reach the unique books easily. Therefore, we help them in order to reach the books easily. Indeed, by using requested book form, people can reach the unique books they need.

Features of the Project

This part of the project is about creating the foundations of the Kitaphane. People will reach Kitaphane via its web site.

In kitaphane.com, following services will be available for customers:

1. Book purchasing
   a. Hardcover
2. Search books from very weight database.
3. Form for requesting non-existed books.
4. Sing in and sign up the system.

For admins there will be:

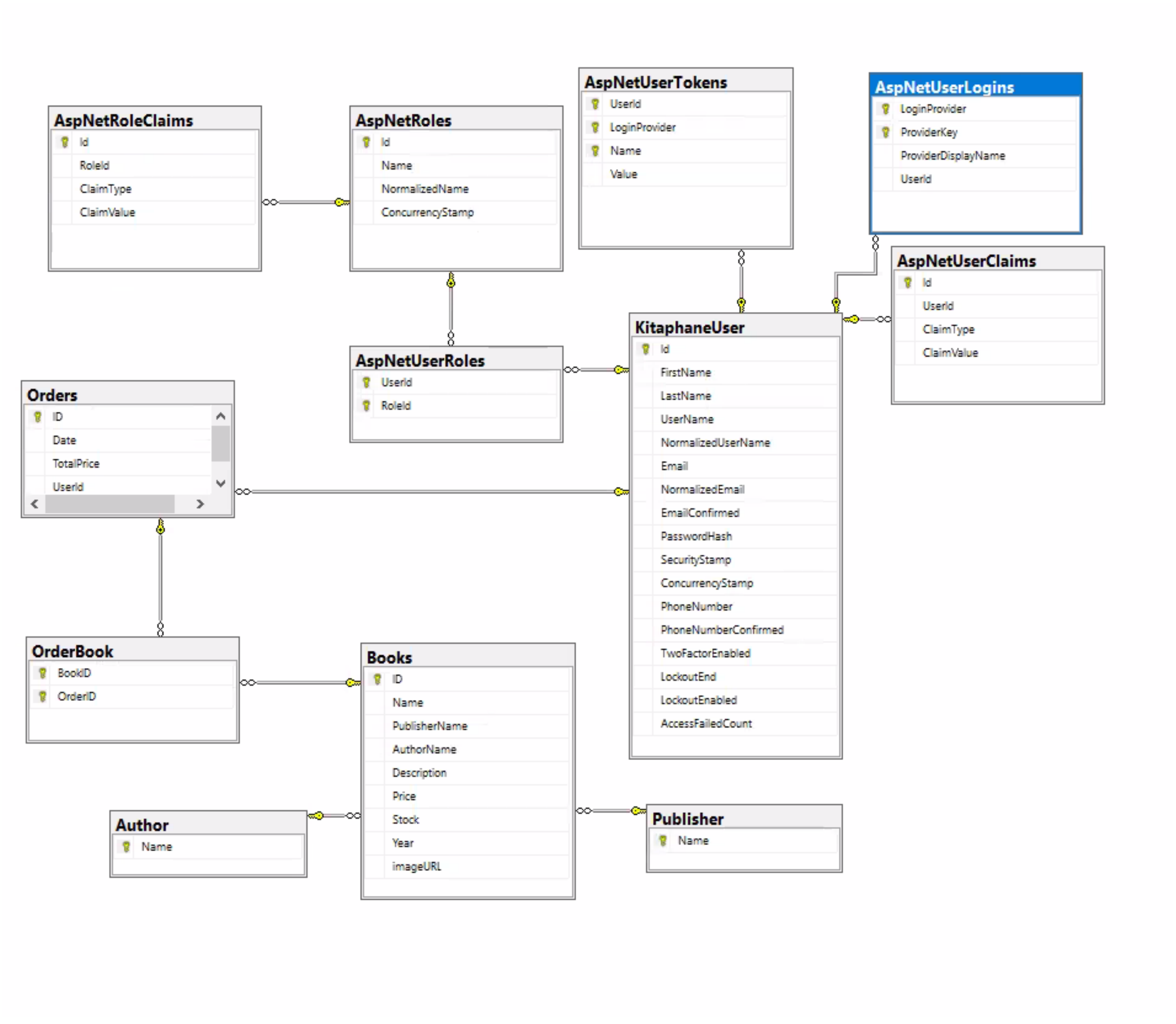
1. Product Insert Page
2. Product delete Page
3. Product edit Page
4. Product detail Page
5. Seeing requested books.

Additionally, there will be a login page, a system for adding to shopping cart, a shopping cart page. Each product will have its own page with the layout system of MVC.

Technologies

In this project, we will use ASP.NET CORE MVC, html to implement the interface and controllers. We will use Entity Framework Core for migrations and communications with the database. We will use bootstrap to design the html pages. We will use JavaScript in the necessary places. Finally, we will use Microsoft SQL Server as database management system. Font awesome are used in some icons.

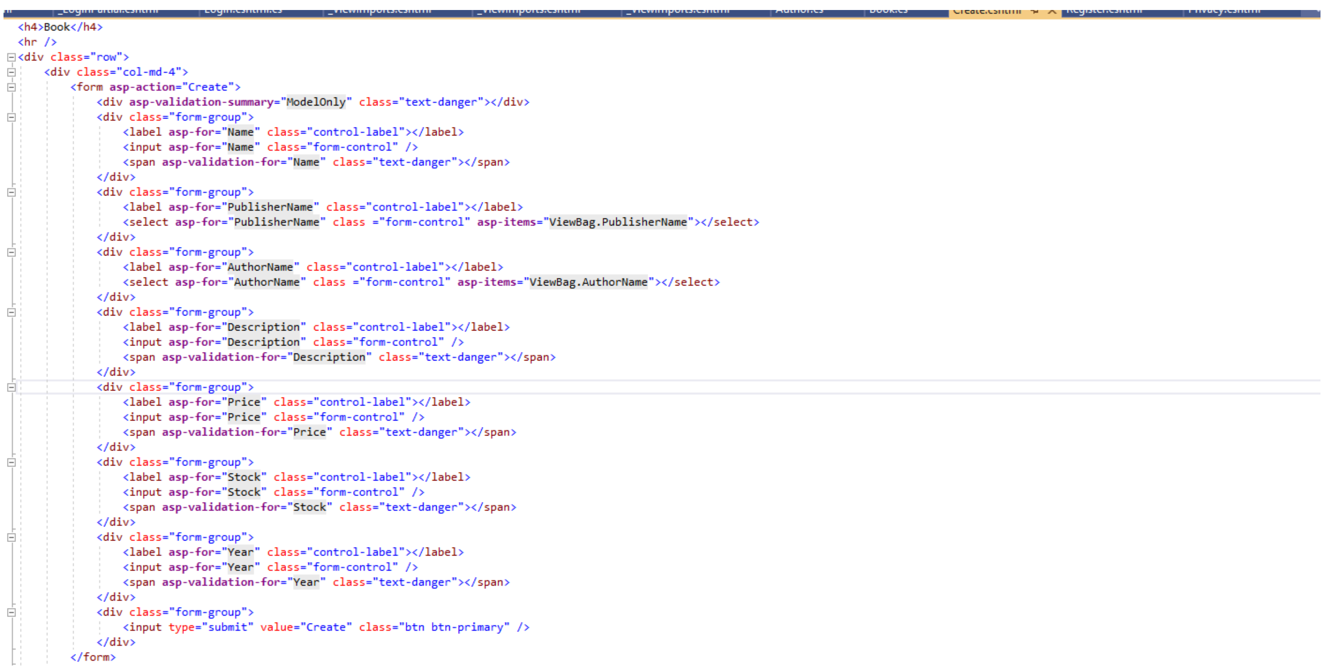
Database


AspNetRoleClaims
Id
RoleId
ClaimType
ClaimValue
AspNetRoles
Id
Name
NormalizedName
ConcurrencyStamp
AspNetUserTokens
UserId
LoginProvider
Name
Value
AspNetUserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId
AspNetUserClaims
Id
UserId
ClaimType
ClaimValue
AspNetUserRoles
UserId
RoleId
KitaphaneUser
Id
FirstName
LastName
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
Orders
ID
Date
TotalPrice
UserId
OrderBook
BookID
OrderID
Books
ID
Name
PublisherName
AuthorName
Description
Price
Stock
Year
imageURL
Author
Name
Publisher
Name

Some code pages from the system

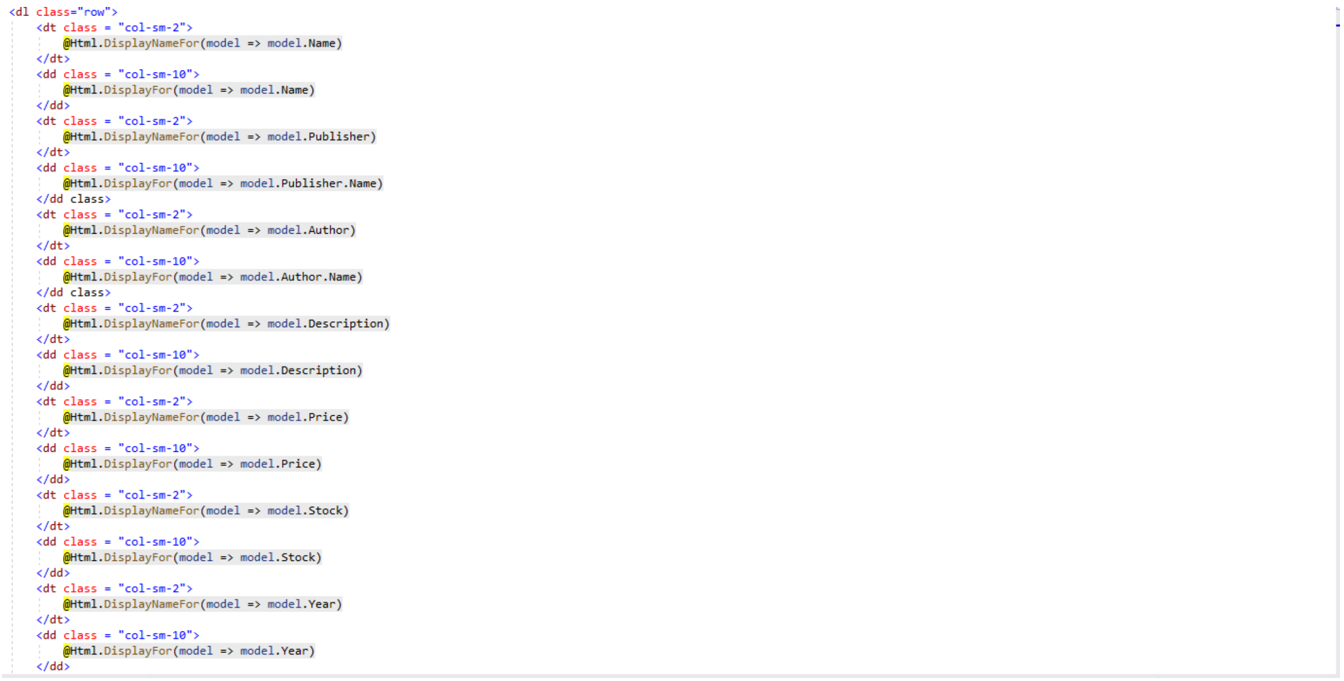
1. View

   1.1 Create: In this page, user can create new books.



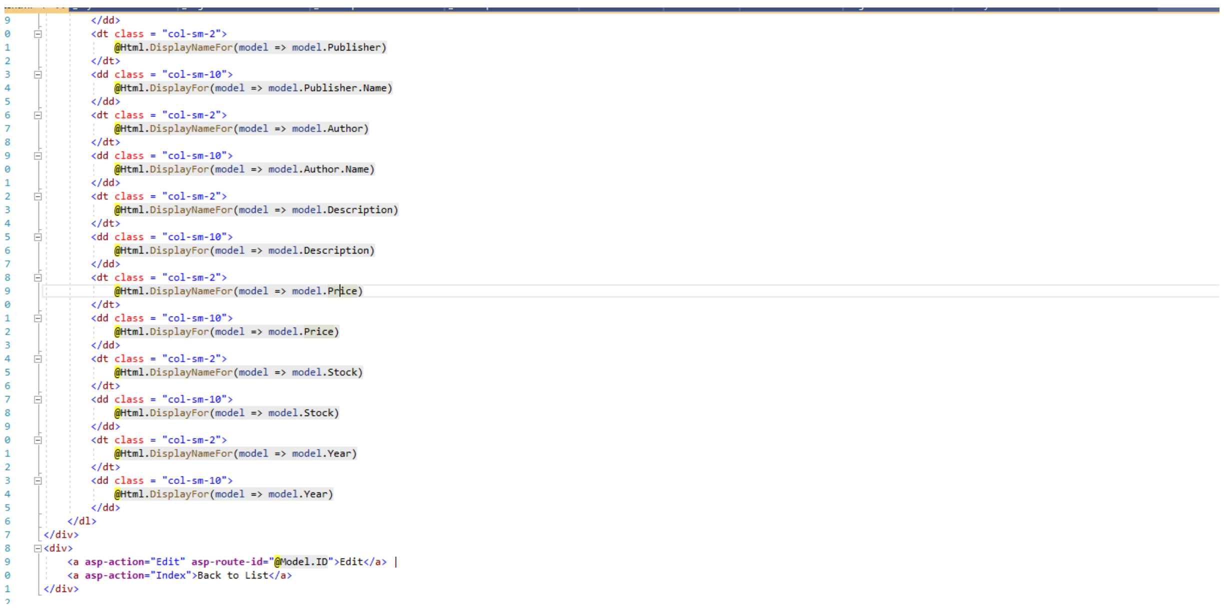
```
<h4>Book</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PublisherName" class="control-label"></label>
                <select asp-for="PublisherName" class ="form-control" asp-items="ViewBag.PublisherName"></select>
            </div>
            <div class="form-group">
                <label asp-for="AuthorName" class="control-label"></label>
                <select asp-for="AuthorName" class ="form-control" asp-items="ViewBag.AuthorName"></select>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Stock" class="control-label"></label>
                <input asp-for="Stock" class="form-control" />
                <span asp-validation-for="Stock" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
```

   1.2 Delete: In this page, user can delete the selected item.



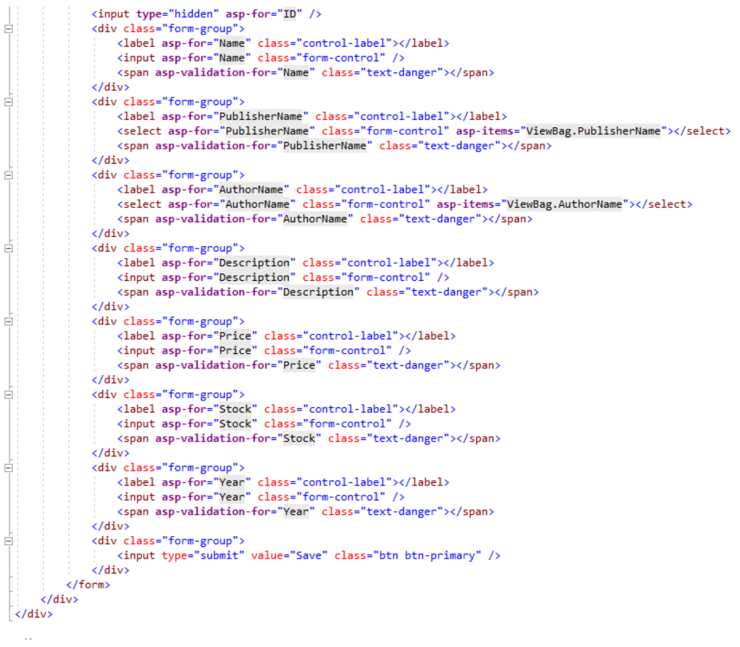
```
<dl class="row">
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Name)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Name)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Publisher)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Publisher.Name)
    </dd class>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Author)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Author.Name)
    </dd class>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Description)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Description)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Price)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Price)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Stock)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Stock)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.Year)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.Year)
    </dd>
```

1.3 Detail: In this page, user can see all the details of books.



```
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Publisher)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Publisher.Name)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Author)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Author.Name)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Description)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Stock)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Stock)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Year)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Year)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model.ID">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

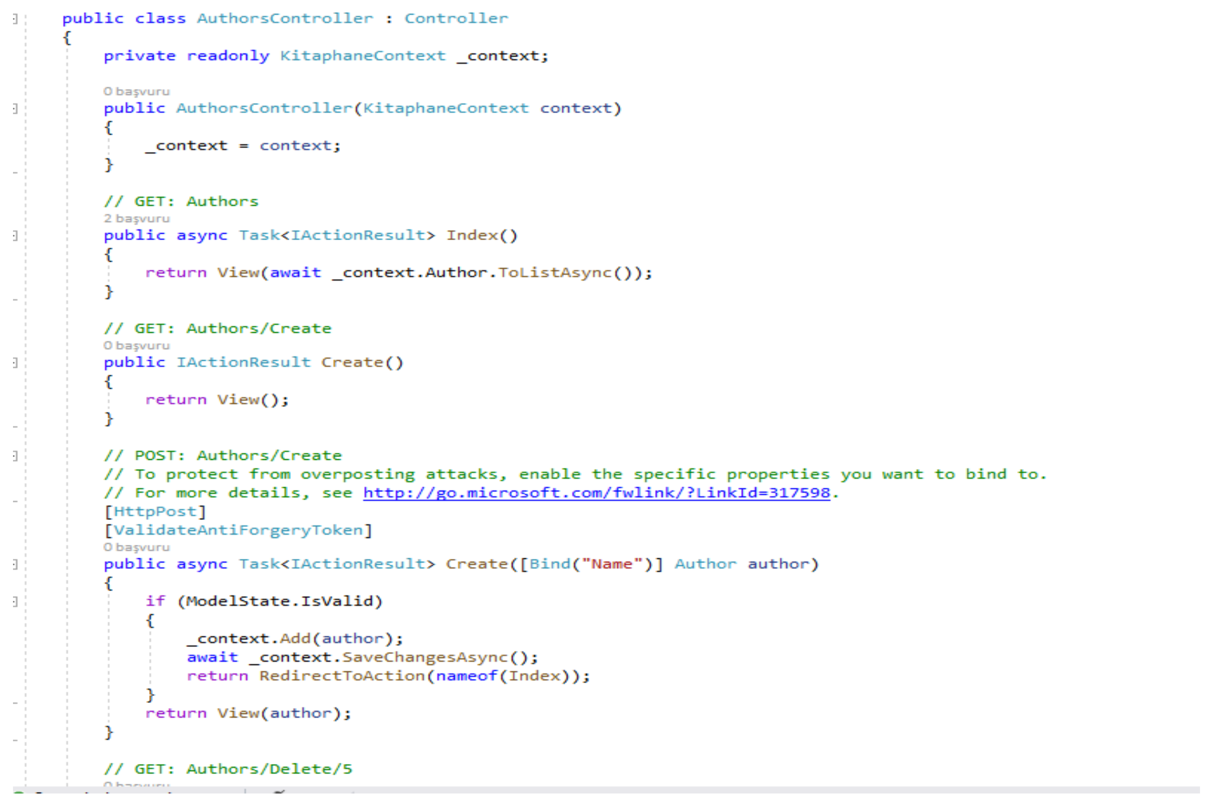
1.4 Edit: In this page, user can edit their data.



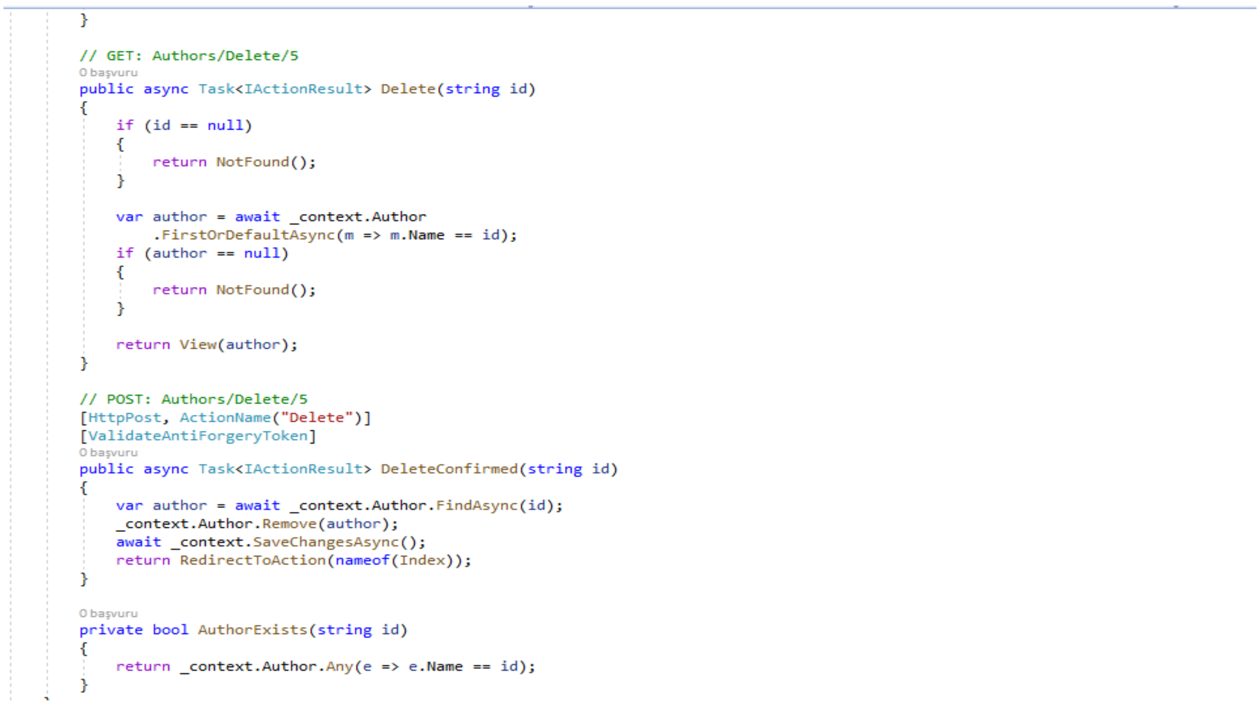
```
            <input type="hidden" asp-for="ID" />
            <div class="form-group">
                <label asp-for="Name" class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PublisherName" class="control-label"></label>
                <select asp-for="PublisherName" class="form-control" asp-items="ViewBag.PublisherName"></select>
                <span asp-validation-for="PublisherName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="AuthorName" class="control-label"></label>
                <select asp-for="AuthorName" class="form-control" asp-items="ViewBag.AuthorName"></select>
                <span asp-validation-for="AuthorName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Stock" class="control-label"></label>
                <input asp-for="Stock" class="form-control" />
                <span asp-validation-for="Stock" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
```

2. Controllers

2.1 Author Controllers: With below code pages, the system can edit, delete, add and show the data of authors from the database.

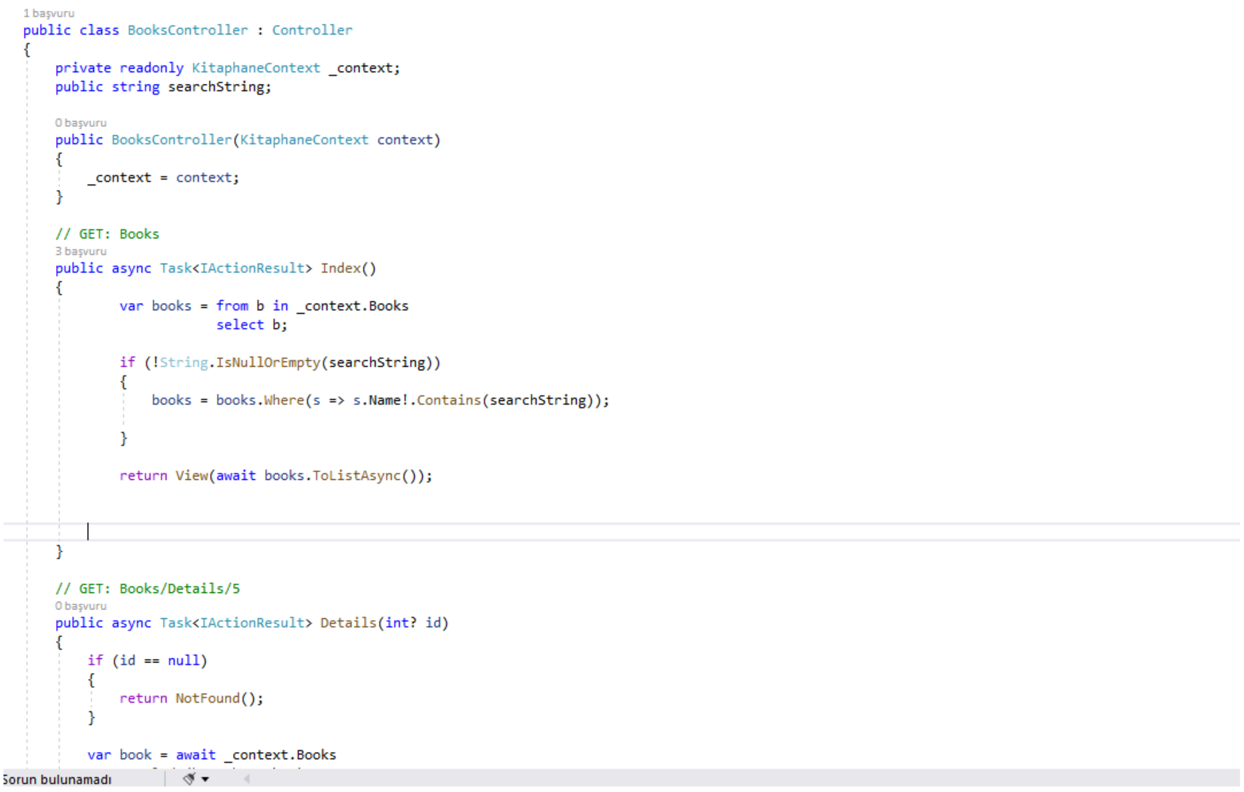


```
public class AuthorsController : Controller
{
    private readonly KitaphaneContext _context;

    0 başvuru
    public AuthorsController(KitaphaneContext context)
    {
        _context = context;
    }

    // GET: Authors
    2 başvuru
    public async Task<IActionResult> Index()
    {
        return View(await _context.Author.ToListAsync());
    }

    // GET: Authors/Create
    0 başvuru
    public IActionResult Create()
    {
        return View();
    }

    // POST: Authors/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 başvuru
    public async Task<IActionResult> Create([Bind("Name")] Author author)
    {
        if (ModelState.IsValid)
        {
            _context.Add(author);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(author);
    }

    // GET: Authors/Delete/5
```



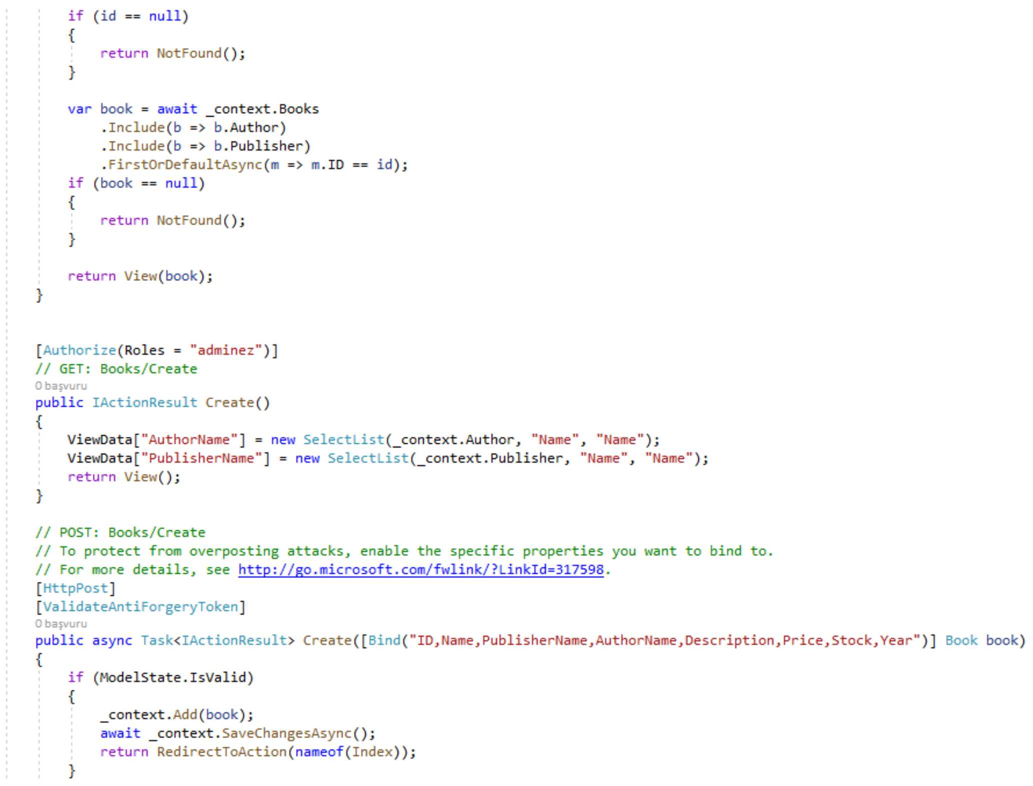
```
    }

    // GET: Authors/Delete/5
    0 başvuru
    public async Task<IActionResult> Delete(string id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var author = await _context.Author
            .FirstOrDefaultAsync(m => m.Name == id);
        if (author == null)
        {
            return NotFound();
        }

        return View(author);
    }

    // POST: Authors/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    0 başvuru
    public async Task<IActionResult> DeleteConfirmed(string id)
    {
        var author = await _context.Author.FindAsync(id);
        _context.Author.Remove(author);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    0 başvuru
    private bool AuthorExists(string id)
    {
        return _context.Author.Any(e => e.Name == id);
    }
```

2.2 Book Controllers: With below code pages, the system can edit, delete, add and show the data of book from the database.

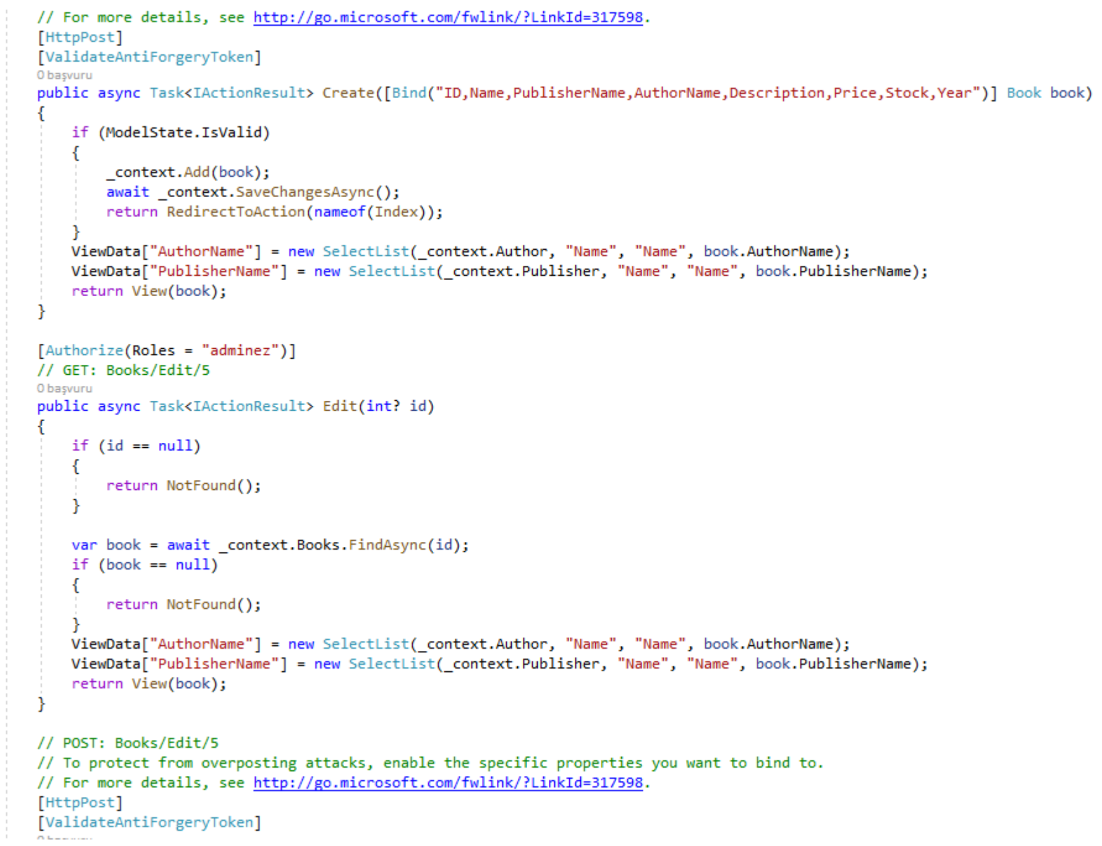


```
1 başvuru
public class BooksController : Controller
{
    private readonly KitaphaneContext _context;
    public string searchString;

    0 başvuru
    public BooksController(KitaphaneContext context)
    {
        _context = context;
    }

    // GET: Books
    3 başvuru
    public async Task<IActionResult> Index()
    {
            var books = from b in _context.Books
                        select b;

            if (!String.IsNullOrEmpty(searchString))
            {
                books = books.Where(s => s.Name!.Contains(searchString));

            }

            return View(await books.ToListAsync());


    }

    // GET: Books/Details/5
    0 başvuru
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var book = await _context.Books
```

Sorun bulunamadı

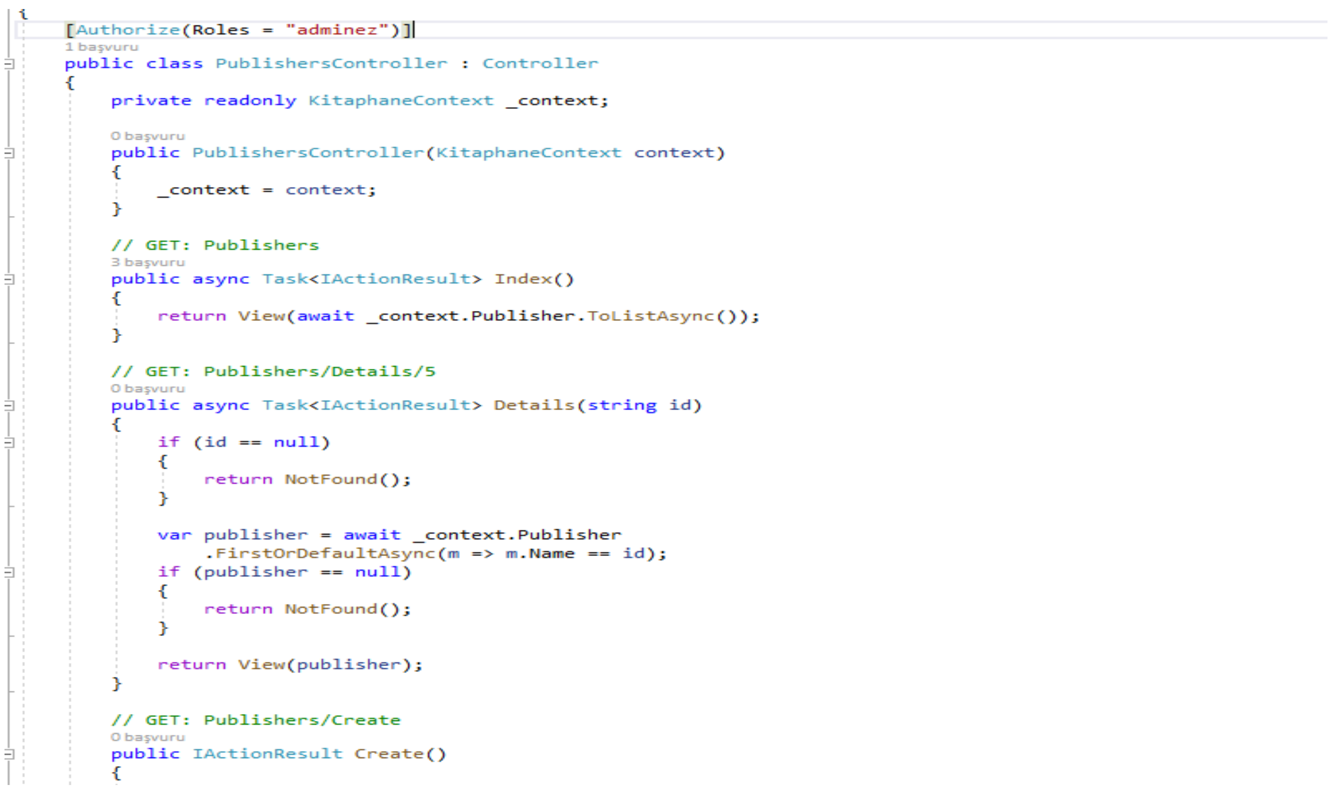


```
        if (id == null)
        {
            return NotFound();
        }

        var book = await _context.Books
            .Include(b => b.Author)
            .Include(b => b.Publisher)
            .FirstOrDefaultAsync(m => m.ID == id);
        if (book == null)
        {
            return NotFound();
        }

        return View(book);
    }


    [Authorize(Roles = "adminez")]
    // GET: Books/Create
    0 başvuru
    public IActionResult Create()
    {
        ViewData["AuthorName"] = new SelectList(_context.Author, "Name", "Name");
        ViewData["PublisherName"] = new SelectList(_context.Publisher, "Name", "Name");
        return View();
    }

    // POST: Books/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 başvuru
    public async Task<IActionResult> Create([Bind("ID,Name,PublisherName,AuthorName,Description,Price,Stock,Year")] Book book)
    {
        if (ModelState.IsValid)
        {
            _context.Add(book);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
```

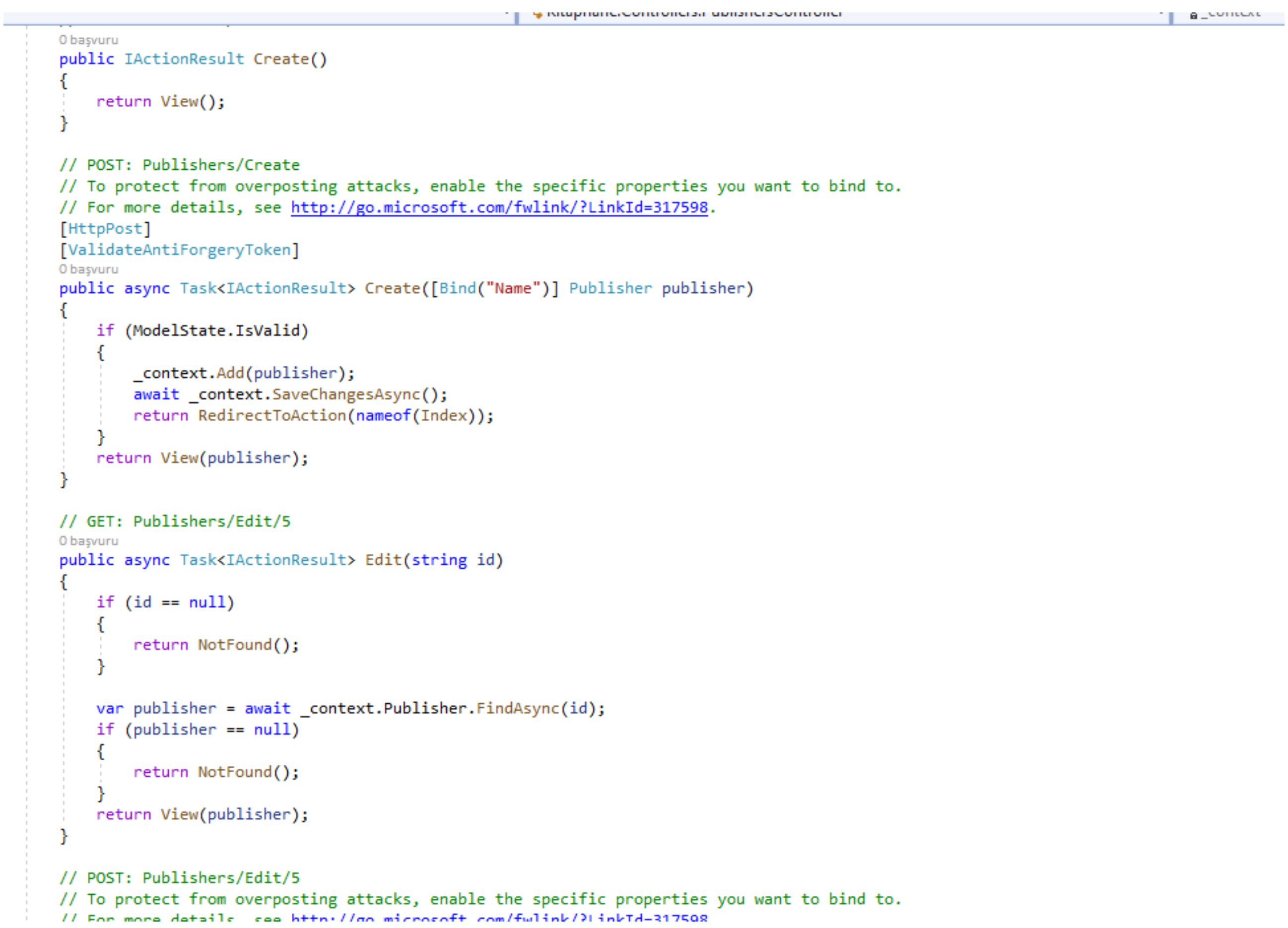


```
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 başvuru
public async Task<IActionResult> Create([Bind("ID,Name,PublisherName,AuthorName,Description,Price,Stock,Year")] Book book)
{
    if (ModelState.IsValid)
    {
        _context.Add(book);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["AuthorName"] = new SelectList(_context.Author, "Name", "Name", book.AuthorName);
    ViewData["PublisherName"] = new SelectList(_context.Publisher, "Name", "Name", book.PublisherName);
    return View(book);
}

[Authorize(Roles = "adminez")]
// GET: Books/Edit/5
0 başvuru
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var book = await _context.Books.FindAsync(id);
    if (book == null)
    {
        return NotFound();
    }
    ViewData["AuthorName"] = new SelectList(_context.Author, "Name", "Name", book.AuthorName);
    ViewData["PublisherName"] = new SelectList(_context.Publisher, "Name", "Name", book.PublisherName);
    return View(book);
}

// POST: Books/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
```

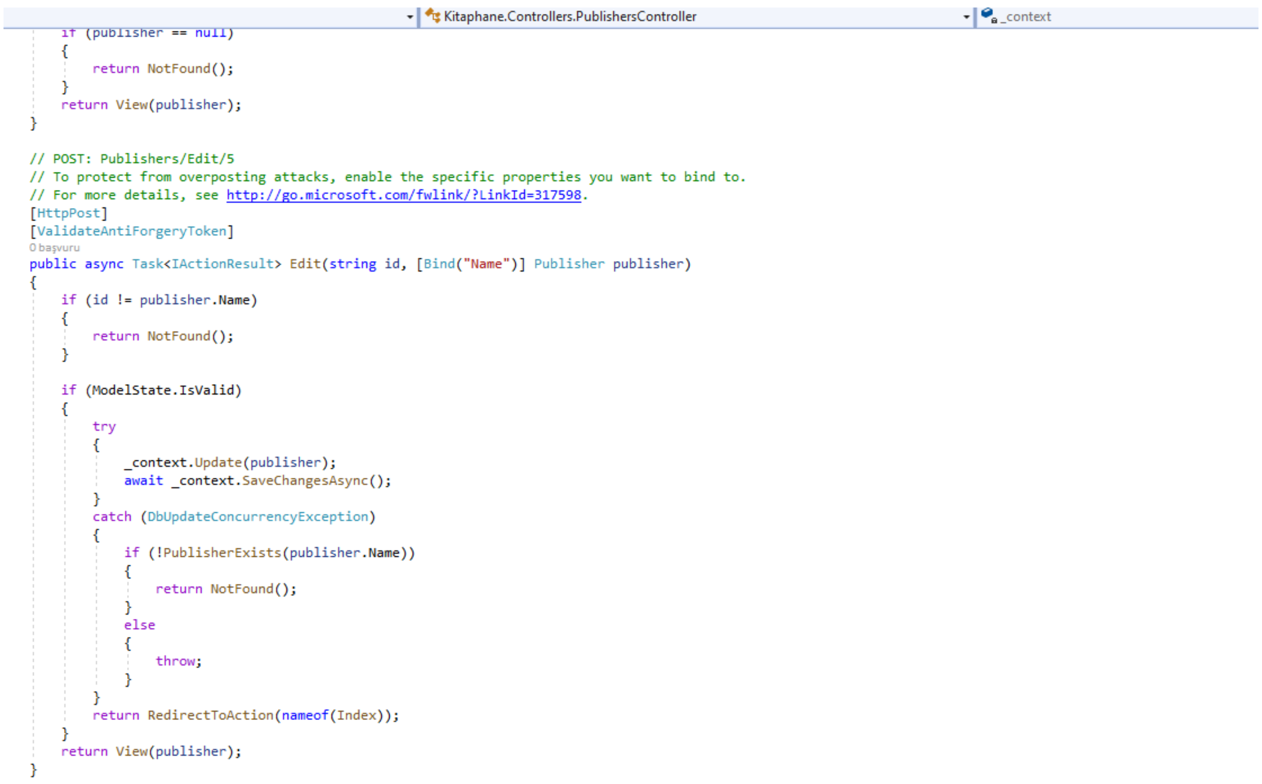
2.3 Publisher Controllers: With below code pages, the system can edit, delete, add and show the data of publisher from the database.



```
{
    [Authorize(Roles = "adminez")]
    1 başvuru
    public class PublishersController : Controller
    {
        private readonly KitaphaneContext _context;

        0 başvuru
        public PublishersController(KitaphaneContext context)
        {
            _context = context;
        }

        // GET: Publishers
        3 başvuru
        public async Task<IActionResult> Index()
        {
            return View(await _context.Publisher.ToListAsync());
        }

        // GET: Publishers/Details/5
        0 başvuru
        public async Task<IActionResult> Details(string id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var publisher = await _context.Publisher
                .FirstOrDefaultAsync(m => m.Name == id);
            if (publisher == null)
            {
                return NotFound();
            }

            return View(publisher);
        }

        // GET: Publishers/Create
        0 başvuru
        public IActionResult Create()
        {
```

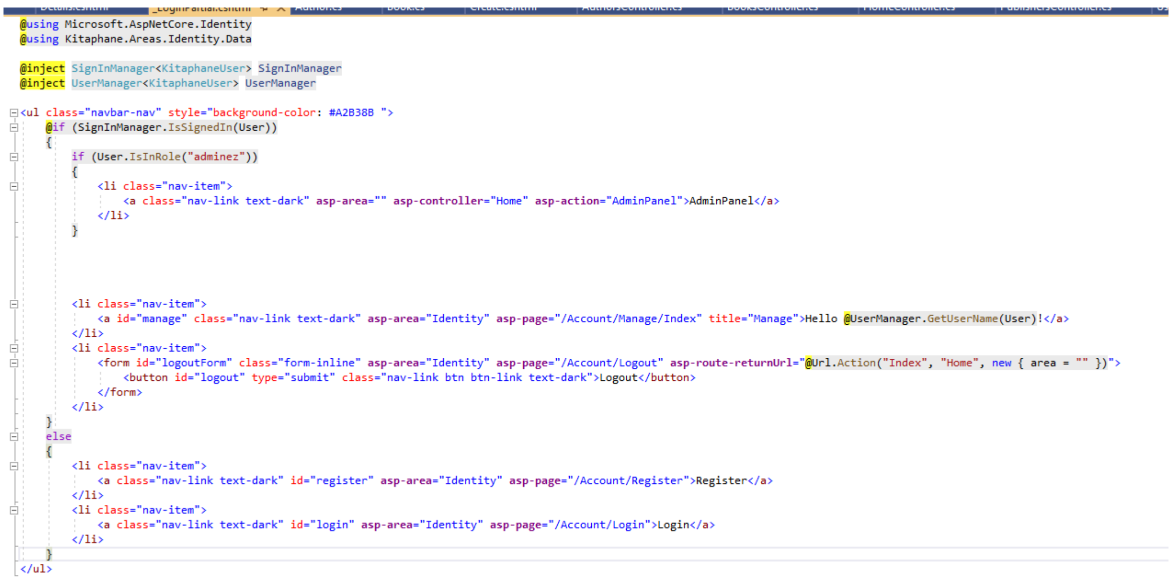


```
0 başvuru
public IActionResult Create()
{
    return View();
}

// POST: Publishers/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 başvuru
public async Task<IActionResult> Create([Bind("Name")] Publisher publisher)
{
    if (ModelState.IsValid)
    {
        _context.Add(publisher);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(publisher);
}

// GET: Publishers/Edit/5
0 başvuru
public async Task<IActionResult> Edit(string id)
{
    if (id == null)
    {
        return NotFound();
    }

    var publisher = await _context.Publisher.FindAsync(id);
    if (publisher == null)
    {
        return NotFound();
    }
    return View(publisher);
}

// POST: Publishers/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598
```



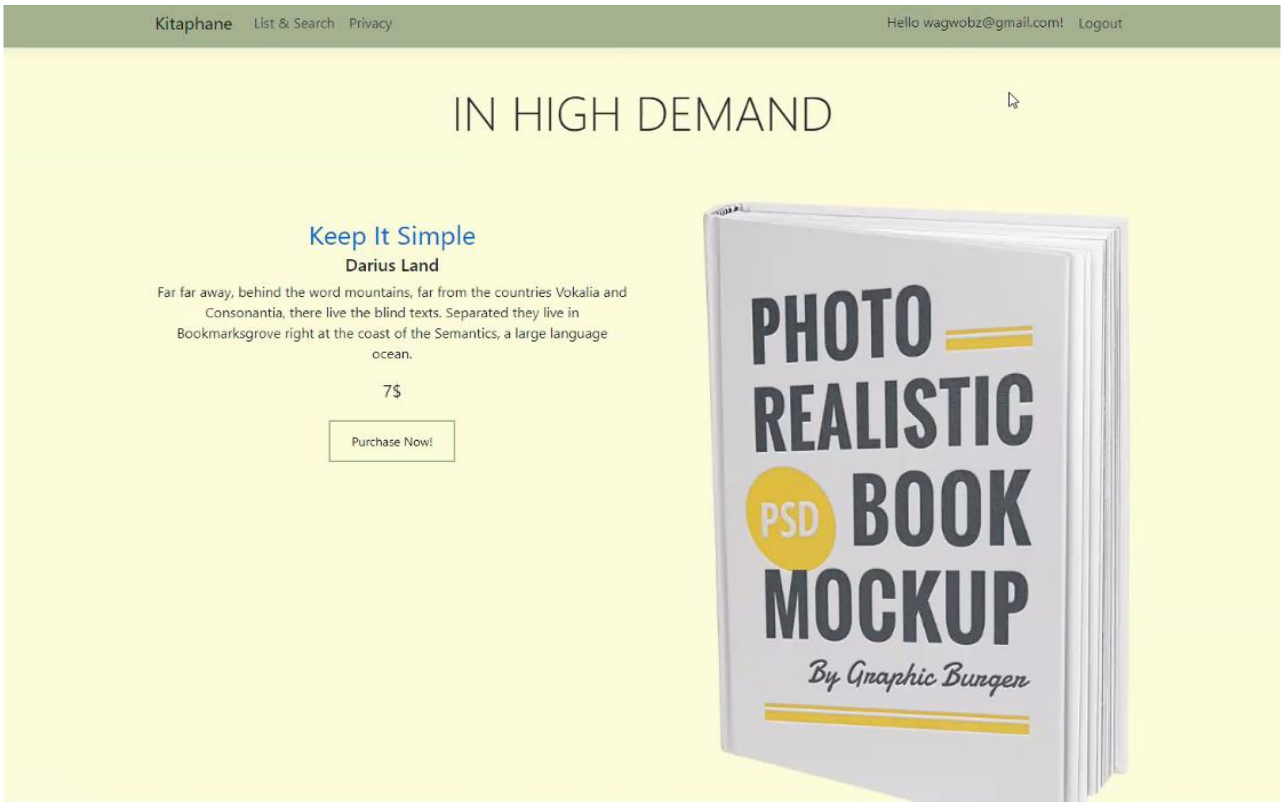
```
    if (publisher == null)
    {
        return NotFound();
    }
    return View(publisher);
}

// POST: Publishers/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 başvuru
public async Task<IActionResult> Edit(string id, [Bind("Name")] Publisher publisher)
{
    if (id != publisher.Name)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(publisher);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!PublisherExists(publisher.Name))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(publisher);
}
```

3. Login: with below code, we both user and customer reach the system.



```
@using Microsoft.AspNetCore.Identity
@using Kitaphane.Areas.Identity.Data

@inject SignInManager<KitaphaneUser> SignInManager
@inject UserManager<KitaphaneUser> UserManager

<ul class="navbar-nav" style="background-color: #A2B38B ">
    @if (SignInManager.IsSignedIn(User))
    {
        if (User.IsInRole("adminez"))
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="AdminPanel">AdminPanel</a>
            </li>
        }



        <li class="nav-item">
            <a id="manage" class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Hello @UserManager.GetUserName(User)!</a>
        </li>
        <li class="nav-item">
            <form id="logoutForm" class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">
                <button id="logout" type="submit" class="nav-link btn btn-link text-dark">Logout</button>
            </form>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" id="register" asp-area="Identity" asp-page="/Account/Register">Register</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" id="login" asp-area="Identity" asp-page="/Account/Login">Login</a>
        </li>
    }
</ul>
```

Especically, I want to mention that on the right-top of the screen, we see the name of the user.



Kitaphane List & Search Privacy Hello wagwobz@gmail.com! Logout

IN HIGH DEMAND

Keep It Simple

Darius Land

Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean.

7$

Purchase Now!

Conclusion

In the end, Kitaphane will be a site that will be working without bugs, will have a responsive interface with bootstrap and It can be used without developers due to the creating of admin pages.