Prueba de Evaluación Técnica – FullSack

En la empresa "Súper Zapatos" quieren realizar una aplicación web para llevar el control de inventario del calzado disponible en su zapatería. Dada esta necesidad, los expertos de tecnología de la empresa consideraron que la mejor opción sería hacer una aplicación ASP NetCore + Angular.

El producto mínimo viable es poder mostrar la información de inventario permitiendo que la vista se ajuste a Móvil / Tablet / PC consumiendo una API Rest de Backnet.

Además, esta aplicación debería de permitir a los administradores de la empresa poder agregar la información en la base de datos.

Dada la necesidad anteriormente planteada, se consultó a un experto de base de datos y él recomendó tener una base de datos compleja para lograr el proyecto. Sin embargo, como el presupuesto del cliente era limitado, se concluyó que la mejor opción era tener únicamente el mantenimiento y los servicios esenciales de la aplicación, lo que implicaba únicamente tener dos tablas básicas, cuyas descripciones son:

table articles id name description price total_in_shelf total_in_vault store_id	**table stores** id name address

Dadas las necesidades de la aplicación, es necesario realizar al menos 3 servicios, cuyos URL y respuestas están descritos en la documentación del API adjunta a esta prueba, estos servicios serán consumidos a futuro por cualquier dispositivo móvil. Además, se requiere crear los mantenimientos para artículos y tiendas en el cual se pueda agregar o modificar una tienda o un artículo vía WEB.

Se esperan como entregables:

- Proyecto basado en NetCore con C# para el backend, EF y SQL Server.
- El proyecto debe contener las capas necesitarías para la aplicación.
- De preferencia utilizando EF Code First, si es Database First incluir script de Base de datos.
- Generar los servicios acordes con la documentación del API adjunta.
- Todas las pantallas tendrán un diseño gráfico sencillo pero agradable basado en Angular 7 o superior utilizando AngularMaterial.
- Entregar todo en mismo repositorio GIT público que nos compartan acceso para revisarlo.

DOCUMENTACION de Services API

Important note: All the services should be created by using JSON as the response format.

General Response Format

All the requests will follow a simple basic format for the errors and another one for the success.

In the case where there was an error from the server side, the return will be:

{**"success"**:false,"error_code"**: XXX, "error_msg"**:"ERROR_MESSAGE"}

Then, if everything is working fine the return will be one of two possible results, the following is presented in the case where the service will return just one element (find by id).

{**"success"**:true,**"MODEL_NAME"**:{**"attr_name1"**:"attr_value1", ...}}

Finally, the following element will present the object that will be returned in the case where the request is for getting a list of elements

{**"success"**:true,**"PLURAL_MODEL_NAME"**:[{**"attr_name1"**:"attr_value1", ...}, {...}, ...]}

Service Error Responses

All the responses for the errors have the following format.

{"success":false,"error_code": XXX, "error_msg":"ERROR_MESSAGE"}

Also, the list of all the error codes and messages are:

400 - Bad request

404 - Record not found

500 - Server Error

Services description

List all stores

Request Method	GET
Path	/services/stores
Parameters	**None**
Sample request	Load all the stores that are stored in the Database.

Response	{
	"stores": [
	
	{
	"id": 1,
	"address": "Somewhere over the rainbow”,
	"name": "Super Store"
	}, { ... }
	],
	"success": true,
	“total_elements”: 2
	
	}

Errors	None

List all the articles

Request Method	GET
Path	/services/articles
Parameters	**None**
Sample request	Load all the articles that are in the Database.

Response	{
	
	"articles": [
	
	{
	
	"id": 1,
	
	"description": "The best quality of shoes in a green color”,
	
	"name": "green shoes",
	
	“price”: 20.15,
	
	“total_in_shelf”: 25,
	
	“total_in_vault”: 40,
	
	“store_name”: “Super Store”
	
	}, { ... }
	
	],
	"success": true,
	“total_elements”: 2
	
	}

Errors	None

List all the articles that are in a specific store

Request Method	GET
Path	/services/articles/stores/:id
Parameters	**id:** a numeric value of the ID of the store
Sample request	Load all the articles from a specific store that are in the Database.

Response	{
	"articles": [
	{
	"id": 1,
	"description": "The best quality of shoes in a green color”,
	"name": "green shoes",
	“price”: 20.15,
	“total_in_shelf”: 25,
	“total_in_vault”: 40,
	“store_name”: “Super Store”
	}, { ... }
	],
	"success": true,
	“total_elements”: 2
	}

Errors	Wrong parameters (id is not a number) {"error_msg": "Bad Request","error_code": 400, "success": false} No store with that ID {"error_msg": "Record not Found", "error_code": 404, "success": false}