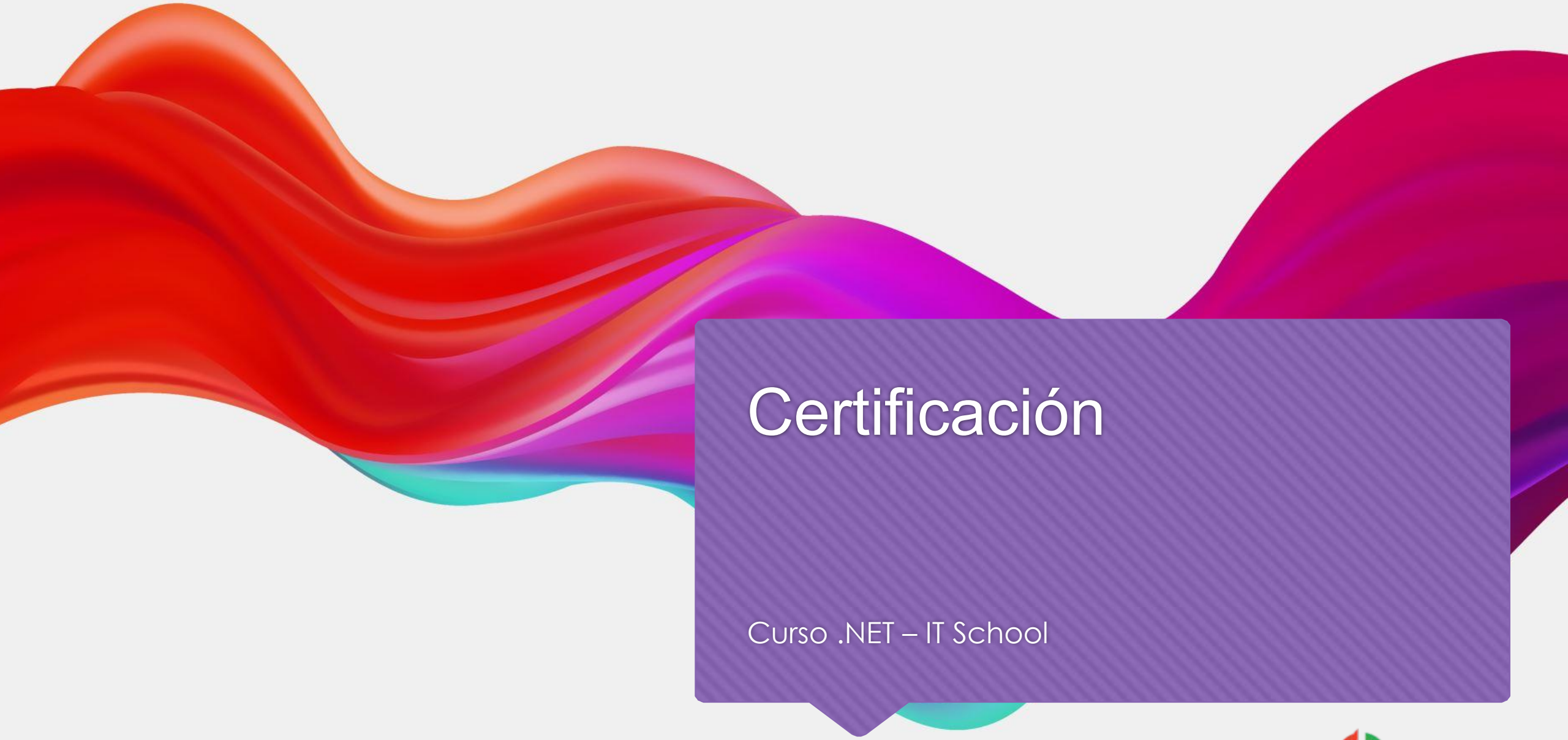


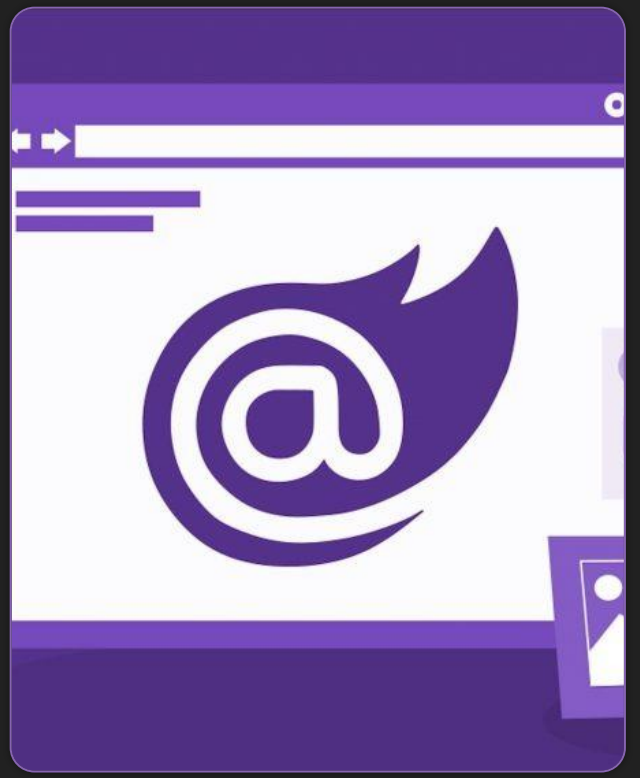
Certificación

Curso .NET – IT School

IT SCHOOL
LA POSTA!

¡Manos a la Obra! El caso Pampazon

- Pampazon es una empresa de logística que necesita una aplicación para mejorar la gestión de sus almacenes.
- Diseñaremos una API que registre e informe el ingreso, almacenamiento y envío de mercadería.
- Luego añadiremos un almacén de datos utilizando Entity Framework y SQL Server
- Y, finalmente, crearemos un front-end con Blazor que permitirá realizar todas estas las operaciones desde una aplicación web.

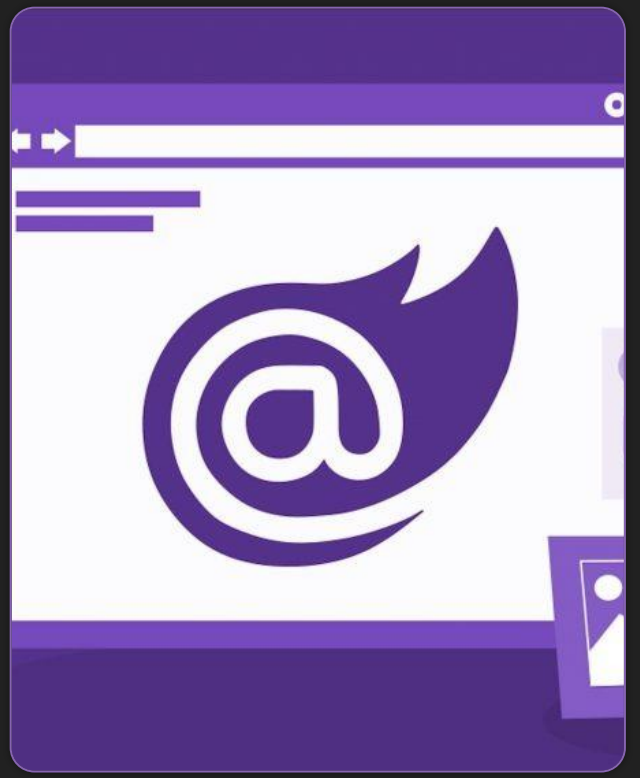


Requerimientos: Productos

¡Sin productos, un almacén no es más que un espacio vacío!

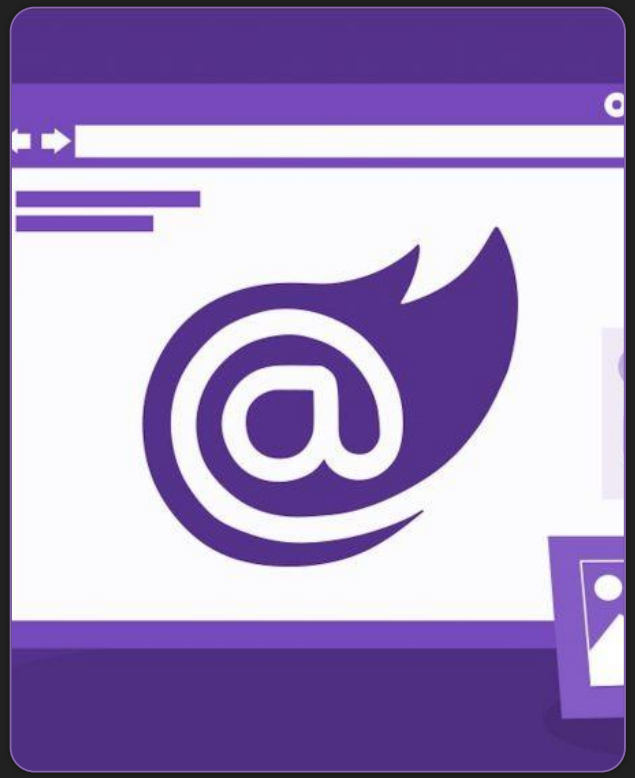
Necesitamos una API que nos informe sobre las características de los productos en el almacén:

- Código de producto
- Descripción
- Dimensiones (alto, ancho y profundidad en cm)



Requerimientos: Productos

- Diseña el modelo para representar un producto
- Crea un controlador llamado ProductosController que implemente PUT, POST, DELETE y GET para dar de alta, modificar, eliminar y obtener un producto a través de la API.
- En esta primera etapa, almacena los datos necesarios para tu API en una lista estática en el controlador correspondiente a cada punto (como en WeatherForecastController)



Probar nuestra API

Antes de seguir avanzando, es un buen momento para probar nuestra API.

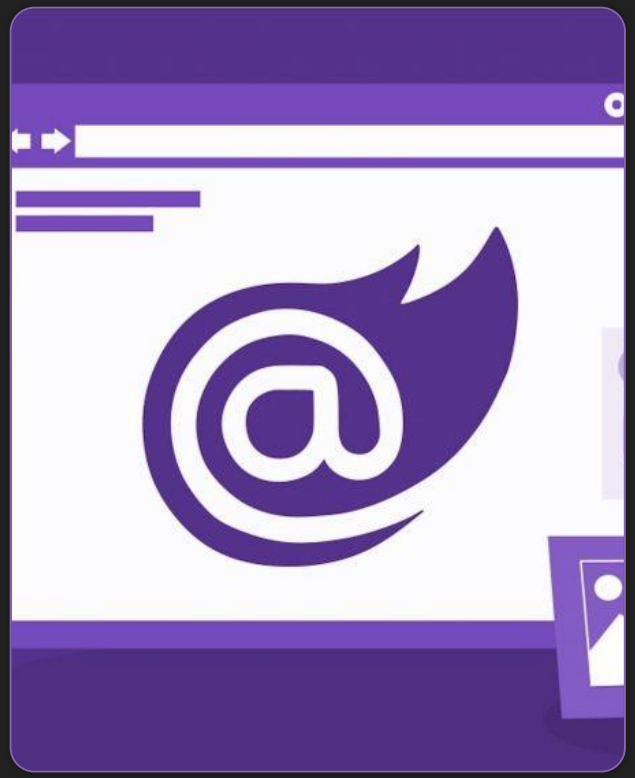
Existen muchas herramientas para ello. Te recomendamos el plugin REST Client para Visual Studio Code.

Otras populares son Postman e Insomnia

Requerimientos: Posiciones

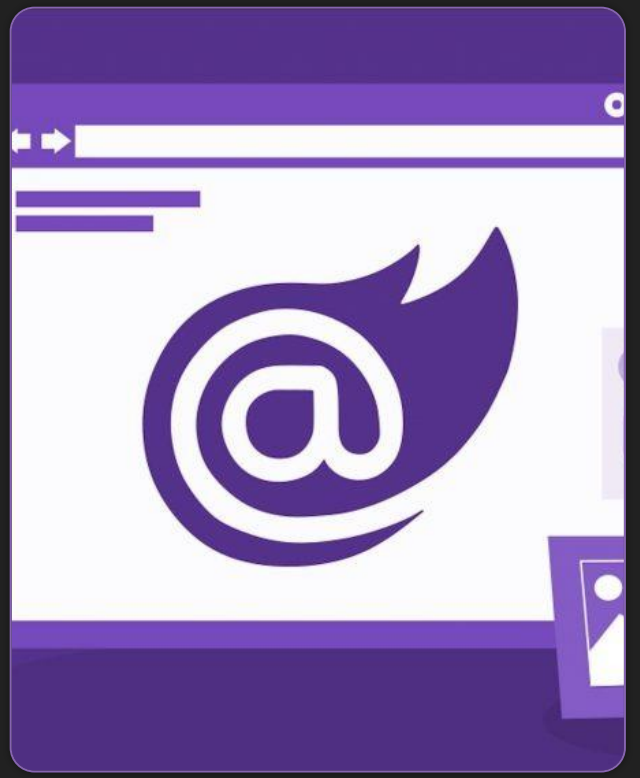
Nuestro almacén tiene que estar ordenado. Creemos posiciones para saber dónde se ubican los productos. Una posición consta de los siguientes datos:

- Pasillo: cada pasillo está identificado con una letra mayúscula, de la A a la Z.
- Seccion: los pasillos están divididos en secciones, y cada una es identificada de 1 a n en cada pasillo.
- Estantería: Las estanterías son las estructuras donde se almacenan los productos. Cada estantería se identifica de 1 a n en cada sección
- Nivel: Las estanterías suelen tener varios niveles o pisos donde se colocan los productos. Los niveles se numeran de abajo hacia arriba, comenzando con el nivel 1 en la base.
- Así una posición podría indicarse como: A.2.7.1 (pasillo A sección 2, estantería 7 nivel 1).
- Cada posición contiene cierta cantidad un único producto. Por ejemplo: 20 Zapatillas.



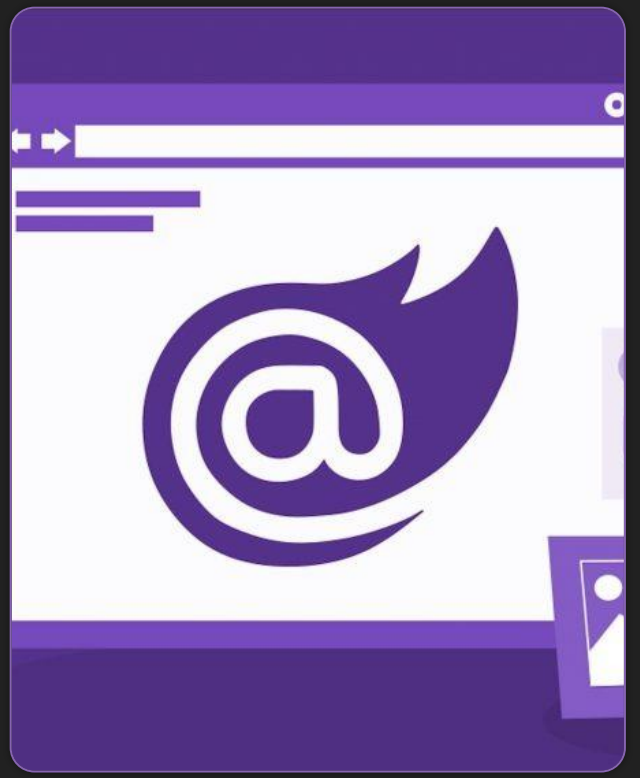
Requerimientos: Posiciones

- Amplía el modelo de productos para registrar en qué cantidad y posición podemos encontrarlos.
- Ten en cuenta que un producto puede estar distribuido en varias posiciones. Por ejemplo, hay 30 cajas de Zapatillas Azules en la posición A.2.4.2, 80 en A.2.4.3 y 120 en B.1.3.2
- Añade un nuevo controlador llamado Stock que permita dar de alta, baja o modificar las cantidades y ubicaciones de los productos.



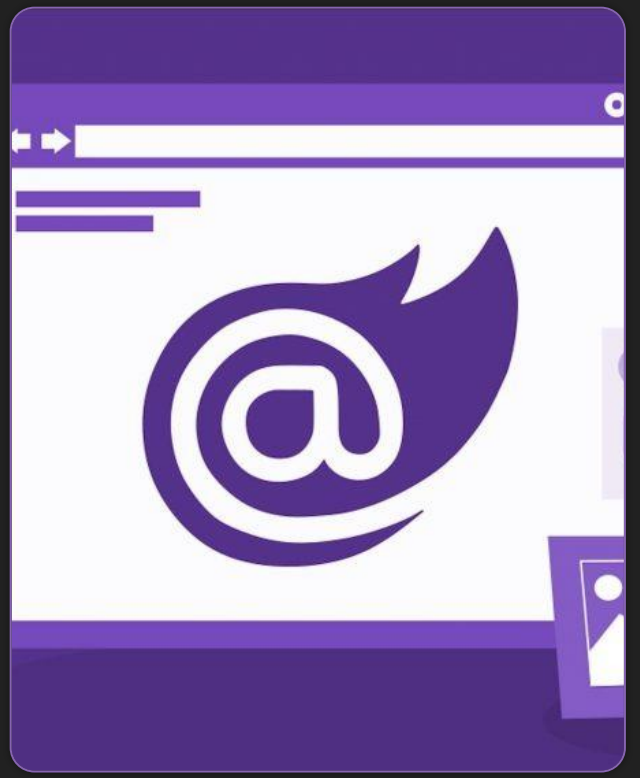
Requerimientos: Clientes

- Los productos tienen dueño, que no es Pampazon sino sus clientes. Ellos son quienes alquilan las posiciones en el almacén para guardarlos hasta tanto deban ser enviados a los compradores.
- Si registramos quién alquila cada posición sabremos a qué cliente pertenece cada ítem en particular.



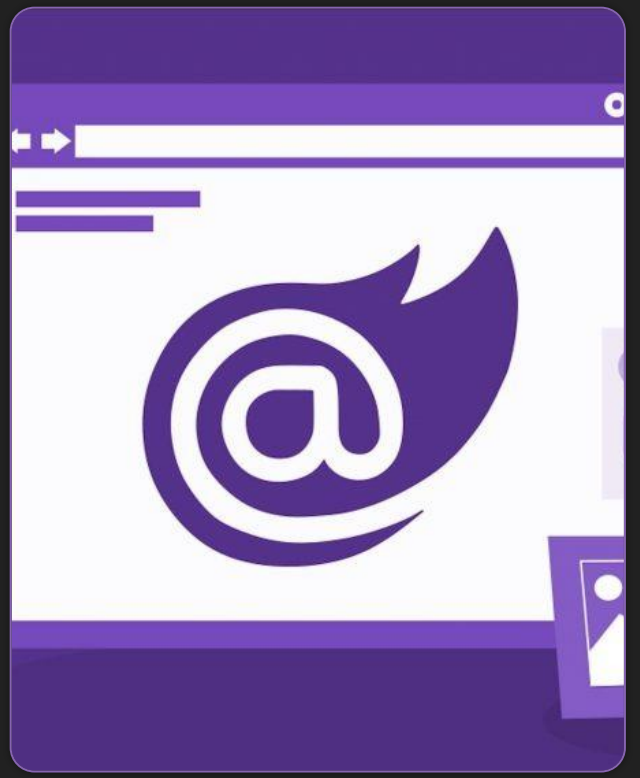
Requerimientos: Clientes

- Pampazon identifica a sus clientes por CUIT y Razon Social.
- Crea un modelo para mantener los datos de cada cliente y la lista de posiciones que alquila.
- Crea un controlador llamado ClientesController para dar de alta, baja y modificar estos datos.



Requerimientos: Ingreso de mercadería

- La mercadería es remitida por los clientes de Pampazon en camiones.
- Junto con la carga, el transportista presenta un remito que indica qué productos y cantidades entrega.
- Pampazon registra el remito pero OJO! no ingresa la mercadería sin antes contarla primero.
- En principio, implementemos en nuestra API el registro de nuevos remitos, que estarán “Pendientes de ingreso”.



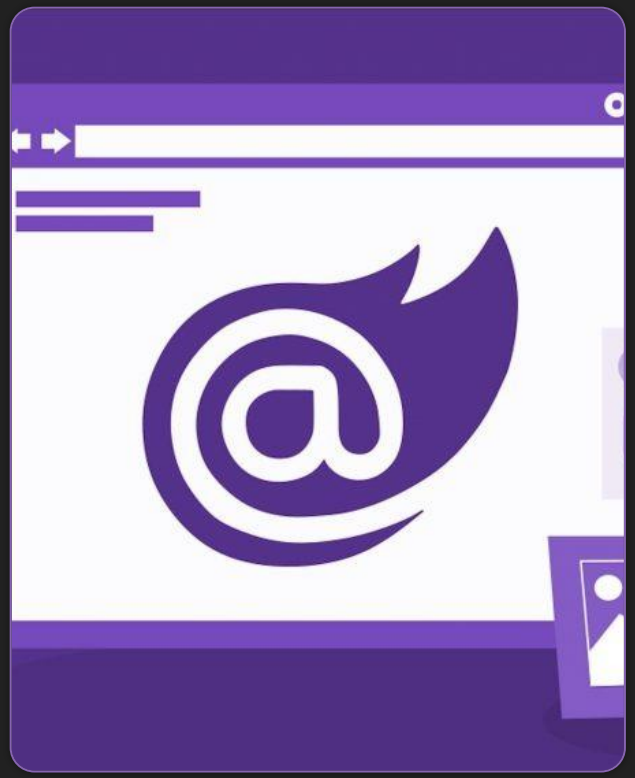
Requerimientos: Ingreso de mercadería

Un remito contiene los siguientes datos:

- Fecha
- CUIT del cliente
- CUIT del transportista
- Lista de códigos de producto y cantidades

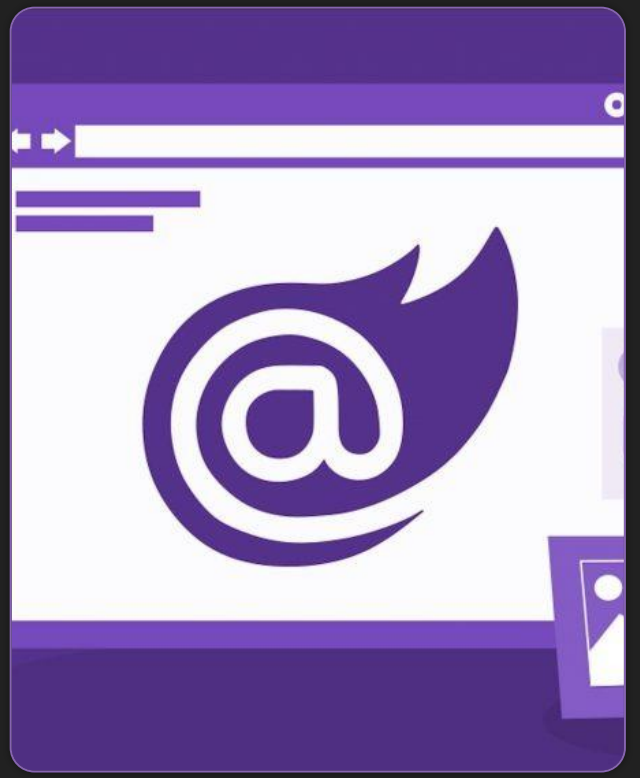
… y debemos tener en cuenta también el dato Estado, que en principio será "Pendiente de ingreso". Utiliza una enumeración, ya que iremos implementando más estados a medida que avancemos.

Por ahora, crea el modelo del remito y un nuevo controlador, RemitosController. Implementaremos el método GET para obtener los datos de un remito, PUT para crear uno nuevo, y POST para cambiar el estado. El resto de los datos no pueden ser modificados.



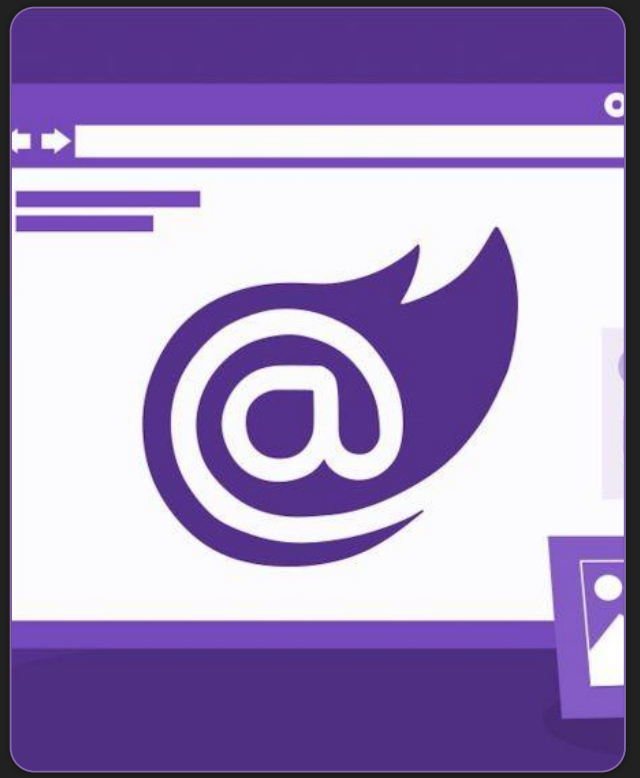
Requerimientos: Ingreso de mercadería

- Hemos registrado el remito pero no ingresamos la mercadería, ya que debíamos esperar a que los operarios de Pampazon la cuenten primero.
- Modifiquemos el método POST para recibir dos nuevos estados: “Ingresado” y “Rechazado”.
- Por ahora no tendremos en cuenta ingresos o rechazos parciales.
- Ten en cuenta que debemos solicitar, en el request, en qué posiciones se han almacenado los productos ingresados, a fin de actualizar el stock.



Requerimientos: Egreso de mercadería

- La mercadería sale del almacén a través de órdenes.
- Una orden es el documento por el cual un cliente de Pampazon (quien alquila el espacio y es dueño de los productos) le ordena al almacén que envíe cierta cantidad de productos a un tercero, por ejemplo un comprador de su tienda virtual.



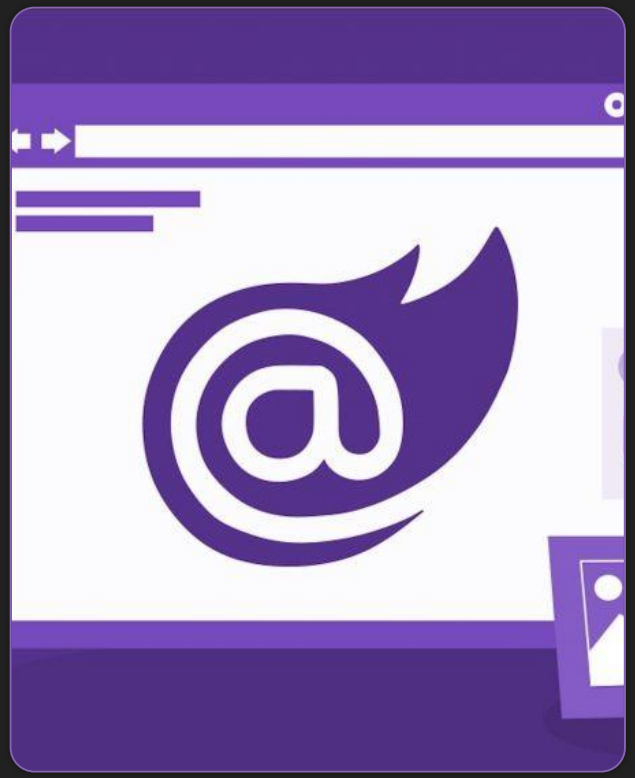
Requerimientos: Egreso de mercadería

La orden tiene los siguientes datos:

- Número de orden
- Fecha
- Cliente
- Lista de códigos de producto y cantidades a enviar
- Nombre del destinatario
- Dirección del destinatario

¡Ya sabes qué hacer! Crea un modelo, un nuevo controlador llamado OrdenesController e implementa la operación PUT para que la API pueda registrar las nuevas órdenes, y GET para consultarlas.

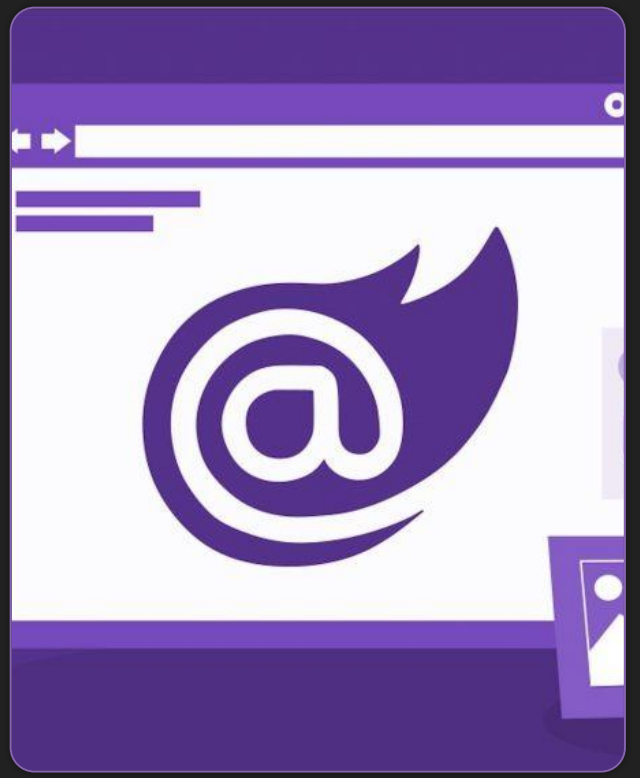
Ten en cuenta que la orden también tendrá un estado, que en principio será "Pendiente".



Requerimientos: Egreso de mercadería

Hemos recibido una orden, ¡hay que prepararla!

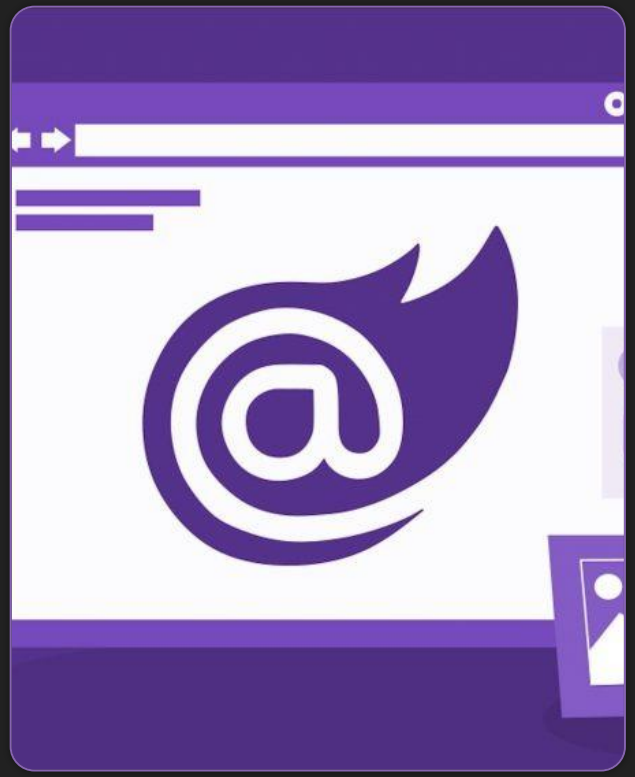
- Preparar una orden implica ir a buscar los productos al almacén, empacarlos y dejarlos en la bahía de salida, donde serán recogidos por un transportista.
- Tendremos que implementar el método POST para cambiar el estado de una orden a “Preparada”.
- Ten en cuenta que debemos solicitar, en el request, de qué posiciones se han retirado los productos a fin de disminuir el stock.



Requerimientos: Despacho

¡Hay ordenes listas para ser despachadas!

- Llega un camión a la bahía de salida. Iniciamos la confección de un nuevo remito, esta vez para entregar al transportista que se llevará la mercadería.
- Lo llamaremos Despacho para no confundirnos con los remitos de mercaderías recibidas que ya hemos implementado.
- Entonces, creamos el despacho cuando llega el camión, lo actualizamos a medida que se cargan las órdenes, y lo finalizamos una vez que el camión parta.



Requerimientos: Despacho

Para crear un nuevo despacho necesitaremos:

- Número de despacho
- Fecha
- CUIT del transportista

y para ingresar una orden al despacho:

- Número de despacho
- Número de orden

Finalmente, cuando nos indiquen que ha finalizado la carga, pasaremos las órdenes a estado “Despachadas” y marcaremos el despacho como finalizado, para que ya no pueda ser modificado.

Implementa estas operaciones en el controlador DespachosController. Utiliza PUT para iniciar, POST para ingresar cada orden y/o finalizar (de acuerdo a los datos que recibas), y el GET para recuperar un despacho.