

## DESARROLLO DE APLICACIONES EMPRESARIALES AVANZADAS

### LABORATORIO N° 02

## Desarrollo de aplicaciones MDI con Windows Forms



<b>Alumno(s):</b>	<i>Juan Escobar Mendoza</i>				<b>Nota</b>	
<b>Grupo:</b>	<i>B</i>	<b>Ciclo: VI</b>				
<b>Criterio de Evaluación</b>	<b>Excelente (4pts)</b>	<b>Bueno (3pts)</b>	<b>Requiere mejora (2pts)</b>	<b>No acept. (0pts)</b>	<b>Puntaje Logrado</b>	
Identifica diferentes tipos de formularios en C#						
Identifica los controles y herramientas para el diseño de interfaz gráfica con Windows Forms						
Emplea conceptos de programación y estructuras de datos						
Realiza las actividades planteadas						
Es puntual y redacta el informe adecuadamente						

## **Laboratorio 02: Desarrollo de aplicaciones MDI con Windows Forms**

### **Objetivos:**

Al finalizar el laboratorio el estudiante será capaz de:

- Realizar el diseño de formularios MDI y formularios estándar
- Reconocer el diseño de la interfaz para sistemas de gestión.
- Aplicar el uso de propiedades, eventos y métodos en C# para diseño de aplicaciones

### **Seguridad:**

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

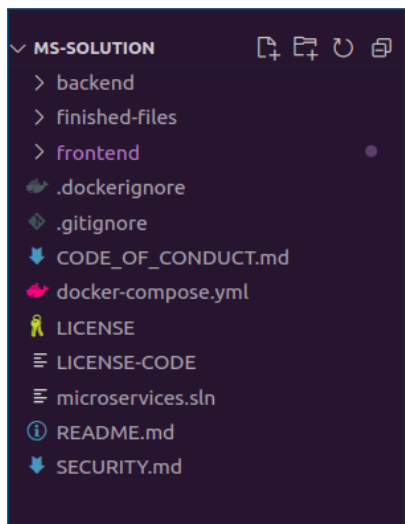
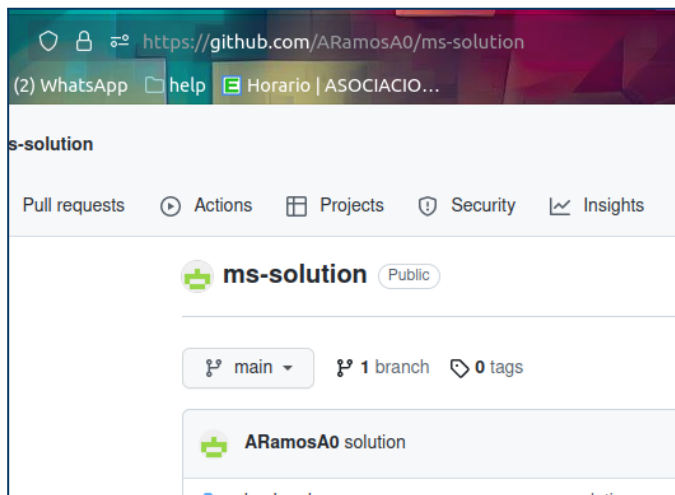
### **Equipos y Materiales:**

- Una computadora con:
  - Windows 7 o superior
  - Conexión a la red del laboratorio
  - Visual Studio 2017 Community Edition

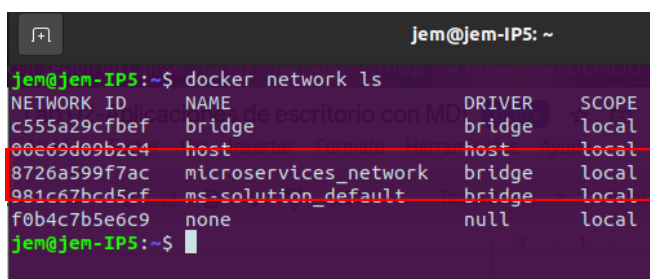
### **Procedimiento:**

## Part 1: PizzaBackend microservices with backend and frontend

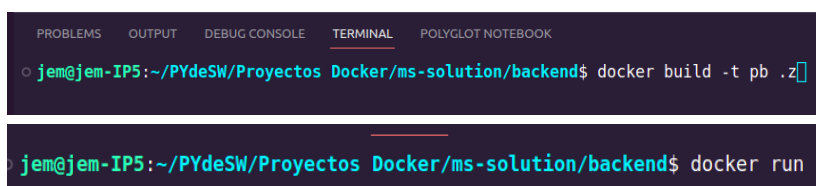
- Cloning the repository of the solution



- We are going to use the installed docker, so we need to create a new network and then attach both containers.



- Creating the backend image and running in port 80



```

jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/backend$ docker start pb
pb
jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/backend$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
a876b8b2177d   pb        "dotnet backend.dll"    30 hours ago  Up 3 seconds  0.0.0.0:80->80/tcp,
jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/backend$

```

- Modifying the PizzaClient.cs to provide the backend url, then we have to do the same in appsettings.Development.json.

```

PizzaClient.cs M X Settings Dockerfile {} appsettings
frontend > PizzaClient.cs
24
25     public async Task<PizzaInfo[]> GetPizzasAsync
26     {
27         try {
28             var responseMessage = await this.client
29             GetAsync("http://pb:80/pizzainfo");
30             if (responseMessage != null)

```

```

Dockerfile {} appsettings.Development.json M X Program.cs
frontend > {} appsettings.Development.json > backendUrl
1  {
2      "Logging": {
3          "LogLevel": {
4              "Default": "Information",
5              "Microsoft": "Warning",
6              "Microsoft.Hosting.Lifetime": "Information"
7          }
8      },
9      "backendUrl": "http://pb:80/pizzainfo"
10 }
11

```

This means the frontend will expect that backend url, but we use pb because later we are going to attach them in the same network, so 'pb' is the name of the backend container, and of course 80 is the port in which it will operate.

- Creating the frontend image and making it run in port 4000. Just to clarify, because we're gonna use the development mode, we need to change that env variable when we're gonna start the container.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POLYGLOT NOTEBOOK
jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/frontend$ docker build -t pf .

```

```

jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/frontend$ docker run -e ASPNETCORE_ENVIRONMENT=Development -p 4000:80 pf

```

```

jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/frontend$ docker start pf
pf
jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/frontend$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
ec9edf33b672   pf        "dotnet frontend.dll"    29 hours ago  Up 3 seconds  443/tcp, 0.0.0.0:4000->80/tcp, :::
a876b8b2177d   pb        "dotnet backend.dll"    30 hours ago  Up 12 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp,
jem@jem-IP5:~/PYdeSW/Proyectos Docker/ms-solution/frontend$

```

- Attaching the containers to the network

```

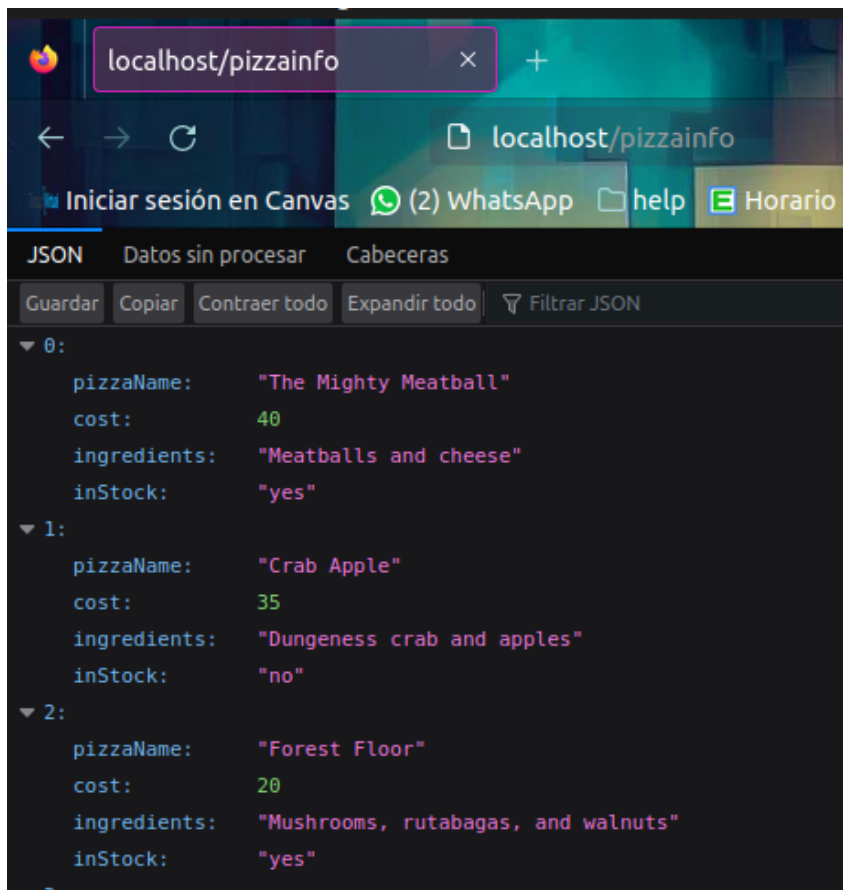
jem@jem-IP5:~$ docker network connect microservices_network pb

```

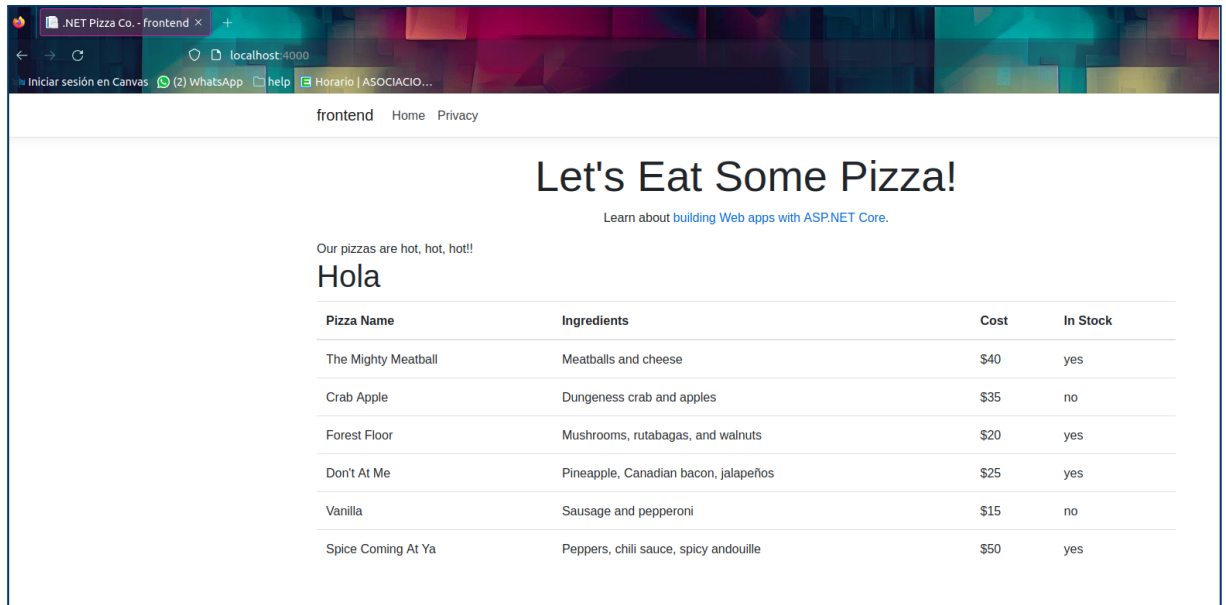
```
jen@jen-IP5:~$ docker network connect microservices_network pf
```

```
},
"ConfigOnly": false,
"Containers": {
  "a876b8b2177df5f61cc9dff689dbe4edfd3a476c3f71727ff4db84b6dff5b8bd": {
    "Name": "pb",
    "EndpointID": "3ffc45f1c85ea38fd7974b132f014254ab608de4e1dca381463249e3454f35f2",
    "MacAddress": "02:42:ac:13:00:02",
    "IPv4Address": "172.19.0.2/16",
    "IPv6Address": ""
  },
  "ec9edf33b672543831b35b75ed3c40117f11bfc888859c045f96aa1661f93ec2": {
    "Name": "pf",
    "EndpointID": "cc55eb403da2f41b421346acf161afa384cd24aa655cbed0380ec4f14ca21693",
    "MacAddress": "02:42:ac:13:00:03",
    "IPv4Address": "172.19.0.3/16",
    "IPv6Address": ""
  }
},
"Options": {},
"Labels": {}
}
```

- Results

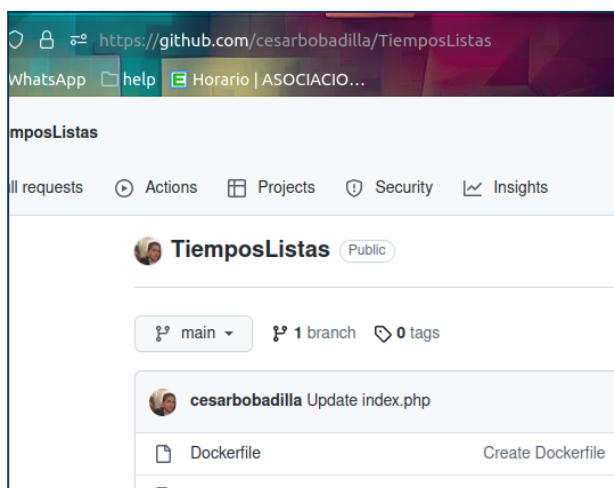


```
{
  "0": {
    "pizzaName": "The Mighty Meatball",
    "cost": 40,
    "ingredients": "Meatballs and cheese",
    "inStock": "yes"
  },
  "1": {
    "pizzaName": "Crab Apple",
    "cost": 35,
    "ingredients": "Dungeness crab and apples",
    "inStock": "no"
  },
  "2": {
    "pizzaName": "Forest Floor",
    "cost": 20,
    "ingredients": "Mushrooms, rutabagas, and walnuts",
    "inStock": "yes"
  },
  "3": {
    "pizzaName": "Forest Floor",
    "cost": 20,
    "ingredients": "Mushrooms, rutabagas, and walnuts",
    "inStock": "yes"
  }
}
```



## Part 2: Php gif tenor using pizza backend

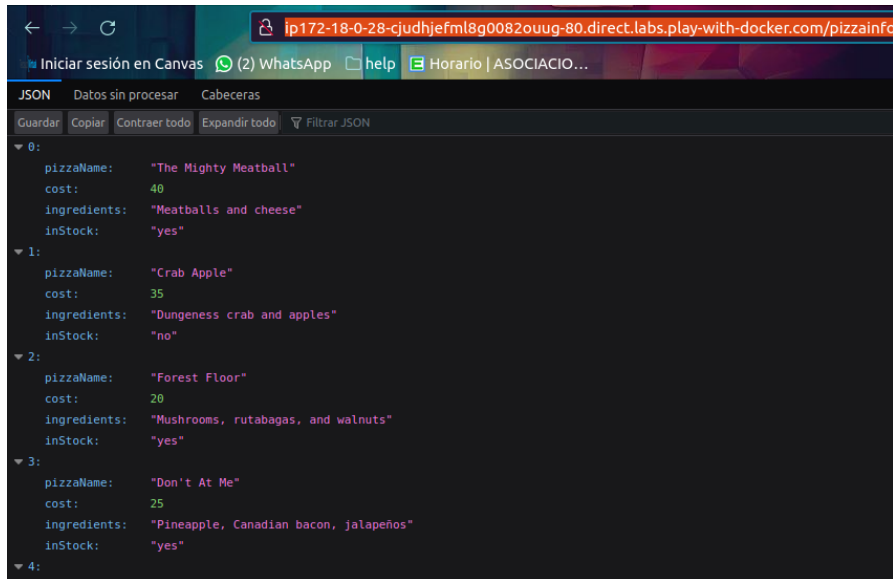
- Cloning the repository



- We're gonna clone it in play with docker, and in replit. And we're gonna use the pizzabackend from the previous step but now in play with docker.


```
[node1] (local) root@192.168.0.8 ~/ms-solution/backend
$ docker build -t pb .
[+] Building 24.7s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 324B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:6.0
=> [build 1/6] FROM mcr.microsoft.com/dotnet/sdk:6.0@sha256:7cb518612321c549bdb488c760ca2d00ec47f469fbafa006477b2
=> => resolve mcr.microsoft.com/dotnet/sdk:6.0@sha256:7cb518612321c549bdb488c760ca2d00ec47f469fbafa006477b2
=> => sha256:85aa8dbad730a716d1f72315f8a09ba7e3c69e065b482fb8f59886ebdf898e4 2.01kB / 2.01kB
=> => sha256:558f94638358037a58438553a1d3ab31af5a273a12b4485a55cd53ab7bc348 7.17kB / 7.17kB
```

```
[node1] (local) root@192.168.0.8 ~/ms-solution/backend
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
pb            latest   46efc5bb7b81   31 minutes ago 212MB
[node1] (local) root@192.168.0.8 ~/ms-solution/backend
$ docker run -p 80:80 pb
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```



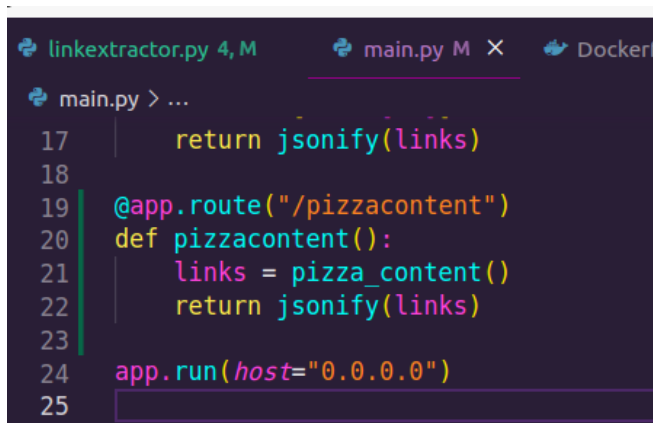
```
{
  "0": {
    "pizzaName": "The Mighty Meatball",
    "cost": 40,
    "ingredients": "Meatballs and cheese",
    "inStock": "yes"
  },
  "1": {
    "pizzaName": "Crab Apple",
    "cost": 35,
    "ingredients": "Dungeness crab and apples",
    "inStock": "no"
  },
  "2": {
    "pizzaName": "Forest Floor",
    "cost": 20,
    "ingredients": "Mushrooms, rutabagas, and walnuts",
    "inStock": "yes"
  },
  "3": {
    "pizzaName": "Don't At Me",
    "cost": 25,
    "ingredients": "Pineapple, Canadian bacon, jalape\u00f1os",
    "inStock": "yes"
  },
  "4": {
    "pizzaName": "Don't At Me",
    "cost": 25,
    "ingredients": "Pineapple, Canadian bacon, jalape\u00f1os",
    "inStock": "yes"
  }
}
```

Now to consume the pizzabackend we need to modify the next files.



```
linkextractor.py 4, M
main.py M
Dockerfile
Index.php

linkextractor.py > ...
19 //
20 return links
21
22 def pizza_content():
23     r = requests.get("http://ip172-19-0-25-cjtm5amfml8g00enlqp0-80.direct.labs.play-with-docker.com/pizzainfo")
24     print(r.content)
25     links = json.loads(r.content)
26     return links
27
28 if __name__ == "__main__":
29     if len(sys.argv) != 2:
30         print(f"Usage: {sys.argv[0]} <url>")
31         sys.exit(1)
```

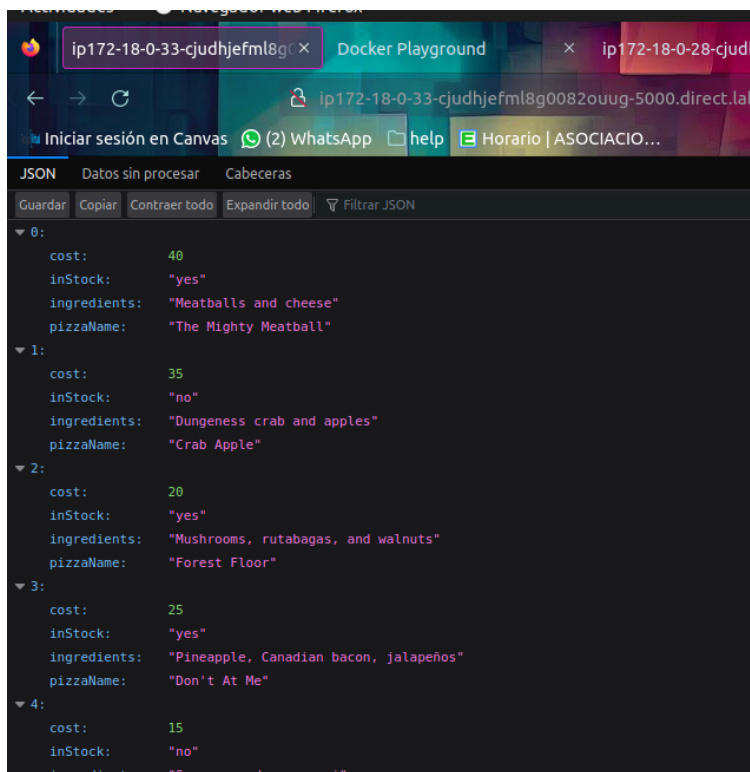


```
linkextractor.py 4, M
main.py M
Dockerfile

main.py > ...
17 return jsonify(links)
18
19 @app.route("/pizzacontent")
20 def pizzacontent():
21     links = pizza_content()
22     return jsonify(links)
23
24 app.run(host="0.0.0.0")
25
```

```
[node2] (local) root@192.168.0.7 ~/TiemposListas
$ docker rmi pc
Untagged: pc:latest
Deleted: sha256:46a3069941a78b925d204ee89514a09bf9cec4ffb3bbe8637efall1e001b010e1
[node2] (local) root@192.168.0.7 ~/TiemposListas
$ docker build -t pc .
[+] Building 0.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 228B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3
=> [1/6] FROM docker.io/library/python:3@sha256:8488a4b1a393b0b2cb479a2da0a0d11cf816a77c0f927820
=> [internal] load build context
=> => transferring context: 1.18kB
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] COPY requirements.txt /app/
=> CACHED [4/6] RUN pip install -r requirements.txt
=> [5/6] COPY *.py /app/
=> [6/6] RUN chmod a+x *.py
=> exporting to image
=> => exporting layers
=> => writing image sha256:df24b669fb941e236d6d67a765eff143c72ddb66d43c8f0538e1b1016de85a17
=> => naming to docker.io/library/pc
[node2] (local) root@192.168.0.7 ~/TiemposListas
```

```
[node2] (local) root@192.168.0.7 ~/TiemposListas
$ docker run -p 5000:5000 pc
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.18.0.1 - - [09/Sep/2023 21:59:59] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [09/Sep/2023 22:00:05] "GET /pizzacontent HTTP/1.1" 200 -
172.18.0.1 - - [09/Sep/2023 22:37:46] "GET /pizzacontent HTTP/1.1" 200 -
172.18.0.1 - - [09/Sep/2023 22:37:46] "GET /pizzacontent HTTP/1.1" 200 -
172.18.0.1 - - [09/Sep/2023 22:37:47] "GET /pizzacontent HTTP/1.1" 200 -
```



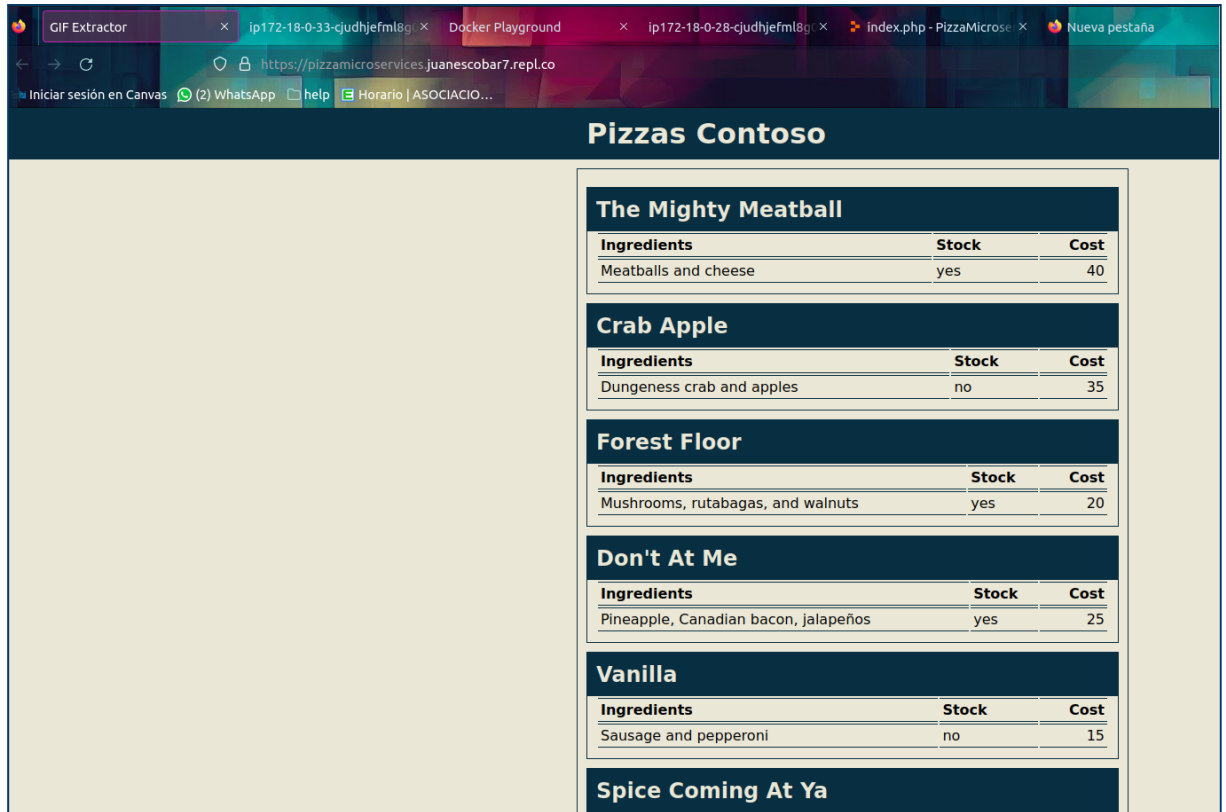
```
JSON  Datos sin procesar  Cabeceras
Guardar Copiar Contraer todo Expandir todo Filtrar JSON
[
  {
    "cost": 40,
    "inStock": "yes",
    "ingredients": "Meatballs and cheese",
    "pizzaName": "The Mighty Meatball"
  },
  {
    "cost": 35,
    "inStock": "no",
    "ingredients": "Dungeness crab and apples",
    "pizzaName": "Crab Apple"
  },
  {
    "cost": 20,
    "inStock": "yes",
    "ingredients": "Mushrooms, rutabagas, and walnuts",
    "pizzaName": "Forest Floor"
  },
  {
    "cost": 25,
    "inStock": "yes",
    "ingredients": "Pineapple, Canadian bacon, jalape\u00f1os",
    "pizzaName": "Don't At Me"
  },
  {
    "cost": 15,
    "inStock": "no",
    "ingredients": "Sausage and pepperoni"
  }
]
```



- Now in our replit we're gonna use the php template.

<https://replit.com/@JuanEscobar7/PizzaMicroservices>

```
php index.php x +
php index.php
1 <!DOCTYPE html>
2 <?php
3     ini_set("allow_url_fopen", 1);
4     $api_endpoint = "http://ip172-18-0-33-cjudhjefml8g0082ouug-5000.direct.labs.play-
with-docker.com/pizzacontent"; //reemplaze el dominio, por ejemplo http://localhost
quedando algo como $api_endpoint = "http://localhost/api/";
5     $json = @file_get_contents($api_endpoint);
6     if($json == false) {
7         $err = "Something is wrong with the data: " . $url;
8     } else {
9         $pizzas = json_decode($json, true);
10    }
11    ?>
12
13 <html>
14 <head>
15     <meta charset="utf-8">
16     <title>GIF Extractor</title>
17     <style media="screen">
18     html {
19         background: #EAE7D6;
20         font-family: sans-serif;
21     }
22     body {
23         margin: 0;
24     }
25     h1 {
26         padding: 10px;
27         margin: 0 auto;
28         color: #EAE7D6;
29         max-width: 600px;
30     }
31     h1 a {
32         text-decoration: none;
```



Pizzas Contoso		
<b>The Mighty Meatball</b>		
Ingredients	Stock	Cost
Meatballs and cheese	yes	40
<b>Crab Apple</b>		
Ingredients	Stock	Cost
Dungeness crab and apples	no	35
<b>Forest Floor</b>		
Ingredients	Stock	Cost
Mushrooms, rutabagas, and walnuts	yes	20
<b>Don't At Me</b>		
Ingredients	Stock	Cost
Pineapple, Canadian bacon, jalapeños	yes	25
<b>Vanilla</b>		
Ingredients	Stock	Cost
Sausage and pepperoni	no	15
<b>Spice Coming At Ya</b>		

### Observaciones y Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.

He utilizado una variedad de tecnologías en mi proyecto, incluyendo Microsoft's microservicio, Docker, C#, Flask, y PHP. Esta diversidad muestra mi habilidad para trabajar con diferentes herramientas y lenguajes.

He implementado un enfoque de microservicios en mi proyecto al crear un backend de pizzas y exponerlo como una API. Esto puede facilitar la escalabilidad y el mantenimiento a medida que mi proyecto crezca.

He logrado integrar diferentes partes de mi proyecto, desde el microservicio en Docker hasta los frontends en C# y PHP. Esta integración demuestra mi capacidad para diseñar un sistema completo y cohesivo.

Mi proyecto muestra una sólida experiencia en el desarrollo web, ya que he trabajado en diferentes capas de una aplicación web, desde la creación de microservicios hasta la implementación de frontends.

El hecho de haber desplegado tu microservicio de pizzas en Docker muestra un enfoque moderno en el desarrollo y despliegue de aplicaciones, lo que puede mejorar la portabilidad y la eficiencia en la gestión de recursos.

Al haber utilizado C#, Flask y PHP en diferentes partes de tu proyecto, has demostrado ser un desarrollador políglota, capaz de trabajar en múltiples lenguajes de programación. Esto es una ventaja en el mundo del desarrollo web, donde la versatilidad es clave.

La transición de un backend de pizzas a una API en Flask muestra una buena estrategia para reutilizar recursos y funcionalidades, lo que puede ahorrar tiempo y esfuerzo en el desarrollo de tu frontend en PHP. Esto también destaca tu capacidad para planificar y optimizar el desarrollo de tu proyecto.