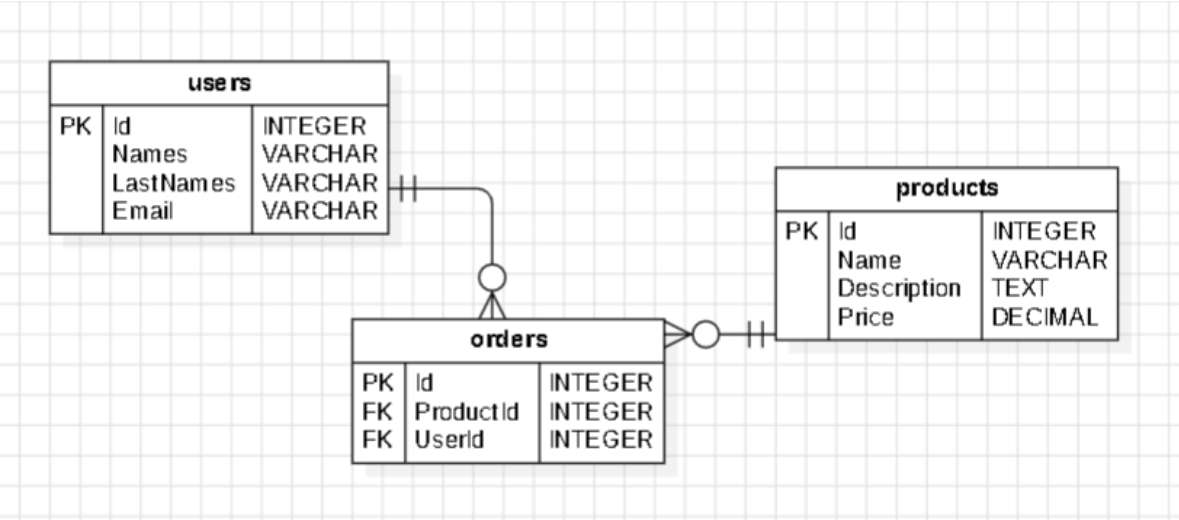
Actividades .NET

19 de septiembre

Objetivo:

Desarrollar una aplicación (webapi) utilizando ASP.NET Core con MySQL que maneje tres entidades relacionadas. Se deberán crear endpoints siguiendo buenas prácticas y documentar el servicio utilizando herramientas como swagger o postman. (*si utilizan swagger se debe evidenciar la manipulación de swagger ya que por defecto viene activado*). La actividad está enfocada en la correcta implementación de relaciones entre modelos (entidades), la creación de endpoints RESTful, la validación de la información y la documentación.



"Si considera que hacen falta campos o tablas para una mejor solución, bienvenidos sean los cambios"

Temas por investigar.

- Relaciones entre modelos **HasOne**, **HasMany**, y **WithMany**

- Métodos adicionales de una relación HasForeignKey, HasPrincipalKey, OnDelete, HasIndex, Ignore, Property, HasAlternateKey, HasDefaultValue, ToTable

- Investigar, **pero no aplicar** Data Annotations

- Aplicar validación de modelos con FluentValidation

Entregables

- **Entidades**:
    - Al menos tres entidades relacionadas con sus respectivas configuraciones en el `DbContext` utilizando `HasOne`, `HasMany`, y `WithMany`.
    - Si es necesario, las relaciones adicionales deben ser configuradas con métodos como `HasForeignKey`, `OnDelete`, y `ToTable`.
- **Endpoints**:
    - Creación de los endpoints RESTful siguiendo buenas prácticas de nomenclatura y diseño (por ejemplo, `/api/usuarios`, `/api/ordenes`, `/api/productos`).
    - Métodos `GET`, `POST`, `PUT`, `DELETE` implementados correctamente en los controladores para cada entidad (*ustedes deciden en que lenguaje mostrar los endpoints (en/es*).
- Implementación de **FluentValidation** para validar los datos de entrada en lugar de usar **Data Annotations**.
- Archivos de validación separados, que deben contener las reglas de validación para cada entidad (por ejemplo, `UsuarioValidator`, `OrdenValidator`).

- **Postman**: deberán entregar una colección exportada que contenga todas las peticiones HTTP para probar los endpoints (GET, POST, PUT, DELETE).
    - La colección de Postman debe estar organizada por recurso (Usuarios, Ordenes, Productos). (es/en)
    - Cada petición debe incluir ejemplos de datos válidos y errores de validación.
    - Los validadores deben estar registrados correctamente en el contenedor de dependencias (`Startup.cs` o `Program.cs`). *"importante hay que destacar que puede que nos encontremos un archivo startup.cs más adelante"*

*Es importante que las relaciones entre las entidades se evidencien de manera clara en los **endpoints**. No quiero que se muestren los identificadores (ID) de las entidades relacionadas, sino que se exponga la información completa de las mismas.*

*Por ejemplo, en el **endpoint de ordenes** (`/api/ordenes`), no solo quiero ver los ID de los usuarios o productos asociados, sino también información detallada de estos, como el nombre del usuario y los detalles de los productos. Esto es necesario para verificar que las relaciones entre entidades están correctamente implementadas y que los datos relacionados se cargan adecuadamente en las respuestas.*