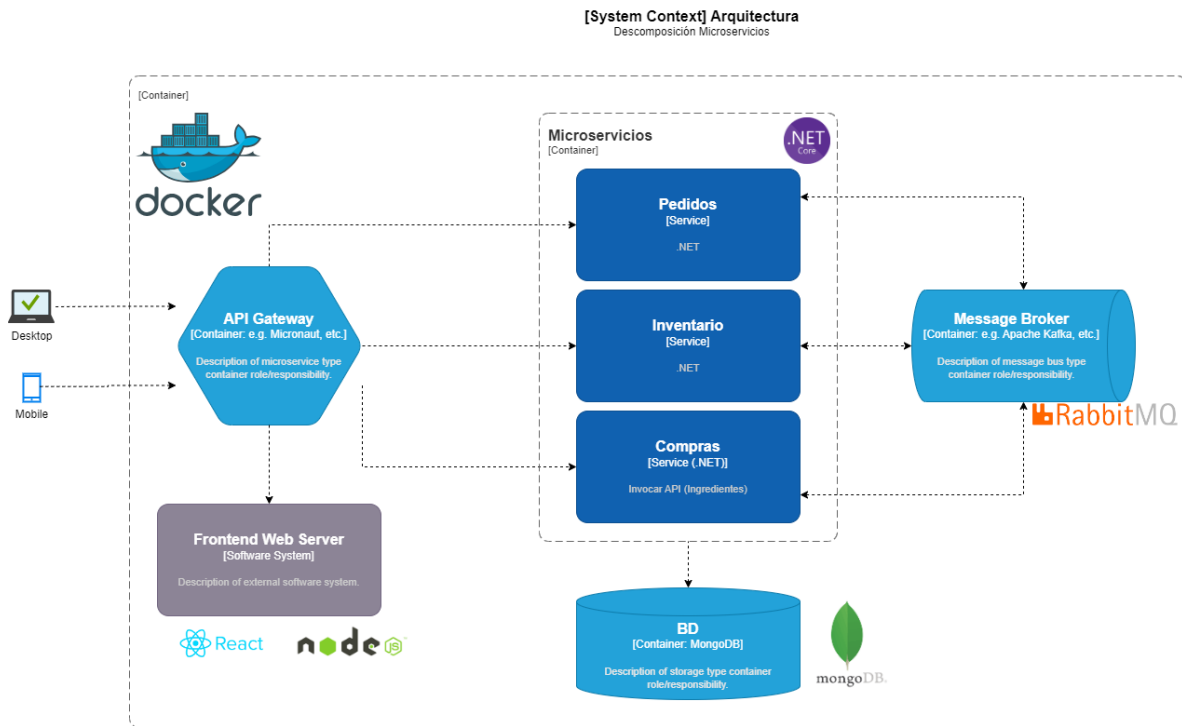


ARQUITECTURA DE SOFTWARE (2S-2023)

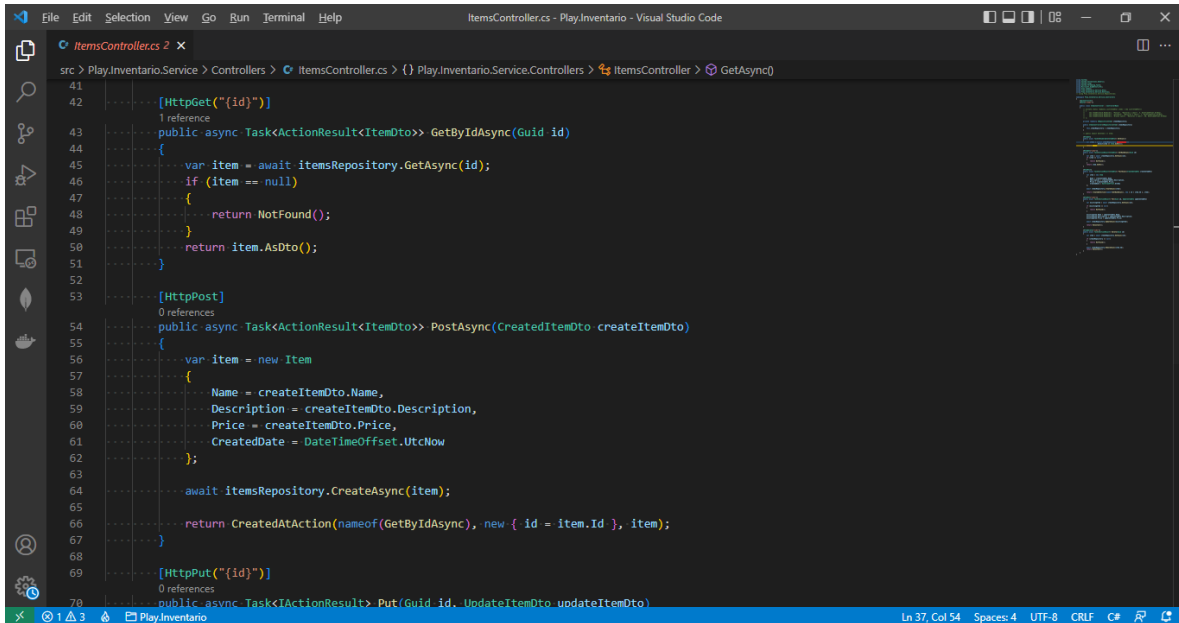
PROYECTO CORTE 2 (MICROSERVICIOS)

EDISSON CABRERA ERASO

Arquitectura Propuesta

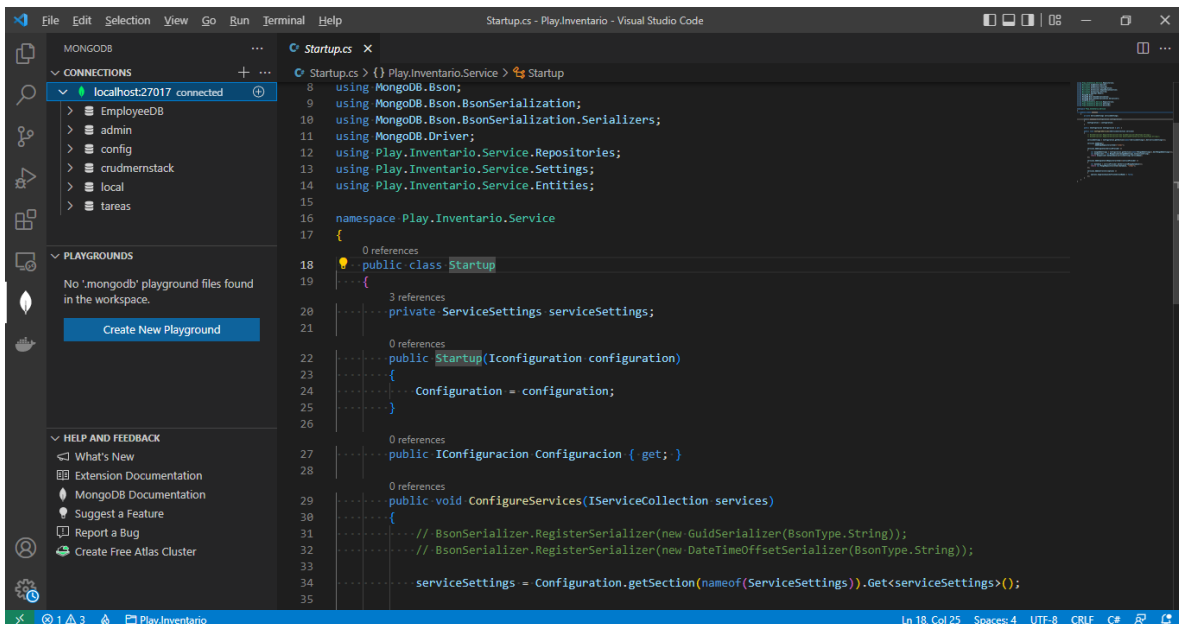


API's



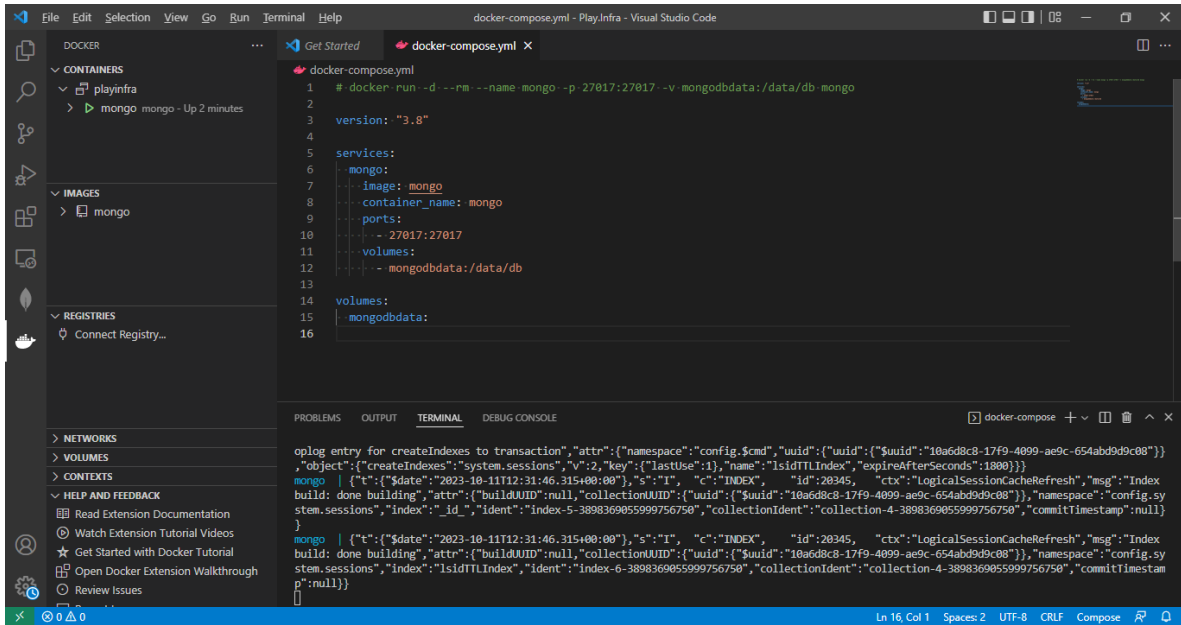
```
41  
42 [HttpGet("{id}")]  
43 public async Task<ActionResult<ItemDto>> GetByIdAsync(Guid id)  
44 {  
45     var item = await itemsRepository.GetAsync(id);  
46     if (item == null)  
47     {  
48         return NotFound();  
49     }  
50     return item.AsDto();  
51 }  
52  
53 [HttpPost] 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

DB Mongo



```
8 using MongoDB.Bson;  
9 using MongoDB.Bson.BsonSerialization;  
10 using MongoDB.Bson.BsonSerialization.Serializers;  
11 using MongoDB.Driver;  
12 using Play.Inventario.Service.Repositories;  
13 using Play.Inventario.Service.Settings;  
14 using Play.Inventario.Service.Entities;  
15  
16 namespace Play.Inventario.Service  
17 {  
18     public class Startup  
19     {  
20         private ServiceSettings serviceSettings;  
21  
22         public Startup(IConfiguration configuration)  
23         {  
24             Configuration = configuration;  
25         }  
26  
27         public IConfiguration Configuration { get; }  
28  
29         public void ConfigureServices(IServiceCollection services)  
30         {  
31             // BsonSerializer.RegisterSerializer(new GuidSerializer(BsonType.String));  
32             // BsonSerializer.RegisterSerializer(new DateTimeOffsetSerializer(BsonType.String));  
33  
34             serviceSettings = Configuration.GetSection(nameof(ServiceSettings)).Get<ServiceSettings>();  
35         }  
36     }  
37 }
```

Docker Compose (yaml)

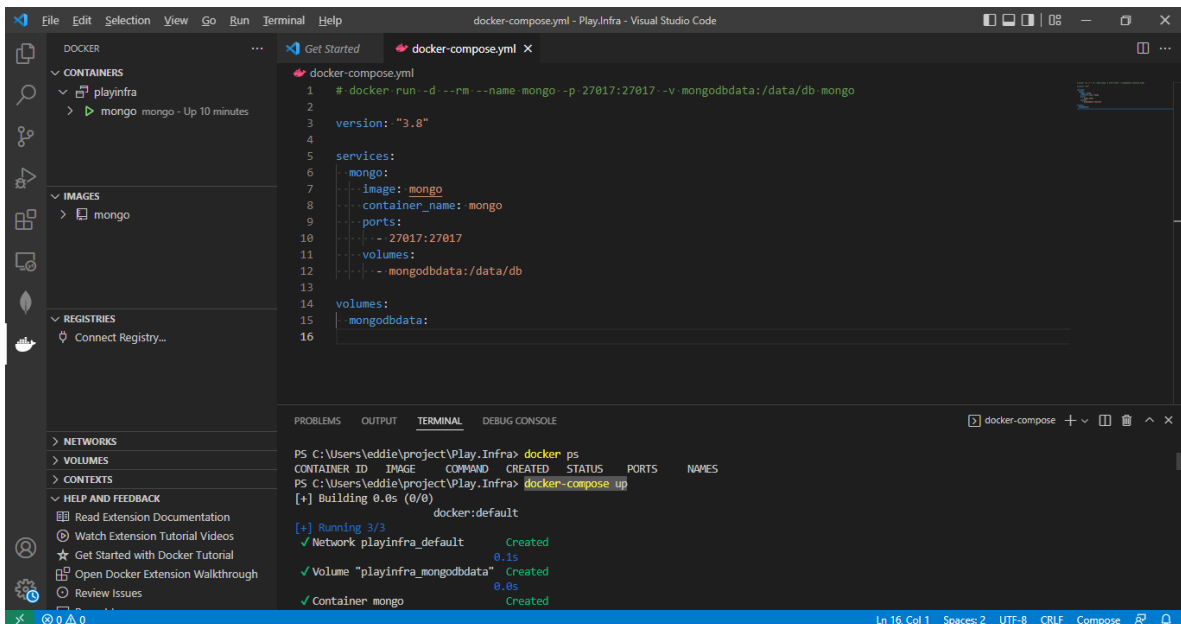


The screenshot shows a Visual Studio Code editor with a Docker Compose file named `docker-compose.yml`. The file is configured to run a MongoDB container. The left sidebar shows the Docker extension interface with 'CONTAINERS', 'IMAGES', and 'REGISTRIES' sections. The 'CONTAINERS' section shows a 'mongo' container that is 'Up 2 minutes'. The 'IMAGES' section shows the 'mongo' image. The 'REGISTRIES' section shows 'Connect Registry...'. The main editor displays the following YAML content:

```
1 # docker run -d --rm --name mongo -p 27017:27017 -v mongoddata:/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      - mongoddata:/data/db
13
14 volumes:
15   mongoddata:
```

The bottom panel shows the 'TERMINAL' output, which includes logs for the container's startup and the creation of the 'mongo' container.

Iniciar Docker Mongo



The screenshot shows the same Visual Studio Code editor with the Docker Compose file. The left sidebar shows the Docker extension interface. The 'CONTAINERS' section shows the 'mongo' container is 'Up 10 minutes'. The 'IMAGES' section shows the 'mongo' image. The 'REGISTRIES' section shows 'Connect Registry...'. The main editor displays the same YAML content as the previous screenshot. The bottom panel shows the 'TERMINAL' output, which includes the command `docker-compose up` and the resulting output:

```
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_mongoddata" Created
  ✓ Container mongo Created
```

Docker Desktop Upgrade plan Search for images, containers, volumes, extensions and more... **Ctrl+K** Sign in

Containers Give feedback

Container CPU usage 0.48% / 400% (4 cores allocated) Container memory usage 80.72MB / 3.63GB [Show charts](#)

Search Only show running containers

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

Showing 1 item

Walkthroughs

What is a container?
5 mins

How do I run a container?
6 mins

[View more in the Learning center](#)

Engine running RAM 1.97 GB CPU 0.25% Not signed in v4.24.0

Comunicación asíncrona entre microservicios

File Edit Selection View Go Run Terminal Help Startup.cs - Play.Inventory - Visual Studio Code

```

src > Play.Inventory.Service > Startup.cs > {} Play.Inventory.Service > Play.Inventory.Service.Startup > ConfigureServices(IServiceCollection services)
40 onRetry: (outcome, timespan, retryAttempt) =>
41 {
42     var serviceProvider = services.BuildServiceProvider();
43     serviceProvider.GetService<ILogger<CatalogClient>>().LogWarning($"Delaying for {timespan.TotalSeconds} seconds, then making retry {retryAttempt}");
44 }
45
46 })
47 .AddPolicyHandler(Policy.TimeoutAsync<HttpResponseMessage>(1));
48
49 services.AddControllers();
50 services.AddSwaggerGen(c =>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: dotnet, dotnet

```

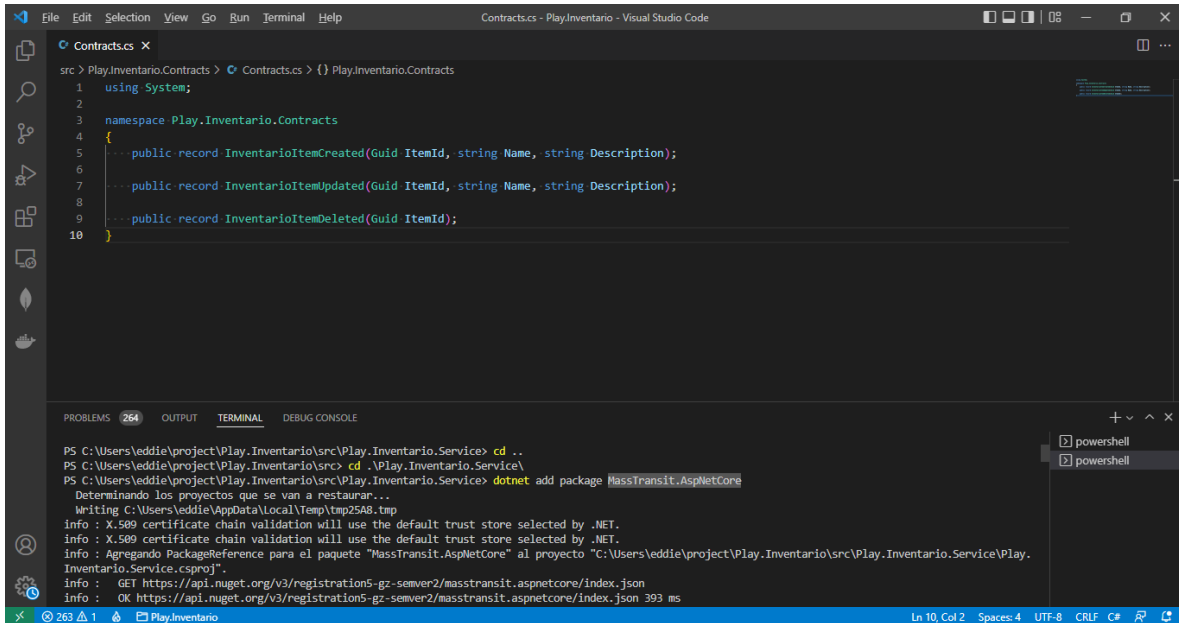
warn: Play.Inventory.Service.Clients.CatalogClient[0]
      Delaying for 4 seconds, then making retry Polly.Context
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[100]
      Sending HTTP request GET https://localhost:5001/items
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[101]
      Received HTTP response headers after 28.8285ms - 500
warn: Play.Inventory.Service.Clients.CatalogClient[0]
      Delaying for 8 seconds, then making retry Polly.Context
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[100]
      Sending HTTP request GET https://localhost:5001/items
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[101]
      Received HTTP response headers after 2.9418ms - 500
warn: Play.Inventory.Service.Clients.CatalogClient[0]
      Delaying for 16 seconds, then making retry Polly.Context
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[100]
      Sending HTTP request GET https://localhost:5001/items
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[101]
      Received HTTP response headers after 88.0146ms - 200
info: System.Net.Http.HttpClient.CatalogClient.LogicalHandler[101]
      End processing HTTP request after 32214.9848ms - 200

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
Request 1: Starting...
Request 1: Delaying...
Request 2: Starting...
Request 2: Delaying...
Request 3: Starting...
Request 3: 500 (Internal Server Error).
Request 3: 500 (Internal Server Error).
Request 3: 500 (Internal Server Error).
Request 4: Starting...
Request 4: 500 (Internal Server Error).
Request 5: Starting...
Request 5: 200 (OK).

```

Ln 45, Col 18 Spaces: 4 UTF-8 CRLF C#

Message Broker (RabbitMQ + MassTransit)



The screenshot shows the Visual Studio Code interface with a C# file named `Contracts.cs` in the `src > Play.Inventario.Contracts > Contracts.cs` path. The code defines three interfaces: `InventoryItemCreated`, `InventoryItemUpdated`, and `InventoryItemDeleted`. The `TERMINAL` tab is active, displaying the command prompt output for installing `MassTransit.AspNetCore` using `dotnet add package`. The status bar at the bottom indicates the file is at line 10, column 2, with 4 spaces, in UTF-8 encoding, CRLF line endings, and C# language.

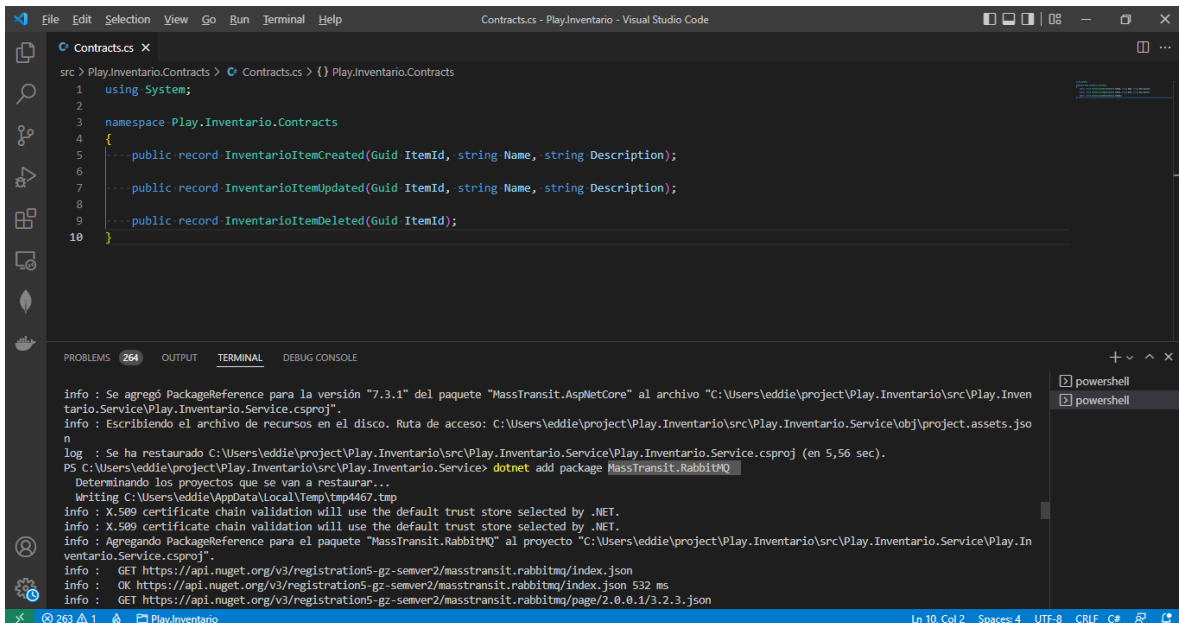
```
Contracts.cs - Play.Inventario - Visual Studio Code

src > Play.Inventario.Contracts > Contracts.cs > {} Play.Inventario.Contracts
1  using System;
2
3  namespace Play.Inventario.Contracts
4  {
5      public record InventoryItemCreated(Guid ItemId, string Name, string Description);
6
7      public record InventoryItemUpdated(Guid ItemId, string Name, string Description);
8
9      public record InventoryItemDeleted(Guid ItemId);
10 }
```

PROBLEMS 264 OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service> cd ..
PS C:\Users\eddie\project\Play.Inventario\src> cd .\Play.Inventario.Service\
PS C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service> dotnet add package MassTransit.AspNetCore
Determinando los proyectos que se van a restaurar...
Writing C:\Users\eddie\AppData\Local\Temp\tmp2548.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.AspNetCore" 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.aspnetcore/index.json
Info : OK https://api.nuget.org/v3/registration5-gz-semver2/masstransit.aspnetcore/index.json 393 ms
```

Ln 10, Col 2 Spaces: 4 UTF-8 CRLF C#



This screenshot shows the next step in the installation process. The `Contracts.cs` file remains the same. The `TERMINAL` tab now shows the command to install `MassTransit.RabbitMQ` using `dotnet add package`. The output indicates that the package was successfully added to the project. The status bar at the bottom remains the same as in the previous screenshot.

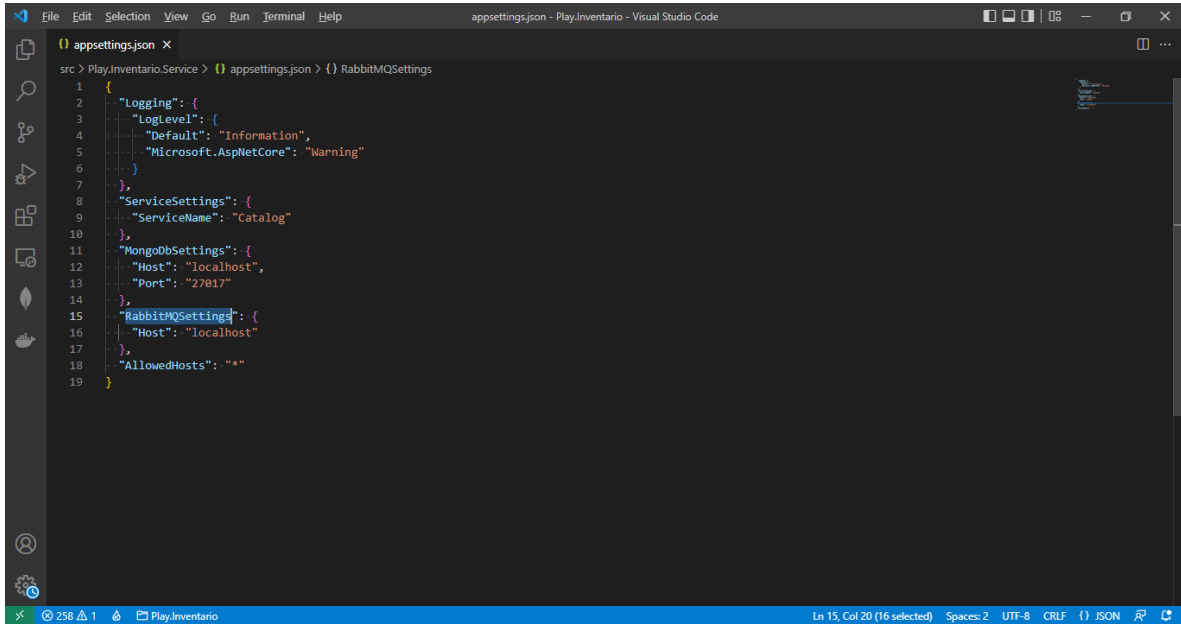
```
Contracts.cs - Play.Inventario - Visual Studio Code

src > Play.Inventario.Contracts > Contracts.cs > {} Play.Inventario.Contracts
1  using System;
2
3  namespace Play.Inventario.Contracts
4  {
5      public record InventoryItemCreated(Guid ItemId, string Name, string Description);
6
7      public record InventoryItemUpdated(Guid ItemId, string Name, string Description);
8
9      public record InventoryItemDeleted(Guid ItemId);
10 }
```

PROBLEMS 264 OUTPUT TERMINAL DEBUG CONSOLE

```
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
```

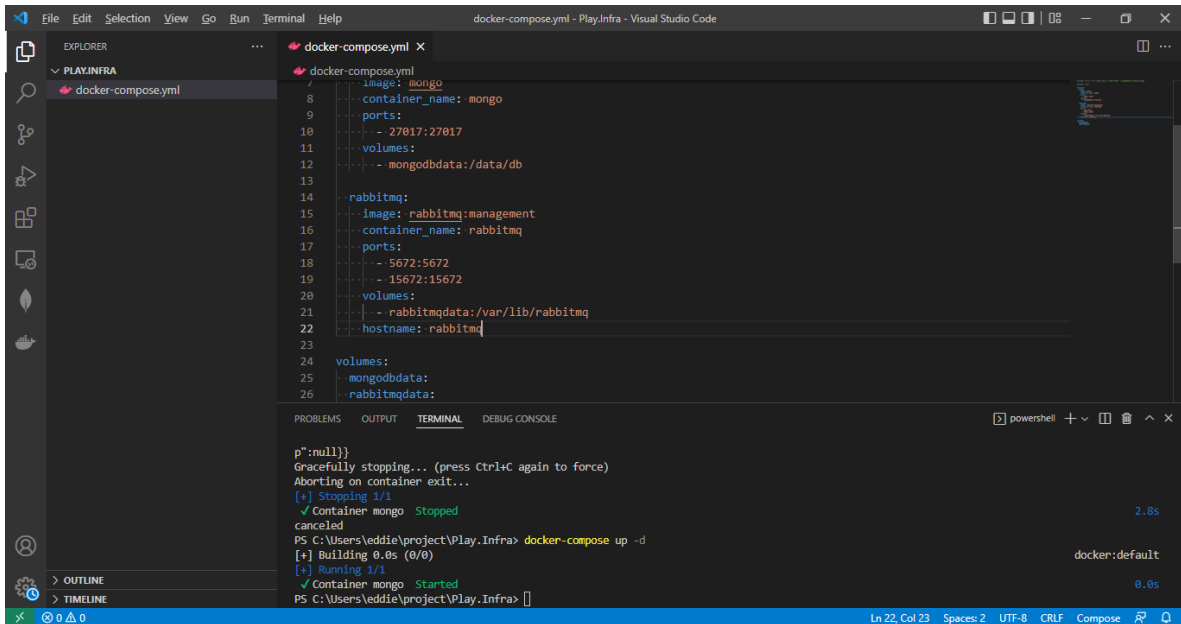
Ln 10, Col 2 Spaces: 4 UTF-8 CRLF C#



The screenshot shows the Visual Studio Code editor with the file `appsettings.json` open. The file is a JSON configuration for a .NET application. The `RabbitMQSettings` section is highlighted, showing the `Host` set to `localhost` and `AllowedHosts` set to `*`.

```
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 }
```

The status bar at the bottom indicates the file is at line 15, column 20, with 16 characters selected. The encoding is UTF-8 and the line endings are CRLF. The file is in JSON format.



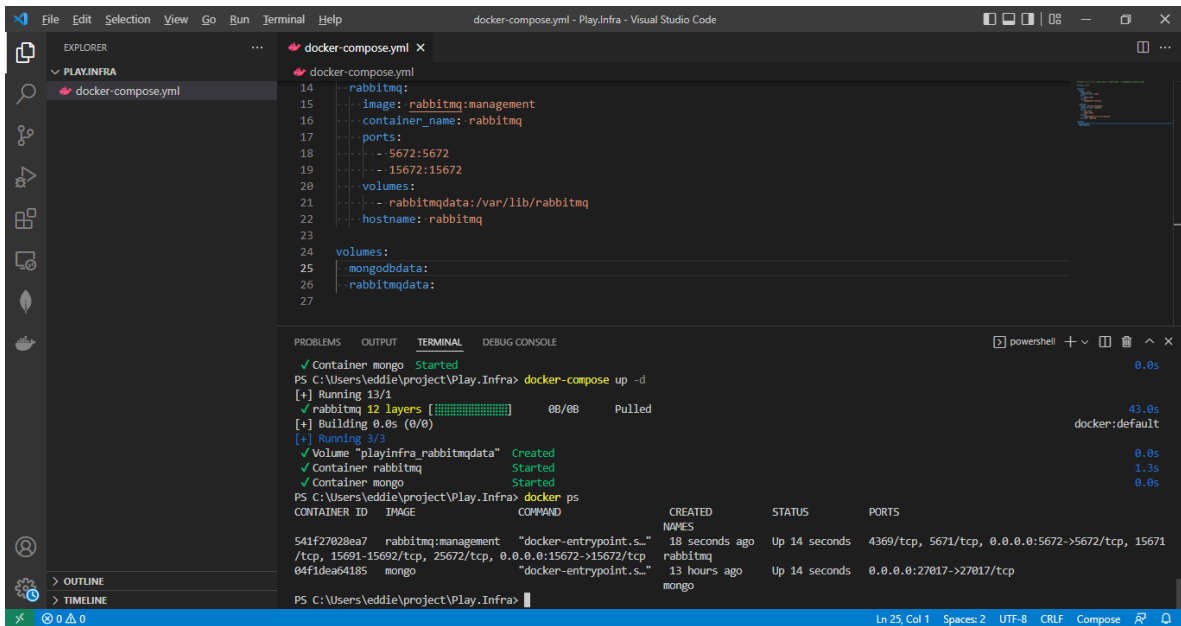
The screenshot shows the Visual Studio Code editor with the file `docker-compose.yml` open. The file defines a multi-container Docker application with three services: `mongo`, `rabbitmq`, and `mongodbdata`. The `mongo` service is configured with the `mongo` image, port 27017, and a volume `mongodbdata:/data/db`. The `rabbitmq` service is configured with the `rabbitmq:management` image, ports 5672 and 15672, and a volume `rabbitmqdata:/var/lib/rabbitmq`. The `mongodbdata` service is configured with the `mongodbdata` image and a volume `rabbitmqdata`.

```
1 image: mongo
2 container_name: mongo
3 ports:
4   - 27017:27017
5 volumes:
6   - mongodbdata:/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   mongodbdata:
20   rabbitmqdata:
```

The terminal output shows the command `docker-compose up -d` being executed. The output indicates that the containers are being built and started. The `mongo` container is stopped, and the `rabbitmq` container is started.

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

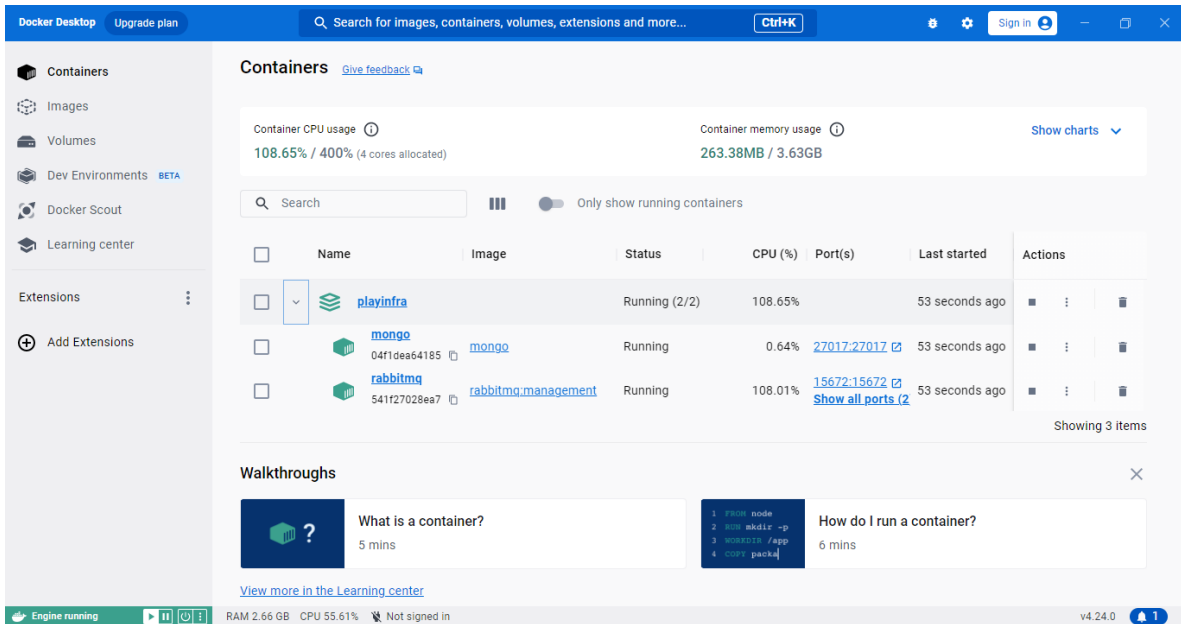
The status bar at the bottom indicates the file is at line 22, column 23, with 2 spaces. The encoding is UTF-8 and the line endings are CRLF. The file is in Compose format.



The image shows a Visual Studio Code editor with a Docker Compose file named `docker-compose.yml` open. The file defines a service named `rabbitmq` using the `rabbitmq:management` image, with ports `5672:5672` and `15672:15672`, and a volume `rabbitmqdata` mapped to `/var/lib/rabbitmq` on the host. The `volumes` section defines `mongoddata` and `rabbitmqdata`.

The terminal output shows the command `docker-compose up -d` being executed. It reports that the `mongo` container is started, and the `rabbitmq` service is running with 12 layers. The output also shows the creation and starting of the `mongo` and `rabbitmq` containers. A table at the bottom of the terminal output provides details for the containers:

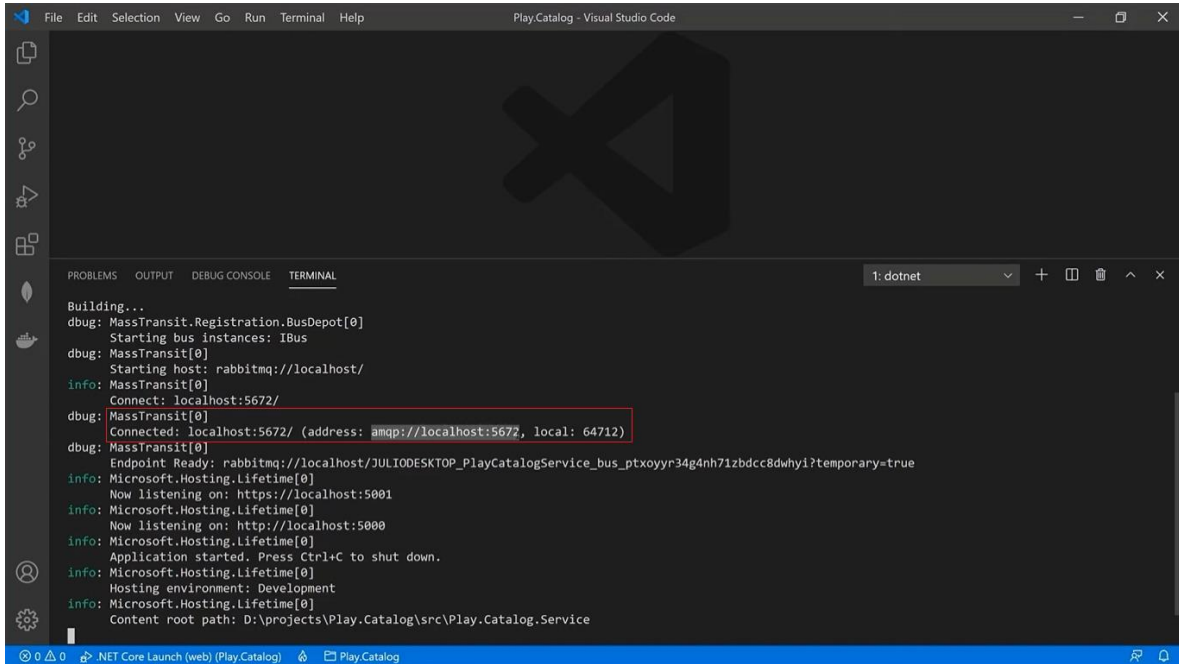
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
541f27028ea7	rabbitmq:management	"docker-entrypoint.s..."	18 seconds ago	Up 14 seconds	4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp
04f1dea64185	mongo	"docker-entrypoint.s..."	13 hours ago	Up 14 seconds	0.0.0.0:27017->27017/tcp



The image shows the Docker Desktop interface. The left sidebar contains navigation options: Containers, Images, Volumes, Dev Environments (BETA), Docker Scout, and Learning center. The main area displays the 'Containers' section, showing a table of running containers. The table includes columns for Name, Image, Status, CPU (%), Port(s), Last started, and Actions. Three containers are listed: `playinfra`, `mongo`, and `rabbitmq`.

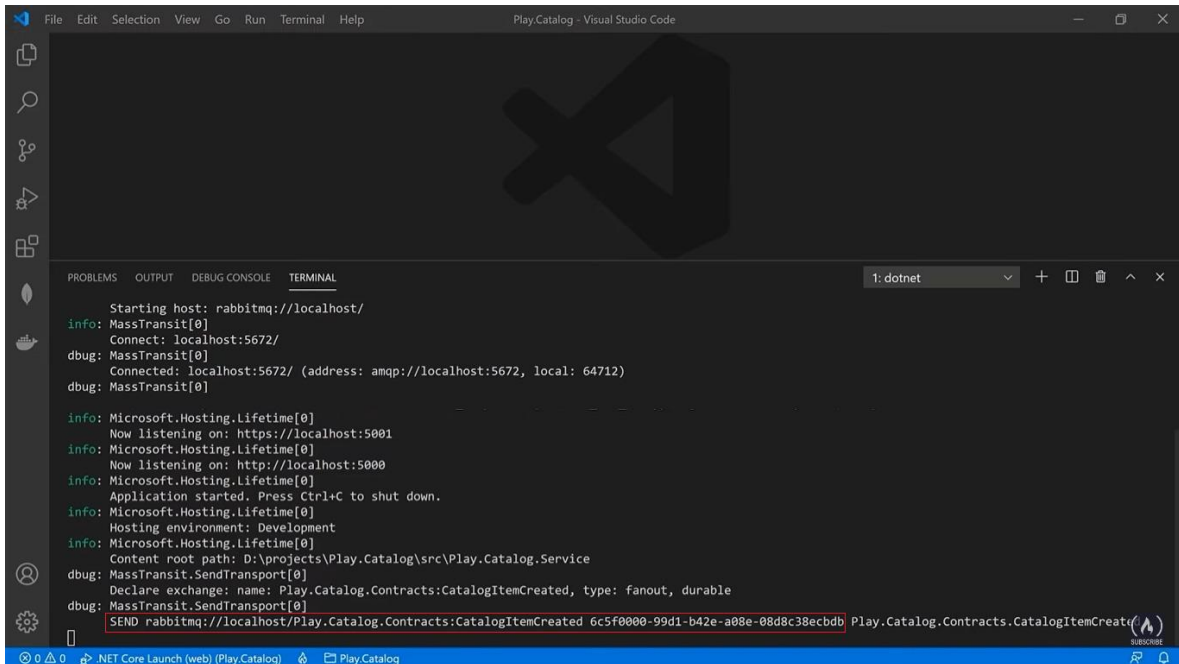
Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
playinfra		Running (2/2)	108.65%		53 seconds ago	[Stop] [Refresh] [Delete]
mongo	mongo	Running	0.64%	27017/27017	53 seconds ago	[Stop] [Refresh] [Delete]
rabbitmq	rabbitmq:management	Running	108.01%	15672:15672	53 seconds ago	[Stop] [Refresh] [Delete]

Below the table, there are 'Walkthroughs' for 'What is a container?' (5 mins) and 'How do I run a container?' (6 mins). The bottom status bar shows 'Engine running', 'RAM 2.66 GB', 'CPU 55.61%', and 'Not signed in'.



```
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_ptxoyr34g4nh71zbdcc8dwhy1?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
```

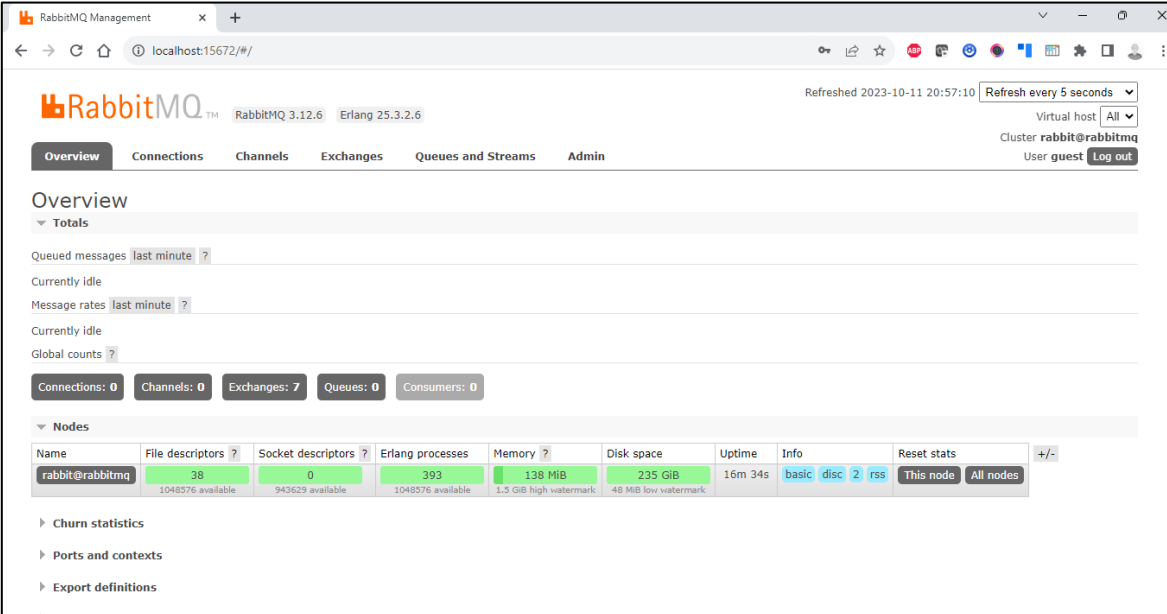
Enviar mensaje mediante RabbitMQ



```
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]

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
dbug: MassTransit.SendTransport[0]
      Declared exchange: name: Play.Catalog.Contracts:CatalogItemCreated, type: fanout, durable
dbug: MassTransit.SendTransport[0]
      SEND rabbitmq://localhost/Play.Catalog.Contracts:CatalogItemCreated 6c5f0000-99d1-b42e-a08e-08d8c38ecbdb Play.Catalog.Contracts.CatalogItemCreated
```


RabbitMQ Console



RabbitMQ Management Console

Overview | Connections | Channels | Exchanges | Queues and Streams | Admin

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

Virtual host: All | Cluster: rabbitmq@rabbitmq | User: guest | Log out

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	+/-
rabbitmq@rabbitmq	38 1048576 available	0 943629 available	393 1048576 available	138 MIB 1.3 GiB high watermark	235 GiB 48 MIB low watermark	16m 34s	basic disc 2 rss	This node All nodes	

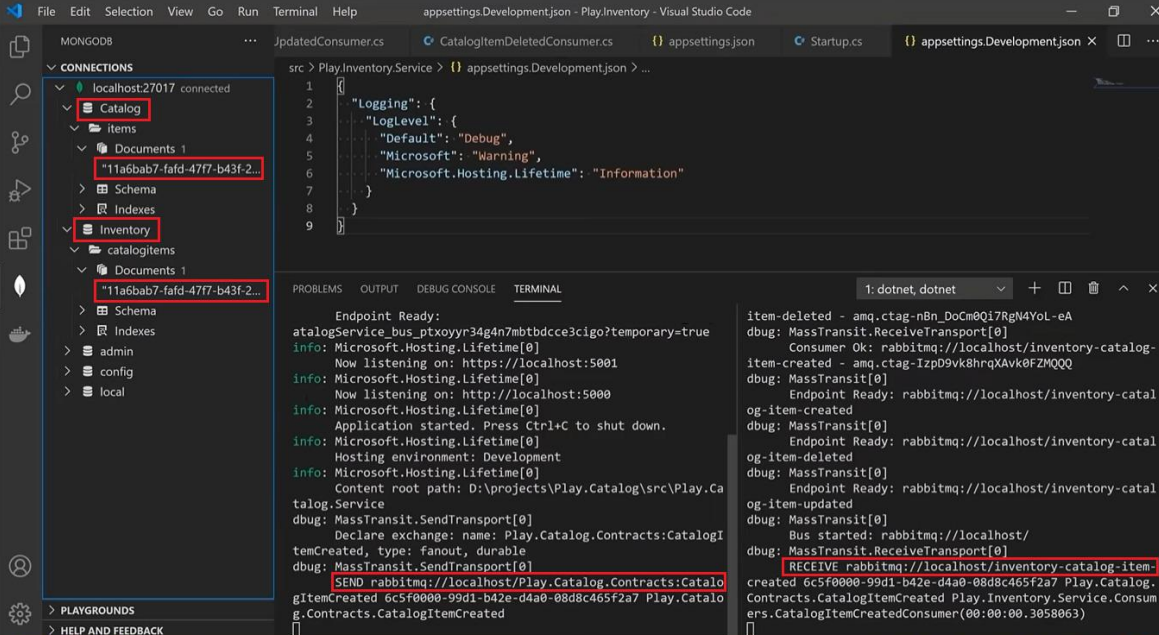
Churn statistics

Ports and contexts

Export definitions

Import definitions

Enviar y recibir mensajes entre microservicios a través de RabbitMQ



Visual Studio Code - Play.Inventory - Visual Studio Code

File Edit Selection View Go Run Terminal Help

appsettings.Development.json - Play.Inventory - Visual Studio Code

src > Play.Inventory.Service > appsettings.Development.json > ...

```

1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Debug",
5        "Microsoft": "Warning",
6        "Microsoft.Hosting.Lifetime": "Information"
7      }
8    }
9  }

```

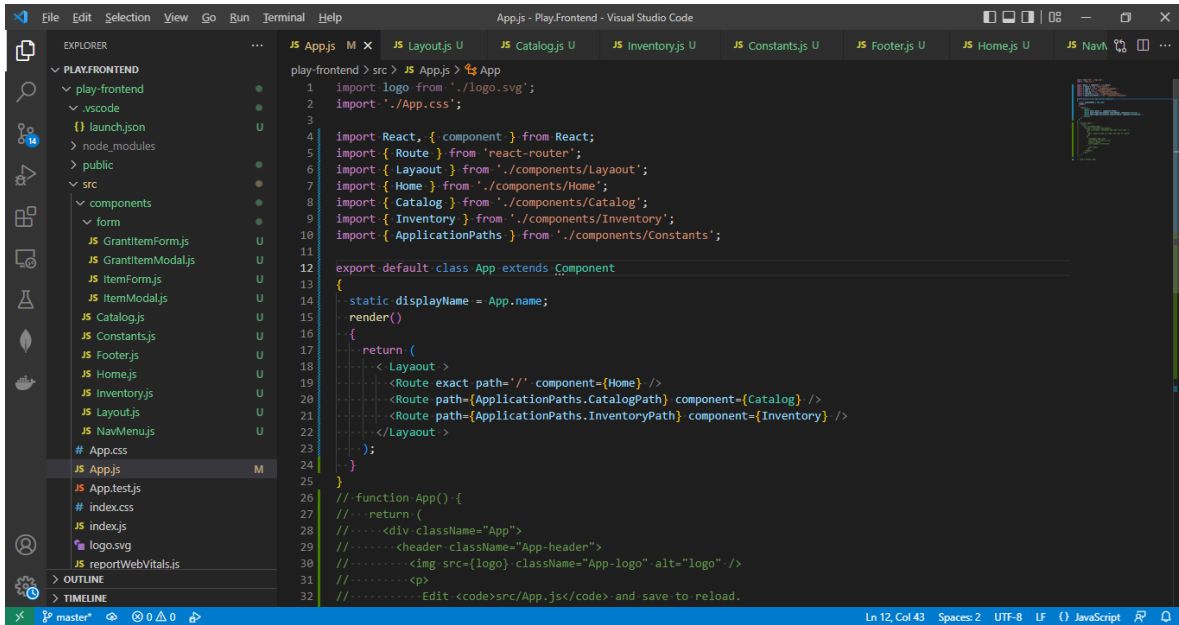
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Endpoint Ready:
atalogService_bus_ptxyoyr34g4n7mbtdcce3cigo?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
Content root path: D:\projects\Play.Catalog\src\Play.Catalog.Service
dbug: MassTransit.SendTransport[0]
Declare exchange: name: Play.Catalog.Contracts:CatalogItemCreated, type: fanout, durable
dbug: MassTransit.SendTransport[0]
SEND rabbitmq://localhost/Play.Catalog.Contracts:CatalogItemCreated 6c5f0000-99d1-b42e-d4a0-08d8c465f2a7 Play.Catalog.Contracts.CatalogItemCreated

item-deleted - amq.ctag-n8n_DoCm0Q17RgN4VoL-eA
dbug: MassTransit.ReceiveTransport[0]
Consumer Ok: rabbitmq://localhost/inventory-catalog-item-created - amq.ctag-IzpD9vk8hrqXAVk0FZMQQQ
dbug: MassTransit[0]
Endpoint Ready: rabbitmq://localhost/inventory-catalog-item-created
dbug: MassTransit[0]
Endpoint Ready: rabbitmq://localhost/inventory-catalog-item-deleted
dbug: MassTransit[0]
Endpoint Ready: rabbitmq://localhost/inventory-catalog-item-updated
dbug: MassTransit[0]
Bus started: rabbitmq://localhost/
dbug: MassTransit.ReceiveTransport[0]
RECEIVE rabbitmq://localhost/inventory-catalog-item-created 6c5f0000-99d1-b42e-d4a0-08d8c465f2a7 Play.Catalog.Contracts.CatalogItemCreated Play.Inventory.Service.Consumers.CatalogItemCreatedConsumer(00:00:00.3058063)

Ln 9, Col 2 Spaces: 2 UTF-8 LF JSON

Frontend (React)



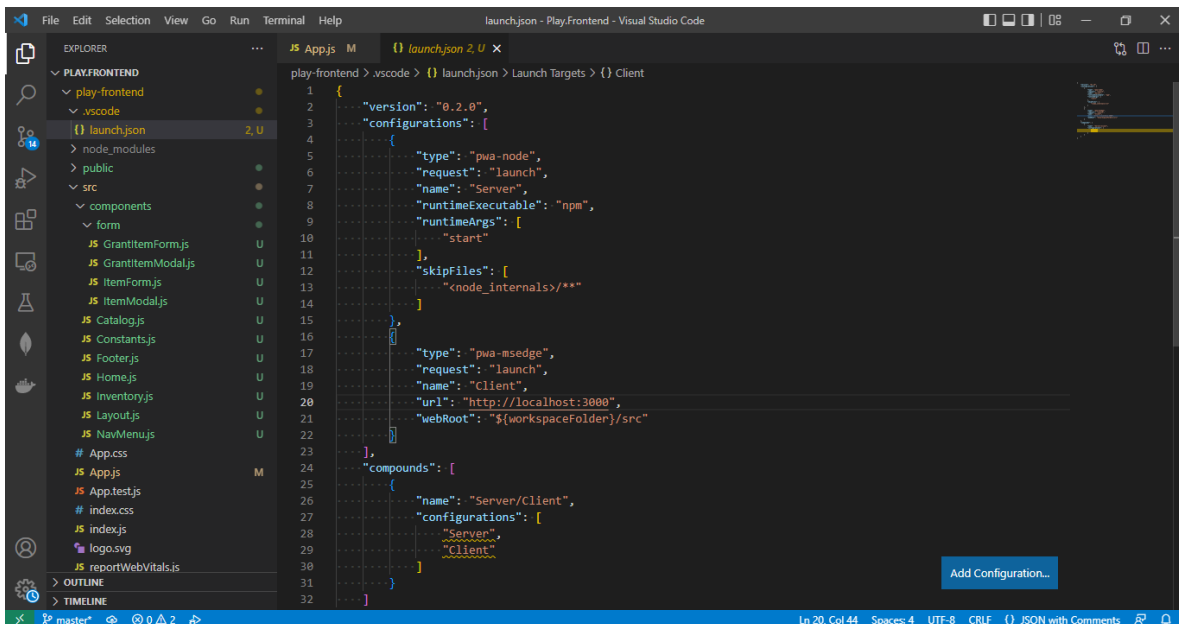
```

File Edit Selection View Go Run Terminal Help
App.js - Play.Frontend - Visual Studio Code

EXPLORER
PLAYFRONTEND
  play-frontend
    .vscode
      launch.json
    node_modules
    public
    src
      components
        form
          GrantItemForm.js
          GrantItemModal.js
          ItemForm.js
          ItemModal.js
          Catalog.js
          Constants.js
          Footer.js
          Home.js
          Inventory.js
          Layout.js
          NavMenu.js
        # App.css
      App.js
      App.test.js
      # index.css
      index.js
      logo.svg
      reportWebVitals.js
    .OUTLINE
    .TIMELINE

play-frontend > src > JS App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3
4 import React, { Component } from React;
5 import { Route } from 'react-router';
6 import { Layout } from './components/Layout';
7 import { Home } from './components/Home';
8 import { Catalog } from './components/Catalog';
9 import { Inventory } from './components/Inventory';
10 import { ApplicationPaths } from './components/Constants';
11
12 export default class App extends Component
13 {
14   static displayName = App.name;
15   render()
16   {
17     return (
18       <Layout>
19         <Route exact path="/" component={Home} />
20         <Route path={ApplicationPaths.CatalogPath} component={Catalog} />
21         <Route path={ApplicationPaths.InventoryPath} component={Inventory} />
22       </Layout>
23     );
24   }
25 }
26
27 // function App() {
28 //   return (
29 //     <div className="App">
30 //       <header className="App-header">
31 //         <img src={logo} className="App-logo" alt="logo" />
32 //       </header>
33 //     </div>
34 //   );
35 // }
36
37 // Edit <code>src/App.js</code> and save to reload.
  
```

Hosted (Node.js) Web Server



```

File Edit Selection View Go Run Terminal Help
launch.json - Play.Frontend - Visual Studio Code

EXPLORER
PLAYFRONTEND
  play-frontend
    .vscode
      launch.json
    node_modules
    public
    src
      components
        form
          GrantItemForm.js
          GrantItemModal.js
          ItemForm.js
          ItemModal.js
          Catalog.js
          Constants.js
          Footer.js
          Home.js
          Inventory.js
          Layout.js
          NavMenu.js
        # App.css
      App.js
      App.test.js
      # index.css
      index.js
      logo.svg
      reportWebVitals.js
    .OUTLINE
    .TIMELINE

play-frontend > .vscode > {} launch.json > Launch Targets > {} Client
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 }
  
```

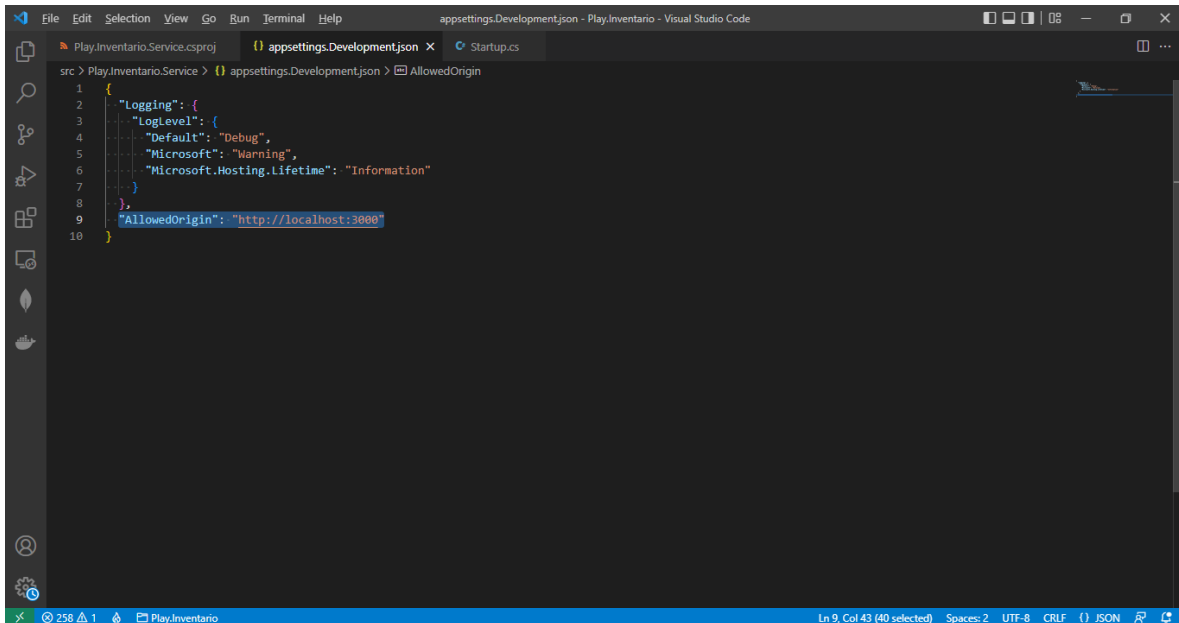
CORS (Cross-Origin Resource Sharing)

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.

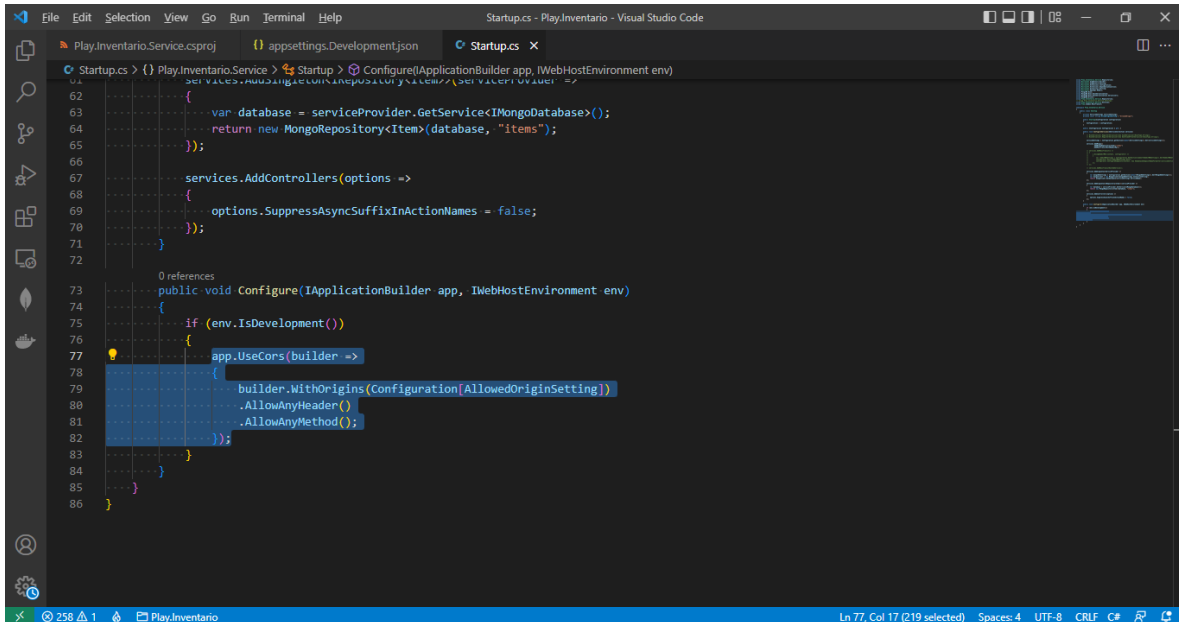
Permitir origen desde otras fuentes



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

The screenshot shows the Visual Studio Code interface with the file `appsettings.Development.json` open. The file contains a JSON configuration for logging and CORS. The `AllowedOrigin` property is set to `"http://localhost:3000"`. The status bar at the bottom indicates the file is in JSON format and the current selection is on line 9, column 43.

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

Startup.cs - Play.Inventario - Visual Studio Code

Play.Inventario.Service.csproj appsettings.Development.json Startup.cs X

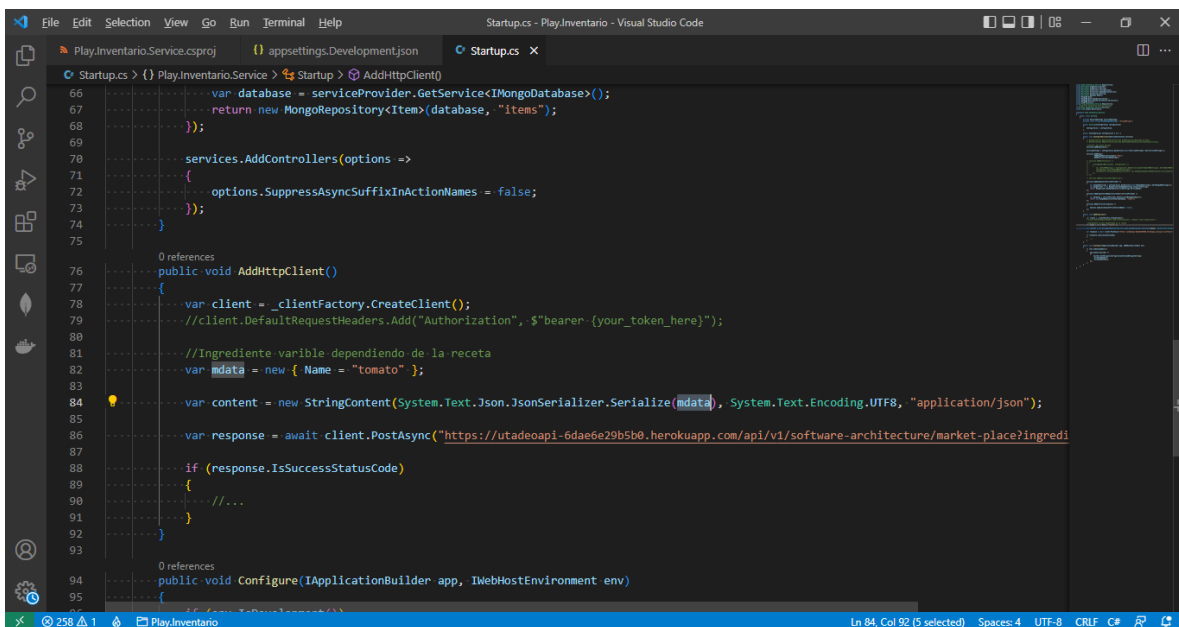
Startup.cs > {} Play.Inventario.Service > Startup > Configure(IApplicationBuilder app, IWebHostEnvironment env)
services.AddSingleton<IMongoRepository>(() => {
    var database = serviceProvider.GetService<IMongoDatabase>();
    return new MongoRepository<Item>(database, "items");
});

services.AddControllers(options =>
{
    options.SuppressAsyncSuffixInActionNames = false;
});

0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseCors(builder =>
        {
            builder.WithOrigins(Configuration[AllowedOriginSetting])
                .AllowAnyHeader()
                .AllowAnyMethod();
        });
    }
}
```

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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

Startup.cs - Play.Inventario - Visual Studio Code

Play.Inventario.Service.csproj appsettings.Development.json Startup.cs X

Startup.cs > {} Play.Inventario.Service > Startup > AddHttpClient()
var database = serviceProvider.GetService<IMongoDatabase>();
return new MongoRepository<Item>(database, "items");

services.AddControllers(options =>
{
    options.SuppressAsyncSuffixInActionNames = false;
});

0 references
public void AddHttpClient()
{
    var client = _clientFactory.CreateClient();
    //client.DefaultRequestHeaders.Add("Authorization", $"bearer {your_token_here}");

    //Ingrediente variable dependiendo de la receta
    var mdata = new { Name = "tomato" };

    var content = new StringContent(System.Text.Json.JsonSerializer.Serialize(mdata), System.Text.Encoding.UTF8, "application/json");

    var response = await client.PostAsync("https://utadeoapi-6dae6e29b5b0.herokuapp.com/api/v1/software-architecture/market-place?ingredient=tomato", content);

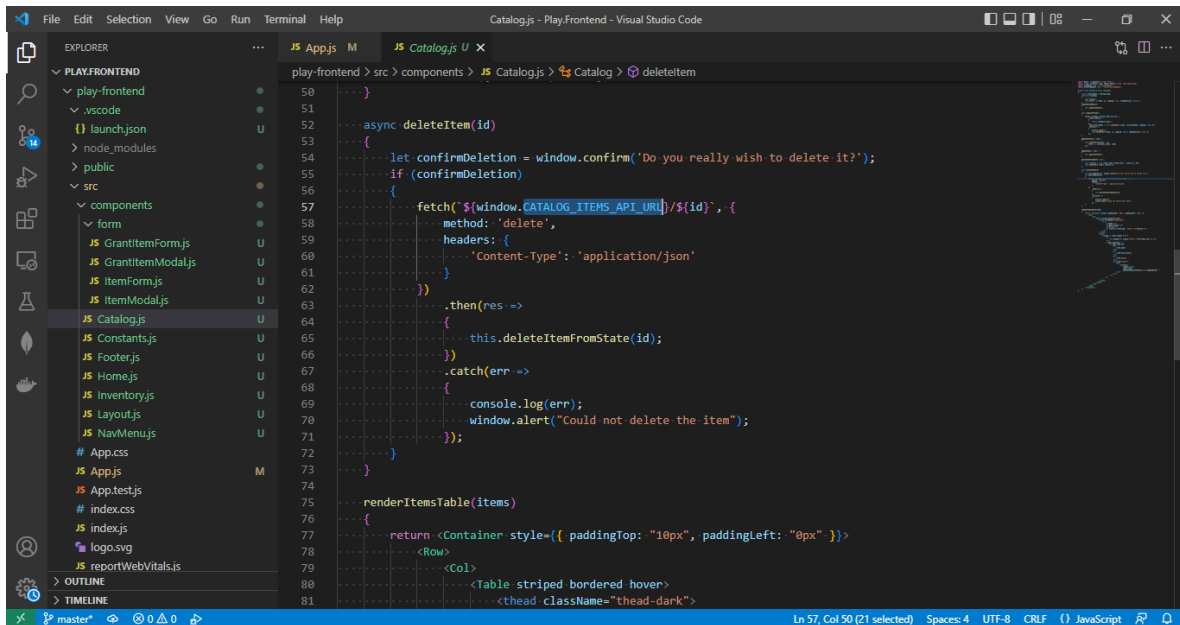
    if (response.IsSuccessStatusCode)
    {
        //...
    }
}

0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    //...
}
```

← → ↻ 🏠 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.

GitHub Repositorio

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