API Tasks

Create a new API Solution

1. Create an ASP.NET Core Web Application
   a. Choose the API template
   b. Don't configure Https or Docker support

Add Swagger tooling (Open API Specification)

1. Add nuget package => Swashbuckle.AspNetCore
2. Configure Swagger Services => services.AddSwaggerGen(c => { c.SwaggerDoc("v1", new Info { Title = "Lists API", Version = "v1" });});
3. Enable Swagger => app.UseSwagger();
4. Configure Swagger UI => app.UseSwaggerUI(c => { c.SwaggerEndpoint("/swagger/v1/swagger.json", "Lists API V1"); });
5. Check Swagger UI => {root}/swagger/index.html

Links

Install Swagger https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-2.2&tabs=visual-studio

Create DocumentDB Repository

1. Add nuget package => Microsoft.Azure.DocumentDB.Core
2. Copy IDocumentDBRepository and DocumentDBRepository classes from quickstart UI solution
3. Copy Item.cs model from quick start UI solution
4. Configure IOC for Repo - services.AddSingleton<IDocumentDBRepository<Item>>(new DocumentDBRepository<Item>());

Create Items Controller

1. Create an Empty API Controller named ItemsController
2. Add Constructor to ItemsController

```
private readonly IDocumentDBRepository<Item> Respository;
public ItemsController(IDocumentDBRepository<Item> Respository)
{
   this.Respository = Respository;
}
```

3. Add REST Methods

```
[HttpGet]
public async Task<IEnumerable<Item>> GetAll()
{
   var items = await Respository.GetItemsAsync(d => !d.Completed);
```

```
    return items;
}

[HttpGet("{id}")]
public async Task<ActionResult> GetItem(string id)
{
    var items = await Respository.GetItemsAsync(x => x.Id == id);
    var item = items.FirstOrDefault();
    if (item == null)
        return NotFound();
    return Ok(item);
}

[HttpPost]
public async Task<ActionResult> Post([FromBody] Item value)
{
    var item = await Respository.CreateItemAsync(value);
    return Ok(item.Id);
}

[HttpPut("{id}")]
public async Task<ActionResult> Put(string id, [FromBody] Item value)
{
    await Respository.UpdateItemAsync(id, value);
    return Ok();

}

[HttpDelete("{id}")]
public async Task<ActionResult> Delete(string id)
{
    await Respository.DeleteItemAsync(id);
    return Ok();
}
```

4. Delete the Values Controller