Ejercicios de Reclutamiento Desarrollador.

Freddy Palma Muñoz

**Ejercicios SQL.**

**Ejercicio 1.**

CREATE TABLE ciudad(

id INT IDENTITY(1,1) NOT NULL,

nombre VARCHAR(255) NOT NULL,

CONSTRAINT id\_ciudad\_pk PRIMARY KEY (id)

);

CREATE TABLE comuna(

id INT IDENTITY(1,1) NOT NULL,

nombre VARCHAR(255) NOT NULL,

CONSTRAINT id\_comuna\_pk PRIMARY KEY (id)

);

CREATE TABLE proveedor (

rut INT NOT NULL,

nombre VARCHAR(255) NOT NULL,

telefono BIGINT NOT NULL,

pagina\_web VARCHAR(500) NOT NULL,

direccion VARCHAR(255) NOT NULL,

id\_comuna INT NOT NULL,

id\_ciudad INT NOT NULL,

CONSTRAINT rut\_proveedor\_pk PRIMARY KEY (rut),

CONSTRAINT id\_comuna\_proveedor\_fk FOREIGN KEY (id\_comuna) REFERENCES comuna(id),

CONSTRAINT id\_ciudad\_proveedor\_fk FOREIGN KEY (id\_ciudad) REFERENCES ciudad(id)

);

CREATE TABLE cliente(

rut INT NOT NULL,

nombre VARCHAR(255) NOT NULL,

direccion VARCHAR(255) NOT NULL,

id\_comuna INT NOT NULL,

id\_ciudad INT NOT NULL,

CONSTRAINT rut\_cliente\_pk PRIMARY KEY (rut),

CONSTRAINT id\_comuna\_cliente\_fk FOREIGN KEY (id\_comuna) REFERENCES comuna(id),

CONSTRAINT id\_ciudad\_cliente\_fk FOREIGN KEY (id\_ciudad) REFERENCES ciudad(id)

);

CREATE TABLE telefono\_contacto(

id INT IDENTITY(1,1) NOT NULL,

numero BIGINT NOT NULL,

rut\_cliente INT NOT NULL,

CONSTRAINT id\_telefono\_contacto\_pk PRIMARY KEY (id),

CONSTRAINT rut\_cliente\_telefono\_contacto\_fk FOREIGN KEY (rut\_cliente) REFERENCES cliente(rut)

);

CREATE TABLE categoria(

id INT IDENTITY(1,1) NOT NULL,

nombre VARCHAR(255) NOT NULL,

descripcion VARCHAR(255) NOT NULL,

CONSTRAINT id\_categoria\_pk PRIMARY KEY (id)

);

CREATE TABLE producto(

id INT IDENTITY(1,1) NOT NULL,

nombre VARCHAR(255) NOT NULL,

precio\_actual INT NOT NULL,

stock INT NOT NULL,

rut\_proveedor INT NOT NULL,

id\_categoria INT NOT NULL,

CONSTRAINT id\_producto\_pk PRIMARY KEY (id),

CONSTRAINT rut\_proveedor\_\_producto\_fk FOREIGN KEY (rut\_proveedor) REFERENCES proveedor(rut),

CONSTRAINT id\_categoria\_\_producto\_fk FOREIGN KEY (id\_categoria) REFERENCES categoria(id)

);

CREATE TABLE venta(

id INT IDENTITY(1,1) NOT NULL,

fecha DATE NOT NULL,

rut\_cliente INT NOT NULL,

descuento INT NOT NULL,

monto\_final INT NOT NULL,

CONSTRAINT id\_venta\_pk PRIMARY KEY (id),

CONSTRAINT rut\_cliente\_venta\_fk FOREIGN KEY (rut\_cliente) REFERENCES cliente(rut)

);

CREATE TABLE venta\_detalle(

id INT IDENTITY(1,1) NOT NULL,

id\_venta INT NOT NULL,

id\_producto INT NOT NULL,

cantidad\_vendida\_producto INT NOT NULL,

monto\_total\_producto INT NOT NULL,

CONSTRAINT id\_venta\_detalle\_pk PRIMARY KEY (id),

CONSTRAINT id\_venta\_venta\_detalle\_fk FOREIGN KEY (id\_venta) REFERENCES venta(id),

CONSTRAINT id\_producto\_venta\_detalle\_fk FOREIGN KEY (id\_producto) REFERENCES producto(id)

);

INSERT INTO ciudad (nombre) values ('Santiago');

INSERT INTO comuna (nombre) values ('Maipu');

INSERT INTO proveedor (rut, nombre, telefono, pagina\_web, direccion, id\_comuna, id\_ciudad) values (196498931, 'Freddy', 930799532, 'www.pricinprueba.cl','Cuatro poniente 322',1,1);

INSERT INTO proveedor (rut, nombre, telefono, pagina\_web, direccion, id\_comuna, id\_ciudad) values (195845921, 'Barbara', 930799532, 'www.pricinprueba.cl','Cuatro poniente 322',1,1);

INSERT INTO categoria (nombre, descripcion) values ('Chatarra', 'Comida deliciosa');

INSERT INTO producto (nombre, precio\_actual, stock, rut\_proveedor, id\_categoria) values ('Papas fritas', 1500, 10, 196498931, 1);

INSERT INTO producto (nombre, precio\_actual, stock, rut\_proveedor, id\_categoria) values ('Hamburguesa', 1000, 10, 195845921, 1);

INSERT INTO cliente (rut, nombre, direccion, id\_comuna, id\_ciudad) values (190566293, 'Hector', 'Pje. Catapilco 1731', 1, 1);

INSERT INTO venta (fecha, rut\_cliente, descuento, monto\_final) values (convert(datetime,'28-01-20 10:30:00 PM',5), 190566293, 0, 1500);

INSERT INTO venta\_detalle (id\_venta, id\_producto, cantidad\_vendida\_producto, monto\_total\_producto) values (10, 1, 1, 1500);

INSERT INTO venta (fecha, rut\_cliente, descuento, monto\_final) values (convert(datetime,'28-01-20 10:30:00 PM',5), 190566293, 0, 2000);

INSERT INTO venta\_detalle (id\_venta, id\_producto, cantidad\_vendida\_producto, monto\_total\_producto) values (11, 2, 2, 2000);

INSERT INTO venta (fecha, rut\_cliente, descuento, monto\_final) values (convert(datetime,'28-01-20 10:35:00 PM',5), 190566293, 0, 1500);

INSERT INTO venta\_detalle (id\_venta, id\_producto, cantidad\_vendida\_producto, monto\_total\_producto) values (12, 1, 1, 1500);

**Ejercicio 2.**

**Procedimiento Almacenado:** Es un grupo de instrucciones SQL que sirve para realizar ejecutar tareas en él. **Uso:** Este lo usaría para realizar una ejecución en cadena, como por ejemplo obtener algunos datos realizar alguna operación y con los datos obtenidos ejecutar un Trigger.

**Función Escalar**: Es una función para obtener parámetros y realizar una operación interna y retornar esta. **Uso:** Este lo usaría para cuando necesite realizar alguna operación matemática en base a los datos de la DB y retornar el resultado.

**Función Tabular:** Tabular un texto consiste en separar dos cadenas de texto un espacio concreto. Este espacio es el comprendido entre un tabulador y otro. Y es que dependiendo del programa que trate el tabulador, esta distancia puede variar.

**Trigger:** Es una instrucción sencilla que espera para que esta sea ejecutada.

**Uso:** Generalmente se utiliza en procedimiento almacenados para poder realizar algo del CRUD.

**Vista:** Es una tabla virtual que está definido por una consulta.

**Uso:** Generalmente se usan para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario.

**Ejercicio 3.**

**A).** La forma 1 es una variable declarada como tabla y la forma 2 es una tabla temporal. La tabla temporal es igual a una tabla normal, en cambio la variable declarada como tabla, pero tiene ciertas restricciones, no se pueden utilizar Insert into, ni tampoco Select into.

**B).** Yo utilizaría la tabla temporal, para poder trabajarla sin problemas y poder “jugar” con ella, además, para mí se me hace más fácil trabajar una tabla y en ella hacer el reporte necesario.

**Ejercicio de C#**

**Ejercicio 2.**

**LINQ:** Es una librería que permite realizar cualquier tipo de consultas a distintas fuentes de datos para no realizarlas de forma directa (base de datos, objetos, xmls, etc).

**Expresiones lambda:** Es una función anónima que normalmente se la utiliza para enviarla como parámetro a un método para ser evaluada en el mismo.

**Ejercicio 3.**

**A) Polimorfismo:** Se refiere a poder definir clases diferentes que tienen métodos o atributos denominados de forma idéntica, pero que se comportan de forma distinta.

**B) Clase abstracta:** Una clase que declara la existencia de métodos, pero no la implementación de dichos métodos, se considera una clase abstracta. Una clase abstracta puede contener métodos no abstractos, pero si o si, uno de ellos debe ser abstracto.

**C) Interfaz:** Una interfaz es un medio común para que los objetos no relacionados se comuniquen entre sí. Estas son definiciones de métodos y valores sobre los cuales los objetos están de acuerdo para cooperar.

**D) Explique: ¿Cuándo usaría una clase abstracta y cuándo una interfaz?**

La clase abstracta la utilizaría para definir subclases y así poder obtener los métodos y los atributos de esta. Y la interfaz, la utilizaría solo para utilizar lo necesario que desee implementar en mis clases.

**Ejercicio ASP.NET MVC**

**Ejercicio 2**

1. **Arquitectura Modelo-Vista-Controlador:** Es un patrón de arquitectura que separa los datos de la lógica de negocio, así se puede reutilizar el código y facilita el desarrollo de las aplicaciones. El Modelo es responsable de acceder al almacenamiento de los datos, el Controlador de la lógica de negocio y las Vistas de recibir los datos y mostrarlos al usuario.
2. **Startup.cs:** La clase Startup es la que arranca los componentes, configuración de la aplicación que se va a ejecutar.
3. **Autorización en ASP.NET MVC:** La autorización permite establecer un tipo de reglas o permiso de manera declarativa de controlador de acciones. De manera nativa esta clase te obliga a poner la regla de autorización en código duro. Al estar en código duro, se requiere que cada vez que se adicionan o modifican los roles tengamos que modificar el código y recompilar.

**Ejercicio 3**

**AJAX:** Es una técnica de comunicación asincrónica que se ejecutan en el cliente para realizar peticiones al servidor (request). Estas request pueden ser de tipo POST, GET, DELETE, PUT. AJAX tiene soporte en diferentes sistemas operativos, así como diferentes navegadores.

**Ejemplo de AJAX GET usando JQUERY:** AJAX de tipo GET recibe tres parámetros, la URL donde haremos la petición (request), los datos y la función. Un ejemplo sería:

$.ajax({

url: 'https://reqres.in/api/users',

success: function(respuesta) {

console.log(respuesta);

},

error: function() {

console.log("No se ha podido obtener la información");

}

});

Esta petición (request) solicita todos los usuarios que los mostraría por consola, en caso de ocurrir algún error, también los mostraría por consola.