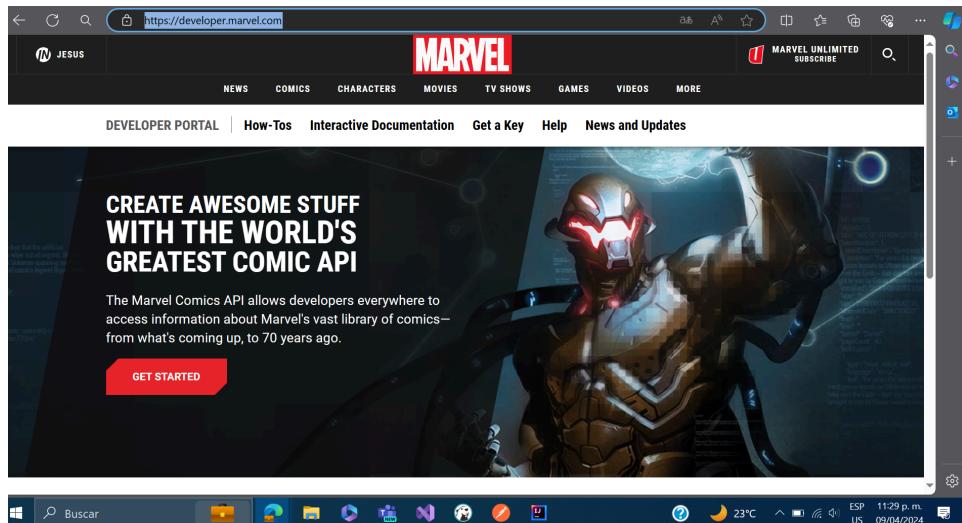


Crear cuenta de developer para marvel

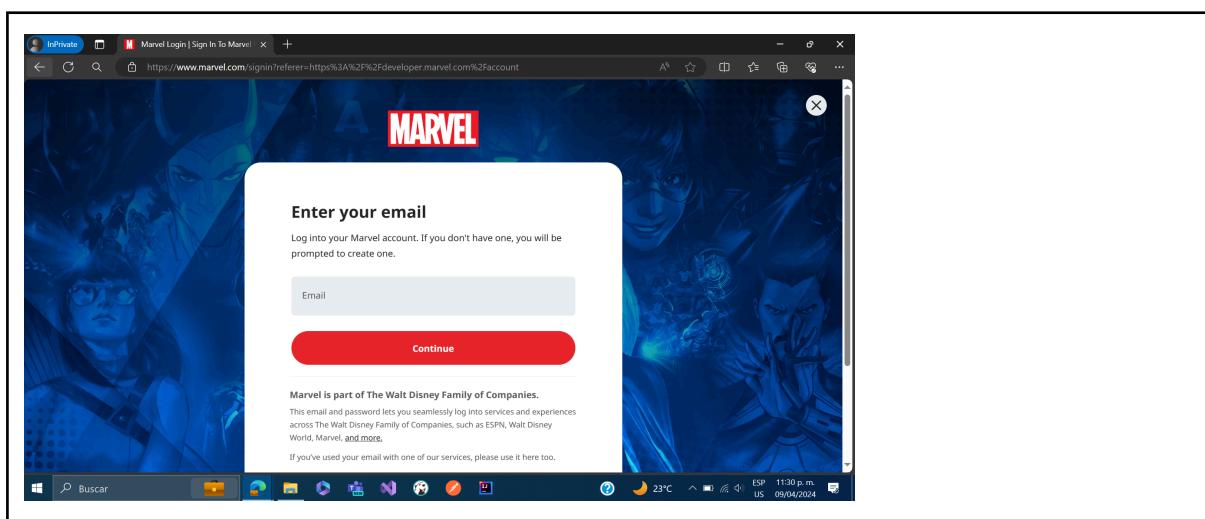
Esto para el consumo del endpoint hacia nuestro backend

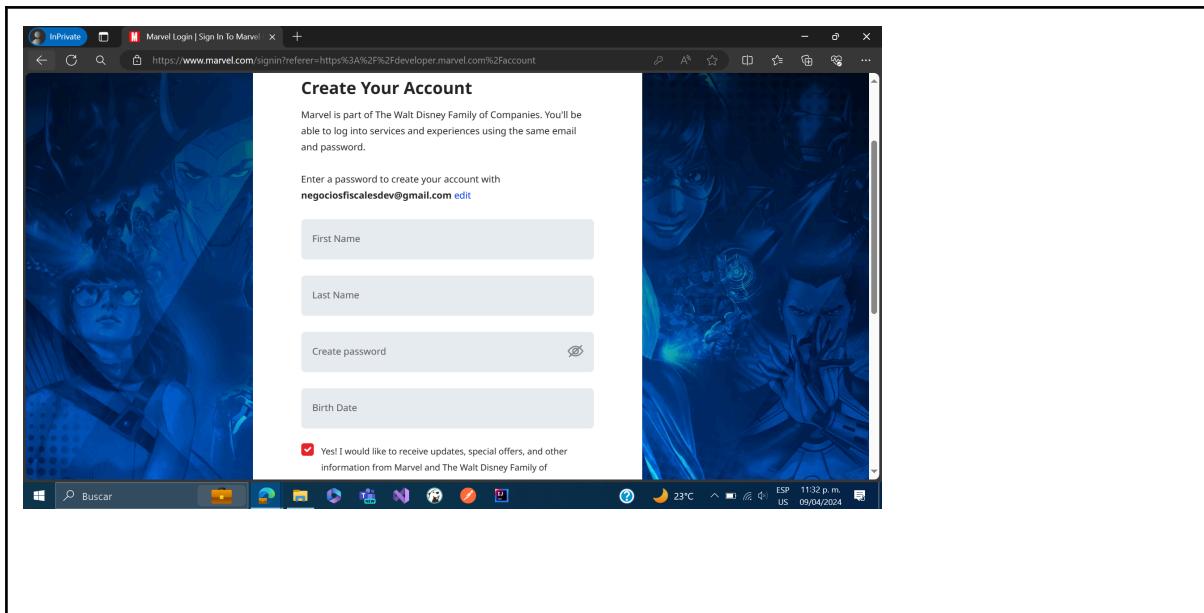
Visita el sitio web de Marvel Developer Portal en <https://developer.marvel.com/>.



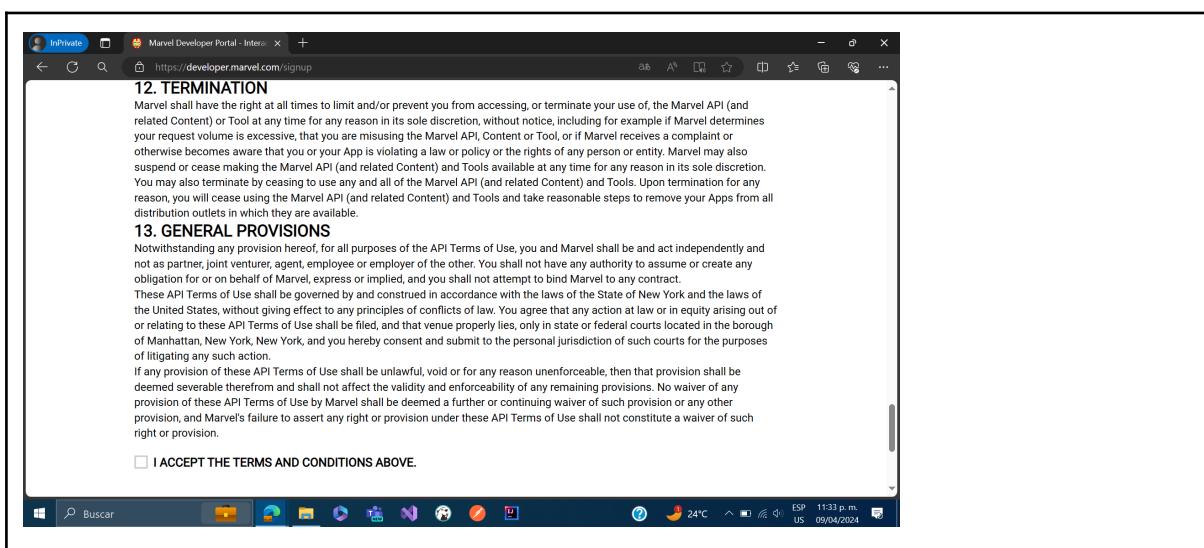
Haz clic en el botón "Get Started" o "Sign Up" (Empezar o Registrarse) que generalmente se encuentra en la esquina superior derecha de la página.

Serás dirigido a una página de registro donde se te pedirá que proporciones información básica como tu nombre, dirección de correo electrónico y contraseña.



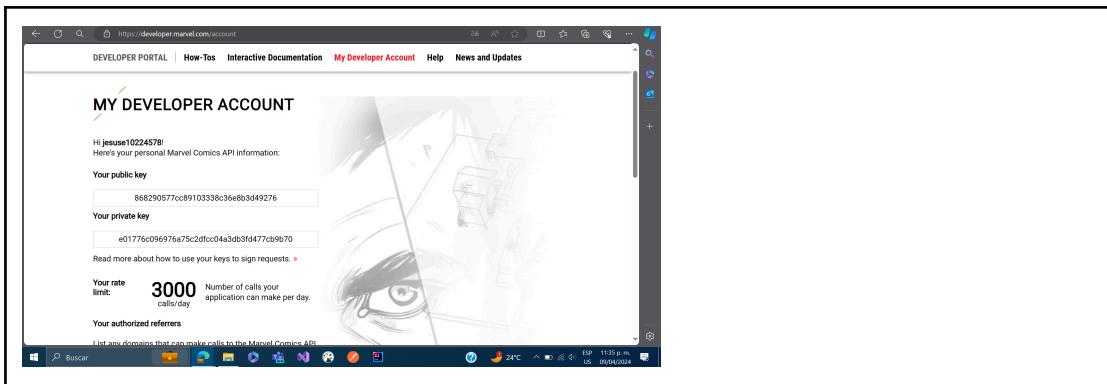


Completa el formulario de registro con la información requerida.
Acepta los términos y condiciones de uso si lo solicitan.



Una vez que hayas completado el registro, es posible que necesites verificar tu dirección de correo electrónico. Revisa tu bandeja de entrada y sigue las instrucciones del correo electrónico de verificación enviado por Marvel Developer Portal.

Después de verificar tu dirección de correo electrónico, deberías poder iniciar sesión en tu cuenta de desarrollador en <https://developer.marvel.com/>. y te aparecerá los detalles de dev



Donde podemos ver nuestras claves que posteriormente necesitaremos en este caso el endpoint que consumimos fue el <https://gateway.marvel.com:443/v1/public/characters> el cual lo puedes encontrar en la ruta documentacion

Parámetros de la solicitud:

- apikey: Tu clave pública de la API proporcionada por Marvel.
- hash: El hash generado para autenticar tu solicitud. Este hash se genera utilizando tu clave privada, tu clave pública y un timestamp (ts).
- ts: Un timestamp que se utiliza en la generación del hash.

URL base: La URL base es la dirección principal a la que estás enviando la solicitud. En este caso, es <https://gateway.marvel.com:443/v1/public/characters>, que es la ruta para obtener información sobre los personajes de Marvel.

Parámetro apikey: Este parámetro es tu clave pública de la API proporcionada por Marvel. Se utiliza para autenticar tu solicitud y permitir el acceso a los datos.

Parámetro hash: Este es un hash generado para autenticar tu solicitud. Se calcula utilizando tu clave privada, tu clave pública y el timestamp (ts). La fórmula para calcular este hash es:

`md5(ts + privateKey + publicKey)`

Donde ts es el timestamp, privateKey es tu clave privada y publicKey es tu clave pública.

Parámetro ts: Este es un timestamp que se utiliza en la generación del hash. Es simplemente un número que representa la hora actual en Unix time (segundos desde el 1 de enero de 1970).

puedes apoyarte con el siguiente sitio para obtener el parámetro hash
[MD5 Hash Generator](#)

donde e1 es igual al ts y los otros datos son combinaciones de la llave pública y privada teniendo estos datos podemos hacer la petición y esta nos debe responder con un código 200

```

1 {
2   "code": 200,
3   "status": "OK",
4   "text": "© 2024 MARVEL",
5   "attributionText": "Data provided by Marvel. © 2024 MARVEL",
6   "attributionHTML": "<a href='http://marvel.com'>Data provided by Marvel. © 2024 MARVEL</a>",
7   "etag": "2d331d7986063411ab3597aa428952487aeff5964",
8   "data": [
9     {
10       "offSet": 0,
11       "limit": 20,
12       "total": 1564,
13       "count": 20,
14       "results": [
15         {
16           "id": 1011334,
17           ...
18         }
19       ]
20     }
21   ]
22 }
```

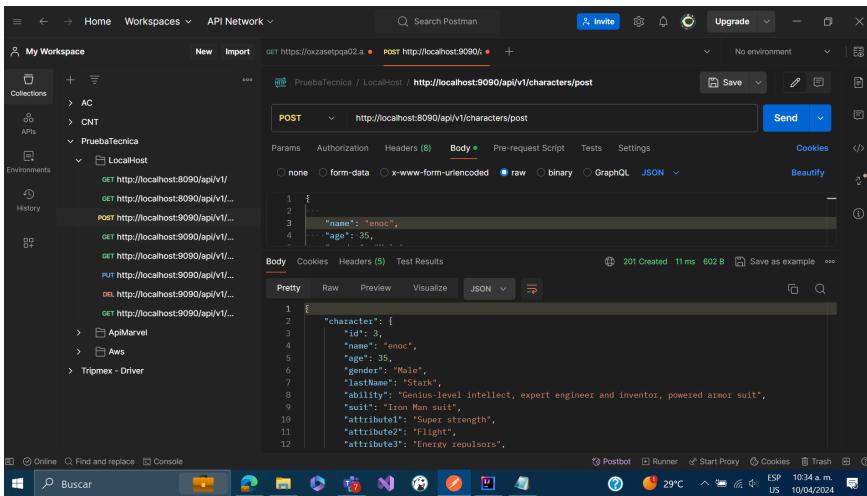
Peticiones Backend

<http://localhost:8090/api/v1/characters/post>

Este método define un endpoint POST /post que inserta superhéroes en BD memoria

Operaciones y Respuestas de la API: Las anotaciones `@Operation` y `@ApiResponses` proporcionan información sobre la operación del endpoint y las posibles respuestas.

- Creación de Personaje: El método `createCharacter` intenta crear un personaje utilizando el objeto `characterService`.
- Si tiene éxito, devuelve un mensaje de éxito junto con el personaje creado en un `ResponseEntity` con estado 200 (OK). Si falla, devuelve un mensaje de error junto con un `CustomError` en un `ResponseEntity` con estado 500 (Error interno del servidor).



The screenshot shows the Postman interface with a POST request to `http://localhost:8090/api/v1/characters/post`. The request body is a JSON object:

```
{ "name": "enoc", "age": 35, "gender": "Male", "lastName": "Stark", "ability": "Genius-level intellect, expert engineer and inventor, powered armor suit", "suit": "Iron Man suit", "attribute1": "Super strength", "attribute2": "Flight", "attribute3": "Energy repulsors", "attribute4": "Missile launchers", "attribute5": "Regenerative life support", "attribute6": "Holographic heads-up display", "message": "Character created successfully" }
```

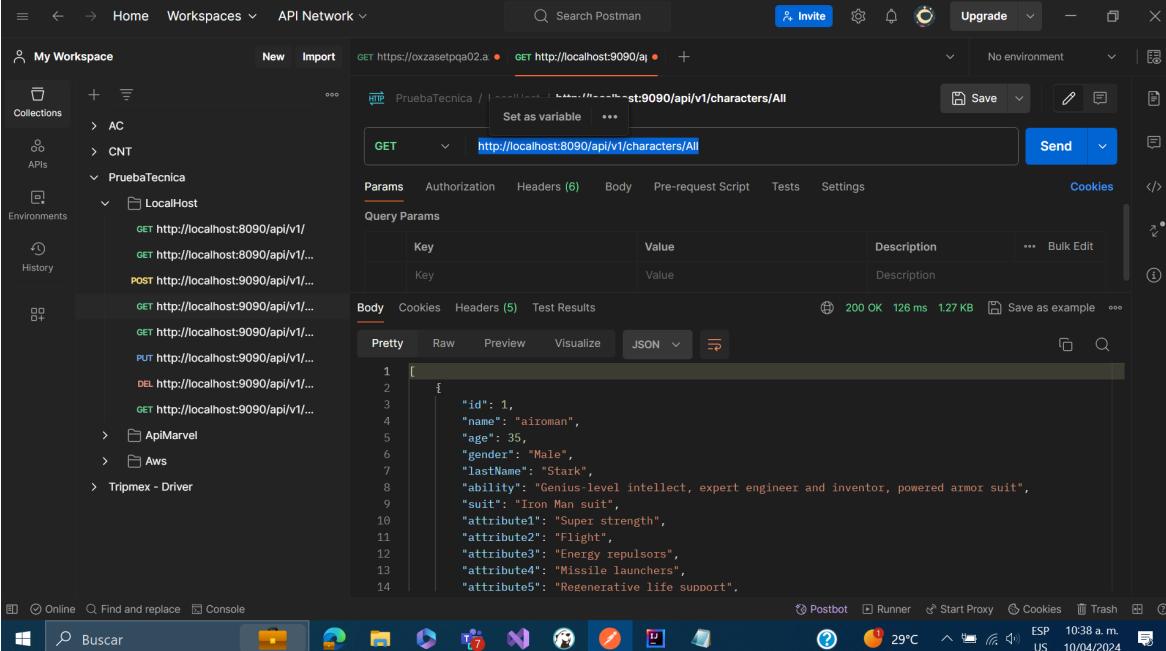
Body:

```
{  
  
    "name": "enoc",  
    "age": 35,  
    "gender": "Male",  
    "lastName": "Stark",  
    "ability": "Genius-level intellect, expert engineer and inventor, powered  
armor suit",  
    "suit": "Iron Man suit",  
    "attribute1": "Super strength",  
    "attribute2": "Flight",  
    "attribute3": "Energy repulsors",  
    "attribute4": "Missile launchers",  
    "attribute5": "Regenerative life support",  
    "attribute6": "Holographic heads-up display",  
    "message": "Character created successfully"  
}
```

<http://localhost:8090/api/v1/characters/All>

Este método define un endpoint GET /All que devuelve todos los superhéroes de la base de datos

- Operaciones y Respuestas de la API: Las anotaciones @Operation y @ApiResponses proporcionan información sobre la operación del endpoint y las posibles respuestas.
- Obtención de Todos los Personajes: El método getAllCharacters obtiene todos los personajes utilizando el servicio characterService. Si la lista de personajes está vacía, devuelve un mensaje indicando que no se encontraron personajes en un ResponseEntity con estado 404 (Not Found). Si la lista contiene personajes, los devuelve en un ResponseEntity con estado 200 (OK).



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'AC', 'CNT', and 'PruebaTecnica'. Under 'PruebaTecnica', there's a 'LocalHost' folder with several requests listed. The main area shows a single request: a GET request to 'http://localhost:8090/api/v1/characters/All'. The 'Body' tab of the request details panel is selected, displaying the following JSON response:

```
1 [  
2 {  
3     "id": 1,  
4     "name": "airman",  
5     "age": 35,  
6     "gender": "Male",  
7     "lastName": "Stark",  
8     "ability": "Genius-level intellect, expert engineer and inventor, powered armor suit",  
9     "suit": "Iron Man suit",  
10    "attribute1": "Super strength",  
11    "attribute2": "Flight",  
12    "attribute3": "Energy repulsors",  
13    "attribute4": "Missile launchers",  
14    "attribute5": "Regenerative life support".  
]
```

<http://localhost:8090/api/v1/characters/get/1>

Este método define un endpoint GET /get/{id} que permite obtener un superhéroe por su ID en la base de datos.

Operaciones y Respuestas de la API: Las anotaciones @Operation y @ApiResponses proporcionan información sobre la operación del endpoint y las posibles respuestas.
Obtención de un Personaje por ID: El método getCharacterById obtiene un personaje utilizando su ID mediante el servicio characterService. Si el personaje con el ID proporcionado existe, devuelve el personaje en un ResponseEntity con estado 200 (OK). Si no se encuentra ningún personaje con el ID proporcionado, devuelve un ResponseEntity con estado 404 (Not Found).

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'AC', 'CNT', and 'PruebaTecnica'. Under 'PruebaTecnica', there's a 'LocalHost' folder with several API endpoints listed. The main panel shows a request for 'GET https://oxzasetpqa02.a...'. The 'Params' tab is selected, showing a single parameter 'id' with a value of '1'. The 'Body' tab shows the JSON response:

```

1
2   "id": 1,
3   "name": "airoman",
4   "age": 35,
5   "gender": "Male",
6   "lastName": "Stark",
7   "ability": "Genius-level intellect, expert engineer and inventor, powered armor suit",
8   "suit": "Iron Man suit",
9   "attribute1": "Super strength",
10  "attribute2": "Flight",
11  "attribute3": "Energy repulsors",
12  "attribute4": "Missile launchers",
13  "attribute5": "Regenerative life support",
14  "attribute6": "Holographic heads-up display"

```

The status bar at the bottom indicates a 200 OK response with 41 ms and 543 B.

<http://localhost:8090/api/v1/characters/getCharacterById?id=1>

Este método define un endpoint GET /getCharacterById que permite obtener datos específicos de un superhéroe por su ID.

Operaciones y Respuestas de la API: Las anotaciones `@Operation` y `@ApiResponses` proporcionan información sobre la operación del endpoint y las posibles respuestas.

- Obtención de Datos Específicos de un Personaje: El método `getCharacterMarvelById` obtiene datos específicos de un personaje utilizando su ID mediante el servicio `characterService`. Si el personaje con el ID proporcionado existe, devuelve los datos en un `ResponseEntity` con estado 200 (OK). Si no se encuentra ningún personaje con el ID proporcionado, devuelve un `ResponseEntity` con estado 404 (Not Found).

The screenshot shows the Postman application interface, similar to the previous one but with a failed request. The 'Params' tab shows a parameter 'id' with a value of '1'. The 'Body' tab shows the JSON response:

```

1
2   "name": null,
3   "suit": null

```

The status bar at the bottom indicates a 200 OK response with 66 ms and 189 B.

<http://localhost:8090/api/v1/characters/put/1>

Este método define un endpoint PUT /put/{id} que permite actualizar un superhéroe por su ID.

- Operaciones y Respuestas de la API: Las anotaciones @Operation y @ApiResponses proporcionan información sobre la operación del endpoint y las posibles respuestas.
- Actualización de un Personaje: El método updateCharacter intenta actualizar un superhéroe utilizando su ID y el objeto updatedCharacter. Si la actualización es exitosa, devuelve un ResponseEntity con estado 200 (OK) y un mensaje indicando que el personaje se actualizó correctamente. Si no se encuentra ningún personaje con el ID proporcionado, devuelve un ResponseEntity con estado 404 (Not Found). Si ocurre un error interno, devuelve un ResponseEntity con estado 500 (Error interno del servidor) y un mensaje indicando el error.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing environments like 'PruebaTecnica' and 'LocalHost'. The main area shows a 'PUT' request to 'http://localhost:8090/api/v1/characters/put/1'. The 'Body' tab is selected, showing a JSON response:

```
1 {  
2     "id": 1,  
3     "message": "Character updated successfully"  
4 }
```

<http://localhost:8090/api/v1/characters/delete/1>

Este método define un endpoint DELETE /delete/{id} que permite eliminar un superhéroe por su ID.

- Operaciones y Respuestas de la API: Las anotaciones @Operation y @ApiResponses proporcionan información sobre la operación del endpoint y las posibles respuestas.
- Eliminación de un Personaje: El método deleteCharacter intenta eliminar un superhéroe utilizando su ID. Si la eliminación es exitosa, devuelve un ResponseEntity con estado 200 (OK) y un mensaje indicando que el personaje se eliminó correctamente. Si ocurre un error al intentar eliminar el personaje, devuelve un ResponseEntity con estado 500 (Error interno del servidor) y un mensaje indicando el error.

My Workspace

HTTP PruebaTecnica / Local Host - http://localhost:8090/api/v1/characters/delete/1

DELETE http://localhost:8090/api/v1/characters/delete/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Character with id 1 deleted successfully"
3 }

```

200 OK 24 ms 218 B Save as example

Postbot Runner Start Proxy Cookies Trash

Online Find and replace Console Buscar

Postman 10:51 a.m. 10/04/2024

<http://localhost:8090/api/v1/marvel/characters>

Este método obtiene contenido de Marvel a través del servicio marvelApiService. Devuelve un ResponseEntity con estado 200 (OK) si la solicitud se procesa correctamente, y un ResponseEntity con estado 500 (Error interno del servidor) si hay algún problema durante el procesamiento de la solicitud.

My Workspace

HTTP PruebaTecnica / Local Host - http://localhost:8090/api/v1/marvel/characters

GET http://localhost:8090/api/v1/marvel/characters

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "code": 200,
3   "status": "Ok",
4   "copyright": "\u00a9 2024 MARVEL",
5   "attributionText": "Data provided by Marvel. \u00a9 2024 MARVEL",
6   "attributionHTML": "<a href=\"http://marvel.com\">Data provided by Marvel. \u00a9 2024 MARVEL</a>",
7   "etag": "2d33fd798b663411ab3507aa428952487ae5f5964",
8   "data": [
9     {
10       "offset": 0,
11       "limit": 20,
12       "total": 1564,
13       "count": 20,
14       "results": [
15         {}
16       ]
17     }
18   ]
19 }

```

200 OK 4.73 s 97.58 KB Save as example

Postbot Runner Start Proxy Cookies Trash

Online Find and replace Console Buscar

Postman 10:53 a.m. 10/04/2024

<http://localhost:8090/api/v1/hello>

Este método devuelve un mensaje de bienvenida generado por el servicio helloService. Devuelve un Map con una clave "message" que contiene el mensaje de bienvenida. Si la solicitud se procesa correctamente, devuelve un ResponseEntity con estado 200 (OK). Si

hay algún problema durante el procesamiento de la solicitud, devuelve un `ResponseType` con estado 500 (Error interno del servidor).

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'AC', 'CNT', and 'PruebaTecnica'. Under 'PruebaTecnica', there's a 'LocalHost' folder with several API endpoints listed. One endpoint, 'GET http://localhost:8090/api/v1/hello', is selected and highlighted in orange. The main workspace shows a request configuration for this endpoint: Method 'GET', URL 'http://localhost:8090/api/v1/hello', Headers (6), Body, Pre-request Script, Tests, and Settings. Below the request, the response body is displayed in JSON format:

```
1  "message": "Hi, this is my tech test"
```

At the bottom of the screen, there's a taskbar with various icons and system status information.

Swagger UI

```
/**  
 * Endpoint para acceder a la interfaz de usuario de Swagger.  
 * Esta ruta proporciona documentación interactiva para la API.  
 * Puedes acceder a ella en http://localhost:8090/api/v1/swagger-ui/index.html  
 */
```

The screenshot shows the Swagger UI interface. At the top, it says 'localhost:8090/api/v1 - Generated server url'. Below that, there are two sections: 'Endpoint characters' and 'Endpoint Api-Marvel'.

Endpoint characters

- PUT** /characters/put/{id} Endpoint Updating Superheroes
- POST** /characters/post Endpoint inserts superheroes
- GET** /characters/getCharacterById Endpoint that yields specific data
- GET** /characters/get/{id} Endpoint Throwing Superheroes By id
- GET** /characters/All Endpoint that brings all the superheroes ...
- DELETE** /characters/delete/{id} Endpoint That Eliminates Superheroes

Endpoint Api-Marvel

- GET** /marvel/characters Endpoint que regresa contenido de marvel

Endpoint Hello

At the bottom of the screen, there's a taskbar with various icons and system status information.

Diagramas de clases Api

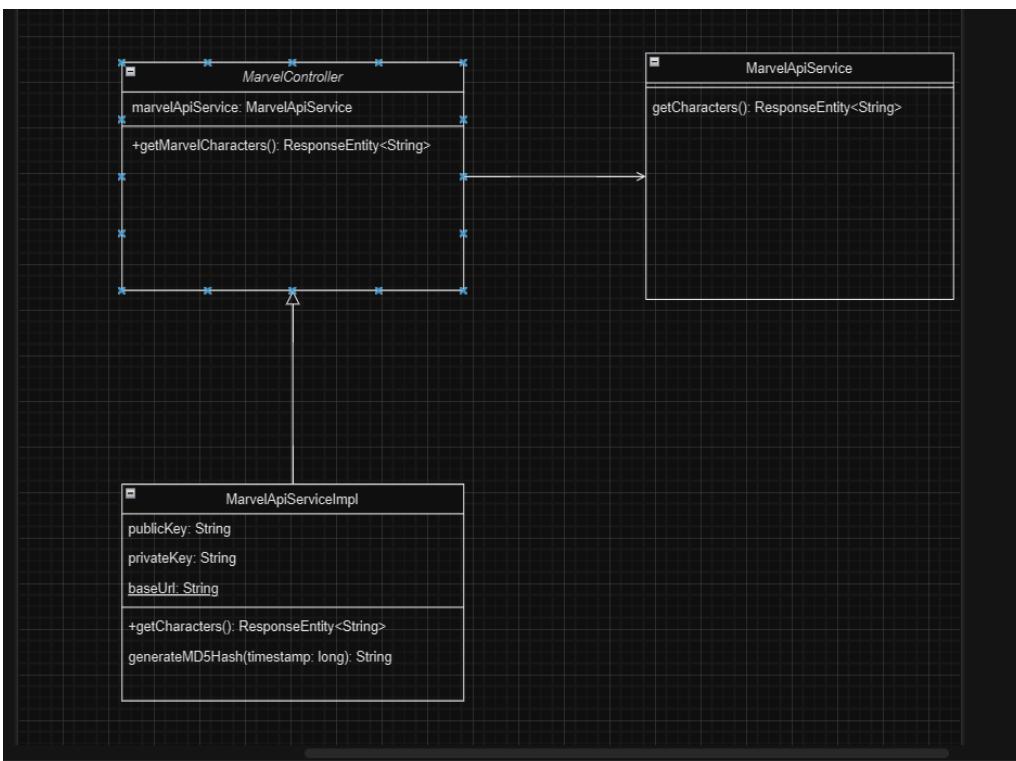


Diagrama de clase consumo de endpoint marvel

Este diagrama de clases representa la estructura básica y las relaciones entre las clases `MarvelController`, `Marvel ApiService`, y `Marvel ApiService Impl`. En el diagrama, los métodos públicos están precedidos por un signo más (+), los atributos privados por un guion (-), y las dependencias entre las clases se muestran con flechas.

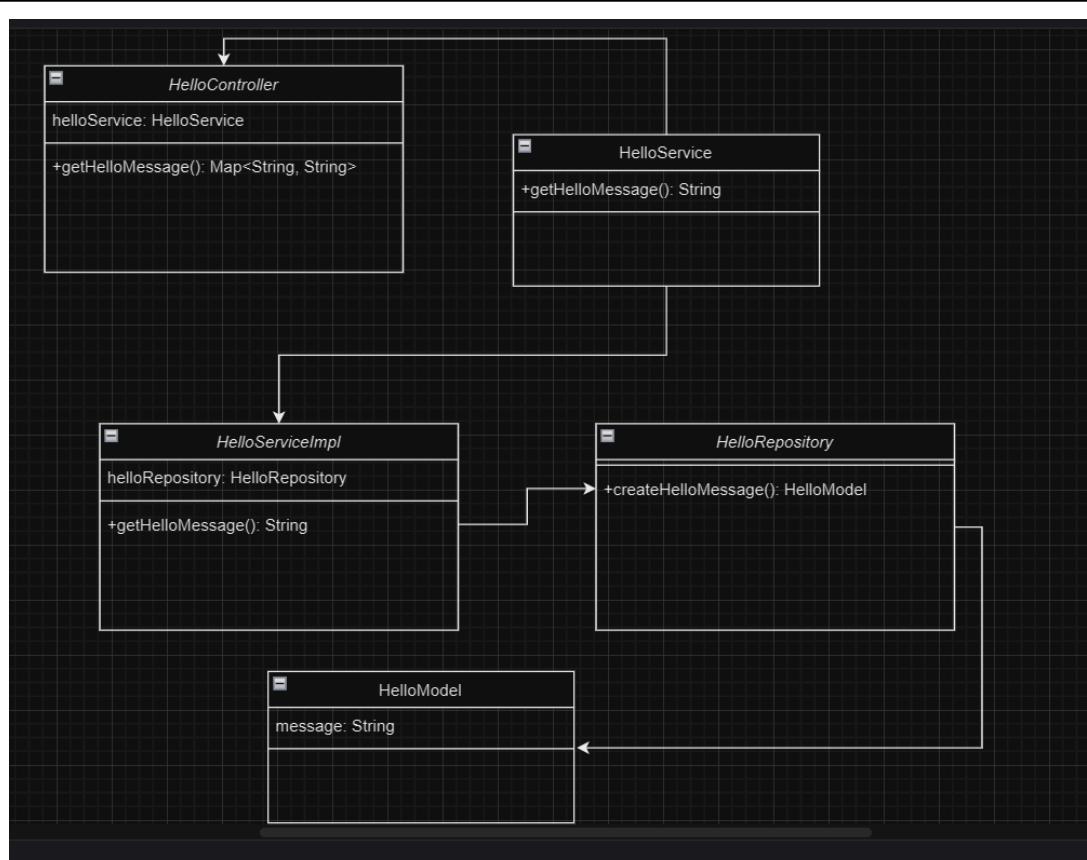


Diagrama de clase endpoint hello

Este diagrama de clases UML muestra la estructura y las relaciones entre las clases en tu aplicación. Las flechas indican las dependencias entre las clases y los métodos públicos están representados en el diagrama

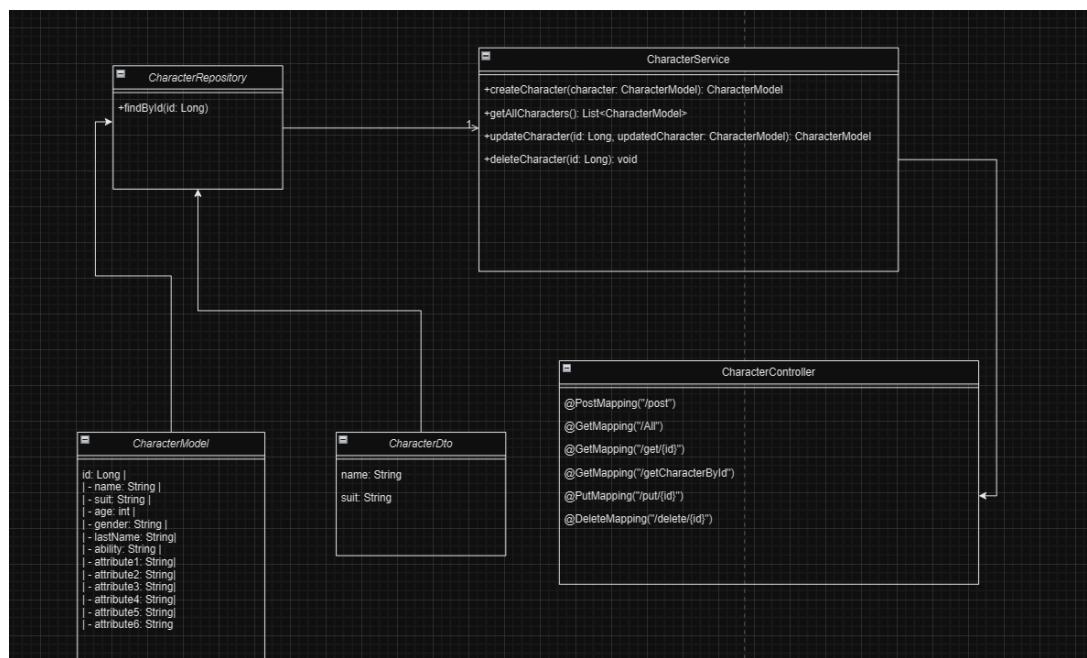


Diagrama de clase endpoint Character

En este diagrama de clases UML:

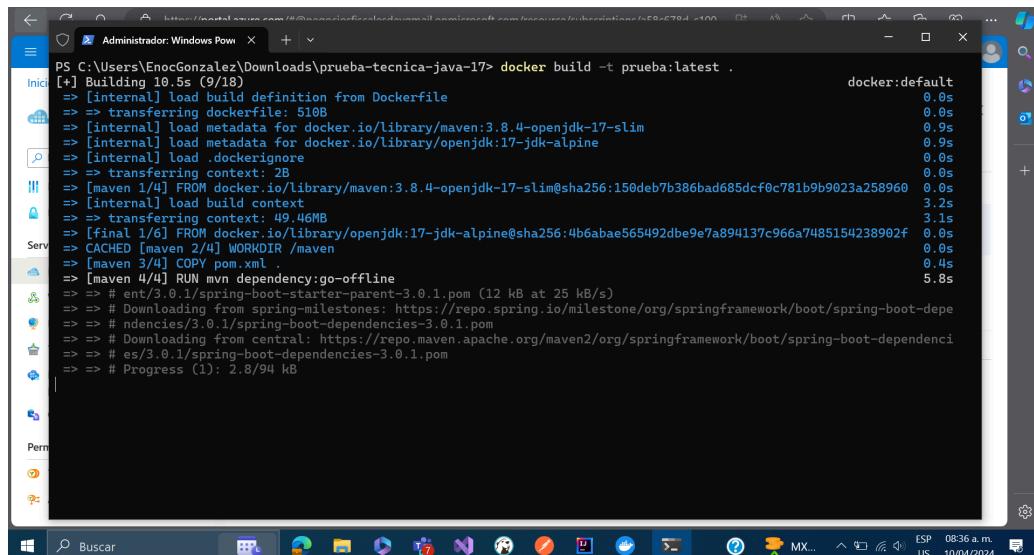
- CharacterRepository y CharacterService son clases que representan la capa de acceso a datos y el servicio de gestión de personajes, respectivamente.
- CharacterModel es una entidad JPA que representa la estructura de datos de un personaje en la base de datos.
- CharacterDto es un DTO que representa una proyección de datos específicos de un personaje.
- Las flechas indican la dependencia entre las clases. Por ejemplo, CharacterService depende de CharacterRepository para acceder a los datos de los personajes.

Despliegue en Nube azure

preparación del contenedor:

Asegúrate de que tu aplicación está empaquetada en un contenedor Docker. Esto implica realizar los siguientes pasos:

- docker build -t prueba:latest . construye la imagen docker debes estar en la raíz



```
PS C:\Users\EnocGonzalez\Downloads\prueba-tecnica-java-17> docker build -t prueba:latest .
[+] Building 10.5s (9/18)
--> [internal] load build definition from Dockerfile          docker:default
--> [internal] load metadata from docker.io/library/maven:3.8.4-openjdk-17-slim
--> [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine
--> [internal] load .dockerignore
--> [internal] load context: 2B
--> [maven 1/4] FROM docker.io/library/maven:3.8.4-openjdk-17-slim@sha256:150deb7b386bad685dcf0c781b9b9023a258960 0.0s
--> [internal] load build context                           0.0s
--> [internal] transfer context: 49.46MB                   3.2s
--> [final 1/6] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f 0.0s
--> [internal] CACHED [maven 2/4] WORKDIR /maven           0.0s
--> [maven 3/4] COPY pom.xml                            0.4s
--> [maven 4/4] RUN mvn dependency:go-offline            5.8s
--> # ent/3.0.1/spring-boot-starter-parent-3.0.1.pom (12 kB at 25 kB/s)
--> # Downloading from spring-milestones: https://repo.spring.io/milestone/org/springframework/boot/spring-boot-dep
--> # ndencies/3.0.1/spring-boot-dependencies-3.0.1.pom
--> # Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependenci
--> # es/3.0.1/spring-boot-dependencies-3.0.1.pom
--> # Progress (1): 2.8/94 kB
```

Verificamos que nuestra imagen se haya construido correctamente con el siguiente comando.

```
docker images
```

```
Administrator: Windows Pow x + ▾
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

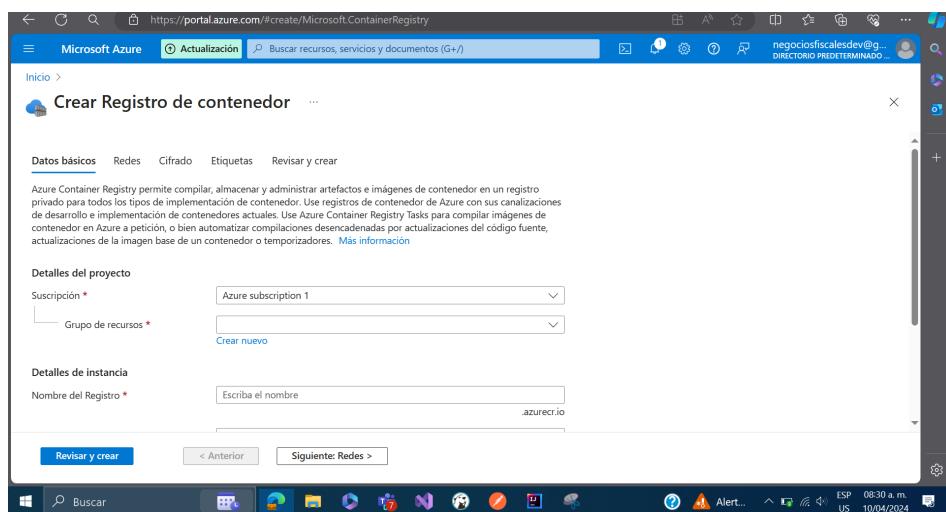
PS C:\Users\Instalaciones> docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
prueba              latest   c1cda8fe9fcc  3 hours ago  424MB
pruebadev.azurecr.io/prueba    latest   c1cda8fe9fcc  3 hours ago  424MB
oxzkkvetpd01.azurecr.io/nueva_autentificacion  latest   9d05908af0a3  11 days ago  595MB
PS C:\Users\Instalaciones> |
```

Acceso a Azure Portal:

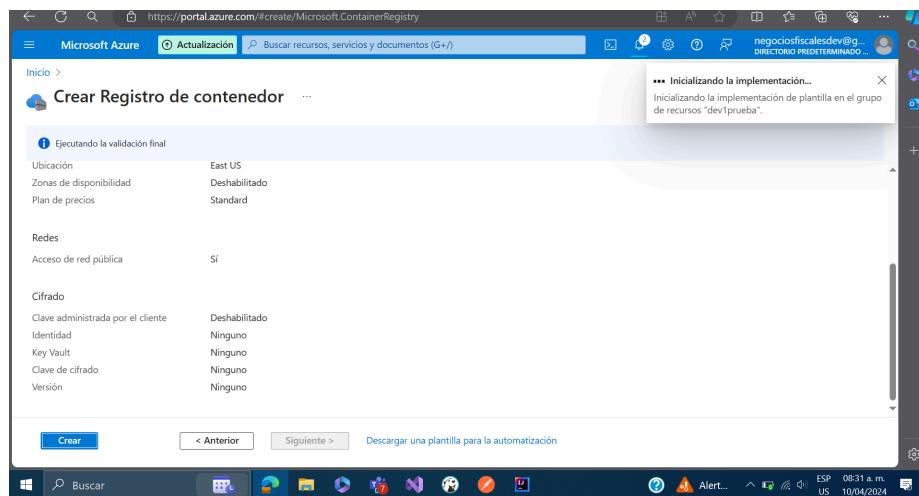
Inicia sesión en el [Portal de Azure](#).

Crea un registro de contenedores:

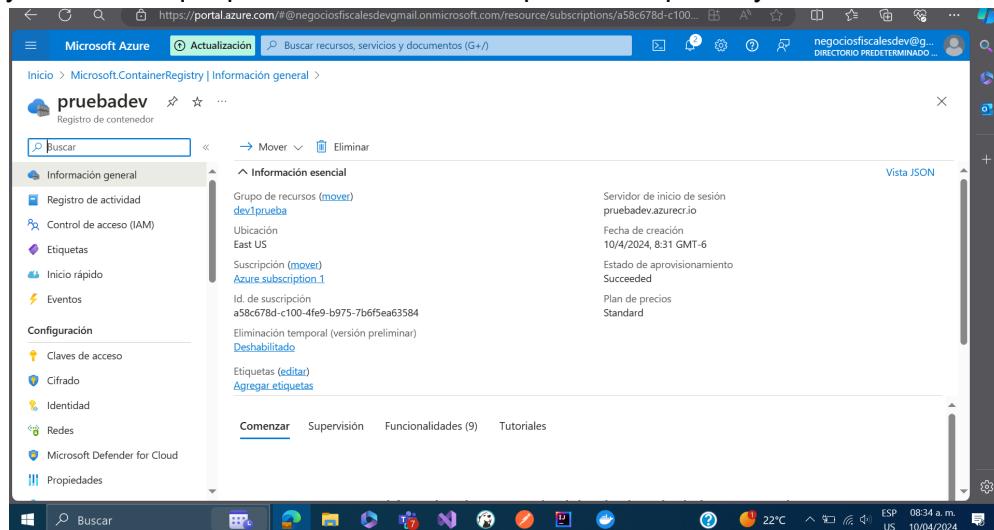
Ve al servicio Azure Container Registry en el portal de Azure y crea un nuevo registro de contenedores



Definimos nuestro grupo de recursos y nombre de registro



Verificamos que nuestro container se haya registrado adecuadamente y revisamos que podamos acceder a la parte de repository



no redirigimos a nuestra consola y ejecutamos el siguiente comando
docker tag prueba pruebadev.azurecr.io/prueba donde 'pruebadev' es el nombre de recursos de nuestro container verificamos que se haya ejecutado corrractamente con el siguiente comando

```
Administrator: Windows Pow x + ▾
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Instalaciones> docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
prueba              latest   c1cda8fe9fcc  3 hours ago  424MB
pruebadev.azurecr.io/prueba    latest   c1cda8fe9fcc  3 hours ago  424MB
oxzkvetpd01.azurecr.io/nueva_autentificacion  latest   9d05908af0a3  11 days ago  595MB
PS C:\Users\Instalaciones> |
```

Luego ejecutamos az login esto nos pedirá en el navegador que nos conectemos con una cuenta asociada

```
Administrator: Windows Pow x + ▾
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Instalaciones> docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
prueba              latest   c1cda8fe9fcc  3 hours ago  424MB
pruebadev.azurecr.io/prueba    latest   c1cda8fe9fcc  3 hours ago  424MB
oxzkvetpd01.azurecr.io/nueva_autentificacion  latest   9d05908af0a3  11 days ago  595MB
PS C:\Users\Instalaciones> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue
the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow w
ith 'az login --use-device-code'.
```

Una vez no logueamos ejecutamos el siguiente comando
az acr login -n pruebadev donde pruebadev es el nombre de los recursos

```

Administrator: Windows Pow x + v
Prueba la nueva tecnologia PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Instalaciones> docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
prueba              latest   c1cda8fe9fcc  3 hours ago  424MB
pruebadev.azurecr.io/prueba    latest   c1cda8fe9fcc  3 hours ago  424MB
oxzkvetpd01.azurecr.io/nueva_autentificacion latest   9d05908af0a3  11 days ago  595MB
PS C:\Users\Instalaciones> az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue
the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow w
ith 'az login --use-device-code'.
[{"cloudName": "AzureCloud",
"homeTenantId": "e1b07ab2-0492-479a-b295-f0b983525a55",
"id": "a58c678d-c100-4fe9-b975-7b6f5ea63584",
"isDefault": true,
"managedByTenants": [],
"name": "Azure subscription 1",
"state": "Enabled",
"tenantId": "e1b07ab2-0492-479a-b295-f0b983525a55",
"user": {
  "name": "negociosfiscalesdev@gmail.com",
  "type": "user"
}}
]
PS C:\Users\Instalaciones> az acr login -n pruebadev
Login Succeeded
PS C:\Users\Instalaciones>

```

con estos pasos nos hemos conectado azure, queda ejecutar el siguiente comando para subir nuestra imagen al container register

docker push pruebadev.azurecr.io/prueba donde es el tag que hemos creado previamente

```

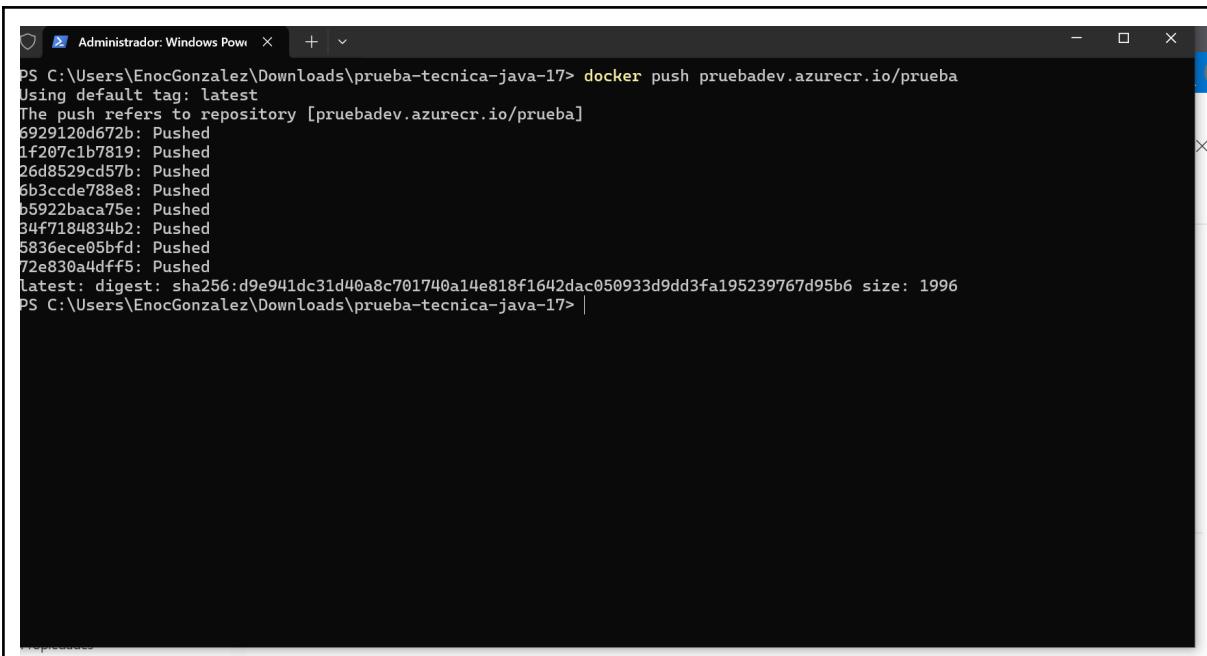
Administrator: Windows Pow x + v
PS C:\Users\EnocGonzalez\Downloads\prueba-tecnica-java-17> docker push pruebadev.azurecr.io/prueba
Using default tag: latest
The push refers to repository [pruebadev.azurecr.io/prueba]
6929120d672b: Pushing [=====] 10.45MB/49.25MB
1f207c1b7819: Pushed
26d8529cd57b: Pushed
6b3ccde788e8: Pushing [=====] 15.17MB/49.25MB
b5922baca75e: Pushed
34f7184834b2: Pushing [>] 2.738MB/317.6MB
5836ece05bfd: Pushing [=====] 2.65MB
72e830a4dff5: Waiting

Ex
@

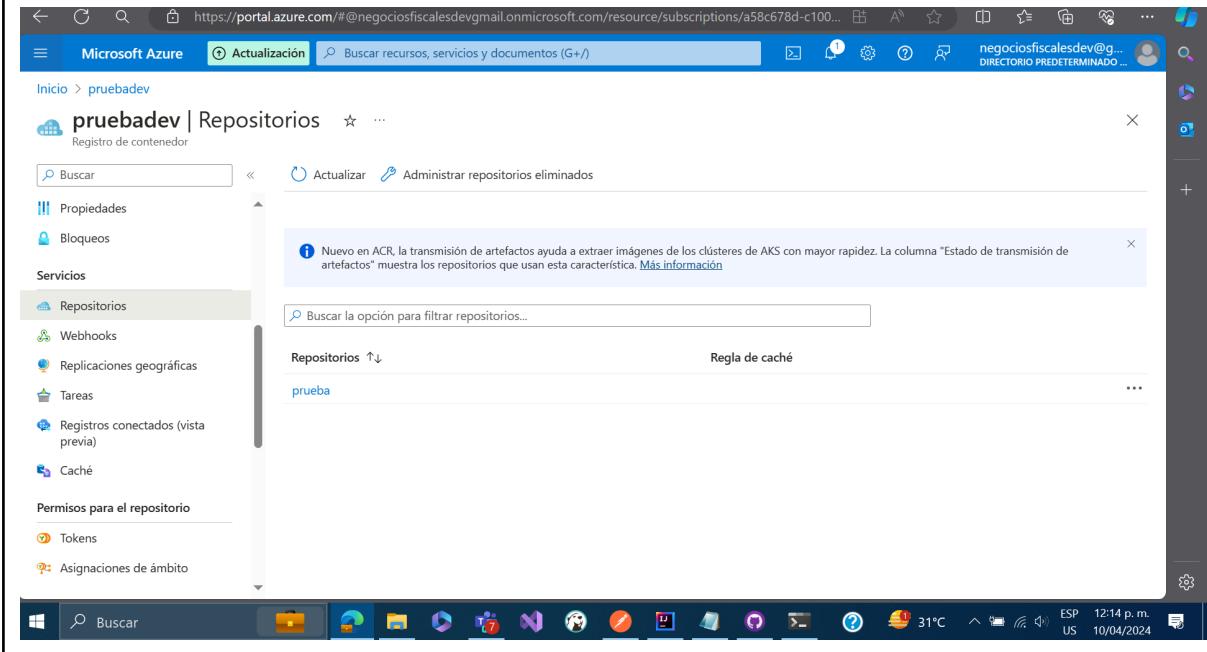
Engine running v4.28.0

```

una vez creada nos lanzara lo siguiente en consola y podemos de igual manera verificar en azure container repositoroy



```
Administrator: Windows Pow + x
PS C:\Users\EnocGonzalez\Downloads\prueba-tecnica-java-17> docker push pruebadev.azurecr.io/prueba
Using default tag: latest
The push refers to repository [pruebadev.azurecr.io/prueba]
6929120d672b: Pushed
1f207c1b7819: Pushed
26d8529cd57b: Pushed
6b3ccde788e8: Pushed
b5922baca75e: Pushed
34f7184834b2: Pushed
5836ece05bfd: Pushed
72e830a4dff5: Pushed
latest: digest: sha256:d9e941dc31d40a8c701740a14e818f1642dac050933d9dd3fa195239767d95b6 size: 1996
PS C:\Users\EnocGonzalez\Downloads\prueba-tecnica-java-17> |
```



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes links for 'Inicio' and 'pruebadev'. The main content area is titled 'pruebadev | Repositorios' and shows a list of repositories. A message box is displayed stating: 'Nuevo en ACR, la transmisión de artefactos ayuda a extraer imágenes de los clústeres de AKS con mayor rapidez. La columna "Estado de transmisión de artefactos" muestra los repositorios que usan esta característica.' Below this, there is a search bar and a table with one row labeled 'prueba'. On the left sidebar, under 'Servicios', 'Repositorios' is selected. Other options include 'Webhooks', 'Replicaciones geográficas', 'Tareas', 'Registros conectados (vista previa)', and 'Caché'. At the bottom, there is a taskbar with various icons and system status information.

Lo siguiente es crear una nueva app services en el aparto de azure

Crear aplicación web

Detalles de instancia

Nombre * pruebadev17 .azurewebsites.net

Publicar * Container

Sistema operativo * Linux

Región * East US

Planes de precios

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Revisar y crear < Anterior Siguiente: Base de datos >

le damos en el apartado de container y linux

Crear aplicación web

Detalles de instancia

Nombre * pruebadev17 .azurewebsites.net

Publicar * Container

Sistema operativo * Linux

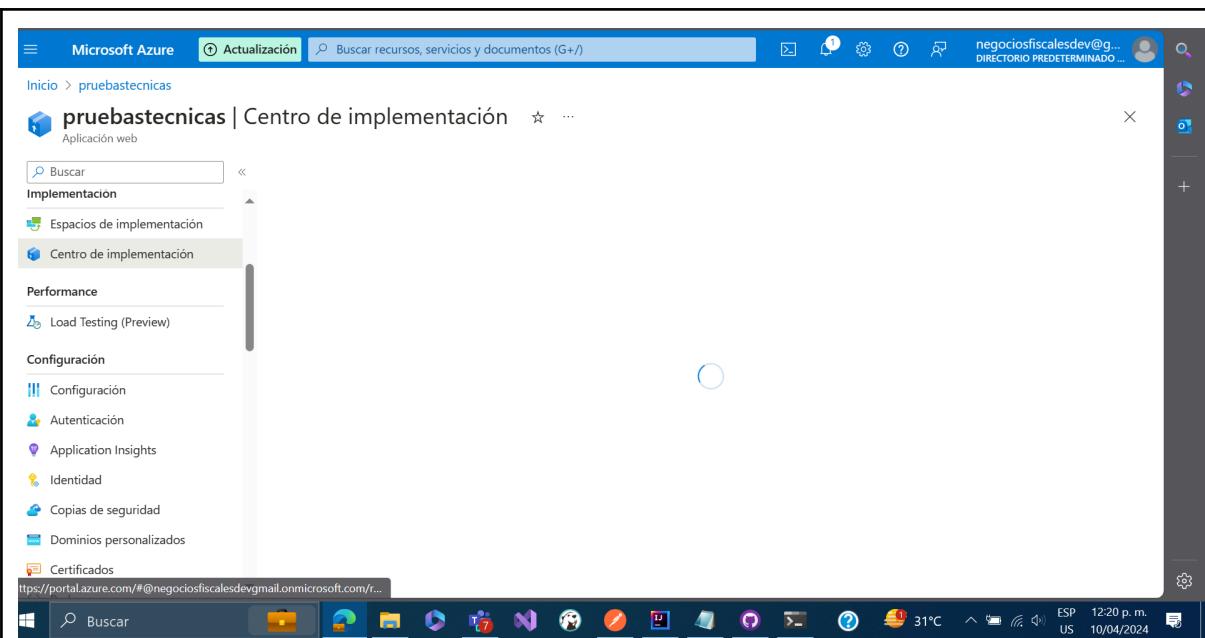
Región * East US

Planes de precios

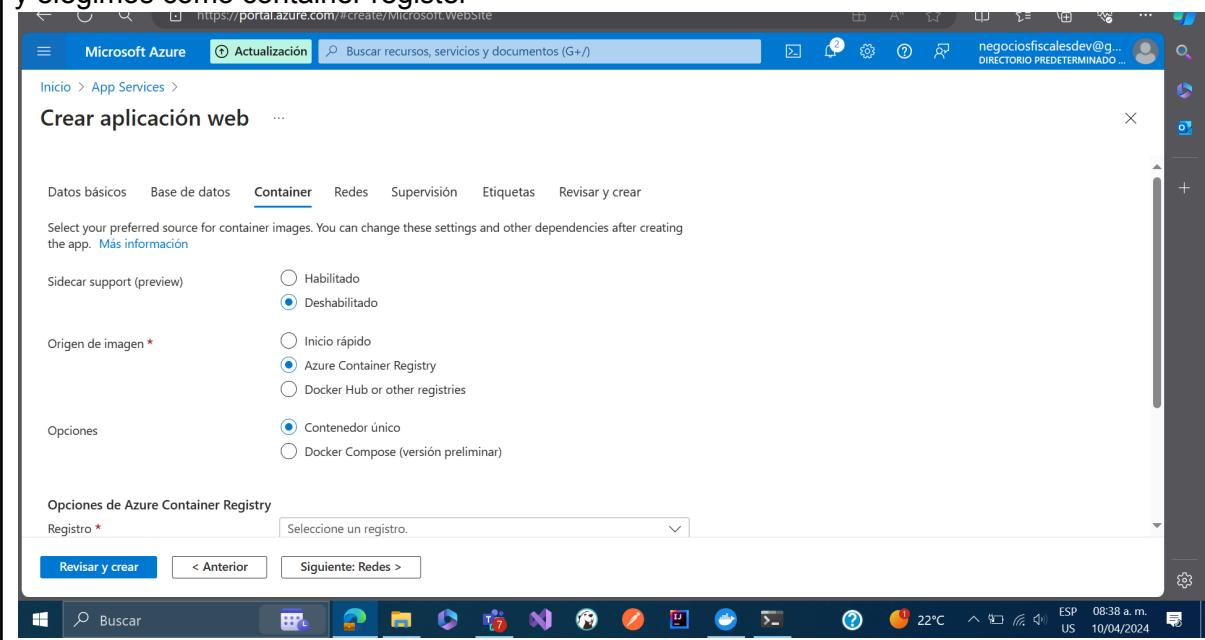
El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Revisar y crear < Anterior Siguiente: Base de datos >

elegimos la región de preferencia dejarla en US y por ultimo le damos en revisar y crear
posteriormente abrimos el app services y nos dirigimos a centros de implementación



y elegimos como container register



escogemos nuestro grupo de recurso y sigamos a nuestro repositorio previamente asignado le damos en revisar y crear.

posteriormente nos dirigimos a introducción para conocer nuestro dominio y hacer peticiones en este caso el dominio es pruebastecnicas.azurewebsites.net

The screenshot shows the Microsoft Azure portal interface. The URL in the address bar is <https://portal.azure.com/#@negociosfiscalesdev@gmail.onmicrosoft.com/resource/subscriptions/a58c678...>. The page title is "Microsoft Azure". The main content area shows the configuration of a web application named "pruebastecnicas". The left sidebar has sections for "Introducción", "Registro de actividad", "Control de acceso (IAM)", "Etiquetas", "Diagnosticar y solucionar problemas", "Microsoft Defender for Cloud", "Eventos (versión preliminar)", "Implementación", "Espacios de implementación", "Centro de implementación", "Performance", and "Load Testing (Preview)". The right pane is titled "Essentials" and shows the following details:

Configuración	Detalles
Grupo de recursos (mover)	dev1prueba
Estado	En ejecución
Ubicación (mover)	East US
Suscripción (mover)	Azure subscription 1
Id. de suscripción	a58c678d-c100-4fe9-b975-7b6f5ea63584
Etiquetas (editar)	Agregar etiquetas
Propiedades	Supervisión Registras Funcionalidades Notificaciones Recomendaciones

The status bar at the bottom shows the date and time as 10/04/2024 12:21 p.m. and the weather as 31°C.

Podemos acceder para verificar que todo este en orden
pruebastecnicas.azurewebsites.net/api/v1/swagger-ui/index.html

The screenshot shows a browser window with the URL pruebastecnicas.azurewebsites.net/api/v1/swagger-ui/index.html. The page title is "Swagger UI". The main content area is titled "OpenAPI definition v0 OAS3" and shows the following endpoints under "Endpoint characters":

- PUT /characters/put/{id} Endpoint Updating Superheroes
- POST /characters/post Endpoint inserts superheroes
- GET /characters/getCharacterByID Endpoint that yields specific data
- GET /characters/get/{id} Endpoint Throwing Superheroes By id

The status bar at the bottom shows the date and time as 10/04/2024 12:25 p.m. and the weather as 31°C.

Notas:

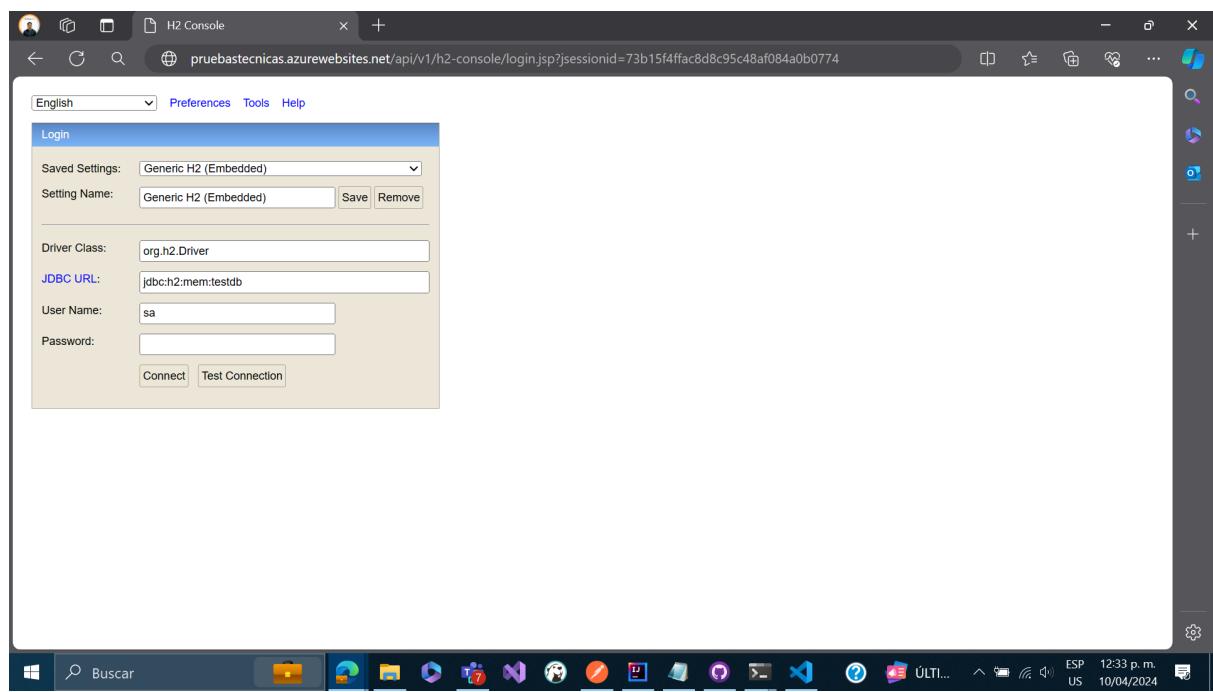
Se utilizó una BD en memoria por lo que no hay datos a la hora de ingresar por lo que es recomendable ejecutar el siguiente script para tener unos datos con el cual trabajar o de lo contrario hacer varias peticiones con el método post

`INSERT_CHARACTER.sql`

ruta local: [H2 Console](#)

ruta en nube:

pruebastecnicas.azurewebsites.net/api/v1/h2-console/login.jsp?jsessionid=73b15f4ffac8d8c95c48af084a0b0774



Usuario: sa

Password:password

[JDBC URL:jdbc:h2:mem:testdb](#)

