PO03_01. Web API de GameStore

1. Aplicación Web API en .NET, Entity Framework Core de gestión de Videojuegos.

El objetivo de esta actividad será el de desarrollar una API con ASP.NET en el que se trabaje persistencia de datos, gestión de usuarios (autenticación de estos y autorización en base a roles de usuario) y prueba y testeo de estos servicios con Postman.

En este caso se tienen los test en postman que han de cumplir los servicios de la API, para introducirse en la metodología TDD.

El puerto donde estén disponibles los servicios será el que está definido en la colección de Postman.

2. Base de datos

Se utilizará la base de datos de GameStoreAPI que se incluye en el repositorio de Github Classroom de la actividad.

3. Casos de uso

La API a desarrollar tendrá que tener los servicios suficientes para ofrecer toda la funcionalidad que se detallará a continuación, pero en general se tratará de poder realizar parte de las operaciones CRUD sobre los modelos implicados, así como algunas acciones propias de la lógica de negocio de esta solución en particular. Igualmente habrá que gestionar procesos de autenticación y autorización de usuarios personalizados, utilizando para ello tokens de tipo JWT.

MODELOS

- Game: Define los juegos y se corresponde con la tabla Game de la BD. Ha de tener las restricciones que se puedan deducir de la definición de la BD.
- Genre: Está relacionada con Game en una relación de uno a muchos. No tendremos servicios que interactúen directamente con este modelo, pero si se podrá obtener el género de un juego de forma indirecta.

- Pedido: Es el modelo que se utilizará para crear pedidos sobre juegos, por lo tanto estarán relacionados con ellos. Tendrá que tener, al menos, los siguientes campos (se han de respetar los nombres propuestos de cara a los test con los que se valorará la actividad):
    - Amount: Indica el número de juegos que se quieren comprar en este mismo pedido. Su valor mínimo es de 1 y el máximo de 10.
    - Address: Campo obligatorio que contiene la dirección de envío del pedido y que tiene una longitud mínima de 5 caracteres.
    - GameId: Id del juego sobre el que se realiza el pedido.
    - ComercialId: Id del comercial que realiza el pedido.
- GameStoreUser: Usuario propio de nuestra aplicación, que servirá para ampliar la información que trae por defecto un usuario de Identity. Tendrá los siguientes campos, todos ellos obligatorios.
    - FullName: Nombre completo. Longitud mínima de 10 caracteres.
    - IsActive: Campo booleano que indica el estado activo/inactivo del usuario. Solo usuarios activos podrán crear nuevos pedidos.

GESTIÓN DE USUARIOS

Como se ha indicado más arriba, se utilizará Identity para la gestión de usuarios y sus roles, pero en este caso los usuario serán personalizados para poder incluir los campos que se han detallado para el modelo GameStoreUser.

Igualmente se ha dicho que se usarán tokens de tipo JWT para autenticar a los usuarios en las distintas peticiones a servicios.

Será obligatorio que los usuarios estén en una BD independiente de la que contiene los datos.

Existirán 2 tipos de roles y podrán hacer lo siguiente:

- **Comercial**: Puede realizar todo tipo de acciones a excepción de aquellas que sean exclusivas de un usuario Admin.
- **Admin**: Este tipo de usuario es el único que puede cambiar el estado de otro usuario y ver detalles de un pedido en base al id del pedido.

Se crearán 3 usuarios por defecto, todos activados, relacionados con los roles detallados:

- comercial1@gamestore.com

- comercial2@gamestore.com
- admin@gamestore.com

Para todos ellos, el password de acceso será la cadena "Qwer123!".

El token que se genera para el usuario autenticado ha de tener solo los siguientes *claims* y características:

- Identificador único del token.
- Fecha de emisión del token.
- El claim que define el portador del token tendrá que contener el UserName del usuario.
- Como proveedor de identidad del token tendrá que figurar el texto "2DAWA".
- La audiencia será "GameStoreClient".
- El token tendrá una caducidad de 2 horas.
- Tendrá que tener un claim que contenga todos los roles del usuario y sirva para autorizar las acciones en base a roles.

SERVICIOS

Los *endpoints* de todos los servicios han de accederse mediante la siguiente URL:

https://<host>:<port>/api

A esta URL se le concatenará los siguientes fragmentos, bajo los cuales se ofrecen distintos servicios

- /Games : Se tendrán que ofrecer los siguientes servicios
    - Listado de todas los juegos. Tras ordenarlos en base al precio del juego se mostrarán los 15 primeros.

```
[
    {
        "id": 19,
        "title": "Extreme Football",
        "description": "Juego de fútbol callejero.",
        "stock": 6,
        "price": 19.99,
        "genreId": 4,
        "genre": null,
        "pedidos": []
    },
    {
        "id": 10,
        "title": "Battle Tactics",
        "description": "Planea tu estrategia y vence al enemigo.",
        "stock": 9,
        "price": 24.99,
        "genreId": 5,
        "genre": null,
        "pedidos": []
    },
```


- Detalles de un juego, que ha de contener información del género al que pertenece y con los pedidos creados para este juego.


```
{
    "id": 1,
    "title": "Super Action Hero",
    "description": "Juego de acción trepidante con múltiples niveles.",
    "stock": 10,
    "price": 49.99,
    "genreId": 1,
    "genre": {
        "id": 1,
        "name": "Acción",
        "description": "Juegos centrados en combates, desafíos físicos y reflejos rápidos.",
        "games": [
            null
        ]
    },
    "pedidos": [
        {
            "id": 2,
            "amount": 5,
            "address": "Calle Mayor 7",
            "gameId": 1,
            "game": null,
            "comercialId": ""
        }
```

- /Pedidos: Se tendrán que ofrecer los siguientes servicios
  - /ComercialPedidos: Listado de pedidos de un comercial. Se mostrarán todos los pedidos creados por un usuario cuyo email se pasa como último segmento en la URL. Un comercial solo podrá solicitar sus propios pedidos. Si trata de pedir los de otro usuario se devolverá el error definido en los test.


```
[
    {
        "id": 25,
        "amount": 5,
        "address": "296 Marquardt Cliff",
        "gameId": 1,
        "game": null,
        "comercialId": "bb8f8fce-2bdd-47ab-aeaa-c5dcfb27f7d7"
    },
    {
        "id": 29,
        "amount": 10,
        "address": "Calle Mayor 7",
        "gameId": 1,
        "game": null,
        "comercialId": "bb8f8fce-2bdd-47ab-aeaa-c5dcfb27f7d7"
```


  - Detalle de un pedido junto con la información del juego para quien fue creado. Solo lo puede ejecutar un usuario administrador.


```
{
    "id": 57,
    "amount": 10,
    "address": "Calle Mayor 7",
    "gameId": 1,
    "game": {
        "id": 1,
        "title": "Super Action Hero",
        "description": "Juego de acción trepidante con múltiples niveles.",
        "stock": 10,
        "price": 49.99,
        "genreId": 1,
        "genre": null,
        "pedidos": [
            null
        ]
    },
    "comercialId": "bb8f8fce-2bdd-47ab-aeaa-c5dcfb27f7d7"
}
```

- Edición de un pedido. Sólo lo puede realizar un usuario administrador (Retorna 204)

  JSON enviado en la petición:

```
{
  "id": {{newPedidoId}},
  "amount": 5,
  "address": "{{randomAddres}}",
  "gameId": 1,
  "comercialId": "{{comercial1Id}}"
}
```

- Creación de un pedido para un juego (Retorna 201 en caso de éxito y devuelve el elemento creado). Si el usuario que intenta crear el pedido no está activo, se retornará un código de respuesta 403 (Forbiden).

  JSON enviado en la petición:

```
{
  "amount": 10,
  "address": "Calle Mayor 7",
  "gameId": 1
}
```

  Respuesta obtenida en caso de éxito:

```
{
    "id": 57,
    "amount": 10,
    "address": "Calle Mayor 7",
    "gameId": 1,
    "game": null,
    "comercialId": "bb8f8fce-2bdd-47ab-aeaa-c5dcfb27f7d7"
}
```

- Eliminación de un pedido (Retorna 204 en caso de éxito). Esta acción solo la podrá realizar un administrador del sistema (Si no retorna 403).

- /Auth : A este segmento se le añadirá otro más para ofrecer los siguientes servicios:
  - /Register : permite la creación de un usuario. Necesitará recibir los datos en el cuerpo de la petición, en formato Json, conteniendo todos los campos del modelo GameStoreUser junto con los campos “email” y “password”, para los campos UserName y Password del usuario de Identity. Devuelve 201 en caso de éxito.

JSON Enviado en la petición:

```
{
  "email": "{{newUserEmailTest}}",
  "password": "{{userPassTest}}",
  "fullName": "{{userFullNameTest}}",
  "isActive": false
}
```

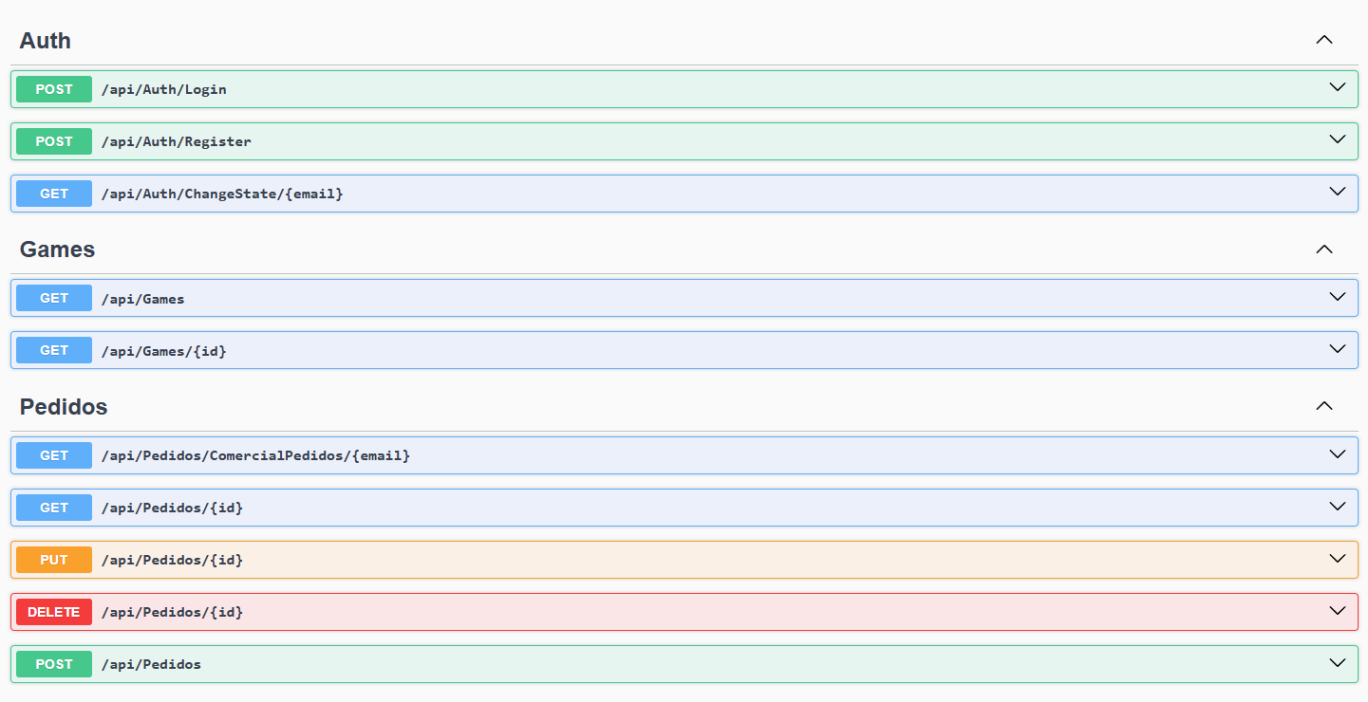
- /Login: Permite verificar la autenticidad de un usuario mediante los campos “email” y “password”. Al igual que en el caso anterior, estos datos se han de pasar en formato Json en el cuerpo de la petición.

  JSON de la petición:

```
{
  "email": "admin@gamestore.com",
  "password": "Qwer123!"
}
```

  La respuesta es texto del token directamente, sin encontrarse dentro de ningún json.

- /ChangeState: Este servicio, solo permitido para usuarios administradores, que recibe el email de un usuario como último segmento de la URL, cambia el estado de un usuario (de forma persistente) al opuesto del que tiene antes de la llamada. Retorna 200 en caso de exito y 403 si el token no es de un usuario administrador. El cuerpo de la respuesta puede no contener nada.



POSTMAN

En el repositorio se incluye un archivo de consultas de postman, en forma de colección, con todas las consultas que hay que probar junto con sus test, que serán las que se evaluarán en este caso, por lo que se recomienda poner especial atención a las consultas y test de esta colección.

4. Penalizaciones:

No funciona -6 puntos.