Adding the CORS middleware

Let's now fix the CORS related errors we are getting in the browser by configuring the CORS middleware in the request pipeline of our microservices.

In Play.Catalog

1. Update appsettings.Development.json

```
{
  "Logging": {
    …
  },
  "AllowedOrigin": "http://localhost:3000"
}
```

"I'm using appsettings.Development.json, and not appsettings.json, because the origin of the frontend is just a development server at this point, and we will not need a CORS policy for that origin in production."

2. Add AllowedOriginSetting constant to Startup:

```
public class Startup
{
    private const string AllowedOriginSetting = "AllowedOrigin";
…
```

3. Update Configure method in Startup:

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        ...
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Play.Catalog.Service
v1"));

        app.UseCors(builder =>
        {
            builder.WithOrigins(Configuration[AllowedOriginSetting])
                .AllowAnyHeader()
                .AllowAnyMethod();
        });
    }
    ...
}
```

In Play.Frontend

4. Reload Catalog section in the browser

5. Open the browser console → Network tab

6. Show the headers sent and received for CORS

7. Add a new item
   Name: Hi-Potion
   Description: Restores a medium amount of HP
   Price: 9

8. Show the preflight in action in the console

In Play.Inventory

9. Repeat 1-3

In the next lesson we will explore in more detail all the scenarios enabled in the front-end and how its react components are interacting with the microservices.