Adding a basic health check

Health checks are a great way to report the health of your ASP.NET Core apps, and orchestrators like Kubernetes can take great advantage of them in your Production environment.

So, let's add a basic health check to our Identity microservice.

In Play.Identity repo

1. Update Startup:

```
public class Startup
{
   public void ConfigureServices(IServiceCollection services)
   {
      …
      services.AddSwaggerGen(c =>
      {
         …
      });

      services.AddHealthChecks();
   }

   public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
   {
      …
      app.UseEndpoints(endpoints =>
      {
         …
         endpoints.MapRazorPages();
         endpoints.MapHealthChecks("/health");
      });
   }
}
```

2. Start the service.

3. Run a GET request against /health in Postman

4. Notice it's healthy

5. Stop the service

6. Run a GET request against /health in Postman again

7. Notice connection refused

8. Commit and push

This can certainly tell us if the microservice is alive or not, but it doesn't say much about the services it depends on, like MongoDB or Cosmos DB.

So, in the next lesson you will create a custom database health check.