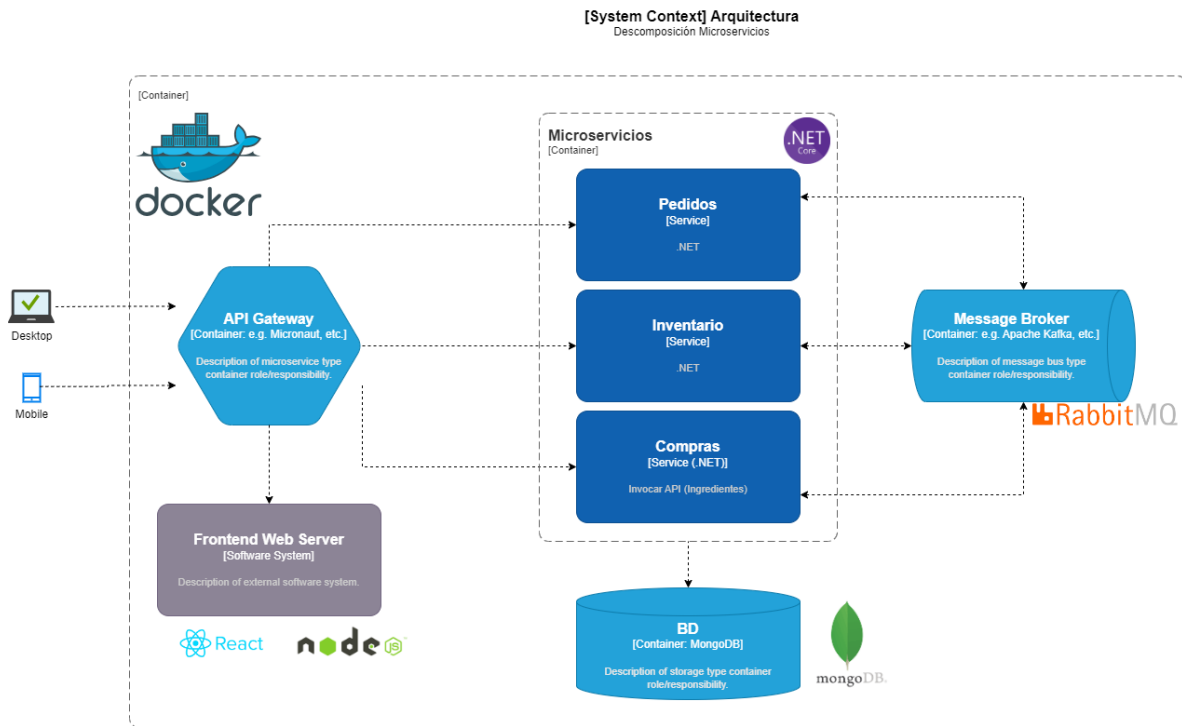


ARQUITECTURA DE SOFTWARE (2S-2023)

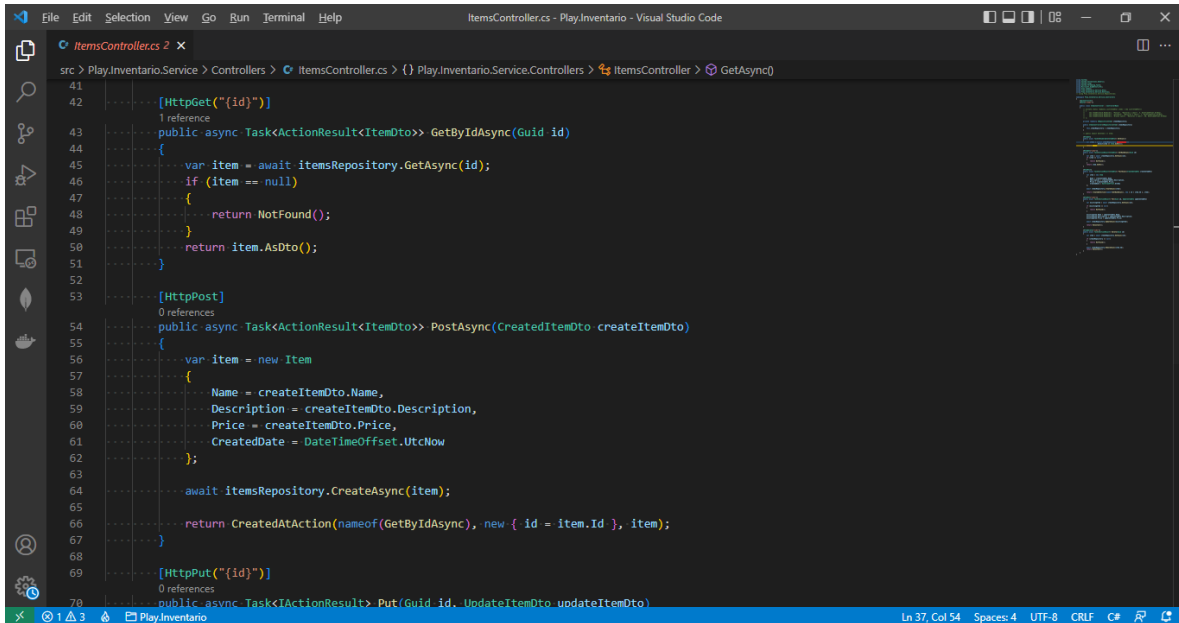
PROYECTO CORTE 2 (MICROSERVICIOS)

EDISSON CABRERA ERASO

Arquitectura Propuesta

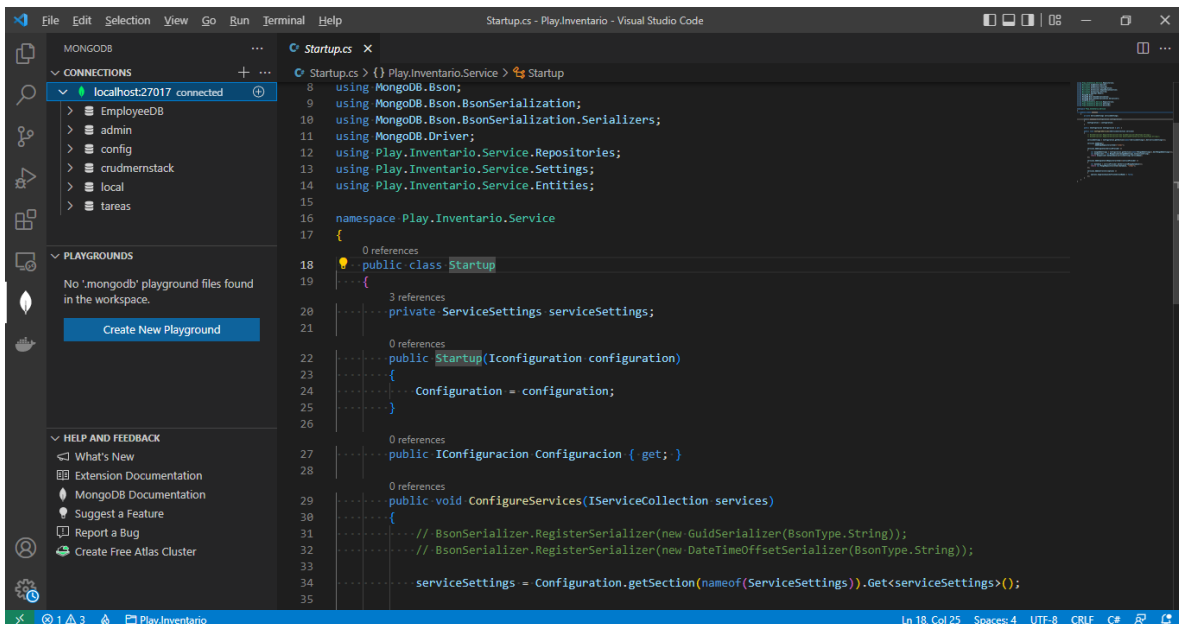


API's



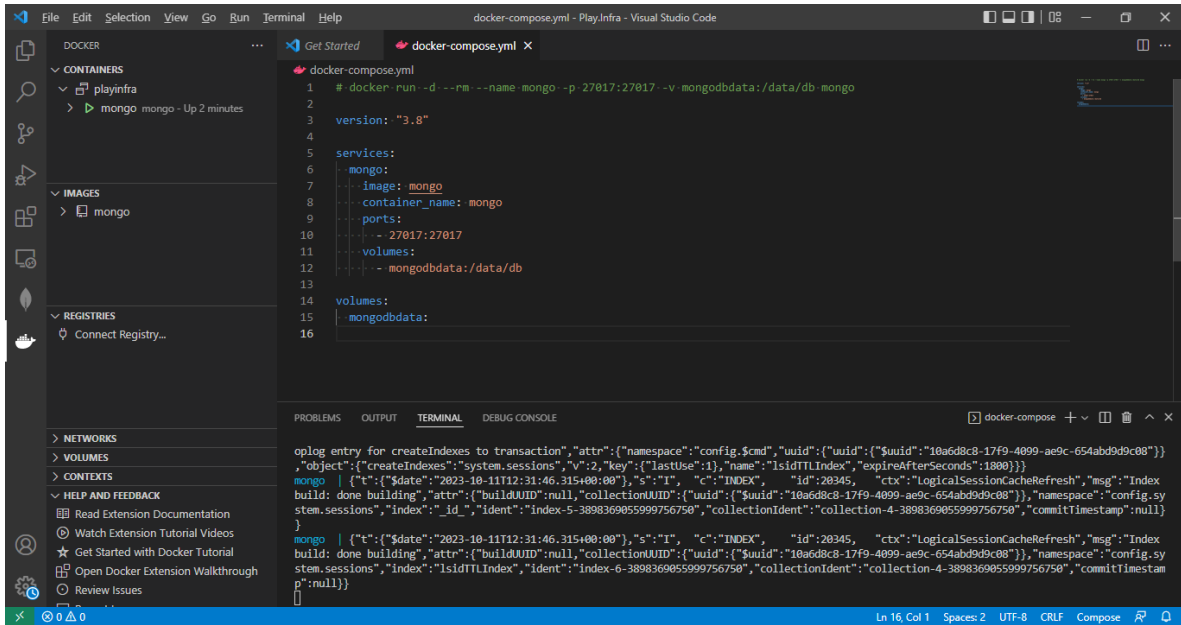
```
41  
42 [HttpGet("{id}")]  
43 public async Task<ActionResult<ItemDto>> GetByIdAsync(Guid id)  
44 {  
45     var item = await itemsRepository.GetAsync(id);  
46     if (item == null)  
47     {  
48         return NotFound();  
49     }  
50     return item.AsDto();  
51 }  
52  
53 [HttpPost]  
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 }
```

Docker Compose (yaml)

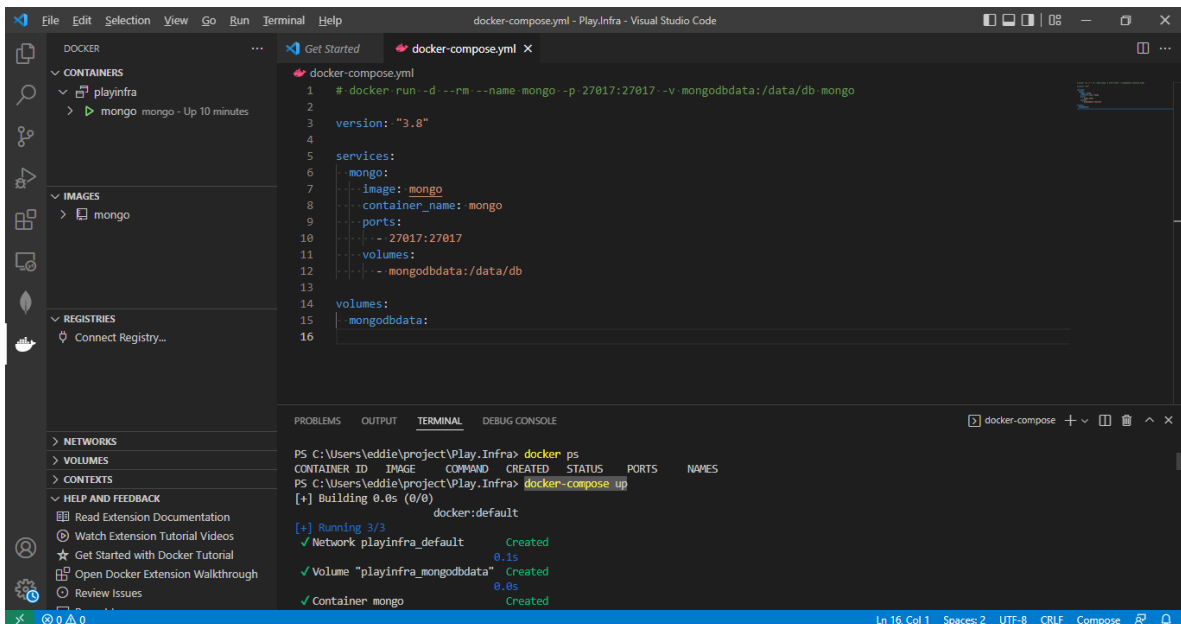


The screenshot shows a Visual Studio Code editor with a Docker Compose file named `docker-compose.yml`. The file is configured to run a MongoDB container. The left sidebar shows the Docker extension interface with 'CONTAINERS', 'IMAGES', and 'REGISTRIES' sections. The 'CONTAINERS' section shows a 'mongo' 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
    0.1s
  ✓ Volume "playinfra_mongoddata" Created
    0.0s
  ✓ 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

```

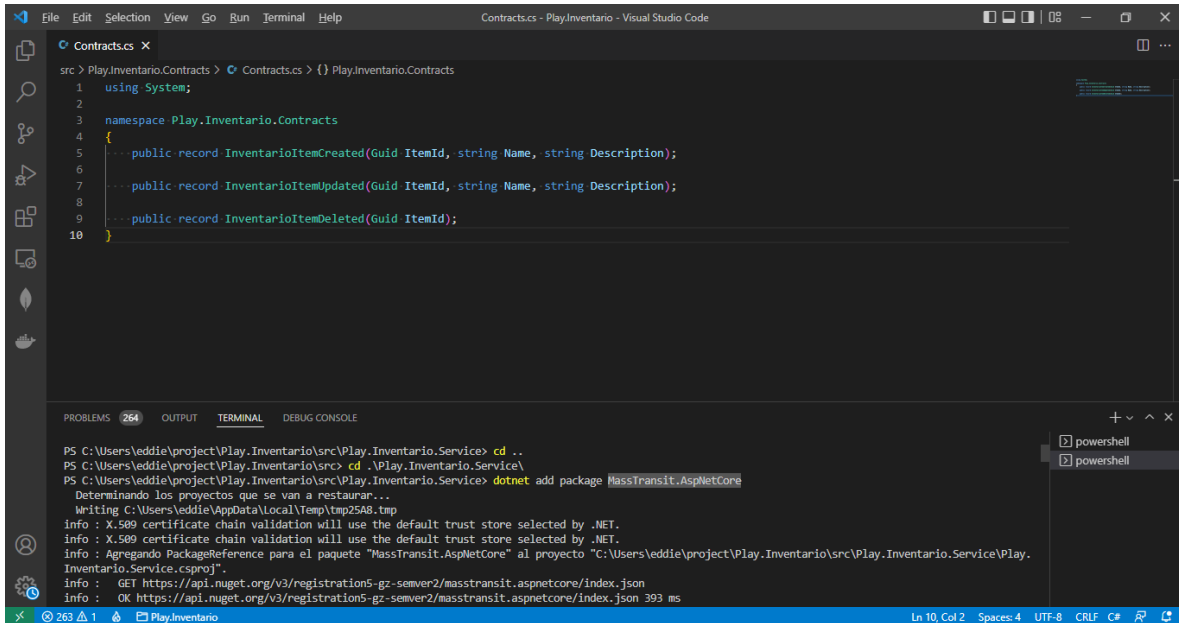
warn: Play.Inventory.Service.Clients.CatalogClient[0]
      Delaying for 4 seconds, then making retry Polly.Context
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[100]
      Sending HTTP request GET https://localhost:5001/items
info: System.Net.Http.HttpClient.CatalogClient.ClientHandler[101]
      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)

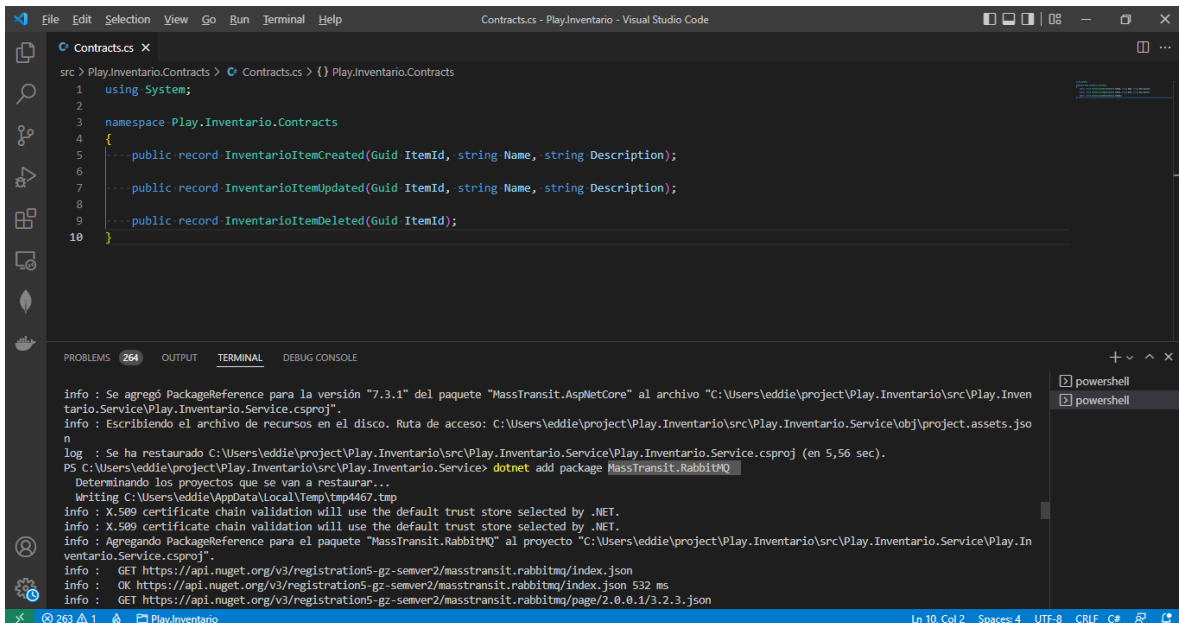


The screenshot shows the Visual Studio Code interface with the 'Contracts.cs' file open. The file contains the following code:

```
src > Play.Inventario.Contracts > Contracts.cs > {} Play.Inventario.Contracts
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 terminal output shows the command to add the package and the resulting output:

```
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
```

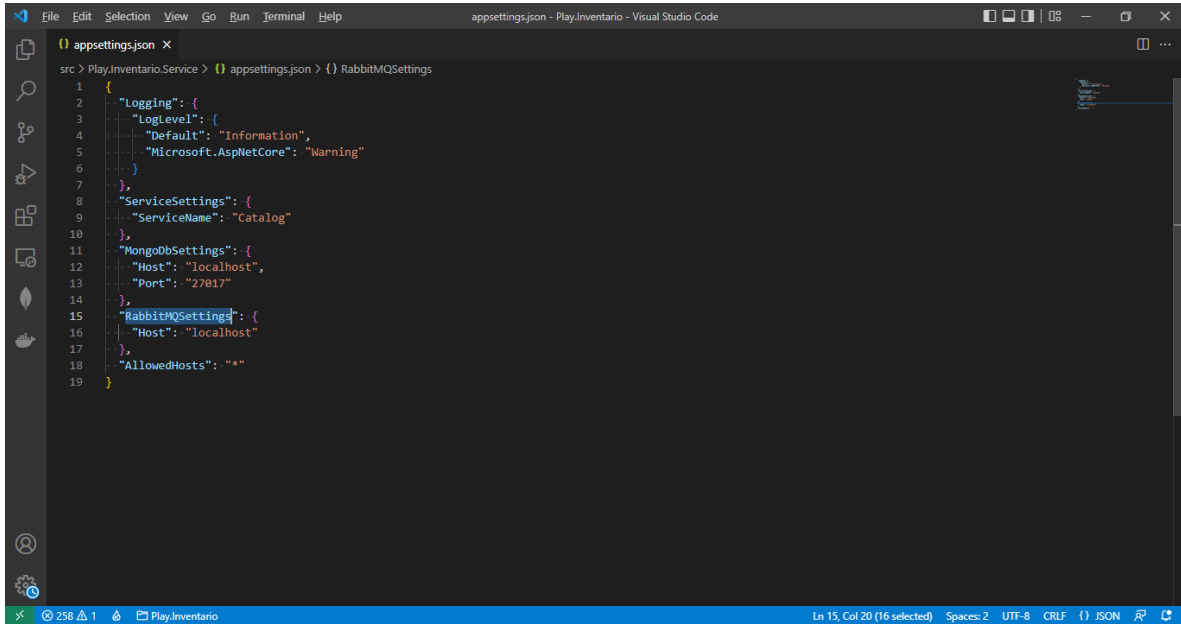


The screenshot shows the Visual Studio Code interface with the 'Contracts.cs' file open. The file contains the following code:

```
src > Play.Inventario.Contracts > Contracts.cs > {} Play.Inventario.Contracts
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 terminal output shows the command to add the package and the resulting output:

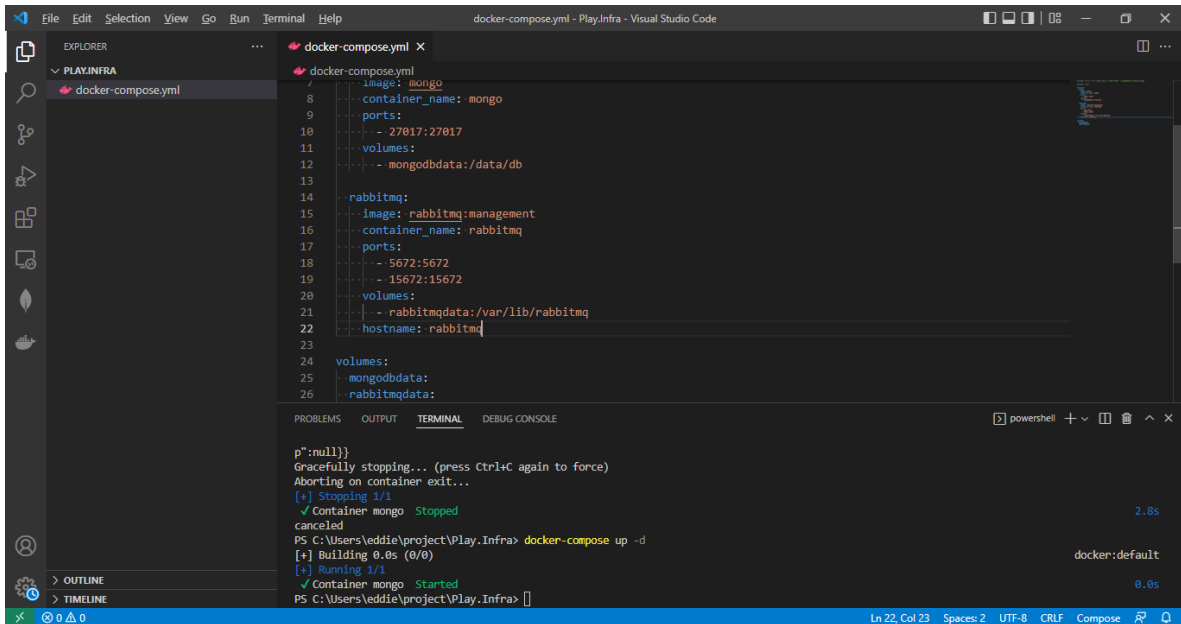
```
Info : Se agregó PackageReference para la versión "7.3.1" del paquete "MassTransit.AspNetCore" al archivo "C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service\Play.Inventario.Service.csproj".
Info : Escribiendo el archivo de recursos en el disco. Ruta de acceso: C:\Users\eddie\project\Play.Inventario\src\Play.Inventario.Service\obj\project.assets.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 editor showing the `appsettings.json` file. The file is located at `src > Play.Inventario.Service > {} appsettings.json > {} RabbitMQSettings`. The content of the file is as follows:

```
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: `Ln 15, Col 20 (16 selected) Spaces: 2 UTF-8 CRLF {} JSON`.



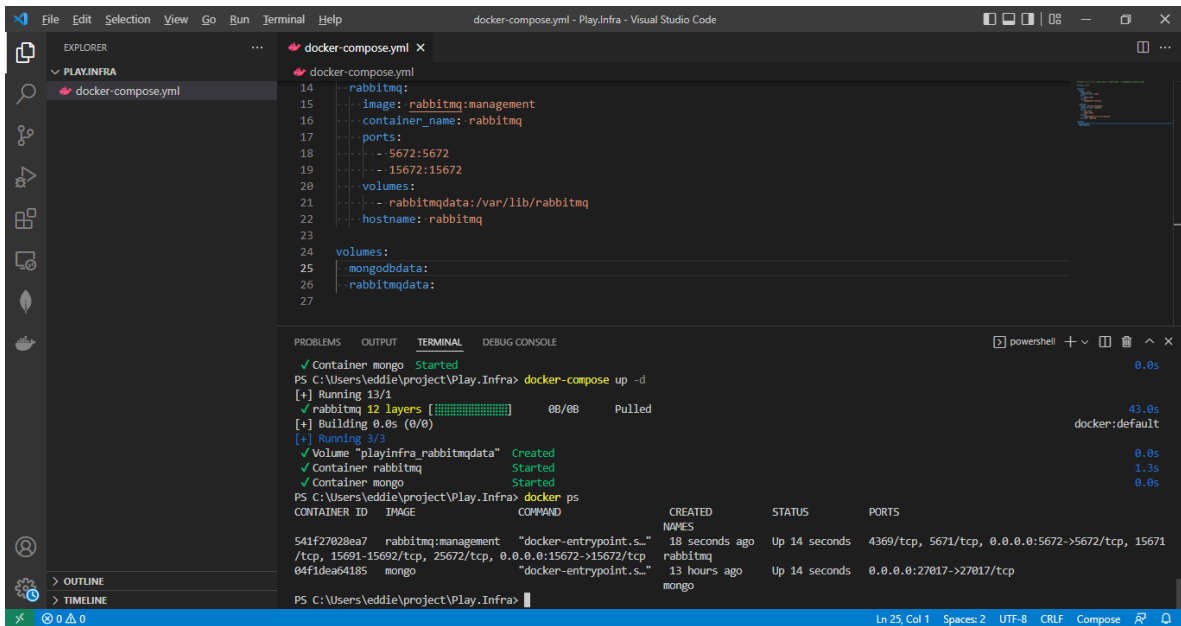
Visual Studio Code editor showing the `docker-compose.yml` file. The file is located at `docker-compose.yml - Play.Infra - Visual Studio Code`. The content of the file is as follows:

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

The terminal output shows the following commands and results:

```
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>
```

The status bar at the bottom indicates: `Ln 22, Col 23 Spaces: 2 UTF-8 CRLF Compose`.



The screenshot shows a Visual Studio Code editor with a Docker Compose file named `docker-compose.yml` open. The file defines three services: `rabbitmq`, `mongo`, and `playinfra`. The `rabbitmq` service uses the `rabbitmq:management` image and maps ports 5672 and 15672. The `mongo` service uses the `mongo` image. The `playinfra` service uses the `mongo` image and maps port 27017. The terminal shows the output of the `docker-compose up -d` command, indicating that all services were successfully started.

```

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  -- mongo:
26  --   - rabbitmqdata:
27

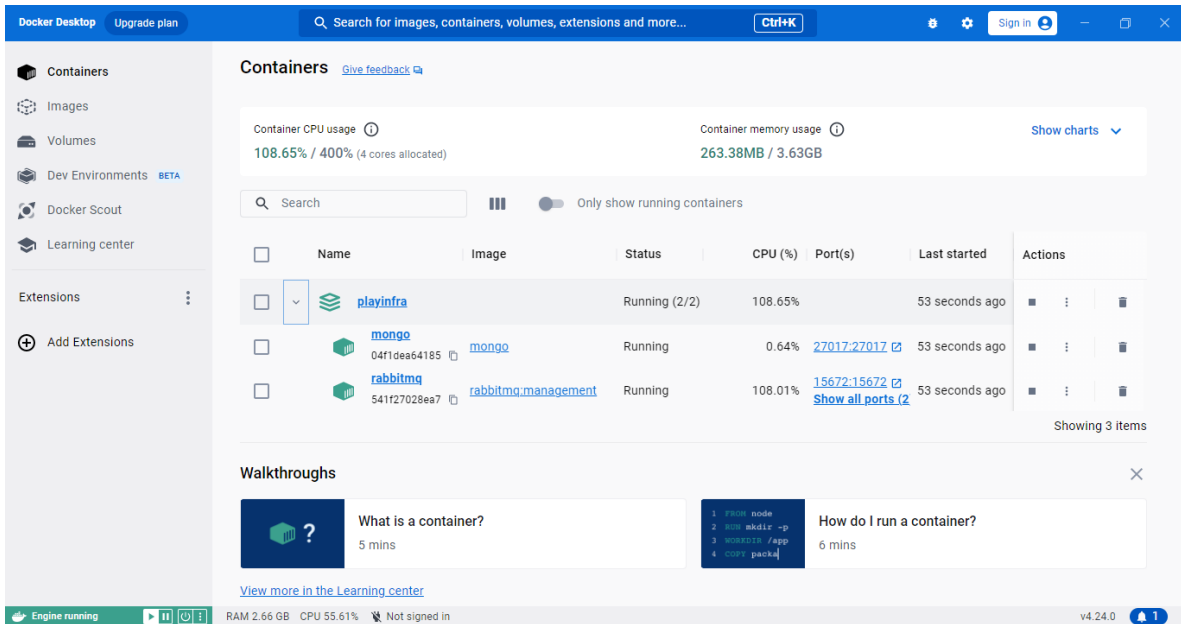
```

Terminal Output:

```

PS C:\Users\eddie\project\Play.Infra> docker-compose up -d
[+] Running 13/1
✓ rabbitmq 12 layers [0/0] 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
541f27028ea7   rabbitmq:management  "docker-entrypoint.s..."  18 seconds ago Up 14 seconds 4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671
04f1dea64185   mongo          "docker-entrypoint.s..."  13 hours ago  Up 14 seconds 0.0.0.0:27017->27017/tcp
mongo

```



The screenshot 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' tab, showing a list of running containers. The system status at the bottom indicates that the engine is running with 2.66 GB of RAM and 55.61% CPU usage.

Containers Summary:

- Container CPU usage: 108.65% / 400% (4 cores allocated)
- Container memory usage: 263.38MB / 3.63GB

Only show running containers

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

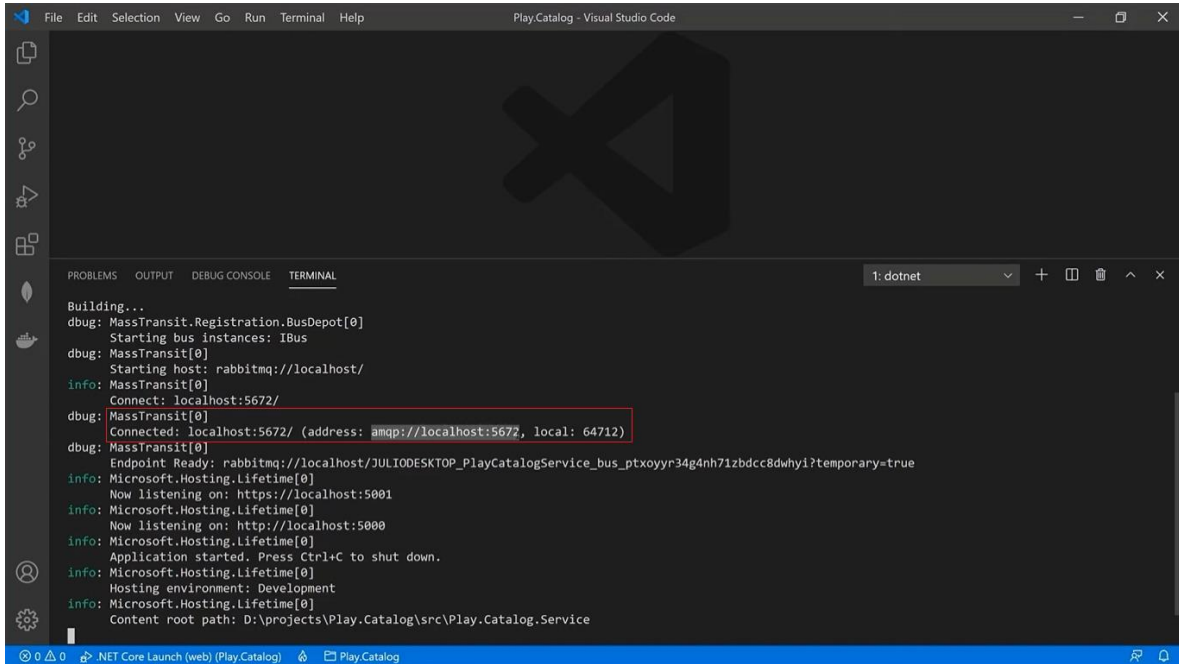
Showing 3 items

Walkthroughs:

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

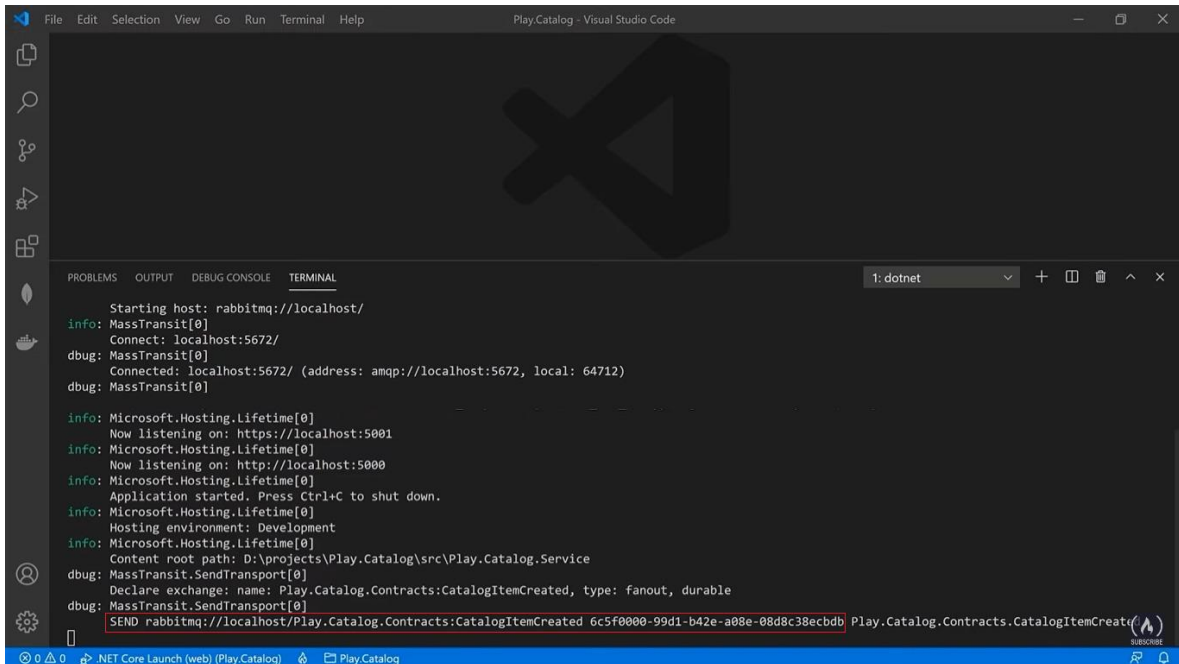
View more in the Learning center

Engine running | RAM 2.66 GB | CPU 55.61% | Not signed in | v4.24.0



```
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
```

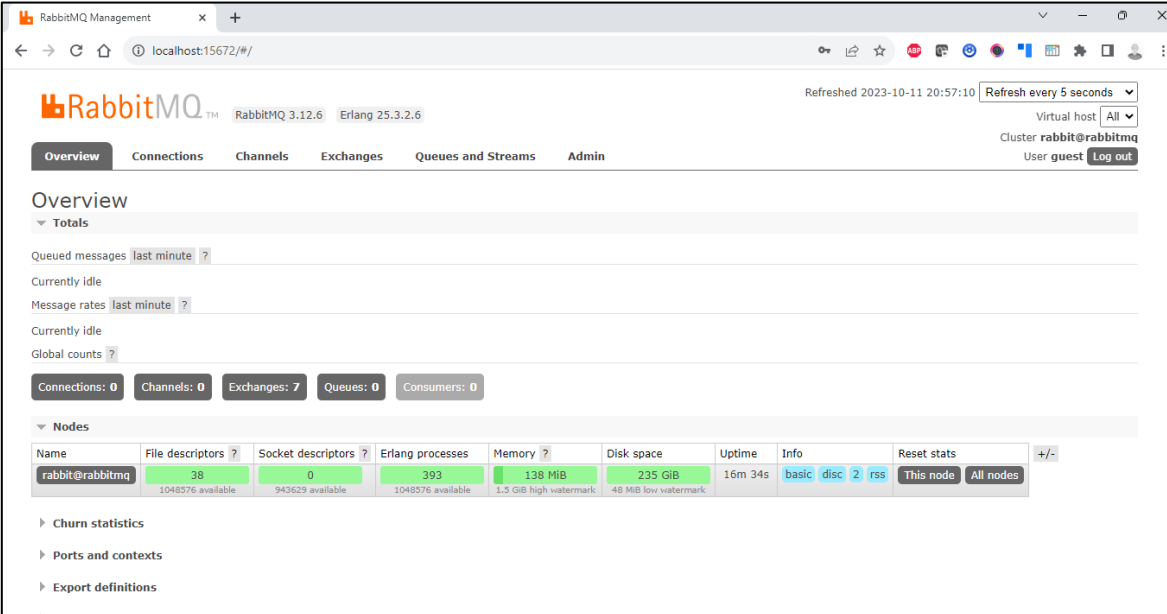
Enviar mensaje mediante RabbitMQ



```
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]

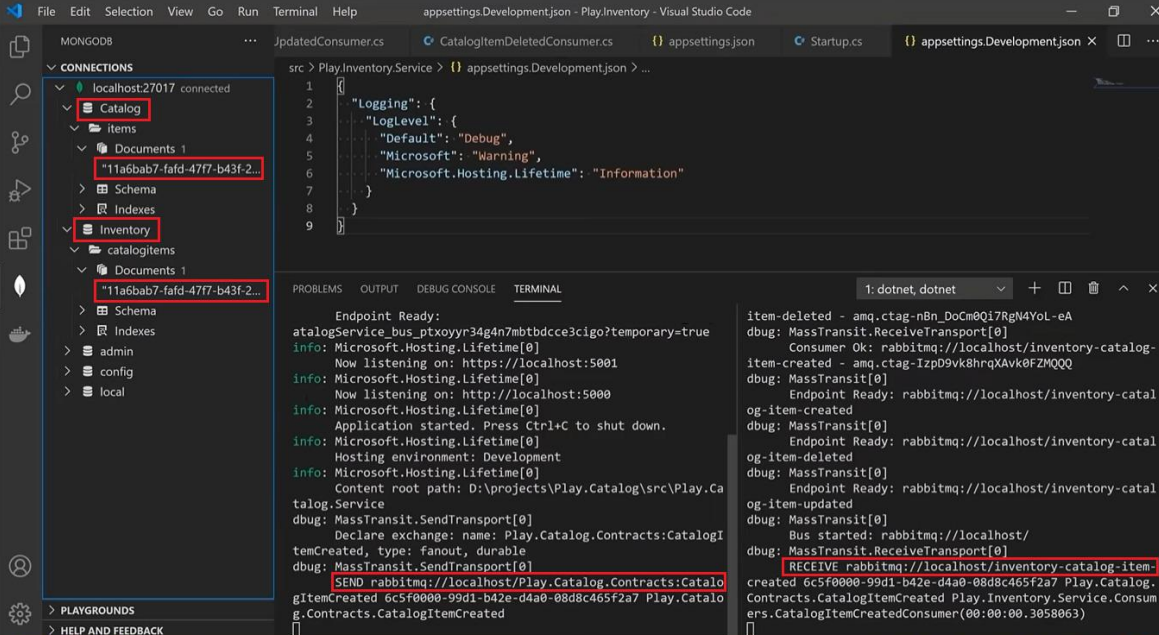
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting 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



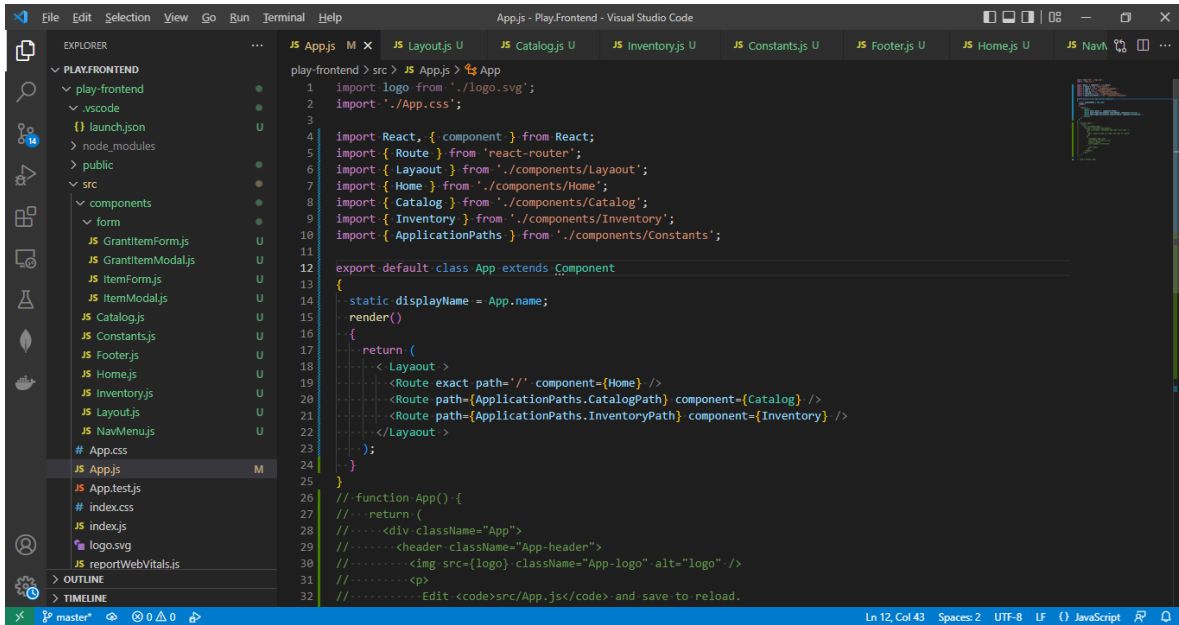
The screenshot shows the RabbitMQ Management Console interface. At the top, it displays 'RabbitMQ 3.12.6' and 'Erlang 25.3.2.6'. The 'Overview' tab is selected, showing a summary of the system's status. The 'Totals' section indicates that there are 0 connections, 0 channels, 7 exchanges, 0 queues, and 0 consumers. Below this, the 'Nodes' section shows a single node named 'rabbit@rabbitmq' with various resource usage metrics. The 'Churn statistics', 'Ports and contexts', 'Export definitions', and 'Import definitions' sections are also visible.

Enviar y recibir mensajes entre microservicios a través de RabbitMQ



The screenshot shows a Visual Studio Code editor with a project named 'Play.Inventory'. The 'CONNECTIONS' sidebar on the left shows a list of connections, with 'localhost:27017' and 'localhost:5000' highlighted. The main editor displays the 'appsettings.json' file, which contains configuration for the application. The 'TERMINAL' pane at the bottom shows the output of the application, including logs for the 'Play.Inventory' service and the 'CatalogItemDeletedConsumer' service. The logs indicate that the application is running and has successfully received a message from the 'Play.Inventory' service.

Frontend (React)



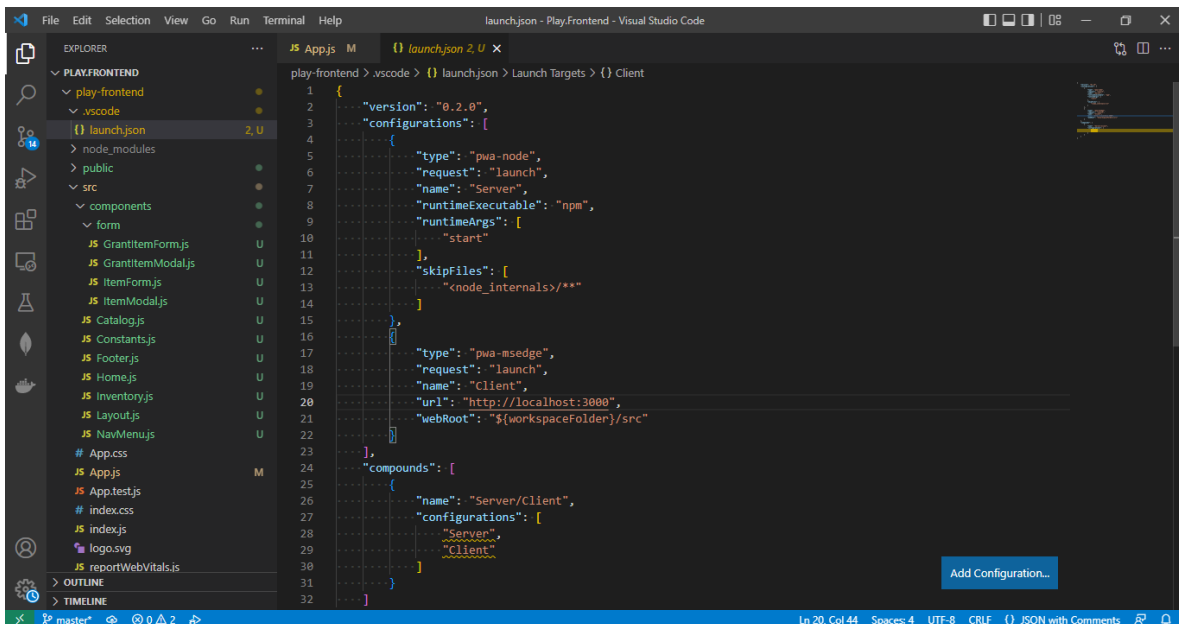
```

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 // 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 }
  
```

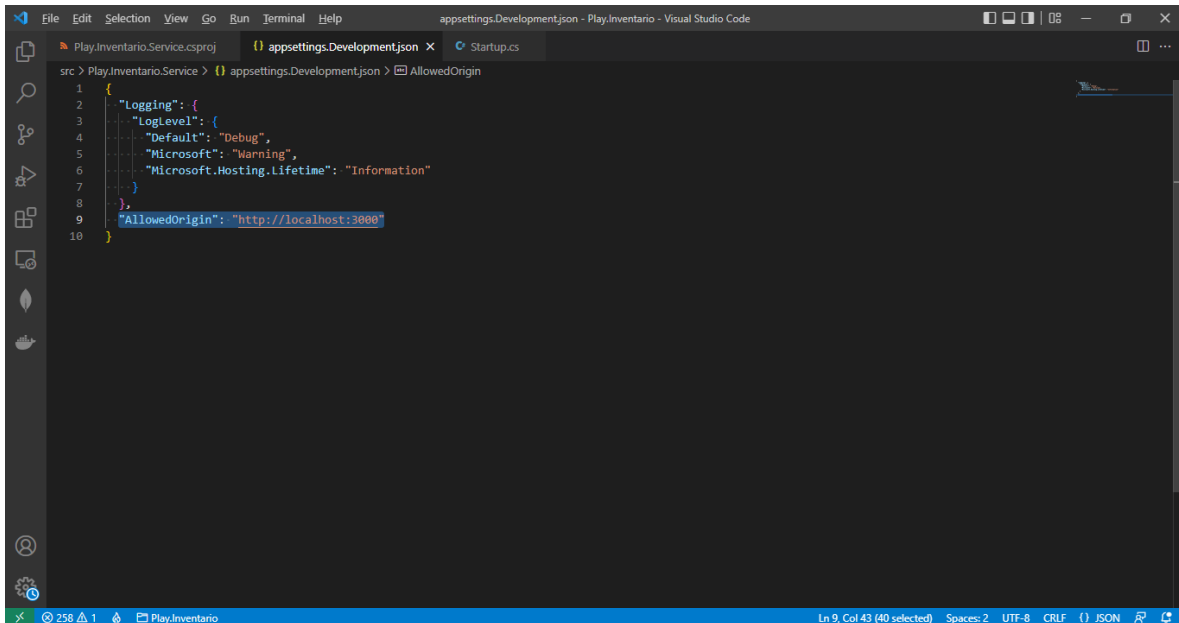
CORS (Cross-Origin Resource Sharing)

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.

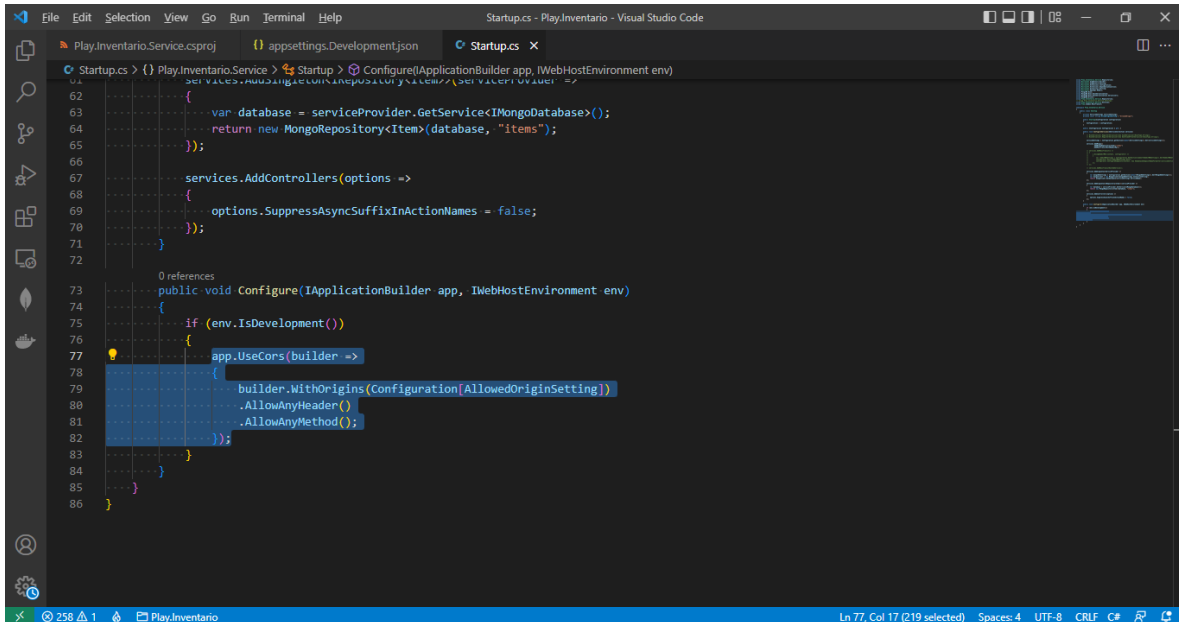
Permitir origen desde otras fuentes



```
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 shows the current line and column.

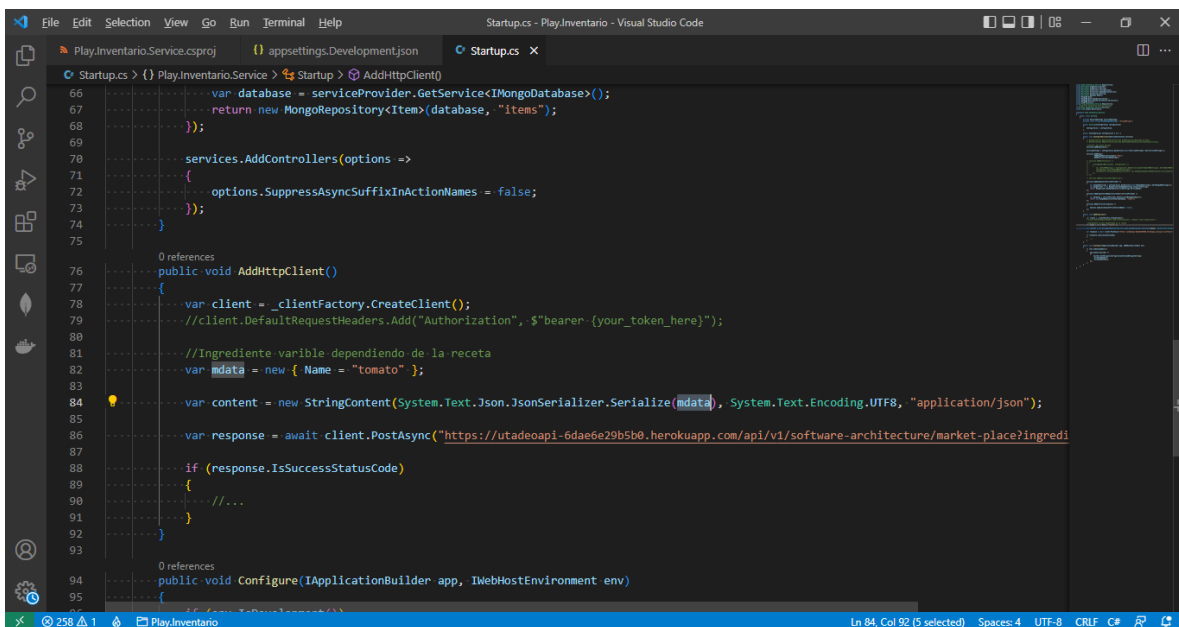
Permitir otros orígenes



```
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>



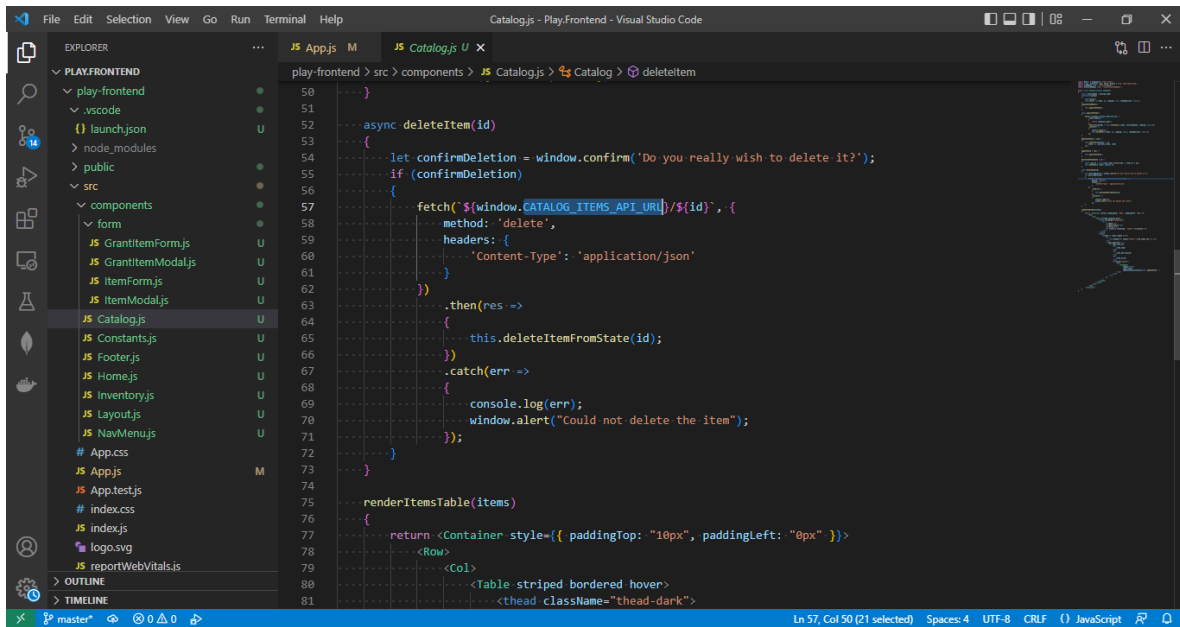
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
```



```
< > ↺ 🏠 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)

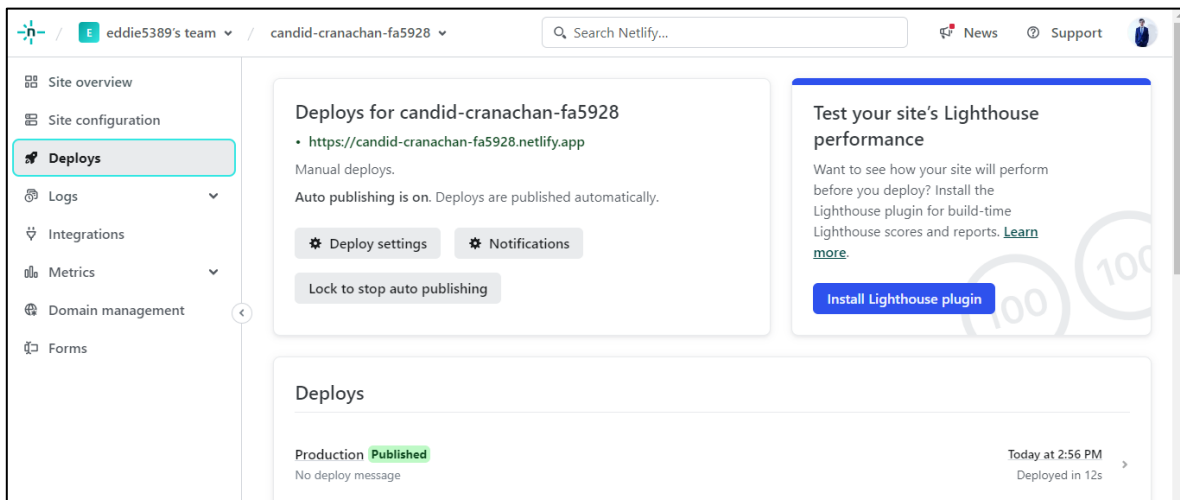
- Tomato (1 unidad)
- Ketchup (1 unidad)
- Preparación: Forma las hamburguesas y cocínalas. Arma las hamburguesas con lechuga, tomate y salsa de ketchup.

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.

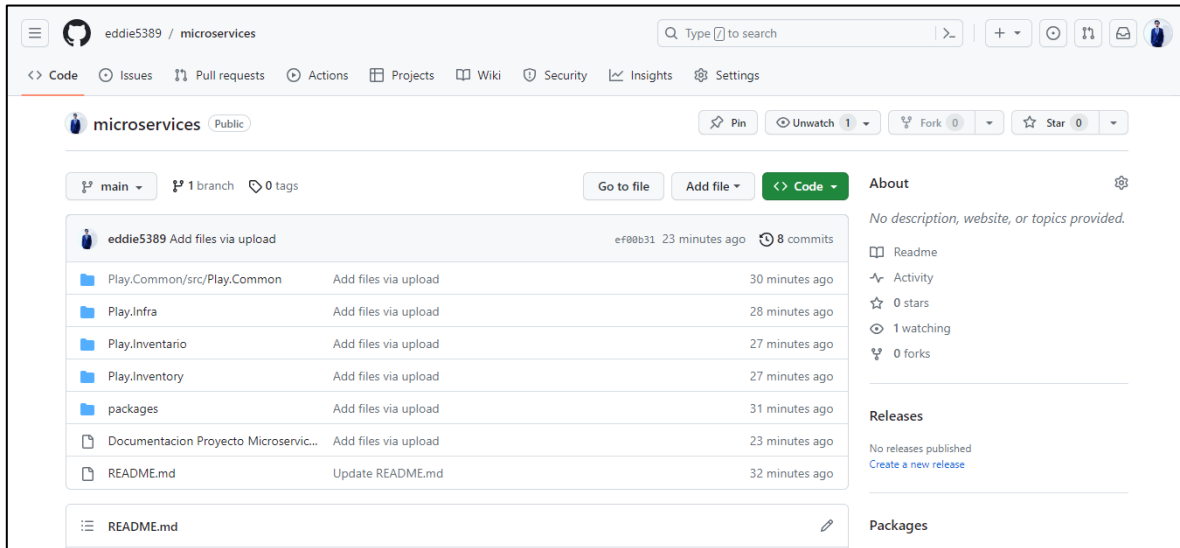
Deploy App NETLIFY

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.

