

ARQUITECTURA DE SOFTWARE (2S-2023)

PROYECTO CORTE 2 (MICROSERVICIOS)

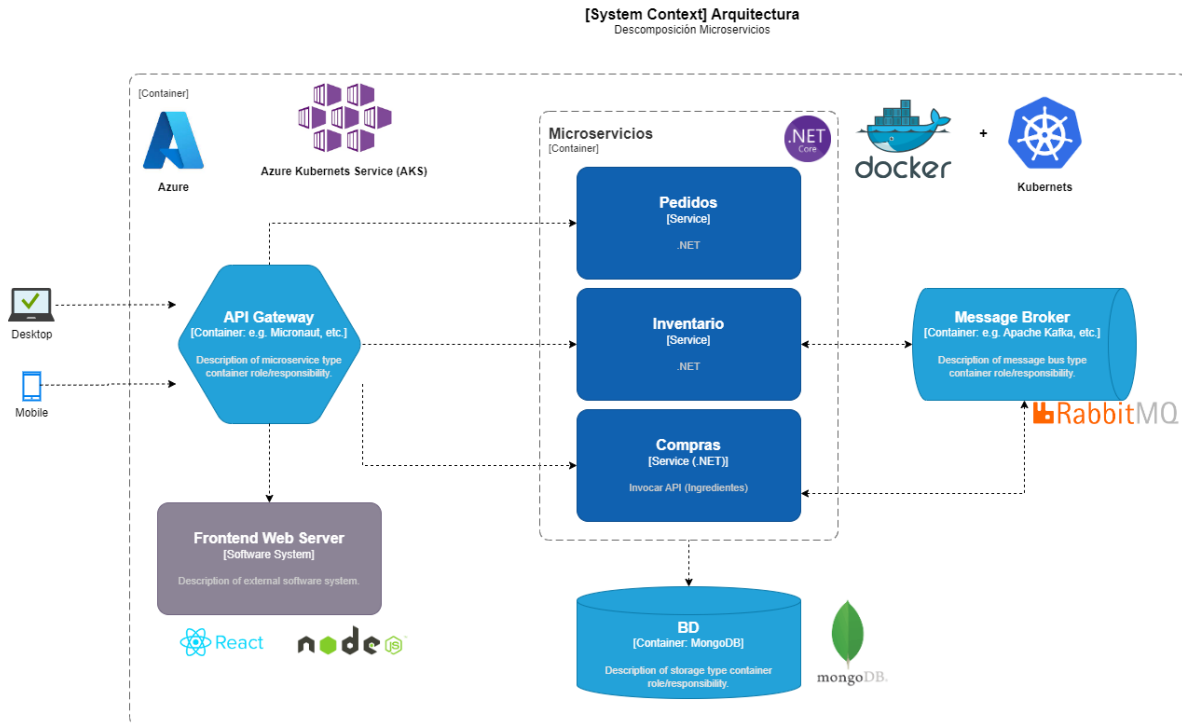
EDISSON CABRERA ERASO

NICOLAS NIÑO BERNAL

Tabla de contenido

Arquitectura Propuesta	2
API's	2
MongoDB.....	3
Docker Compose (yaml)	3
Iniciar Docker Mongo	4
Comunicación asíncrona entre microservicios.....	5
Message Broker (RabbitMQ + MassTransit)	5
Enviar mensaje mediante RabbitMQ	9
RabbitMQ Console	9
Enviar y recibir mensajes entre microservicios a través de RabbitMQ.....	10
Frontend (React).....	10
Hosted (Node.js) Web Server	11
CORS (Cross-Origin Resource Sharing)	11
Para la comunicación entre el frontend y los microservicios.....	11
Consumir API plaza de mercado desde API propia	12
Interconexión entre microservicios.....	13
Recetas	13
Deploy App NETLIFY	15
GitHub Repositorio	16

Arquitectura Propuesta



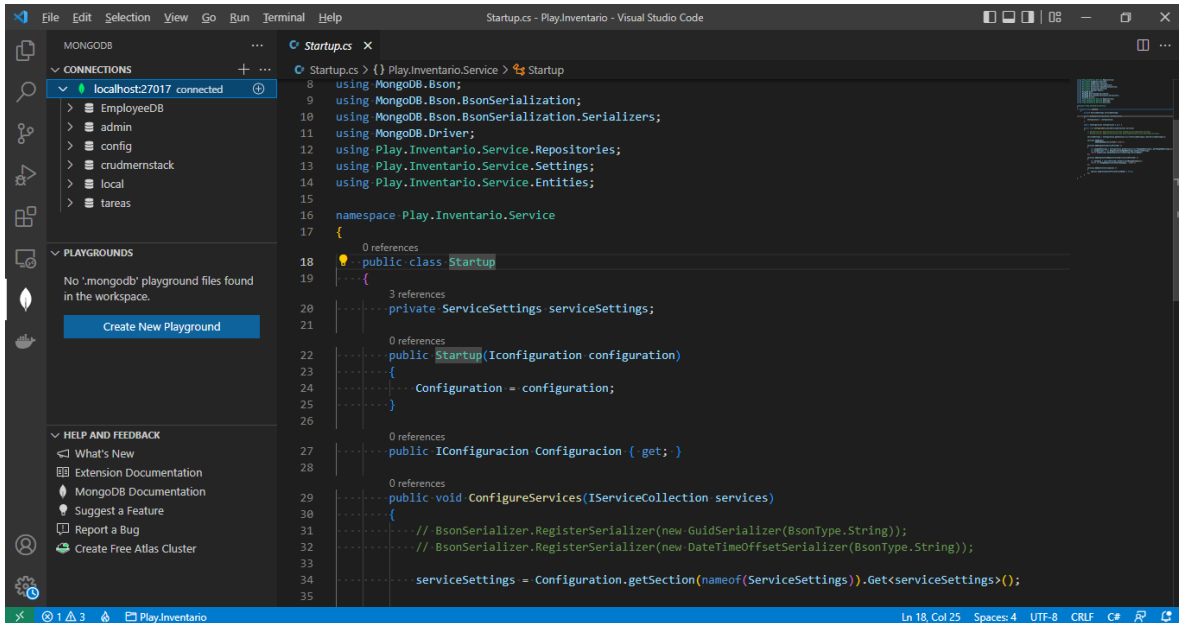
API's

```

File Edit Selection View Go Run Terminal Help
ItemsController.cs - Play.Inventario - Visual Studio Code

src > Play.Inventario.Service > Controllers > ItemsController.cs > {} Play.Inventario.Service.Controllers > ItemsController > GetAsync()
41
42 [HttpGet("{id}")]
43     1 reference
44     public async Task<ActionResult<ItemDto>> GetByIdAsync(Guid id)
45     {
46         var item = await itemsRepository.GetAsync(id);
47         if (item == null)
48             return NotFound();
49         return item.AsDto();
50     }
51
52
53 [HttpPost]
54     0 references
55     public async Task<ActionResult<ItemDto>> PostAsync(CreatedItemDto createItemDto)
56     {
57         var item = new Item
58         {
59             Name = createItemDto.Name,
60             Description = createItemDto.Description,
61             Price = createItemDto.Price,
62             CreatedDate = DateTimeOffset.UtcNow
63         };
64         await itemsRepository.CreateAsync(item);
65         return CreatedAtAction(nameof(GetByIdAsync), new { id = item.Id }, item);
66     }
67
68
69 [HttpPut("{id}")]
70     0 references
71     public async Task<ActionResult> Put(Guid id, UpdateItemDto updateItemDto)
  
```

MongoDB

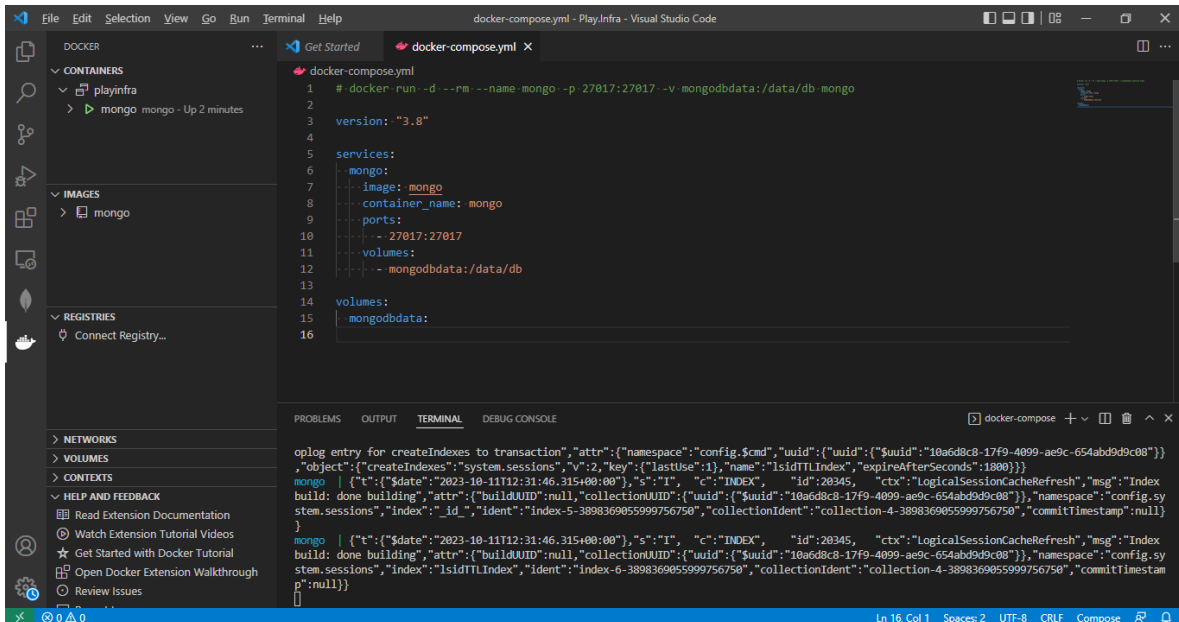


The screenshot shows the Visual Studio Code interface. On the left, the 'MongoDB' sidebar is open, showing a connection to 'localhost:27017' with a database named 'EmployeeDB'. The main editor displays the 'Startup.cs' file for the 'Play.Inventario' project. The code includes MongoDB drivers and a 'Startup' class that configures services and registers serializers.

```

Startup.cs
1 using MongoDB.Bson;
2 using MongoDB.Bson.BsonSerialization;
3 using MongoDB.Bson.BsonSerialization.Serializers;
4 using MongoDB.Driver;
5 using Play.Inventario.Service.Repositories;
6 using Play.Inventario.Service.Settings;
7 using Play.Inventario.Service.Entities;
8
9 namespace Play.Inventario.Service
10 {
11     0 references
12     public class Startup
13     {
14         3 references
15         private ServiceSettings serviceSettings;
16
17         0 references
18         public Startup(IConfiguration configuration)
19         {
20             Configuration = configuration;
21         }
22
23         0 references
24         public IConfiguration Configuracion { get; }
25
26         0 references
27         public void ConfigureServices(IServiceCollection services)
28         {
29             // BsonSerializer.RegisterSerializer(new GuidSerializer(BsonType.String));
30             // BsonSerializer.RegisterSerializer(new DateTimeOffsetSerializer(BsonType.String));
31             serviceSettings = Configuration.GetSection(nameof(ServiceSettings)).Get<ServiceSettings>();
32         }
33     }
34 }
  
```

Docker Compose (yaml)



The screenshot shows the Visual Studio Code interface with a Docker Compose file named 'docker-compose.yml'. The file defines a 'mongo' service using the 'mongo' image, container name, ports, and volumes. The terminal at the bottom shows the output of the 'docker-compose up' command, indicating that the MongoDB service is running successfully.

```

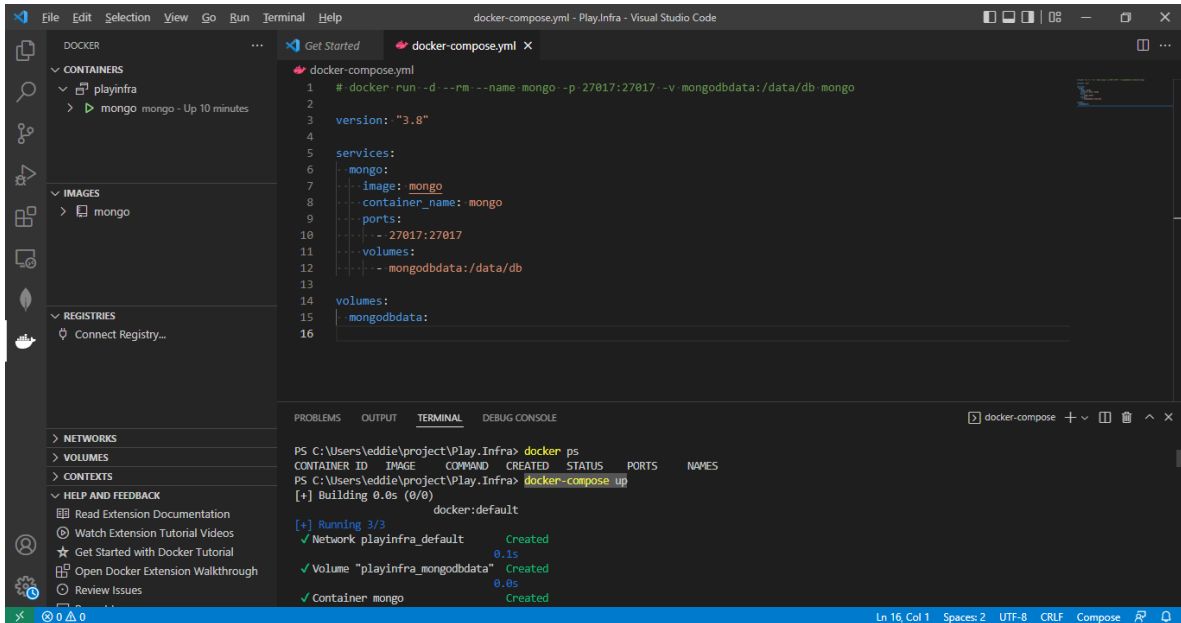
docker-compose.yml
1 # docker run -d --rm --name mongo -p 27017:27017 -v mongodbddata:/data/db mongo
2
3 version: "3.8"
4
5 services:
6   mongo:
7     image: mongo
8     container_name: mongo
9     ports:
10      - 27017:27017
11     volumes:
12      - mongodbddata:/data/db
13
14 volumes:
15   mongodbddata:
16
  
```

Terminal Output:

```

oplog entry for createIndexes to transaction, "attr": {"namespace": "config.$cmd", "uid": {"uid": {"$uid": "10a6d8c8-17f9-4099-ae9c-654abd9d9c08"}}}, "object": {"createIndexes": "system.sessions", "v": 2, "key": {"lastUse": 1}, "name": "lsidTTLIndex", "expireAfterSeconds": 1800}}
mongo | {"t":{"date":"2023-10-11T12:31:46.315400:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh","msg":"Index build: done building", "attr":{"buildUUID":null,"collectionUUID":{"uid":{"$uid":"10a6d8c8-17f9-4099-ae9c-654abd9d9c08"}}, "namespace":"config.system.sessions", "index":{"id","ident":"Index-5-389836985999756750","collectionId":"collection-4-389836985999756750","commitTimestamp":null}}
mongo | {"t":{"date":"2023-10-11T12:31:46.315400:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh","msg":"Index build: done building", "attr":{"buildUUID":null,"collectionUUID":{"uid":{"$uid":"10a6d8c8-17f9-4099-ae9c-654abd9d9c08"}}, "namespace":"config.system.sessions", "index":{"lsidTTLIndex", "ident":"Index-6-389836985999756750","collectionId":"collection-4-389836985999756750","commitTimestamp":null}}
  
```

Iniciar Docker Mongo

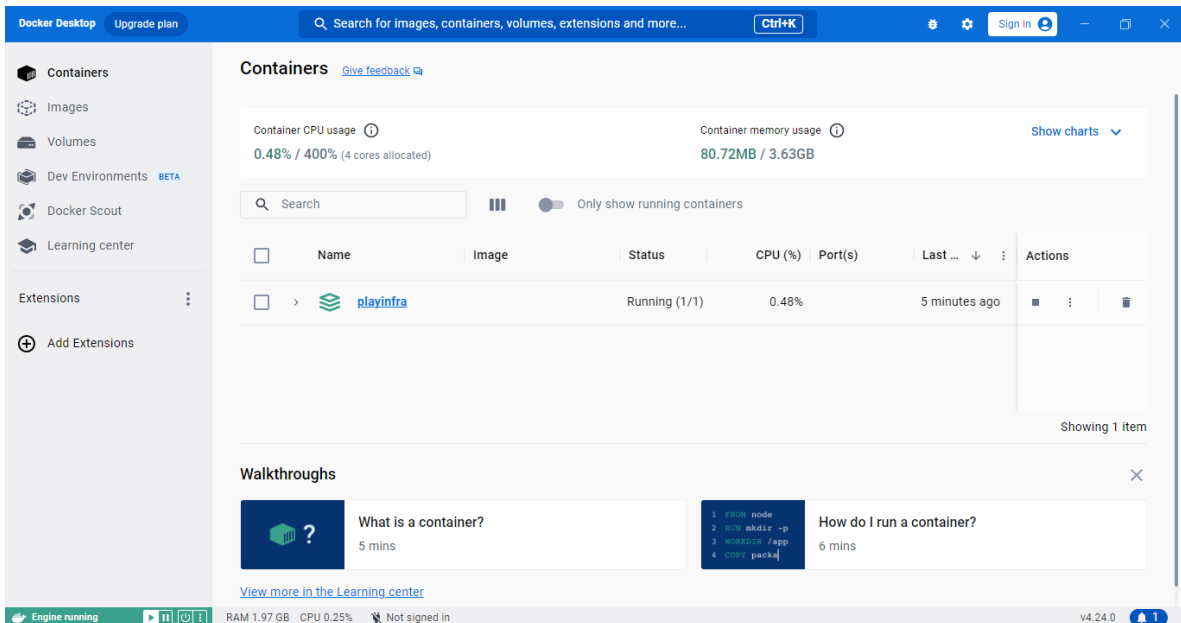


The screenshot shows the Visual Studio Code editor with the `docker-compose.yml` file open. The file contains the following configuration:

```
1 # docker: run -d --rm --name mongo -p 27017:27017 --v mongodbddata:/data/db mongo
2
3 version: "3.8"
4
5 services:
6   mongo:
7     image: mongo
8     container_name: mongo
9     ports:
10      - 27017:27017
11     volumes:
12      - mongodbddata:/data/db
13
14 volumes:
15   mongodbddata:
```

The terminal output shows the command `docker-compose up` being executed, resulting in the creation of the `mongo` container and the `mongodbddata` volume.

```
PS C:\Users\eddie\project\Play.Infra> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Users\eddie\project\Play.Infra> docker-compose up
[+] Building 0.0s (0/0)
docker:default
[+] Running 3/3
  ✓ Network playinfra_default      Created
  ✓ Volume "playinfra_mongodbddata" Created
  ✓ Container mongo                Created
```



The screenshot shows the Docker Desktop interface. The **Containers** tab is active, displaying a table with one running container named `playinfra`.

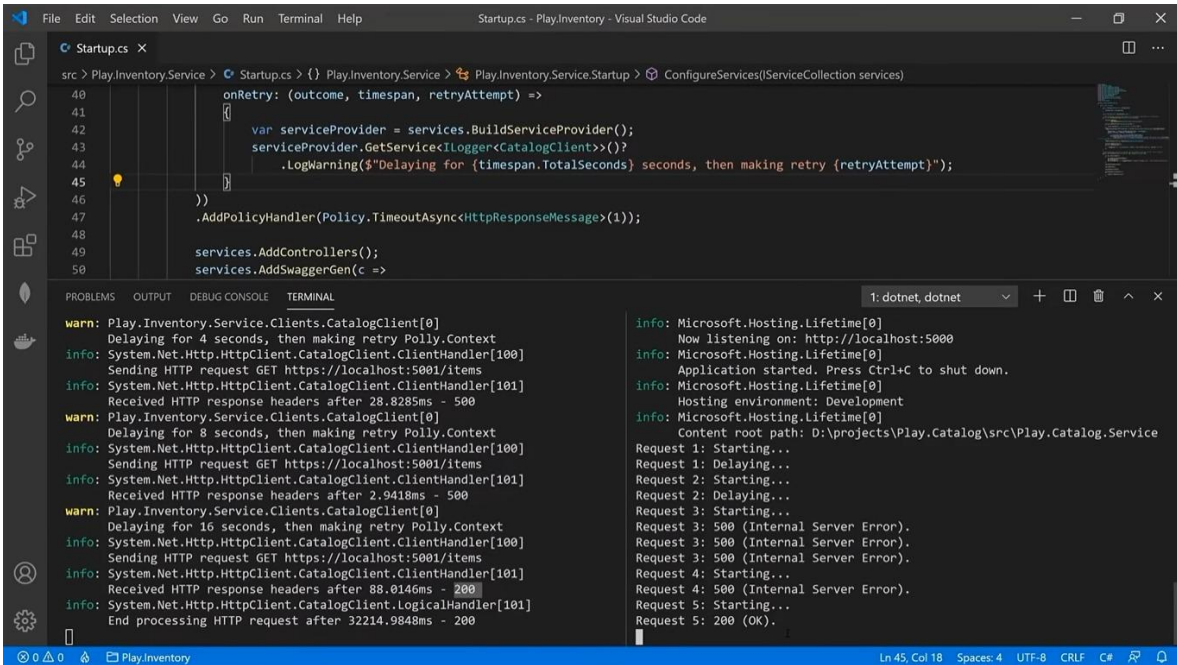
Name	Image	Status	CPU (%)	Port(s)	Last ...	Actions
> playinfra		Running (1/1)	0.48%		5 minutes ago	[Stop] [Restart] [Delete]

Below the table, there are two walkthroughs:

- What is a container?** (5 mins)
- How do I run a container?** (6 mins)

The bottom status bar shows the engine is running, with RAM at 1.97 GB and CPU at 0.25%.

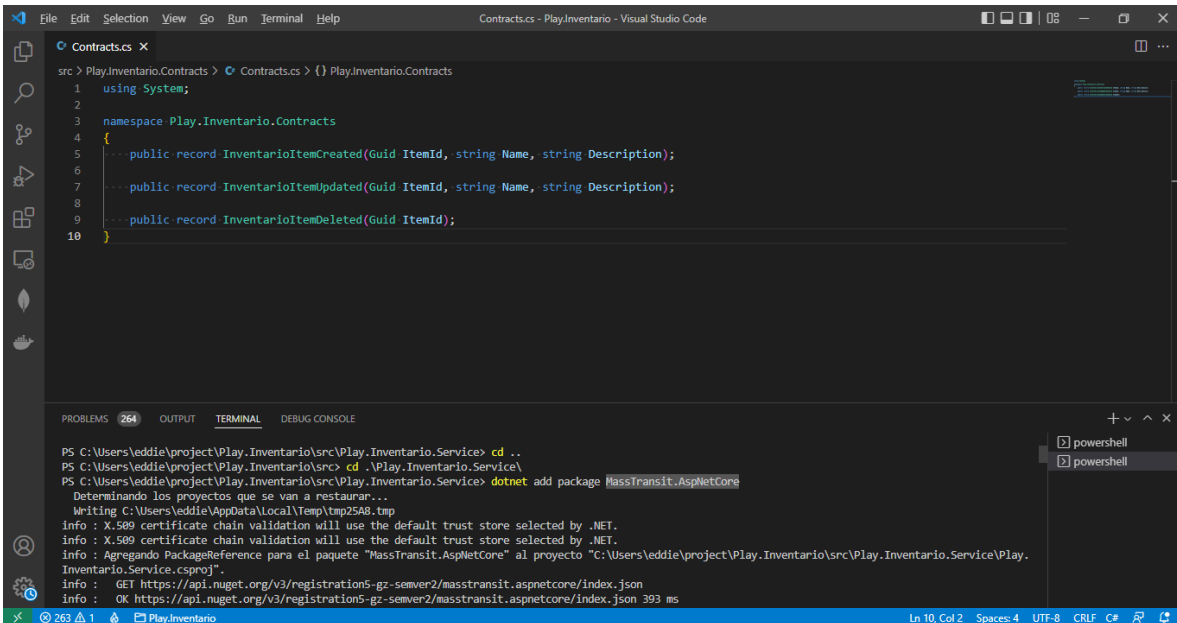
Comunicación asíncrona entre microservicios



The screenshot shows the Visual Studio Code editor with the `Startup.cs` file open. The code defines a `ConfigureServices` method for `Play.Inventory.Service`. It includes a `retryPolicy` for `HttpClient` calls to `CatalogClient`, which delays for 4 seconds and retries up to 5 times. The `services.AddControllers()` and `services.AddSwaggerGen()` are also present.

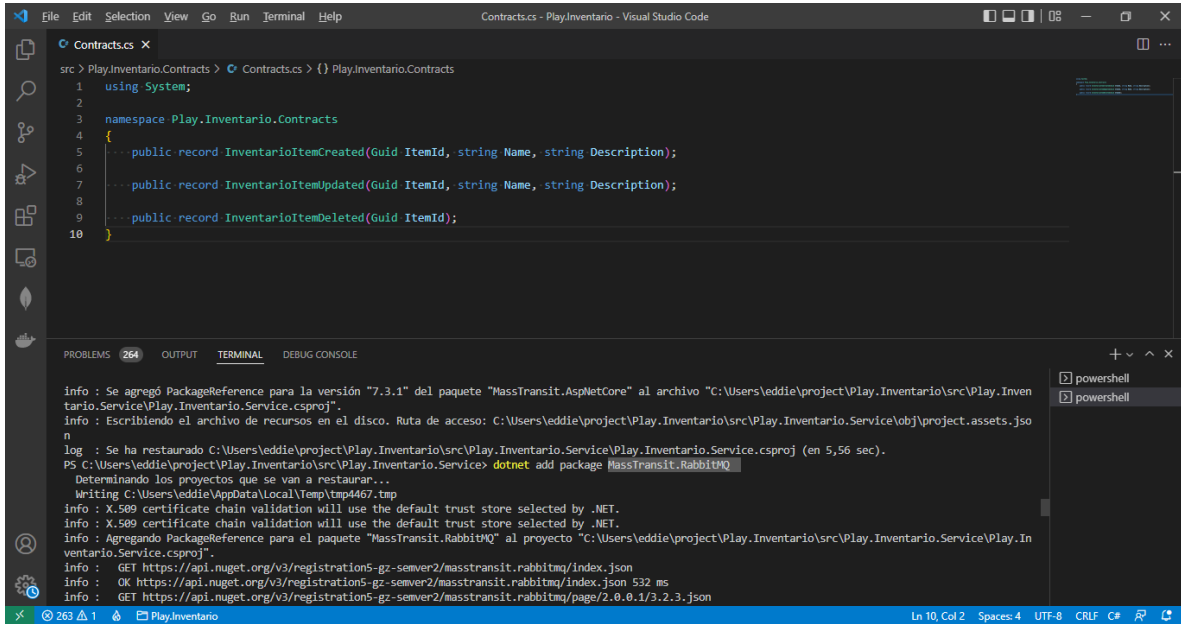
The terminal output shows the application running on `http://localhost:5000`. It displays several log messages, including warnings about delays and retries, and information about the application's startup and the first successful request (Request 5: 200 OK).

Message Broker (RabbitMQ + MassTransit)



The screenshot shows the Visual Studio Code editor with the `Contracts.cs` file open. The code defines a `namespace Play.Inventario.Contracts` and includes three `public record` types: `InventarioItemCreated`, `InventarioItemUpdated`, and `InventarioItemDeleted`.

The terminal output shows the application running on `http://localhost:5000`. It displays several log messages, including warnings about delays and retries, and information about the application's startup and the first successful request (Request 5: 200 OK).

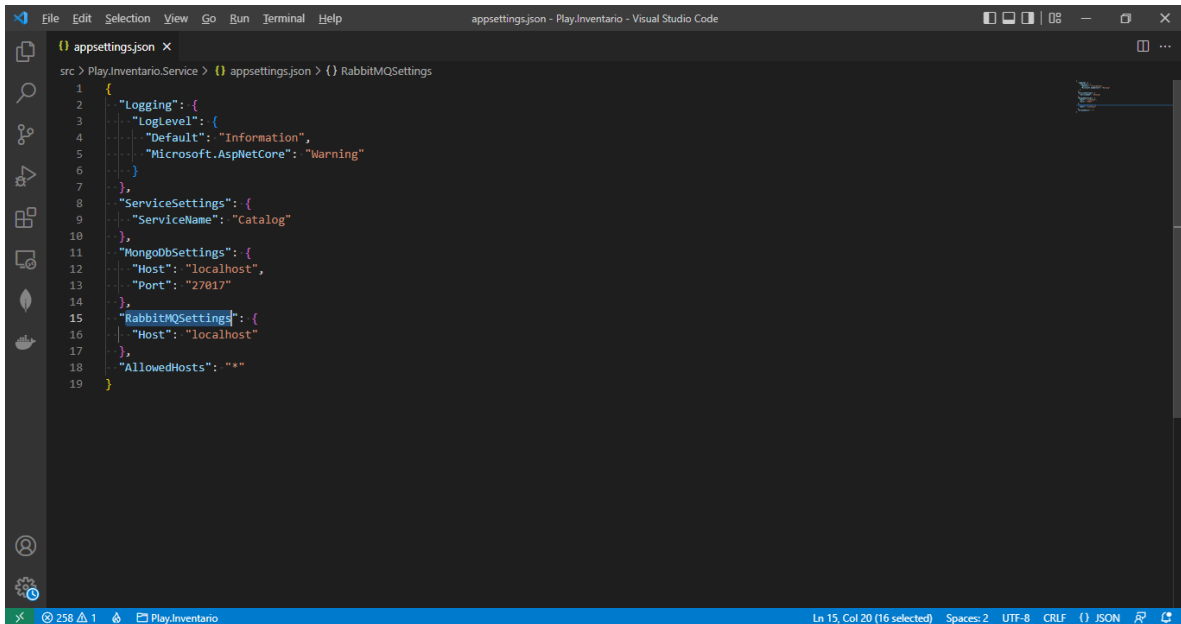


Visual Studio Code interface showing the `Contracts.cs` file in the `Play.Inventario` project. The file contains the following code:

```
1 using System;
2
3 namespace Play.Inventario.Contracts
4 {
5     public record InventarioItemCreated(Guid ItemId, string Name, string Description);
6
7     public record InventarioItemUpdated(Guid ItemId, string Name, string Description);
8
9     public record InventarioItemDeleted(Guid ItemId);
10 }
```

The bottom panel shows the `TERMINAL` output, displaying information about package references and the installation of `MassTransit.RabbitMQ`.

```
info : Se agregó PackageReference para la versión "7.3.1" del paquete "MassTransit.AspNetCore" al archivo "C:\Users\eddie\project\Play.Inventario\src\Play.Inventario\Service\Play.Inventario.Service.csproj".
info : Escribiendo el archivo de recursos en el disco. Ruta de acceso: C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service\obj\project.assets.json
log : Se ha restaurado C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service\Play.Inventario.Service.csproj (en 5,56 sec).
PS C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service> dotnet add package MassTransit.RabbitMQ
Determinando los proyectos que se van a restaurar...
Writing C:\Users\eddie\AppData\Local\Temp\tmp4467.tmp
info : X.509 certificate chain validation will use the default trust store selected by .NET.
info : X.509 certificate chain validation will use the default trust store selected by .NET.
info : Agregando PackageReference para el paquete "MassTransit.RabbitMQ" al proyecto "C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service\Play.Inventario.Service.csproj".
info : GET https://api.nuget.org/v3/registration5-gz-semver2/masstransit.rabbitmq/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/masstransit.rabbitmq/index.json 532 ms
info : GET https://api.nuget.org/v3/registration5-gz-semver2/masstransit.rabbitmq/page/2.0.0.1/3.2.3.json
```



Visual Studio Code interface showing the `appsettings.json` file in the `Play.Inventario` project. The file contains the following configuration:

```
1 {
2   "Logging": {
3     "LogLevel": {
4       "Default": "Information",
5       "Microsoft.AspNetCore": "Warning"
6     }
7   },
8   "ServiceSettings": {
9     "ServiceName": "Catalog"
10  },
11  "MongoDbSettings": {
12    "Host": "localhost",
13    "Port": "27017"
14  },
15  "RabbitMQSettings": {
16    "Host": "localhost"
17  },
18  "AllowedHosts": "*"
19 }
```

docker-compose.yml - Play.Infra - Visual Studio Code

```

1  image: mongo
2  container_name: mongo
3  ports:
4  - 27017:27017
5  volumes:
6  - mongobdata:/data/db
7
8  rabbitmq:
9  image: rabbitmq:management
10 container_name: rabbitmq
11 ports:
12 - 5672:5672
13 - 15672:15672
14 volumes:
15 - rabbitmqdata:/var/lib/rabbitmq
16 hostname: rabbitmq
17
18 volumes:
19 mongobdata:
20 rabbitmqdata:
  
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

p*:null}}
 Gracefully stopping... (press Ctrl+C again to force)
 Aborting on container exit...
 [+] Stopping 1/1
 ✓ Container mongo Stopped 2.8s
 canceled
 PS C:\Users\eddie\project\Play.Infra> docker-compose up -d
 [+] Building 0.0s (0/0)
 [+] Running 1/1
 ✓ Container mongo Started 0.0s
 PS C:\Users\eddie\project\Play.Infra>

Ln 22, Col 23 Spaces: 2 UTF-8 CRLF Compose

docker-compose.yml - Play.Infra - Visual Studio Code

```

14 rabbitmq:
15 image: rabbitmq:management
16 container_name: rabbitmq
17 ports:
18 - 5672:5672
19 - 15672:15672
20 volumes:
21 - rabbitmqdata:/var/lib/rabbitmq
22 hostname: rabbitmq
23
24 volumes:
25 mongobdata:
26 rabbitmqdata:
27
  
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

✓ Container mongo Started 0.0s
 PS C:\Users\eddie\project\Play.Infra> docker-compose up -d
 [+] Running 13/1
 ✓ rabbitmq 12 layers [██████████] 0B/0B Pulled 43.0s
 [+] Building 0.0s (0/0)
 [+] Running 3/3
 ✓ Volume "playinfra_rabbitmqdata" Created 0.0s
 ✓ Container rabbitmq Started 1.3s
 ✓ Container mongo Started 0.0s
 PS C:\Users\eddie\project\Play.Infra> docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
541f27020ea7	rabbitmq:management	"docker-entrypoint.s..."	18 seconds ago	Up 14 seconds	4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp
04f1dea64185	mongo	"docker-entrypoint.s..."	13 hours ago	Up 14 seconds	0.0.0.0:27017->27017/tcp

PS C:\Users\eddie\project\Play.Infra>

Ln 25, Col 1 Spaces: 2 UTF-8 CRLF Compose

Docker Desktop

Upgrade plan

Search for images, containers, volumes, extensions and more...

Ctrl+K

Sign in

Containers

Images

Volumes

Dev Environments BETA

Docker Scout

Learning center

Extensions

+

Add Extensions

Containers

Give feedback

Container CPU usage

108.65% / 400% (4 cores allocated)

Container memory usage

263.38MB / 3.63GB

Show charts

Search

Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	<div>playinfra</div>		Running (2/2)	108.65%		53 seconds ago	<div></div> <div>:</div> <div></div>
<input type="checkbox"/>	<div>mongo</div>	<div>04f1dea64185</div> <div>mongo</div>	Running	0.64%	<div>27017:27017</div>	53 seconds ago	<div></div> <div>:</div> <div></div>
<input type="checkbox"/>	<div>rabbitmq</div>	<div>541f27028ea7</div> <div>rabbitmq:management</div>	Running	108.01%	<div>15672:15672</div> <div>Show all ports (2)</div>	53 seconds ago	<div></div> <div>:</div> <div></div>

Showing 3 items

Walkthroughs

What is a container?

5 mins

How do I run a container?

6 mins

View more in the Learning center

Engine running

RAM 2.66 GB

CPU 55.61%

Not signed in

v4.24.0

1

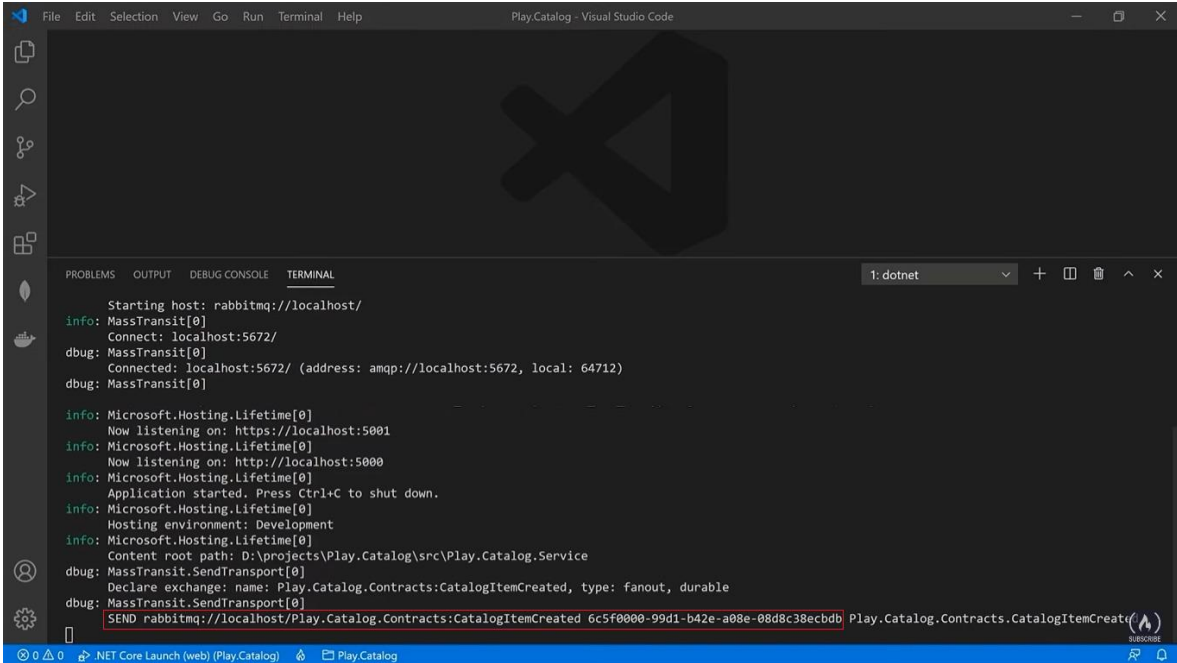
File Edit View Go Run Terminal Help Play.Catalog - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: dotnet

```
Building...
dbug: MassTransit.Registration.BusDepot[0]
      Starting bus instances: IBus
dbug: MassTransit[0]
      Starting host: rabbitmq://localhost/
info: MassTransit[0]
      Connect: localhost:5672/
dbug: MassTransit[0]
      Connected: localhost:5672/ (address: amqp://localhost:5672, local: 64712)
dbug: MassTransit[0]
      Endpoint Ready: rabbitmq://localhost/JULIODESKTOP_PlayCatalogService_bus_ptxoyyr34g4nh71zbdc8dwhyi?temporary=true
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: D:\projects\Play.Catalog\src\Play.Catalog.Service
```

0 0 0 NET Core Launch (web) (Play.Catalog) Play.Catalog

Enviar mensaje mediante RabbitMQ



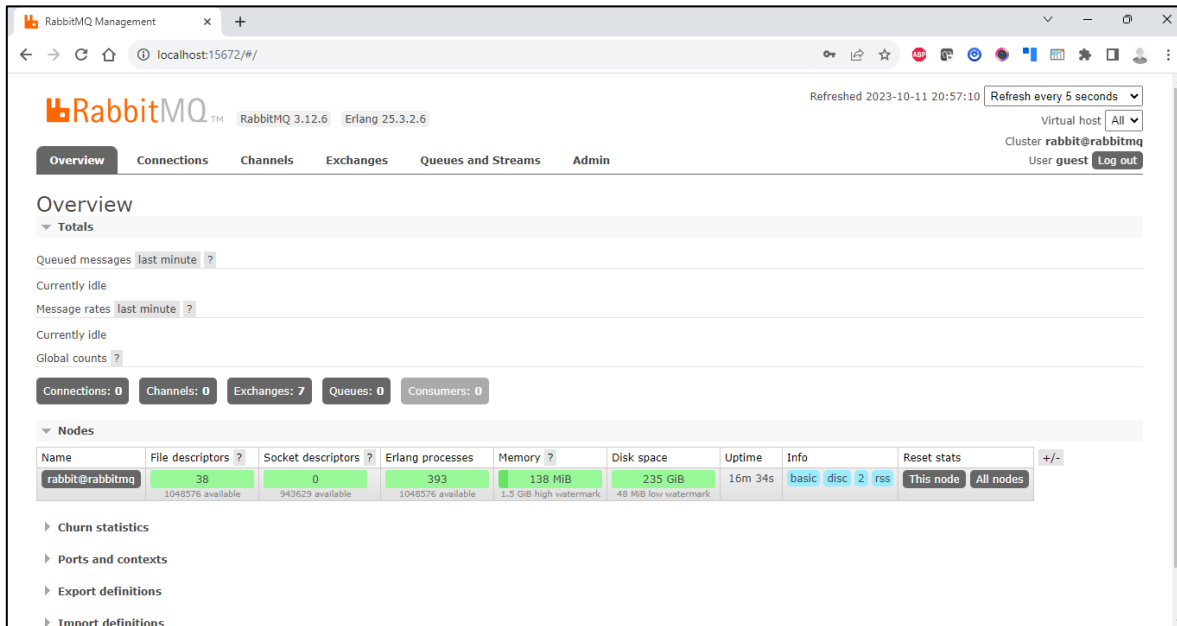
```

File Edit Selection View Go Run Terminal Help
Play.Catalog - Visual Studio Code

Starting host: rabbitmq://localhost/
info: MassTransit[0]
Connect: localhost:5672/
debug: MassTransit[0]
Connected: localhost:5672/ (address: amqp://localhost:5672, local: 64712)
debug: MassTransit[0]

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: D:\projects\Play.Catalog\src\Play.Catalog.Service
debug: MassTransit.SendTransport[0]
Declare exchange: name: Play.Catalog.Contracts:CatalogItemCreated, type: fanout, durable
debug: MassTransit.SendTransport[0]
SEND rabbitmq://localhost/Play.Catalog.Contracts:CatalogItemCreated 6c5f0000-99d1-b42e-a08e-08d8c38ecbdb Play.Catalog.Contracts.CatalogItemCreated
  
```

RabbitMQ Console



RabbitMQ Management

localhost:15672/#

Refreshed 2023-10-11 20:57:10 Refresh every 5 seconds

Virtual host All

Cluster rabbit@rabbitmq

User guest Log out

Overview Connections Channels Exchanges Queues and Streams Admin

Overview

Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@rabbitmq	38 1048576 available	0 943629 available	393 1048576 available	138 MiB 1.5 GiB high watermark	235 GiB 48 MiB low watermark	16m 34s	basic disc 2 rss	This node All nodes

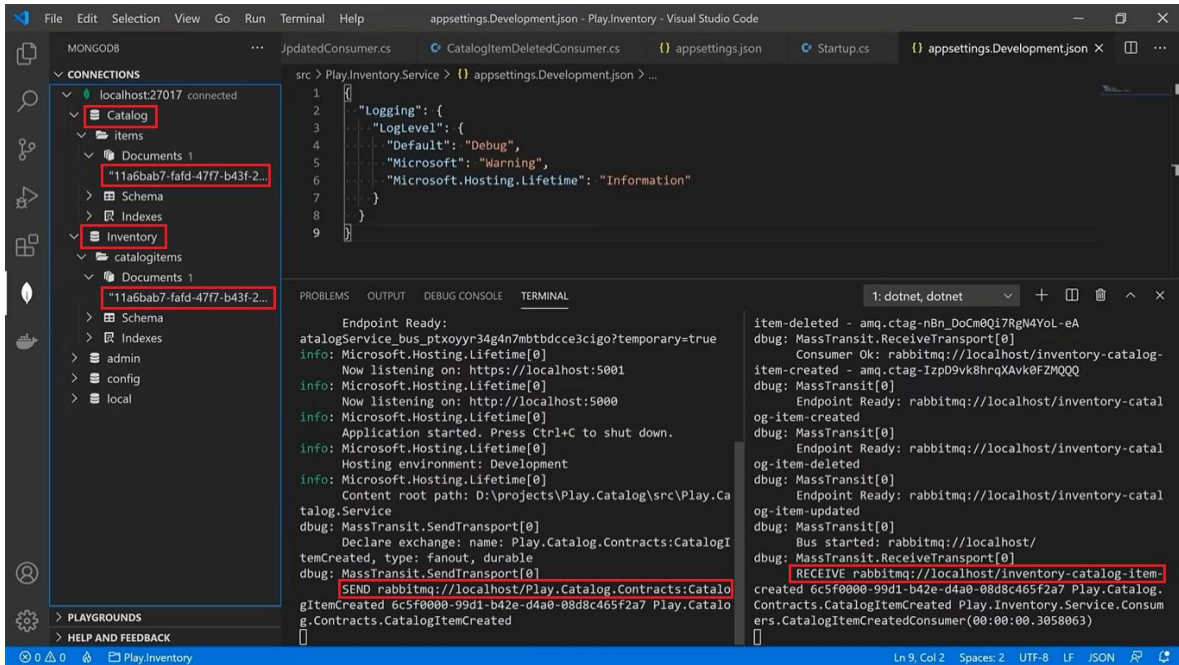
Churn statistics

Ports and contexts

Export definitions

Import definitions

Enviar y recibir mensajes entre microservicios a través de RabbitMQ



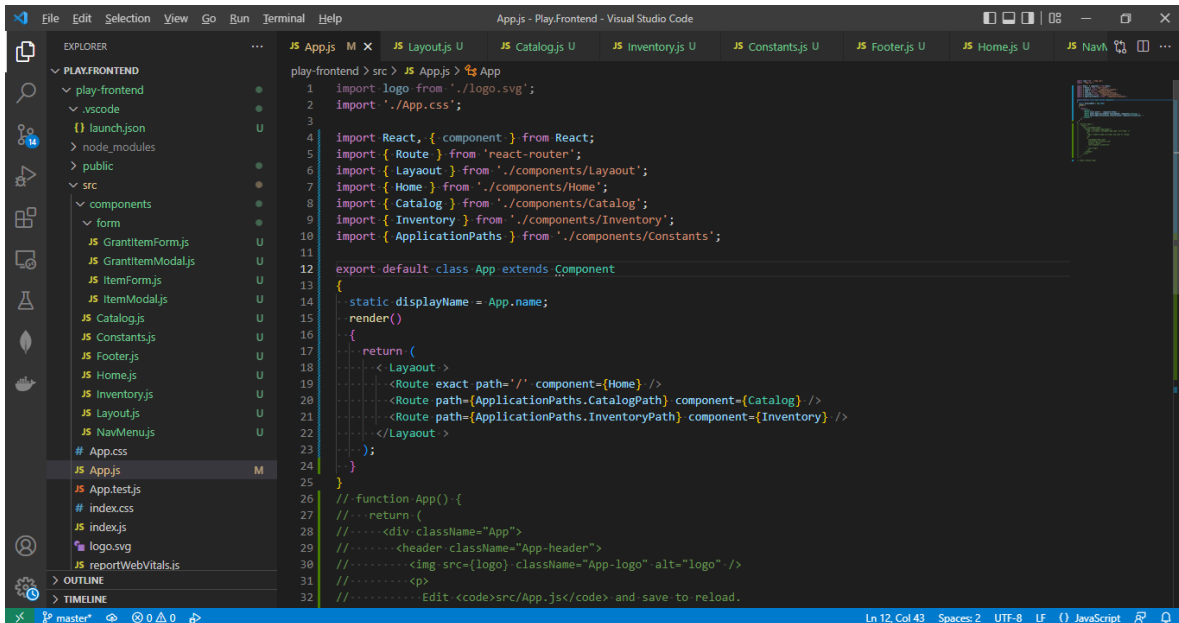
The screenshot shows the Visual Studio Code interface with the following details:

- Left Panel (EXPLORER):** Shows the project structure for 'Play.Inventory'. The 'CONNECTIONS' section is expanded, showing a connection to 'localhost:27017' with a database named 'Catalog'. The 'Inventory' collection is highlighted.
- Editor:** Displays the 'appsettings.Development.json' file with the following configuration:


```

{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```
- Terminal:** Shows the output of the application. It includes messages about the endpoint being ready, the application starting, and the RabbitMQ bus being started. A message is received from the 'rabbitmq://localhost/inventory-catalog-item-created' queue, indicating that a catalog item was created.

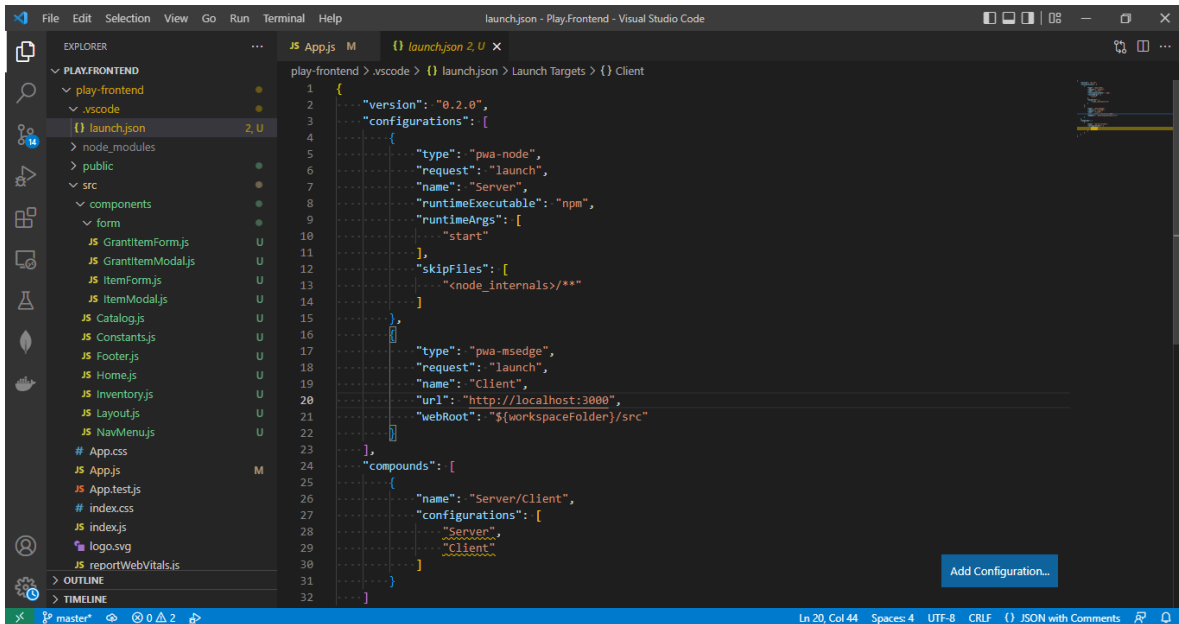
Frontend (React)



The screenshot shows the Visual Studio Code interface with the following details:

- Left Panel (EXPLORER):** Shows the project structure for 'Appjs - Play.Frontend'. The 'PLAYFRONTEND' section is expanded, showing the 'src' directory with various components and files.
- Editor:** Displays the 'App.js' file, which is the main entry point of the application. It includes imports for React, React Router, and various components. The code defines the App component and its render method, which returns a JSX element with a layout and routes for Home, Catalog, and Inventory.

Hosted (Node.js) Web Server



```

1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "type": "pwa-node",
6       "request": "launch",
7       "name": "Server",
8       "runtimeExecutable": "npm",
9       "runtimeArgs": [
10        "start"
11      ],
12      "skipFiles": [
13        "**<node_internals>/**"
14      ],
15    },
16    {
17      "type": "pwa-msedge",
18      "request": "launch",
19      "name": "Client",
20      "url": "http://localhost:3000",
21      "webRoot": "${workspaceFolder}/src"
22    }
23  ],
24  "compounds": [
25    {
26      "name": "Server/Client",
27      "configurations": [
28        "Server",
29        "Client"
30      ]
31    }
32  ]
33 }
  
```

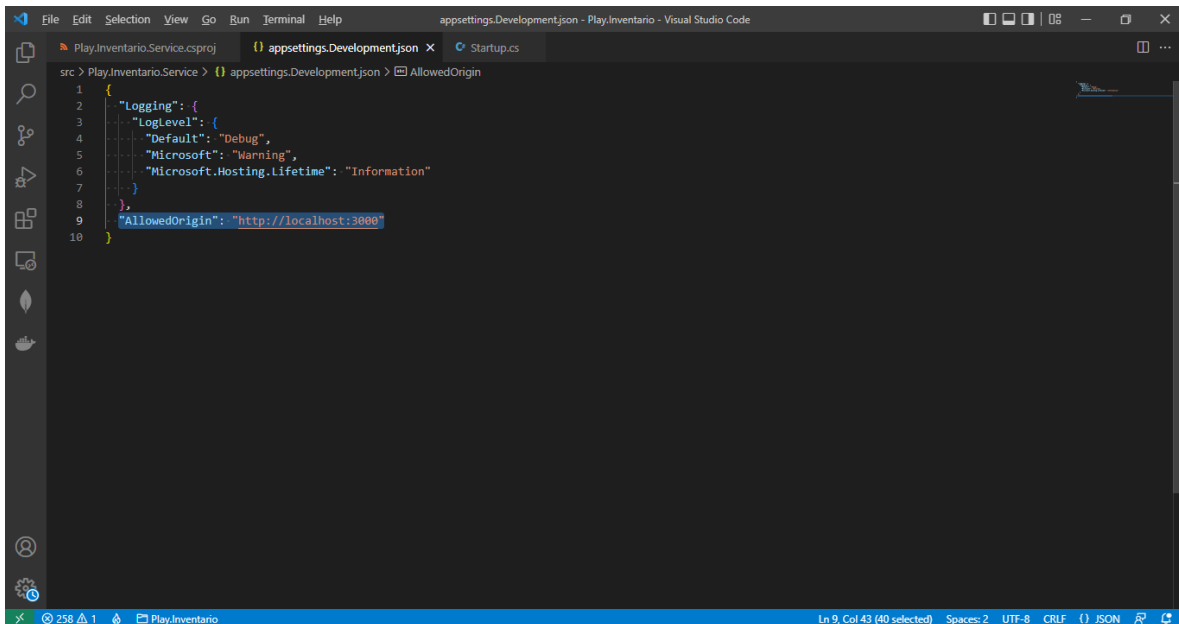
CORS (Cross-Origin Resource Sharing)

Para la comunicación entre el frontend y los microservicios

CORS

Permite a un servidor indicar cualquier otro origen distinto del suyo desde el cual un navegador debería permitir la carga de recursos.

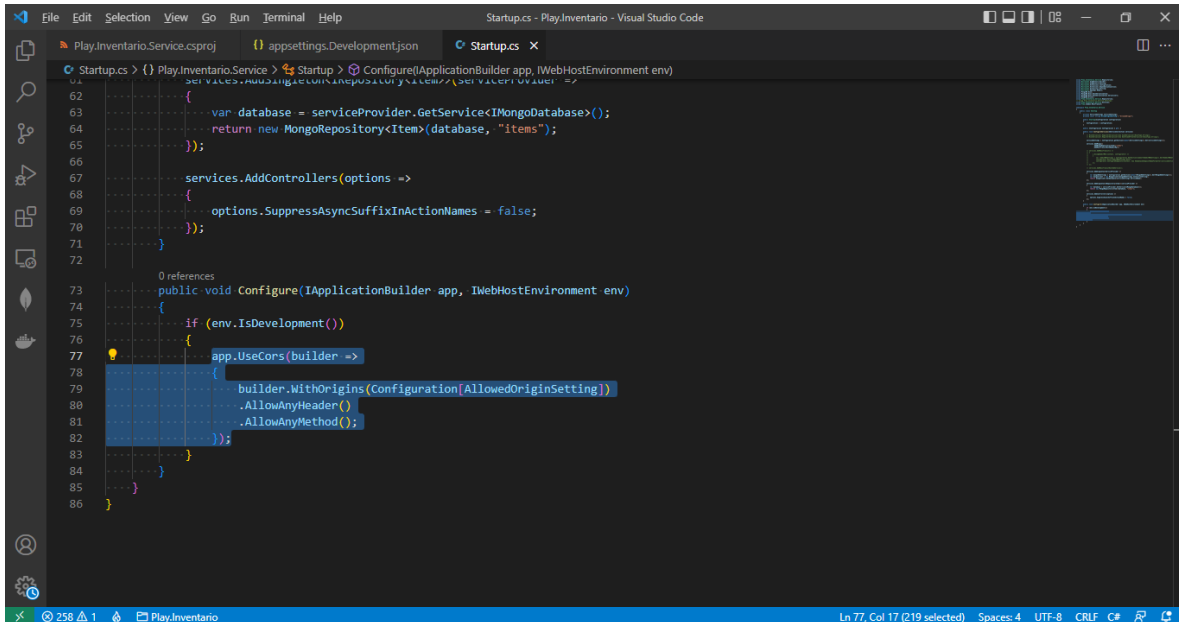
Permitir origen desde otras fuentes



```

1 {
2   "Logging": {
3     "LogLevel": {
4       "Default": "Debug",
5       "Microsoft": "Warning",
6       "Microsoft.Hosting.Lifetime": "Information"
7     }
8   },
9   "AllowedOrigin": "http://localhost:3000"
10 }
  
```

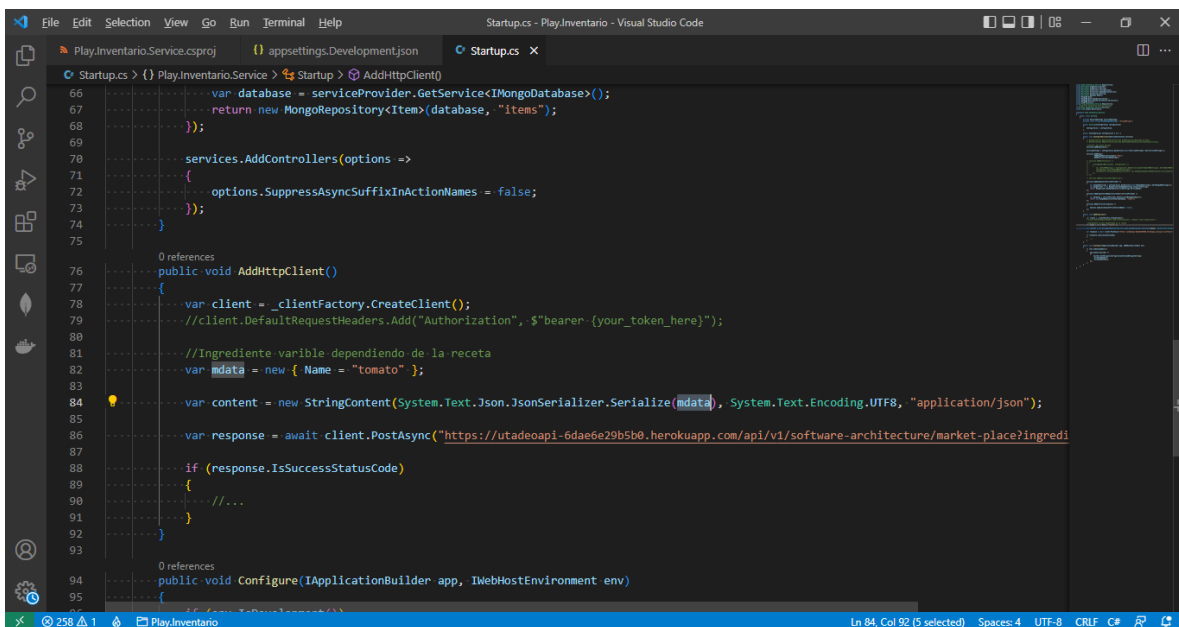
Permitir otros orígenes



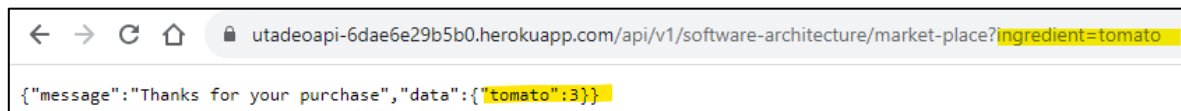
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
```

Consumir API plaza de mercado desde API propia

<https://utadeoapi-6dae6e29b5b0.herokuapp.com/api/v1/software-architecture/market-place?ingredient=tomato>



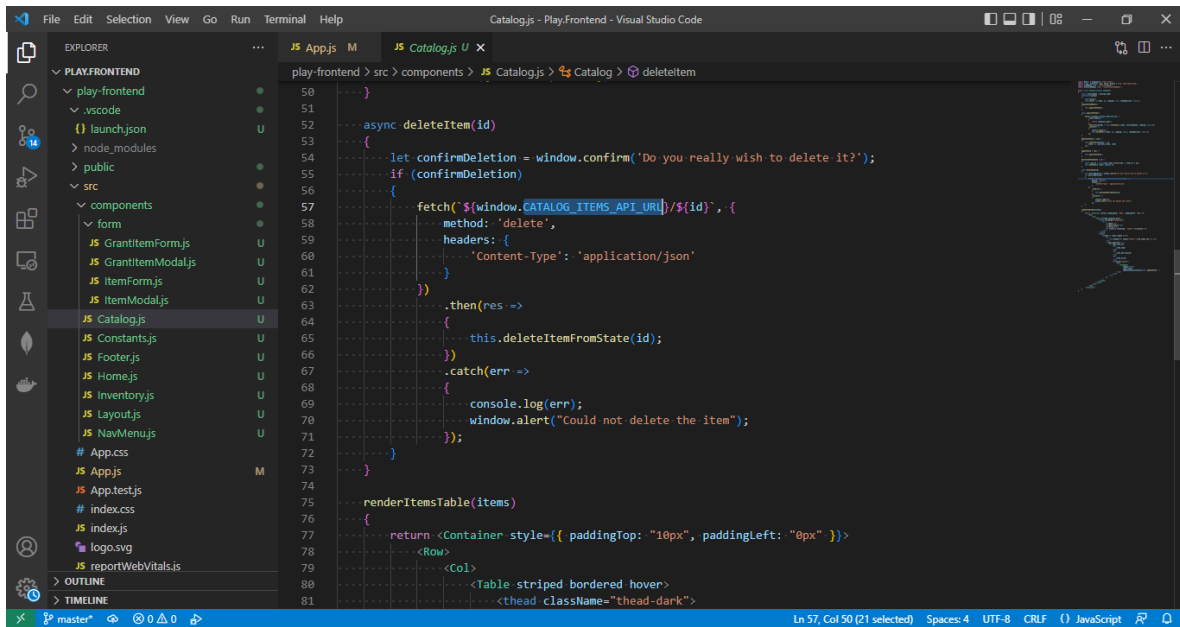
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```



utadeoapi-6dae6e29b5b0.herokuapp.com/api/v1/software-architecture/market-place?ingredient=tomato

```
{\"message\": \"Thanks for your purchase\", \"data\": {\"tomato\": 3}}
```

Interconexión entre microservicios



Recetas

Recetas x 6

(Ingredientes):

Tomato
 Lemon
 Potato
 Rice
 Ketchup
 Lettuce
 Onion
 Cheese
 Meat
 Chicken

A continuación, se describen 6 recetas ficticias utilizando los ingredientes proporcionados.

1. Ensalada de Tomate y Queso:

- Ingredientes:

- Tomato (2 unidades)
- Lettuce (1 unidad)
- Cheese (2 unidades)

- Preparación: Corta los tomates en rodajas, mezcla con la lechuga y el queso. Sirve con tu aderezo favorito.

2. Arroz con Pollo al Limón:

- Ingredientes:

- Rice (1 unidad)
- Chicken (1 unidad)
- Lemon (2 unidades)

- Preparación: Cocina el arroz. En una sartén, cocina el pollo con el jugo de limón. Sirve el pollo sobre el arroz.

3. Papas Gratinadas con Queso:

- Ingredientes:

- Potato (4 unidades)
- Cheese (2 unidades)

- Preparación: Corta las papas en rodajas finas. Coloca en capas en una fuente para horno con queso entre cada capa. Hornea hasta que las papas estén tiernas.

4. Salsa de Tomate Casera:

- Ingredientes:

- Tomato (5 unidades)
- Onion (1 unidad)
- Garlic (2 unidades)

- Preparación: Hierva los tomates y pélalos. Sofríe la cebolla y el ajo, luego agrega los tomates triturados. Cocina a fuego lento hasta obtener una salsa espesa.

5. Hamburguesa con Salsa de Ketchup:

- Ingredientes:

- Meat (1 unidad)
- Lettuce (1 unidad)

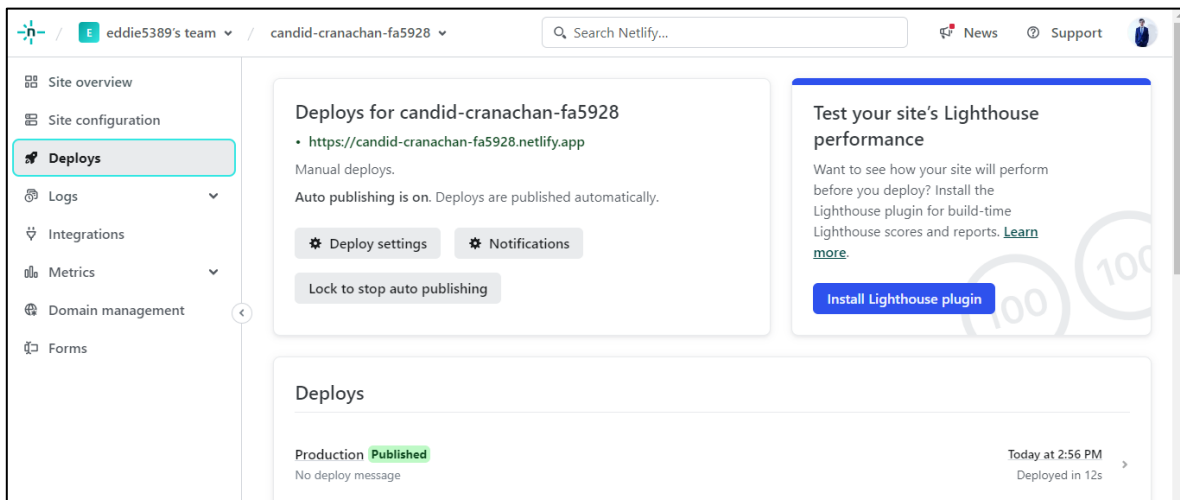
- Tomato (1 unidad)
- Ketchup (1 unidad)
- Preparación: Forma las hamburguesas y cocínalas. Arma las hamburguesas con lechuga, tomate y salsa de ketchup.

6. Pollo con Limón y Queso:

- Ingredientes:
 - Chicken (1 unidad)
 - Lemon (1 unidad)
 - Cheese (2 unidades)
- Preparación: Cocina el pollo a la parrilla o en una sartén. Exprime limón sobre el pollo y agrega queso rallado antes de servir.

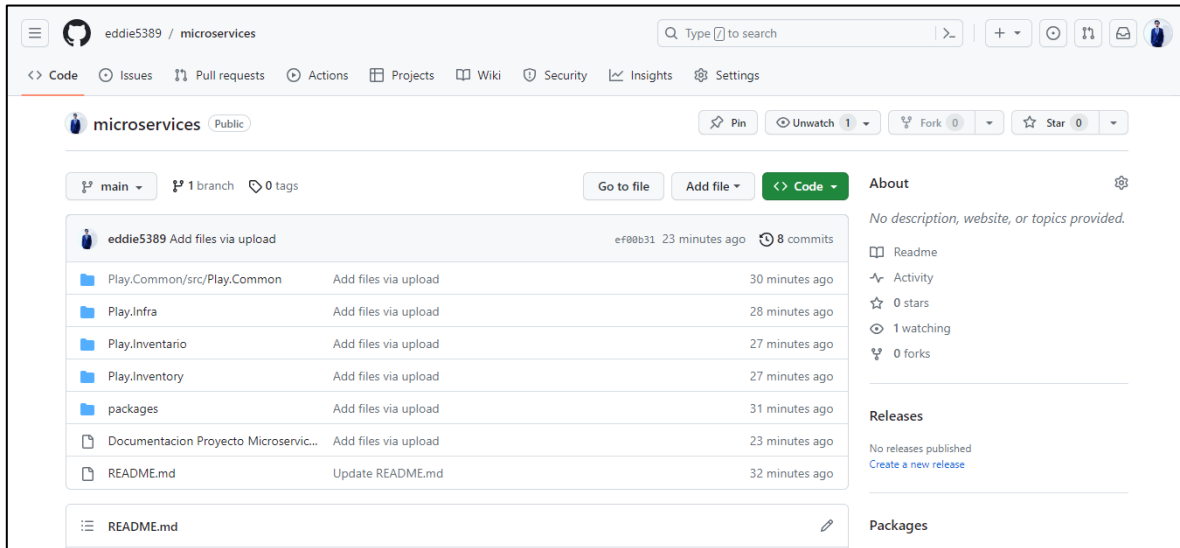
Deploy App NETLIFY

URL: <https://candid-cranachan-fa5928.netlify.app/>



GitHub Repositorio

URL: <https://github.com/eddie5389/microservices>



Nota. Github no permitió subir archivo frontend (react) ya que excede el tamaño permitido (25MB) por lo cual se adjunta el archivo comprimido en ".zip" directamente en Avata.

