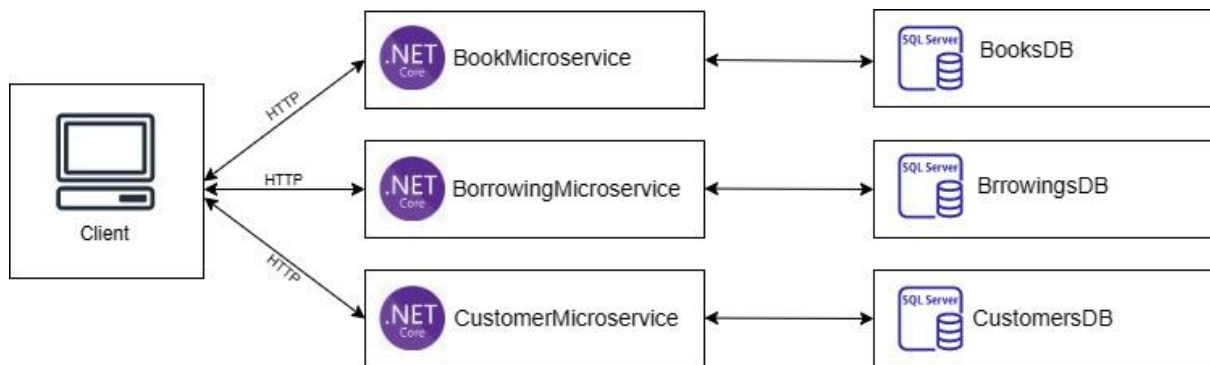


MICROSERVIZI

INFRASTRUTTURA



CREAZIONE DEI DB

I database sono stati creati su Microsoft SQL Server 2018 e sono rispettivamente Books, Borrowings e Customers.

I dati all'interno dei database sono visibili alla cartella Models di ogni microservizio.

CREAZIONE DEI MICROSERVIZI

Il mio progetto utilizza l'approccio Database-First, perciò utilizzerò un comando di scaffold per generare i modelli del database che ho creato .

- launchsettings.js alla pagina index di swagger

```
"$schema": "http://json.schemastore.org/launchsettings.json",
"profiles": {
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "launchUrl": "swagger/index.html",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
}
```

- Collegamento DB al microservizio

ASP.NET Core API Web App in Visual Studio 2019;
Installazione dei seguenti pacchetti:

Microsoft.EntityFrameworkCore.SqlServer -Version 5.0.17
Microsoft.EntityFrameworkCore.Design -Version 5.0.17
Microsoft.EntityFrameworkCore.Tools -Version 5.0.17

Inserimento della stringa di connessione del database nel seguente comando:

```
Scaffold-DbContext "Data Source=DESKTOP-G525CCI\SQLEXPRESS;Initial Catalog=Customers;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Context CoreDbContext -DataAnnotations
```

```
PM> Scaffold-DbContext "Data Source=DESKTOP-G525CCI\SQLEXPRESS;Initial Catalog=Customers;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Context CoreDbContext -DataAnnotations
Build started...
Build succeeded.
```

Con questo comando si crea automaticamente la cartella Models in cui sono presenti le classi che rappresentano i dati nel database.

Inserimento della stringa di connessione all'interno di appsettings.json:

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Information",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    },
9    "ConnectionStrings": {
10     "Database": "Data Source=DESKTOP-G525CCI\SQLEXPRESS;Initial Catalog=Customers;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"
11   },
12   "AllowedHosts": "*"
13 }
14
15
16
```

Nella classe Startup.cs, collegare il json della stringa di connessione tramite il framework di SqlServer:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "CustomerMicroservice", Version = "v1" });
    });
    services.AddDbContext<CustomerContext>(op => op.UseSqlServer(Configuration.GetConnectionString("Database"),
        sqlServerOptionsAction: SqliteDbContextOptionsBuilderExtensions =>
        {
            SqliteDbContextOptionsBuilderExtensions.EnableRetryOnFailure();
        }
    ));
}
```

Do il seguente comando per la migrazione dei dati:

```
dotnet ef migrations add Initial --context CustomerContext --project CustomerMicroservice
```

Ora il database è collegato al microservizio.

DESIGN PATTERN DEL PROGETTO

Il mio progetto segue due design patterns:

. **Feature-based organization:** Organizzazione per componenti: il codice del progetto è organizzato in base alle funzionalità della mia applicazione. Ho suddiviso in cartelle che suddividono le diverse parti dell'applicazione:

Controllers, in cui sono gestite le richieste HTTP

Helper, dentro il quale c'è tutto ciò che riguarda il mapping

Data, in cui è contenuto il contesto di dati a cui farà riferimento la logica di business

Model, generata automaticamente, racchiude le classi che rappresenteranno i record del database

Migrations, generata anch'essa automaticamente durante la fase di migrazione dei dati

Services, all'interno della quale sono contenuti i servizi per gestire la logica di business

. **Separation of Concerns:** Separazione dei concerns: il codice ha delle cartelle che suddividono le componenti in base agli ambiti che bisogna trattare all'interno del microservizio, così che le responsabilità del codice sono isolate e gestite separatamente:

Properties, in cui sono contenute le proprietà che gestiscono i vari ambienti di avvio del progetto

Helper, in cui si gestisce le funzionalità di Automapper

Migrations, in cui si gestisce la migrazione dei dati

VERIFICA FUNZIONAMENTO MICROSERVIZI

Chiamata **GET**

GET /api/Customers

200

Response body

```
[{"nome": "Rossella", "cognome": "Femia", "mail": "ross@gmail.com", "telefono": "3459284574"}, {"id": 2, "nome": "Alek ", "cognome": "Dimitrov", "mail": "alek@gmail.com", "telefono": "3481726354"}, {"id": 3, "nome": "Mattia", "cognome": "Morritti", "mail": "mattia@gmail.com", "telefono": "3485784928"}, {"id": 4, "nome": "Giovanna", "cognome": "Lisanti", "mail": "lisi@gmail.com", "telefono": "347193844"}]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 13 Jan 2023 11:05:12 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Chiamata **POST**

POST

/api/Customers

201

Parameters

No parameters

Request body

```
{
  "id": 5,
  "nome": "Giulia",
  "cognome": "Rosa",
  "mail": "rosa@gmail.com",
  "telefono": "3349283746"
}
```

Response body

5

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 13 Jan 2023 11:12:52 GMT
location: https://localhost:44329/api/Customers?id=5
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Chiamata **GET ONE**

GET

/api/Customers/{id}

Parameters

Name	Description
id ★ required	
integer(\$int32)	5
(path)	

200

Response body

```
{
  "id": 5,
  "nome": "Giulia",
  "cognome": "Rosa",
  "mail": "rosa@gmail.com",
  "telefono": "3349283746"
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 13 Jan 2023 11:14:05 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Chiamata **PUT**

PUT /api/Customers/{id}

Parameters

Cancel

Reset

Name	Description
id * required	
integer(\$int32)	5
(path)	

Request body

application/json

```
{  "id": 5,  "nome": "Giulia",  "cognome": "RosaModified",  "mail": "put@gmail.com",  "telefono": "3746374683"}
```

204

Response headers

```
date: Fri,13 Jan 2023 11:15:28 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```


Responses

(**Get** per verificare la modifica del dato)

200

Response body

```
{  "id": 5,  "nome": "Giulia",  "cognome": "RosaModified",  "mail": "rosa@gmail.com",  "telefono": "3349283746"}
```

 Download

Chiamata **DELETE**

DELETE /api/Customers/{id}

Parameters

Name	Description
id * required	
integer(\$int32)	5
(path)	

204

Response headers

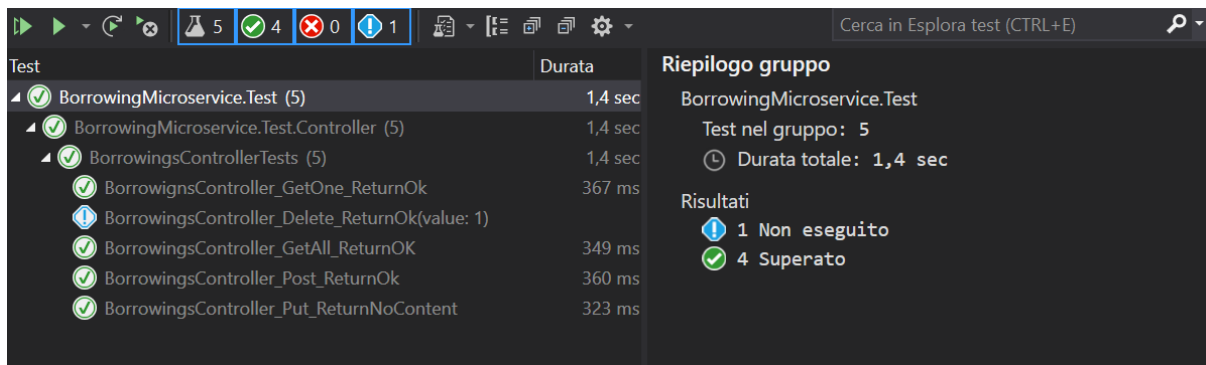
```
date: Fri,13 Jan 2023 11:16:28 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

CREAZIONE UNIT TESTS

Creazione progetto di unit test nella stessa soluzione del microservizio da testare. Ho utilizzato xUnit come Framework con il supporto di FakeItEasy per il mocking dei dati e Fluent Assertions per agevolare la scrittura dei test.

Sarà testato il controller di ogni progetto. Per le asserzioni sarà usato anche Assertions di xUnit.

Ogni test verifica che la response della chiamata testata avvenga correttamente e restituisca il giusto codice.



Tutti i test eseguiti vengono superati, tranne il test per la chiamata Delete che non riesce a venire eseguito. Probabilmente perché Delete rispetto agli altri metodi non è asincrono quindi probabilmente andrebbe scritto diversamente.

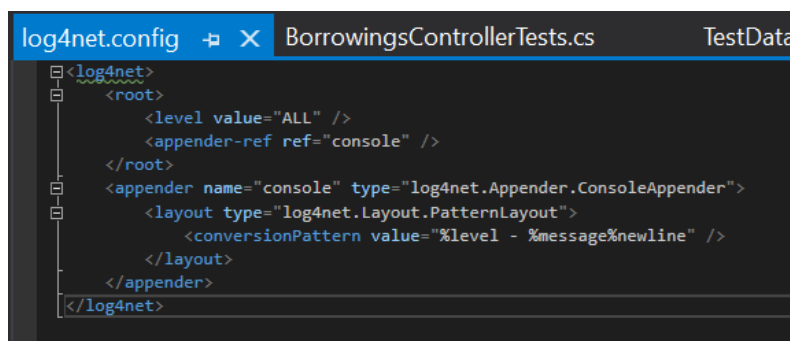
Questa correzione non è stata fatta per una questione logistica.

I test risultano allo stesso modo per tutti e tre i microservizi.

CREAZIONE LOG

Per la creazione dei log viene integrato al microservizio il pacchetto NuGet log4net.

Ho impostato la configurazione di log4net in un documento nella root del progetto



Ho aggiunto una classe AssemblyInfo per la costruzione dei log nella cartella Proprietà del progetto:

```
2
3 // Nei progetti di tipo SDK come questo diversi attributi di assembly che sono stati
4 // definiti cronologicamente in questo file vengono ora aggiunti automaticamente durante
5 // la compilazione e popolati con i valori definiti nelle proprietà del progetto.
6 // Per informazioni dettagliate sugli attributi inclusi e su come personalizzare questo
7 // processo, vedere: https://aka.ms/assembly-info-properties
8
9
10 // Se si imposta ComVisible su false, i tipi in questo assembly non saranno visibili
11 // ai componenti COM. Se è necessario accedere a un tipo in questo assembly da COM,
12 // impostare su true l'attributo ComVisible per tale tipo.
13
14 [assembly: ComVisible(false)]
15
16 // Se il progetto viene esposto a COM, il GUID seguente verrà usato come ID di typelib.
17
18 [assembly: Guid("bd528c1a-be62-4b84-9b0e-96698a1c5040")]
19 [assembly: log4net.Config.XmlConfigurator(ConfigFile = "log4net.config")]
```

Nella classe controller ho aggiunto la seguente riga di codice per integrare i log:

```
public class BorrowingsController : ControllerBase
{
    private readonly IBorrowingServices _services;
    private static log4net.ILog Log = log4net.LogManager.GetLogger(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);
}
```

E poi ad ogni chiamata del controller ed ogni percorso che essa può prendere ho aggiunto un messaggio di log

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
1 riferimento | 1/1 superati
public async Task<ActionResult> Post([FromBody] BorrowingsBaseModel value)
{
    if (value == null)
    {
        Log.Error("BorrowingMicroservice:[Post] Create fields are null");
        return BadRequest("Borrowing null");
    }
    Log.Info("BorrowingMicroservice:[Post] Create succeeded");
    var id = await _services.Create(value);

    return CreatedAtAction(nameof(GetAll), new { id = id }, id);
}
```

Ora eseguendo il microservizio, non solo comparirà la pagina di `swagger/index.html`, ma anche la console che stamperà ogni azione eseguita.

```
C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2\BorrowingMicroservice\BorrowingMicroservice>
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2\BorrowingMicroservice\BorrowingMicroservice
INFO - BorrowingMicroservice:[GetAll] Read all succeeded
INFO - BorrowingMicroservice:[GetAll] Time required to create a record: 0ms
INFO - BorrowingMicroservice:[Post] Create succeeded
INFO - BorrowingMicroservice:[Create] Time required to create a record: 187ms
INFO - BorrowingMicroservice:[GetOne] Read one succeeded. ID: 7
INFO - BorrowingMicroservice:[GetOne] Time required to create a record: 0ms
INFO - BorrowingMicroservice:[GetOne] Time required to create a record: 0ms
INFO - BorrowingMicroservice:[Put] Update one succeeded. ID: 7
INFO - BorrowingMicroservice:[Update] Time required to create a record: 0ms
INFO - BorrowingMicroservice:[GetOne] Read one succeeded. ID: 7
INFO - BorrowingMicroservice:[GetOne] Time required to create a record: 0ms
INFO - BorrowingMicroservice:[Delete] Delete succeeded
INFO - BorrowingMicroservice:[GetOne] Time required to create a record: 0ms
INFO - CustomerMicroservice:[Delete] Time required to delete a record: 0ms
INFO - BorrowingMicroservice:[Delete] Delete succeeded
INFO - BorrowingMicroservice:[GetOne] Time required to create a record: 0ms
ERROR - BorrowingMicroservice:[Delete] Book does not exist
```

CONTAINERIZZAZIONE

Per creare un'immagine docker dei microservizi è necessario creare un dockerfile all'interno della root del progetto

```
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
ENV ASPNETCORE_URLS http://*:5020

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["BookMicroservice/BookMicroservice.csproj", "BookMicroservice/"]
RUN dotnet restore "BookMicroservice/BookMicroservice.csproj"
COPY . .
WORKDIR "/src/BookMicroservice"
RUN dotnet build "BookMicroservice.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "BookMicroservice.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "BookMicroservice.dll"]
```


- . **FROM** prende l'immagine di base per un container di partenza:
mcr.microsoft.com/dotnet/aspnet:3.1;
- . **WORKDIR** specifica il percorso all'interno del container:
/app;
- . **EXPOSE** indica le porte che il container espone:
80 e 443;
- . **ENV** imposta una variabile d'ambiente per il container:
ASPNETCORE_URLS: http://5020
- . **FROM** specifica un'altra immagine di base per la costruzione dell'immagine:
mcr.microsoft.com/dotnet/sdk:3.1
- . **WORKDIR** specifica il percorso in questo container:
/src
- . **COPY** crea una copia del file BookMicroservice.csproj nella cartella BookMicroservice nel container;
- . **RUN** esegue il comando dotnet restore sul file BookMicroservice.csproj così che le dipendenze del progetto siano ripristinate;
- . **COPY . .** copia tutti i file nella directory corrente all'interno del container;
- . **WORKDIR** specifica il percorso di lavoro all'interno del container:
/src/BookMicroservice
- . **RUN** esegue il comando dotnet build sul file BookMicroservice.csproj con l'opzione -c Release e -o /app/build, così da creare la build del progetto
- . **FROM** specifica di utilizzare la fase base con nome final;
- . **WORKDIR** specifica il percorso all'interno del container, final
- . **COPY** copia la cartella publish dalla fase publish all'interno del container
- . **ENTRYPOINT** specifica il comando da eseguire quando il container viene avviato:
dotnet BookMicroservice.dll

Build del container:

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2\BookMicroservice> docker build -t saramorritti/bookmicroservice:v4 -f BookMicroservice/Dockerfile .
[+] Building 0.8s (18/18) FINISHED
```

Run dell'immagine:

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2\BookMicroservice> docker run -it --rm -p 5020:5020 saramorritti/bookmicroservice:v4
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://[::]:5020
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

Avvio l'immagine con il link <http://localhost:5020/swagger/index.html>.

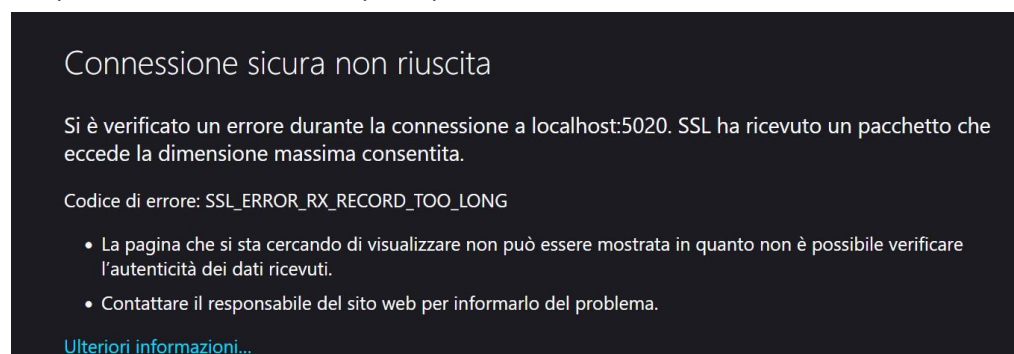
L'errore sul browser Microsoft edge è il seguente:



L'errore avviene poiché il browser accetta solo pagine https. La mia immagine docker non ha un certificato SSL..

La soluzione è o di utilizzare OpenSSL per generare una chiave SSL, o di utilizzare un altro browser, ho scelto Firefox.

In questo caso l'errore è più specifico:



Ciò indica che il server web sta inviando un pacchetto che supera la dimensione massima consentita. Probabilmente il mio server web non è configurato correttamente, quindi anche utilizzando OpenSSL non risolverei il problema. Dovrei integrare i miei microservizi con il server web di nginx, che è compatibile con .net 3.1, ma purtroppo per una questione logistica, non farò in tempo entro il a correggere questo errore senza rinunciare a provare anche le fasi successive di questo corso.

Push dei miei container su docker hub:

```
Microservice> docker push saramorritti/bookmicroservice:v2
The push refers to repository [docker.io/saramorritti/bookmicroservice]
```

```
Microservice> docker push saramorritti/borrowingmicroservice:v2
The push refers to repository [docker.io/saramorritti/borrowingmicroservice]
```

```
Microservice> docker push saramorritti/customermicroservice:v1
The push refers to repository [docker.io/saramorritti/customermicroservice]
```

Le immagini ora sono su docker hub:

The screenshot shows the Docker Hub profile for 'saramorritti'. At the top, there is a search bar with 'saramorritti' entered, a search button, and a dropdown menu set to 'All Content'. To the right is a blue 'Create repository' button. Below this, three repository cards are listed:

- saramorritti / customermicroservice**: Contains: Image | Last pushed: a few seconds ago. Status: Not Scanned, 0 stars, 0 downloads, Public.
- saramorritti / borrowingmicroservice**: Contains: Image | Last pushed: a minute ago. Status: Not Scanned, 0 stars, 0 downloads, Public.
- saramorritti / bookmicroservice**: Contains: Image | Last pushed: 2 minutes ago. Status: Not Scanned, 0 stars, 1 download, Public.

KUBERNETES

Download e installazione di Minikube dal repository di github.

Da powershell, imposto la variabile d'ambiente:

```
PS C:\WINDOWS\system32> $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
>> if ($oldPath.Split(';') -notcontains 'C:\minikube'){ `
>> [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine) `
>> }
```

Do il comando minikube start

```
PS C:\WINDOWS\system32> minikube start
* minikube v1.28.0 on Microsoft Windows 11 Home 10.0.22000 Build 22000
```

(...)

```
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Ora creo il file di configurazione yaml per creare i miei deployment di Kubernetes:

```
! microservices-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: microservices-deployment
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        app: mymicroservices
10   template:
11     metadata:
12       labels:
13         app: mymicroservices
14     spec:
15       containers:
16       - name: bookmicroservice
17         image: saramorritti/bookmicroservice:v4
18       - name: borrowingmicroservice
19         image: saramorritti/borrowingmicroservice:v2
20       - name: customermicroservice
21         image: saramorritti/customermicroservice:v1
```

Applico il deployment:

```
PS C:\Ukubectl apply -f microservices-deployment.yaml
```

```
deployment.apps/microservices-deployment created
```

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl describe deployment microservices-deployment
Name:                microservices-deployment
Namespace:            default
CreationTimestamp:    Fri, 20 Jan 2023 12:21:18 +0100
Labels:               <none>
Annotations:          deployment.kubernetes.io/revision: 1
Selector:             app=mymicroservices
Replicas:             3 desired | 3 updated | 3 total | 0 available | 3 unavailable
StrategyType:         RollingUpdate
MinReadySeconds:      0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=mymicroservices
  Containers:
    Type           Status    Reason
    ----           -
    Progressing    True      NewReplicaSetAvailable
    Available      False     MinimumReplicasUnavailable
OldReplicaSets: <none>
NewReplicaSet:  microservices-deployment-745f89c6d8 (3/3 replicas created)
Events:
  Type     Reason              Age    From                      Message
  ----     -
  Normal   ScalingReplicaSet   6m11s  deployment-controller     Scaled up replica set microservices-deployment-745f89c6d8 to 3
```

Verifico che il mio deployment sia stato creato:

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl get deployment
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
microservices-deployment           0/3      3              0            22s
```

Verifico i pods:

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl get pod
NAME                                READY    STATUS              RESTARTS    AGE
microservices-deployment-745f89c6d8-4mv77    2/3      Error                5 (88s ago)  4m3s
microservices-deployment-745f89c6d8-5wh5p    2/3      CrashLoopBackOff     4 (84s ago)  4m3s
microservices-deployment-745f89c6d8-vm248    2/3      CrashLoopBackOff     4 (88s ago)  4m3s
```

Faccio il log dei pods:

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl logs microservices-deployment-745f89c6d8-4mv77
Defaulted container "bookmicroservice" out of: bookmicroservice, borrowingmicroservice, customermicroservice
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://[::]:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl logs microservices-deployment-745f89c6d8-5wh5p
Defaulted container "bookmicroservice" out of: bookmicroservice, borrowingmicroservice, customermicroservice
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://[::]:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

```
PS C:\Users\MMorr\OneDrive\Desktop\Microservices\Microservices2> kubectl logs microservices-deployment-745f89c6d8-vm248
Defaulted container "bookmicroservice" out of: bookmicroservice, borrowingmicroservice, customermicroservice
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://[::]:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

RABBITMQ

Per implementare il broker di messaggistica RabbitMQ è necessario installare Erlang 25.x o 24.3.x e successivamente RabbitMQ dal sito ufficiale.

Una volta eseguita l'installazione e creato l'account su RabbitMQ è necessario implementare il servizio all'interno dei miei microservizi. Per fare ciò occorre l'installazione del pacchetto NuGet RabbitMQ.Client 6.4.0.

A causa del poco tempo rimasto e dell'effettiva difficoltà dei processi, non sono riuscita ad addentrarmi ulteriormente in questa tecnologia.

Nonostante l'installazione di RabbitMQ e l'impostazione della sua variabile d'ambiente, non sono riuscita a far funzionare RabbitMQ sul mio computer.