# Api (Application Programming Interface) Nedir?

Türkçe karşılığı uygulama geliştirme arayüzü anlamına gelmektedir. Kişinin elindeki dataları dış dünyaya açarak diğer uygulama veya platformların kullanımına sunmak için belirli kurallar çerçevesinde tanımlamalar yaptığı interface yani arayüzdür.

# Güvenlik mekanizmalarının oluşturulmasında kullanılan iki temel kavram vardır. Bunlar:

- **Kullanıcı Doğrulama** (**Authentication**), bir kullanıcının kim olduğunu belirleme işlemidir.
- **Kullanıcı Yetkilendirme** (**Authorization**), belirlenen kullanıcıların hangi kaynaklara erişeceğini belirleme işlemidir.

[Authorize] attiribute user apisine erişebilmek için login yapmış olmak ve token almadan bu methodlara erişemeyeceğimizi gösterir.

```
namespace AspNetCoreWebAPI.Controllers {

    [Authorize]
    [ApiController]
    [Route ("api/[controller]")]
    public class UserController : ControllerBase {

        [HttpGet ("getusers")]
        public ActionResult GetUsers () {
            var users = GetUserList ().ToList ();
            return Ok (users);
        }
```

Bu action'a erişebilmek için Admin yetkisine sahip kullanıcı olmak gerekmektedir.

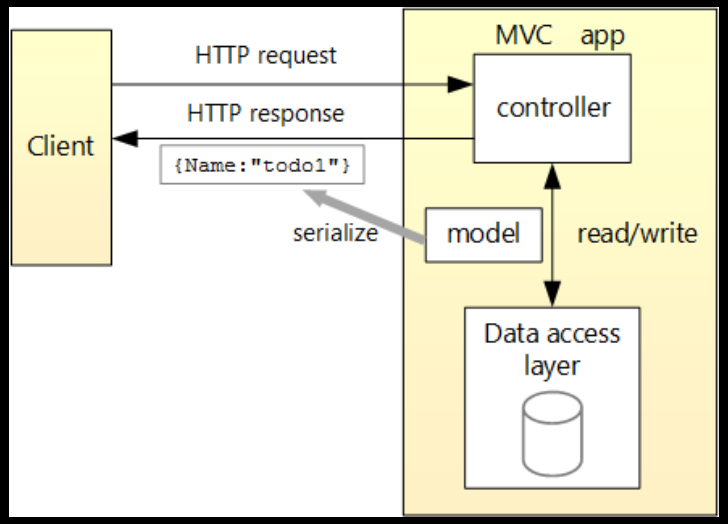
```
    [Authorize (Roles = "Admin")]
    [HttpPost ("setuser")]
    //[Route("setuser")]
    public ActionResult SetUser ([FromForm] UserRequestModel model) {
        var users = GetUserList ();

        User userItem = new User () {
            Id = model.Id,
            Name = model.Name,
            Company = model.Company
        };

        users.Add (userItem);
        return Ok (users);
    }
```

Ancak bazı action metodları sınırlamadan çıkarmak için [**AllowAnonymous**] attribute kullanılabilir.

```
    [AllowAnonymous]
    [HttpPut ("putuser/{id}")]
    public IActionResult PutUser (int Id, UserRequestModel model) {
        var response = GetUserList ().Find (b => b.Id == Id);
        if (response == null) {
            return NotFound ();
        }
        response.Name = model.Name;
        return Ok (response);
    }
```

# Asp .Net Core Web Api

Rest mimarisi üzerinde, restful servisler oluşturmamız için ihtiyacımız olan alt yapıyı içinde barındıran .Net çevresi içerisinde bulunan framework.



# Proje oluşturma adımları

- Öncelikle cd ile source altındaki repos klasörüne gidilmiştir.
- Proje hangi klasörde oluşturulmak isteniyorsa mkdir komutu ile DotNetWebApiDemo adından bir klasör oluşturulmaktadır.
- Cd komutu ile DotNetWebApiDemo klasörüne gidilmektedir.
- dotnet new webapi -n AspNetCoreWebAPI komutu ile proje oluşturulmaktadır.
- AspNetCoreWebAPI klasörüne gidilmektedir.
- dotnet restore komutu ile csproj dosyası oluşturulmaktadır.
- code . komutu ile proje açılabilmektedir.
- dotnet –version komutu ile mevcut sdk versionuna erişilebilmektedir.
- dotnet –info komutu ile yüklü sdk versionlarına erişilebilmektedir.
- dotnet new globaljson --sdk-version 2.2.402 komutu ile version değişikliği yapılabilir. İstenilen versionda proje oluşturabilir.

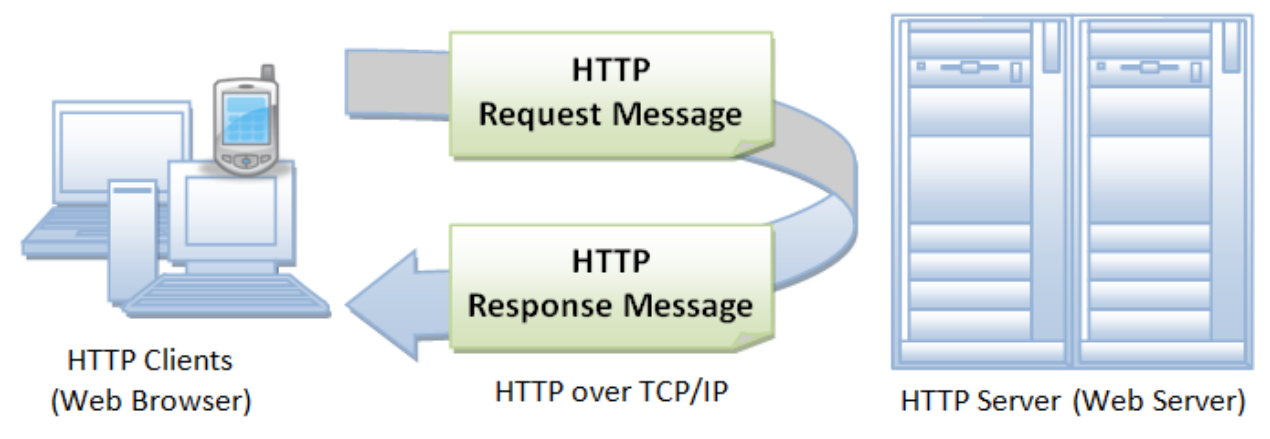
```
C:\Users\seyma>cd source

C:\Users\seyma\source>cd repos

C:\Users\seyma\source\repos>mkdir DotNetWebApiDemo

C:\Users\seyma\source\repos>cd DotNetWebApiDemo

C:\Users\seyma\source\repos\DotNetWebApiDemo>dotnet new webapi -n AspNetCoreWebAPI
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on AspNetCoreWebAPI\AspNetCoreWebAPI.csproj...
  Restore completed in 195.47 ms for C:\Users\seyma\source\repos\DotNetWebApiDemo\AspNetCoreWebAPI\AspNetCoreWebAPI.csproj.

Restore succeeded.


C:\Users\seyma\source\repos\DotNetWebApiDemo>cd AspNetCoreWebAPI

C:\Users\seyma\source\repos\DotNetWebApiDemo\AspNetCoreWebAPI>dotnet restore
  Restore completed in 35.64 ms for C:\Users\seyma\source\repos\DotNetWebApiDemo\AspNetCoreWebAPI\AspNetCoreWebAPI.csproj.

C:\Users\seyma\source\repos\DotNetWebApiDemo\AspNetCoreWebAPI>code .
```

# Extensions

- https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp
- https://marketplace.visualstudio.com/items?itemName=jchannon.csharpextensions
- https://marketplace.visualstudio.com/items?itemName=Leopotam.csharpfixformat

# Http(Hypertext Transfer Protocol) Protokolü Nedir?

- Rest mimarisi kullanan servisler Client-Server arasında Http protokolü üzerinden veri alışverişini sağlar.



# Http Methodları

- GET: URL den verilen bilgiler kullanılarak backend tarafında kayıtlı bir bilgiyi almak için kullanılır. Request Body'si bulunmamakt adır.

https://localhost:8087/api/user/getuser/2

```
[HttpGet ("getuser/{id}")]
public ActionResult GetUser (int id) {
    var user = GetUserList ().Where (c => c.Id == id).FirstOrDefault ();
    return Ok (user);
}
```

Url: https://localhost:8087/api/user/getuser?id=2

```
[HttpGet ("getuser")]
public ActionResult GetUser (int id) {
    var user = GetUserList ().Where (c => c.Id == id).FirstOrDefault ();
    return Ok (user);
}
```

Url: https://localhost:8087/api/user/getusers

```
namespace AspNetCoreWebAPI.Controllers {

    [ApiController]
    [Route ("api/[controller]")]
    public class UserController : ControllerBase {

        [HttpGet ("getusers")]
        public ActionResult GetUsers () {
            var users = GetUserList ().ToList ();
            return Ok (users);
        }
```

Aşağıda gösterildiği şekilde Route attribute eklenerek de kullanılabilir.

```
namespace AspNetCoreWebAPI.Controllers {

    [ApiController]
    [Route ("api/[controller]")]
    public class UserController : ControllerBase {

        [HttpGet]
        [Route ("getusers")]
        public ActionResult GetUsers () {
            var users = GetUserList ().ToList ();
            return Ok (users);
        }
```

- **POST: G**enellikle backend tarafında yeni bir bilgi göndermek bir kayıt oluşturmak amacıyla kullanılır. Request Body'si bulunmaktadır.

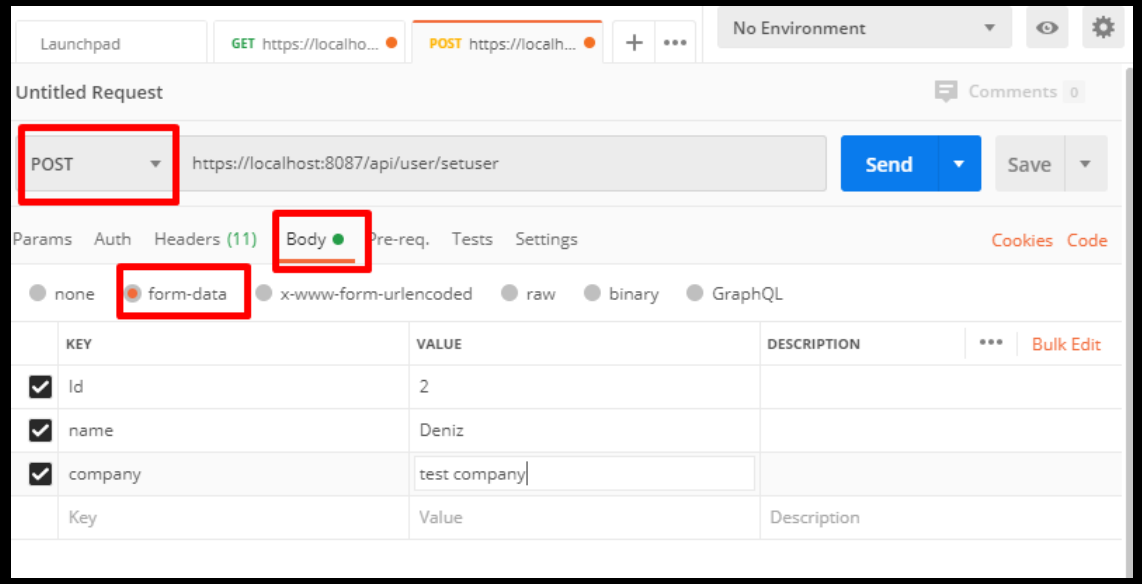
form-data yada x-www-form-urlencoded veri formatında post edileceği zaman [FromForm]UserRequestModel model bu şekilde almalıyız.

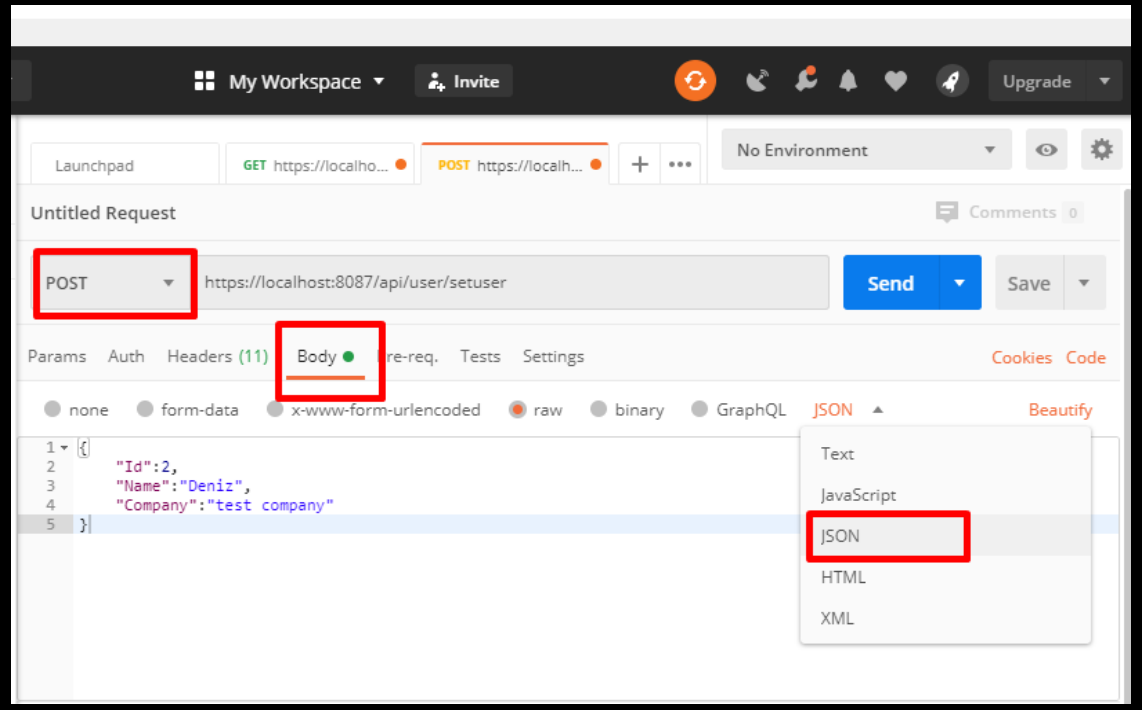
```
[HttpPost ("setuser")]
public ActionResult SetUser ([FromForm] UserRequestModel model) {

    var users = GetUserList ();

    User userItem = new User () {
        Id = model.Id,
        Name = model.Name,
        Company = model.Company
    };

    users.Add (userItem);
    return Ok (users);
}
```

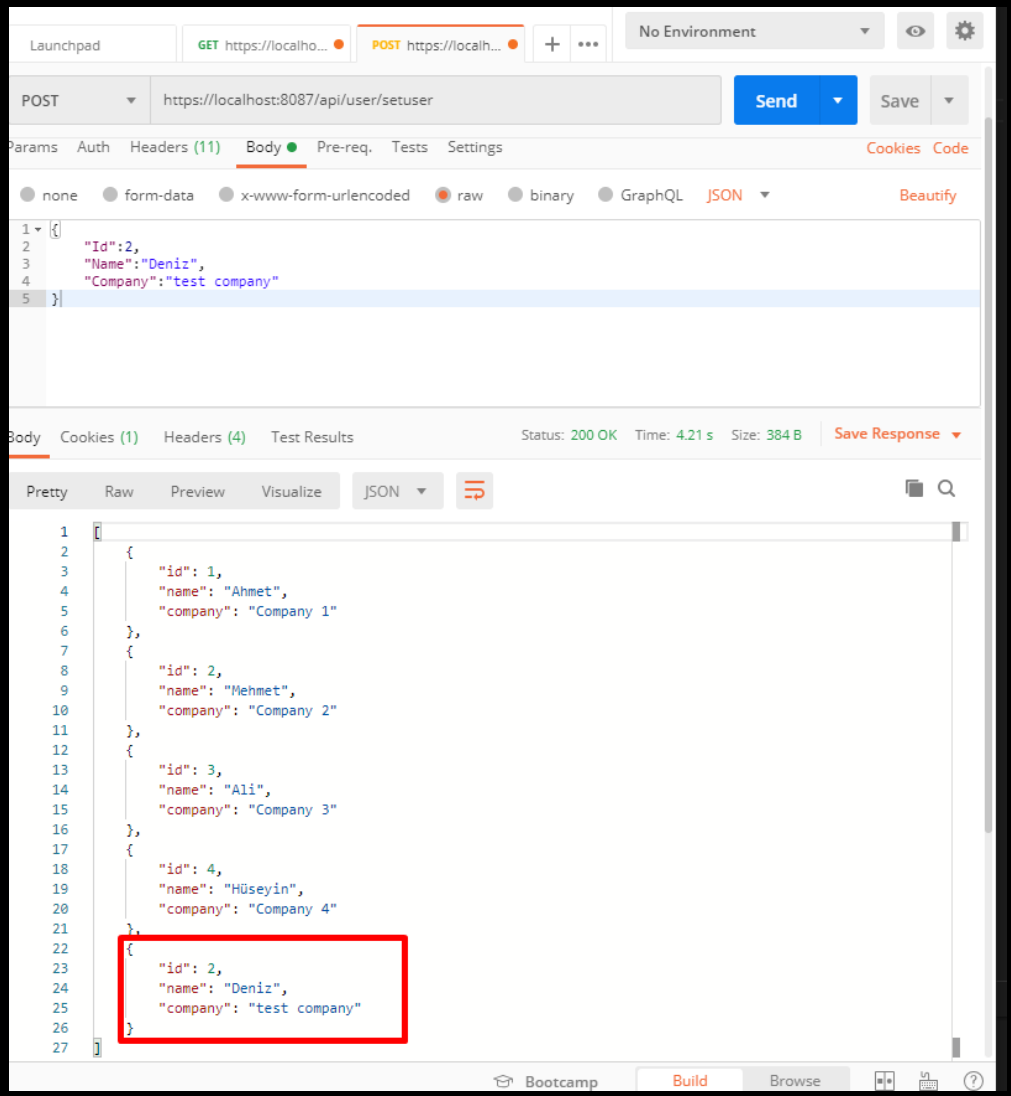
JSON data post edileceği zaman [FromForm] attiributeünü kullanmamalıyız.

```
[HttpPost ("setuser")]
public ActionResult SetUser (UserRequestModel model) {

    var users = GetUserList ();
    User userItem = new User () {
        Id = model.Id,
        Name = model.Name,
        Company = model.Company
    };

    users.Add (userItem);
    return Ok (users);
}
```

# Postman POST Request ve Response örneği

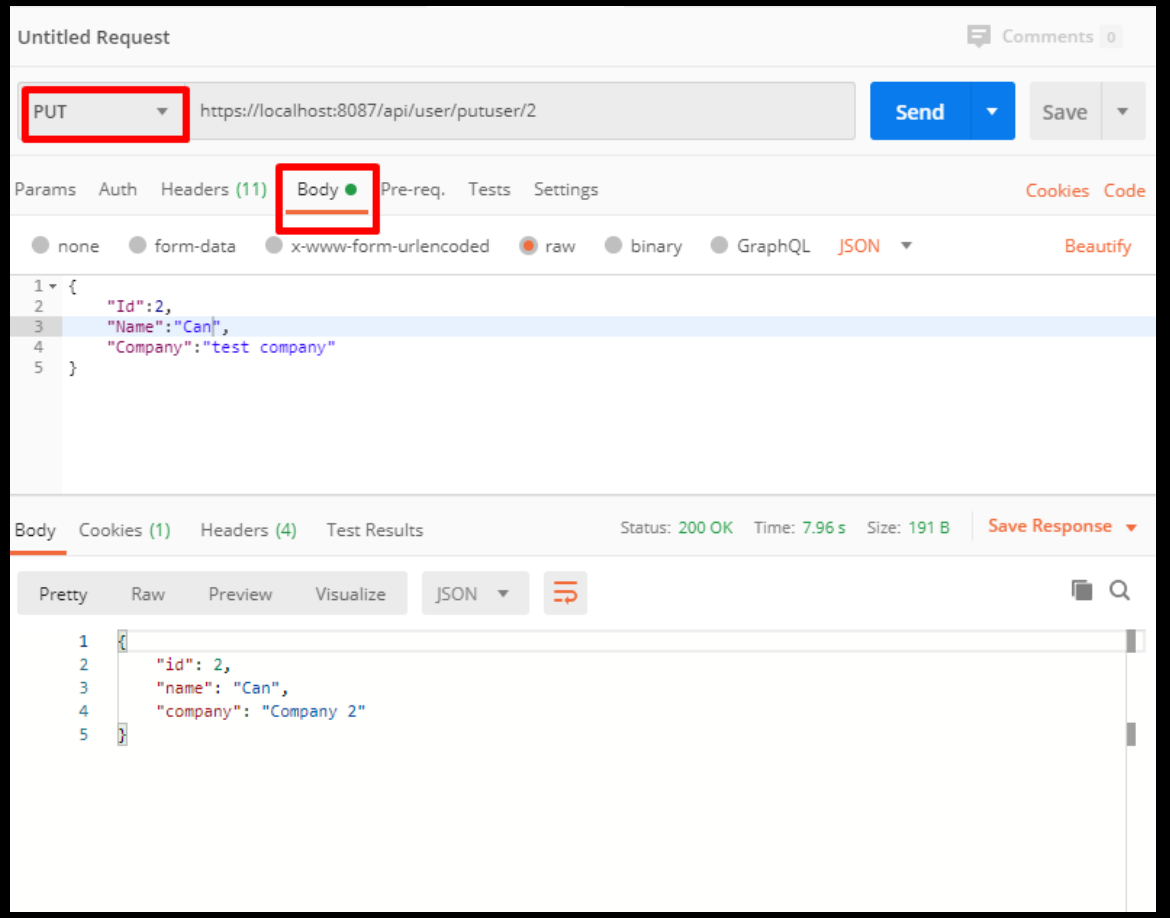


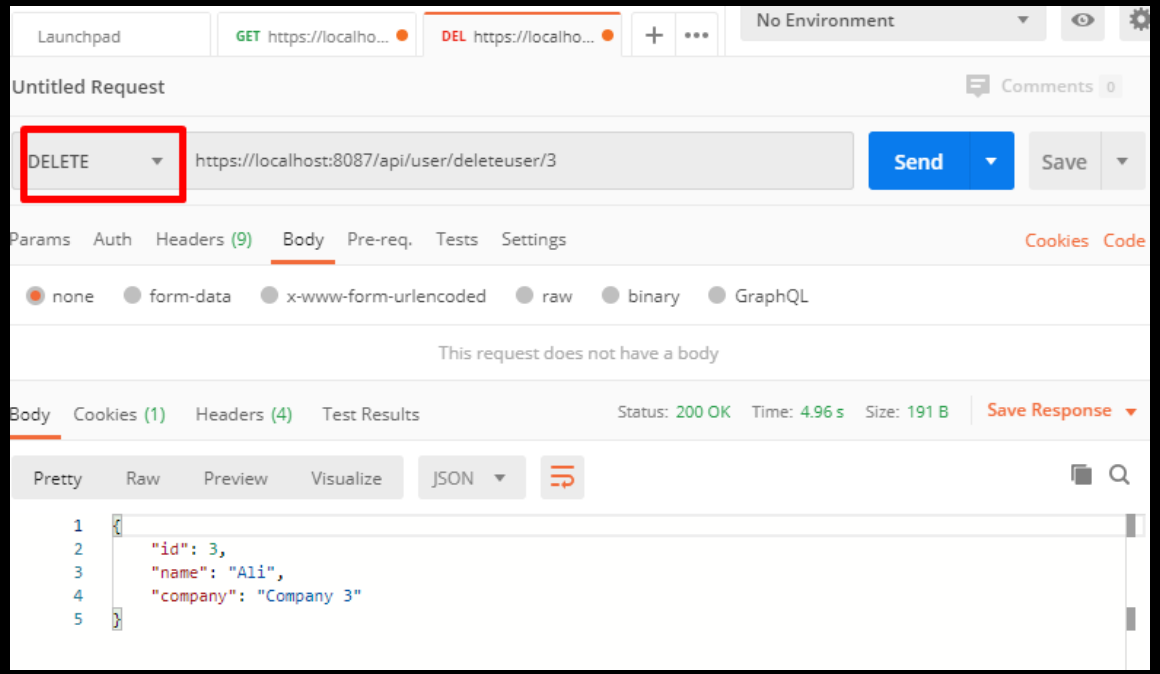
- **PUT: K**ayıtlı bir bilgiyi güncellemek için kullanılır. URL içerisinde güncellenmesi istenen kayda ait id ve Body gönderilir.

```
[HttpPut ("putuser/{id}")]
public IActionResult PutUser (int Id, UserRequestModel model) {
    var response = GetUserList ().Find (b => b.Id == Id);
    if (response == null) {
        return NotFound ();
    }
    response.Name = model.Name;
    return Ok (response);
}
```



- **DELETE:** Kayıtlı bir bilgiyi silmek için kullanılır. Request Body gönderilebilir, fakat genellikle sadece URL içerisinde silinecek bilgiye ait id bilgisi gönderilerek işlem tamamlanmaktadır.

```
[HttpDelete ("deleteuser/{id}")]
public IActionResult DeleteUser (int id) {
    var response = GetUserList ().Find (b => b.Id == id);
    if (response == null) {
        return NotFound ();
    }
    GetUserList ().Remove (response);
    return Ok (response);
}
```

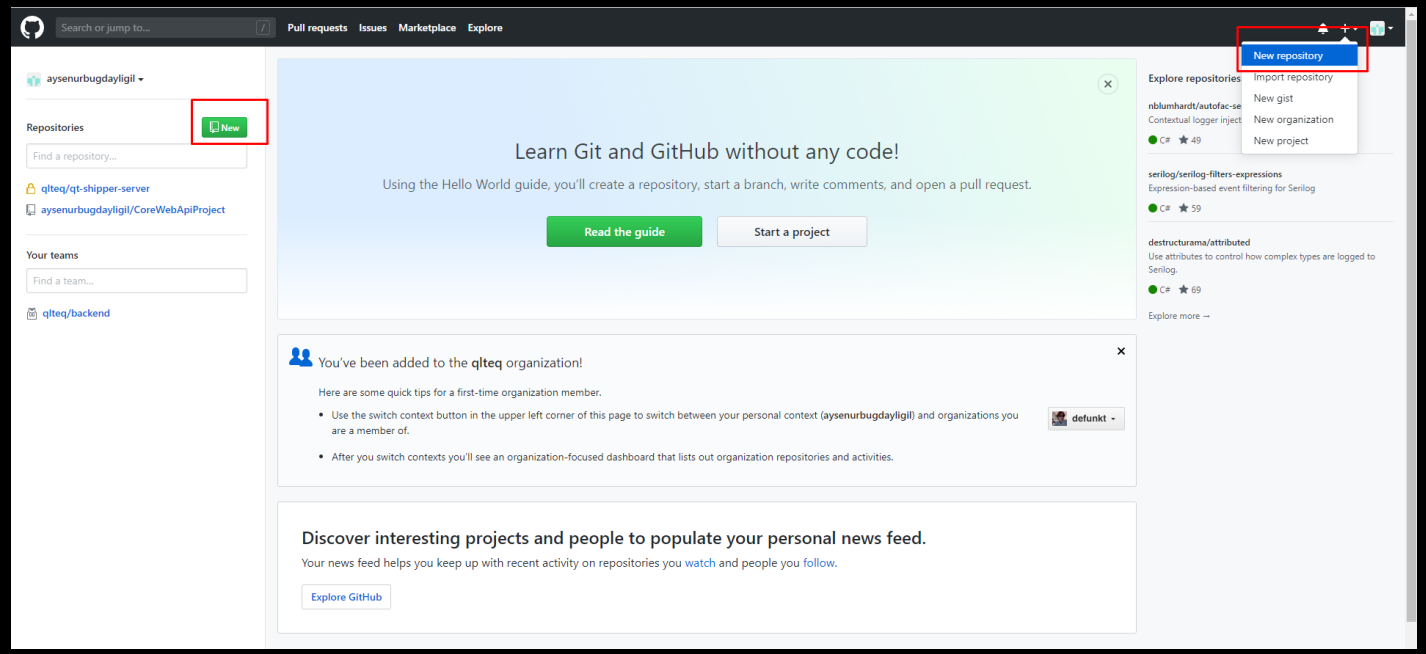


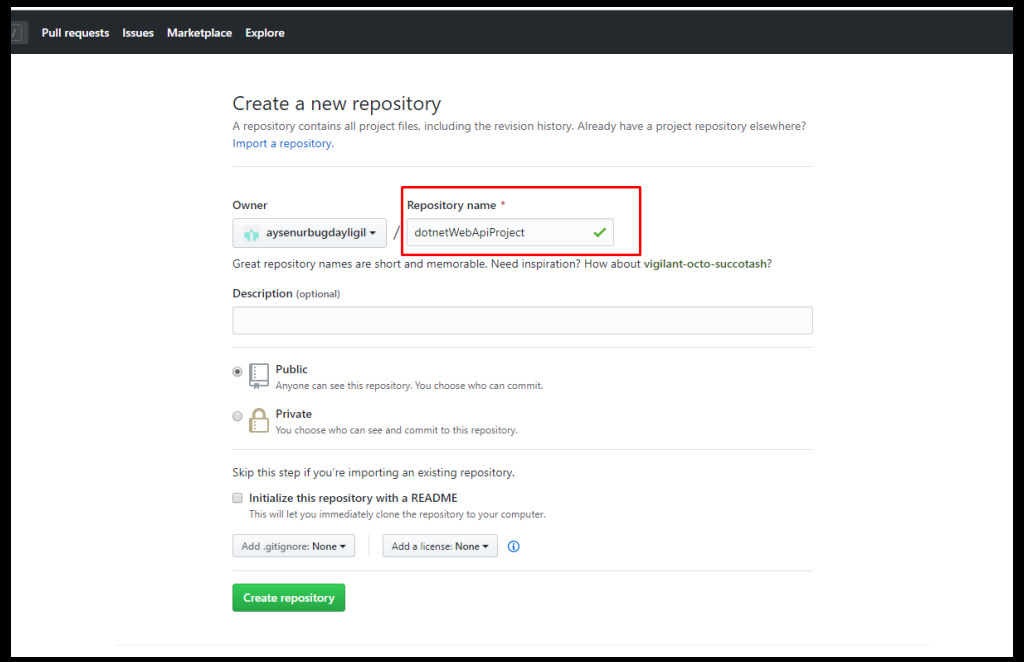
# Projeyi Githuba pushlama adımları

Adım 1: Git ve Visual Studio Code download
->>https://git-scm.com/downloads
->>https://code.visualstudio.com/download

Adım 2: Github hesabı oluşturma (Varolan hesapta kullanabilir.)

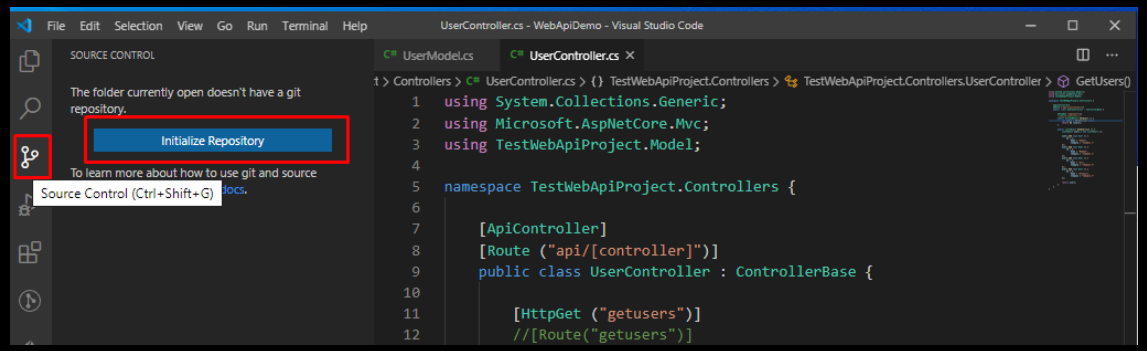
Adım 3: Login olduktan sonra repository oluşturma

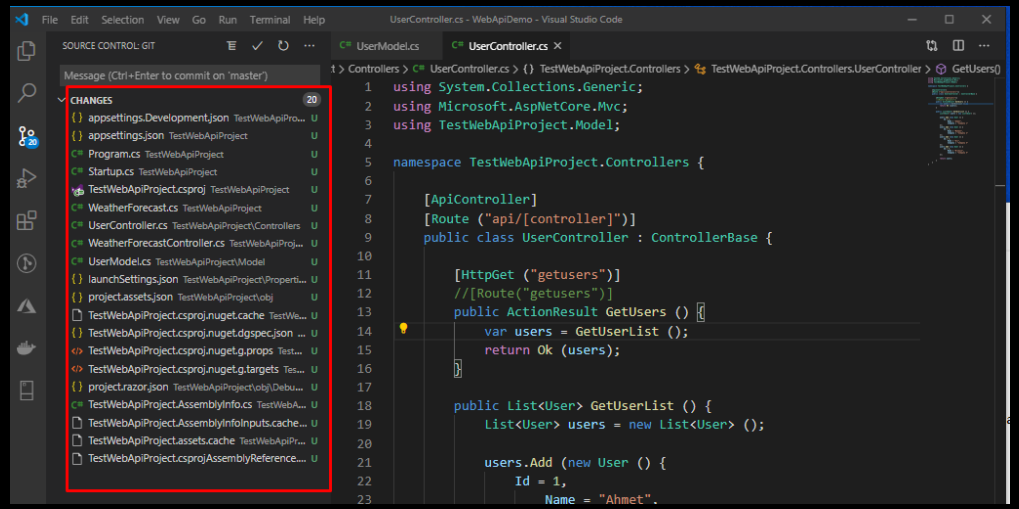


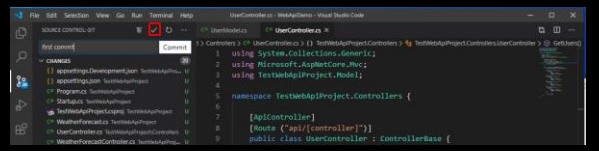
Adım 4: Visual Studio Code file->preferences->settings search arama kısmına git enabled yazıyoruz ve çekli olup olmadığını kontrol ediyoruz. çekli olması gerek.

Adım 5: source control -> initialize Repository butonuna tıklıyoruz. Changesları göreceğiz.



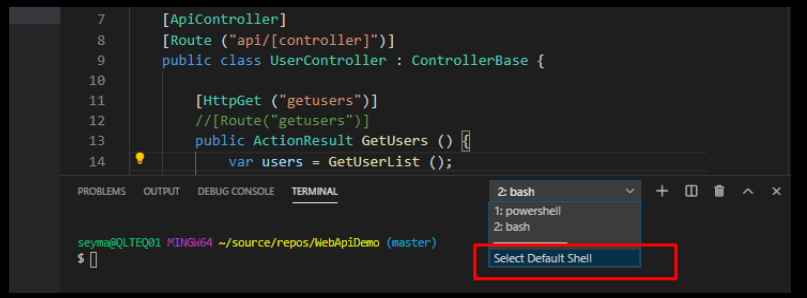



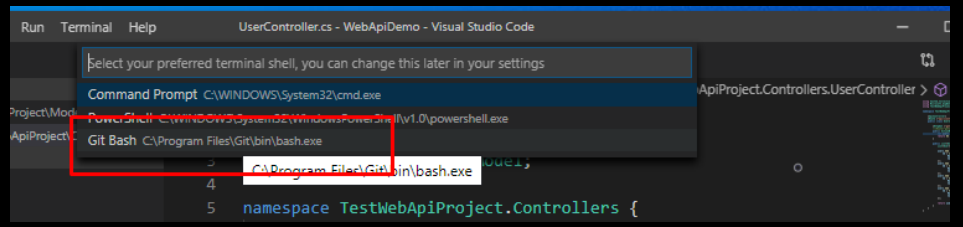

Adım 6: Changesların üzerindeki alana commit message i yazıp commit iconuna tıklanır.



Adım 7: "would you like to automatically stage all your changes and commit them directly" uyarı mesajı için evet i seçiyoruz.

Adım 8: Select Default Shell tıklıyoruz git bash seçilir.



Adım 9: Githuba giriş yapıyoruz aşağıdaki komutlarla

git config — global user.name “type your name here”
git config — global user.email typeyour@email.com
git config — global push.default matching
git config — global alias.co checkout

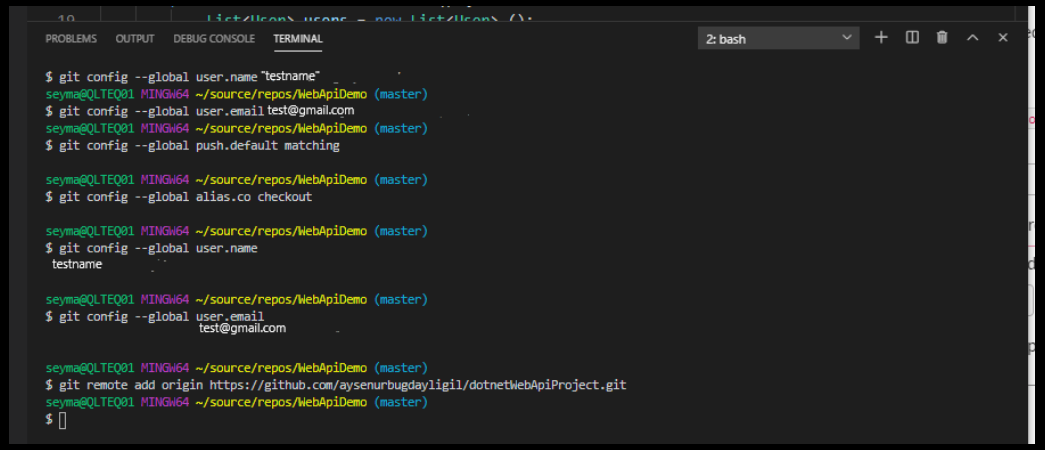
Bunları çalıştırdıktan sonra giriş yapıp yapmadığınızı kontrol etmek için aşağıdaki komutları
kullanabilirsiniz.

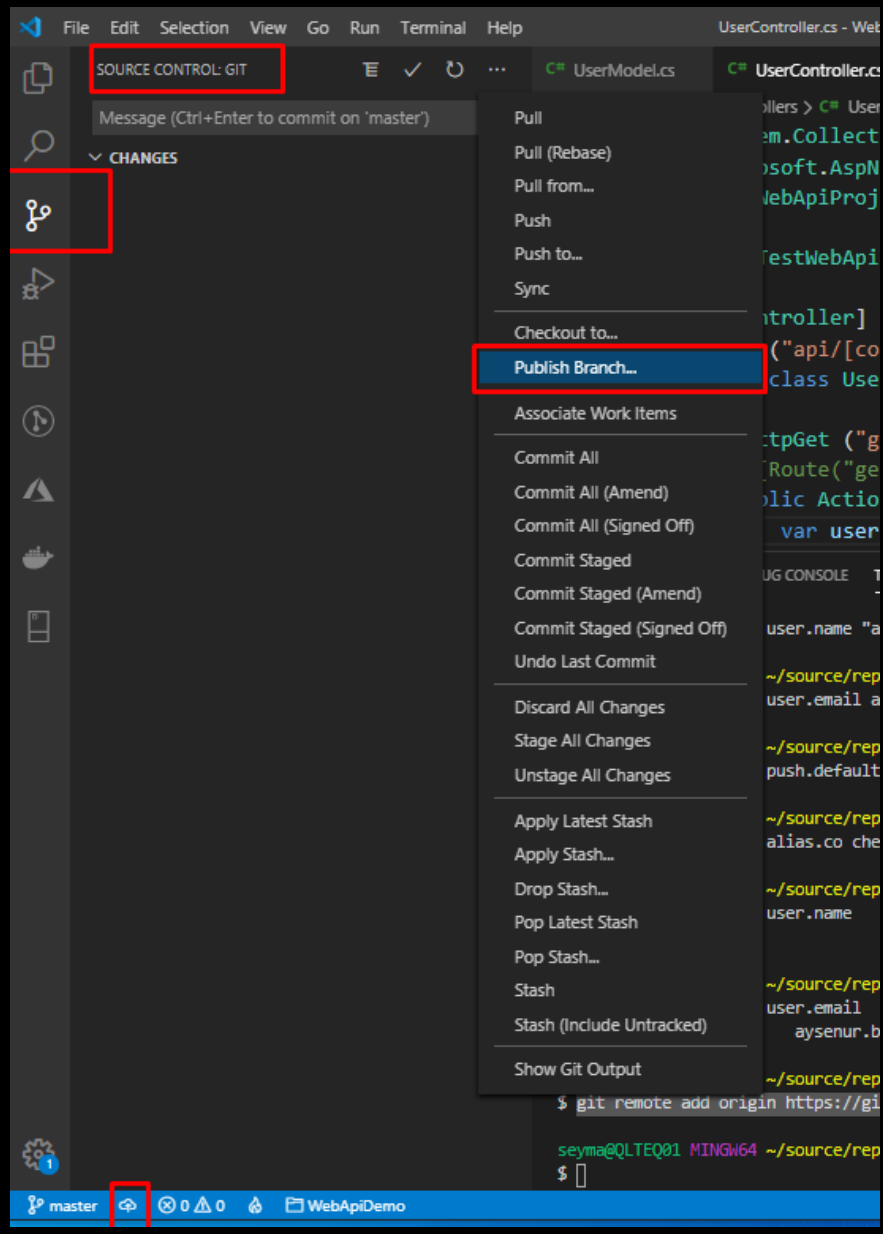
git config — global user.name
git config — global user.email

Adım 10: Add remote repository mizi (github repo) ekliyoruz. Aşağıdaki komutla

git remote add origin https://github.com/aysenurbugdayligil/dotnetWebApiProject.git

Adım 11: Publish branche tıklıyoruz.



# Kaynakça


- https://docs.microsoft.com/tr-tr/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio
- https://docs.microsoft.com/tr-tr/aspnet/core/web-api/?view=aspnetcore-3.1
- https://medium.com/net-core/how-to-build-a-restful-api-with-asp-net-core-fb7dd8d3e5e3
- https://medium.com/@muratsuzen/visual-code-ile-asp-net-core-web-api-ve-entityframework-core-kullan%C4%B1m%C4%B1-90d563132dd7
- https://medium.com/@powerupwebdev/deploy-your-projects-on-github-using-visual-studio-code-and-git-5f7221b272ca

Teşekkürler...

Muhammet Emre Kefelioğlu

Ayşenur Buğdaylıgil