C#

*Lucas Moy*

*https://www.youtube.com/watch?v=TqiysLEBZo4*

Apps para cel con Xamarin

Juegos con Unity para cel o consolas.

Con framework.net servicios web.

Dice que lo mas fácil es arrancar con aplicaciones de escritorio.

Visual Studio.

Dice que no es Visual Studio Code.

Dice que el fondo oscuro cuida mas los ojos.

En el diseño inserta un label y un botón. Luego ingresa al botón y codea esto:

private void Btn\_Cambiar\_Click(object sender, EventArgs e)

{

label\_HolaMundo.Text = "texto cambiado";

}

Al iniciar, cuando se toca el botón, aparece “texto cambiado” allí donde aparecía “hola mundo”.

Constante=/= Variable. No cambian mas de valor.

Nomenclatura: no pueden haber espacios; no pueden empezar con números; no puede ser una palabra reservada (if, while, etc).



Convención.

camelCase. Se usa para declarar variables

Screaming para las constantes.

Snake-case. Solo se mantiene para bases de datos.

Kebab para urls y nombres de carpetas. No es tan respetado esto. Importan mas las 2 primeras.

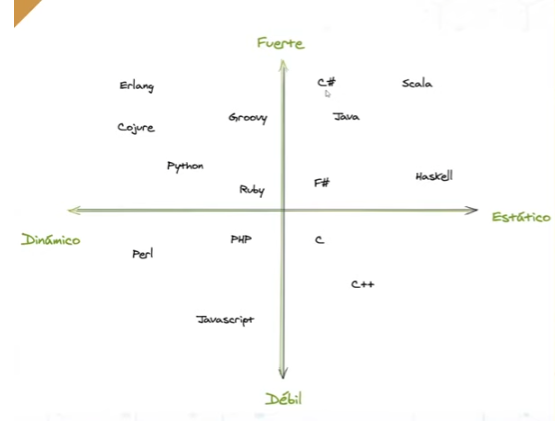
Convención l33t. Se usa para contraseñas, reemplaza algunas letras por números.

Notación húngara. Al ppio del nombre señala que tipo de dato es: strpersona.

Tipado fuerte /débil.

Tipado fuerte. Se debe definir qué tipo de dato es.

Lenguajes de tipado débil. No se declaran las variables especificando el tipo de dato.

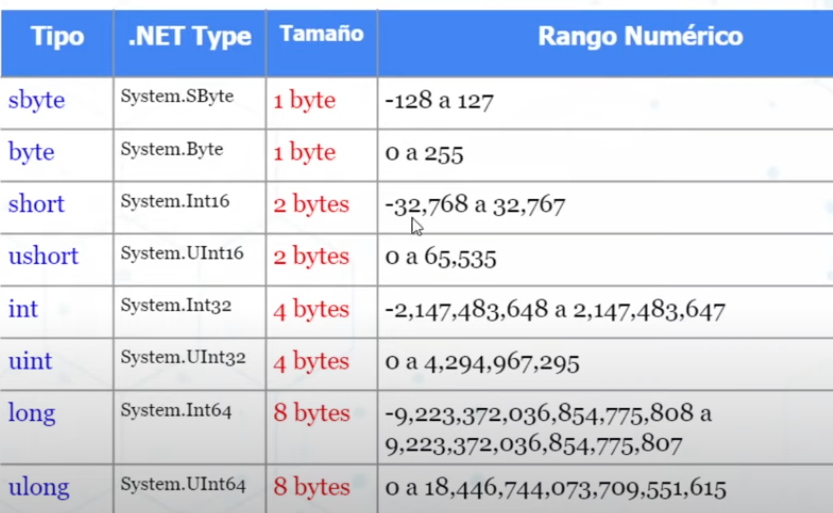


Dinámico/Estático. Estático: se deben crear las variables antes de usarlas.

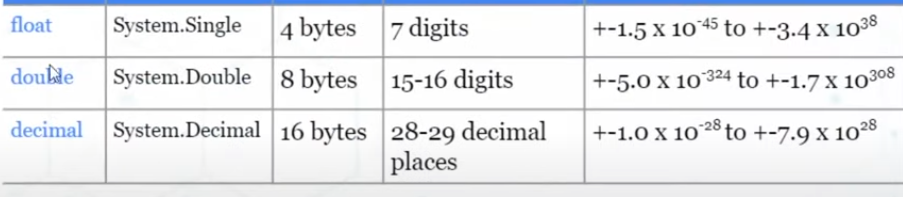
El resto se usa para saber si un número es múltiplo de otro. Como ser par o no.

Sistemas decimal/Octal (solo tiene números del 0 al 7. El 8 no existe, se pasa al 10). Luego está el sistema binario, con solo 0 y 1.

Por qué se usa el sistema binario. Porque el hardware se maneja con corriente eléctrica. Hoy hay corriente o no. 0 o 1.



Dice trabajar con int y long.



int numero = 123;

double numeroConComas = 6000.27;

string prueba = "Prueba de variable: "+ numeroConComas + numero;

Eso da 6000.27123 ya que al ser un string el primer tipo de dato, lo convierte todo a texto.

string prueba = numero + numeroConComas + "Prueba de variable: " ;

Ahí hace la cuenta correctamente.

private void Btn\_Cambiar\_Click(object sender, EventArgs e)

{

int edad = 16;

**bool esMayorDeEdad = edad >= 18; //Ese es el true, implícitamente.**

label\_HolaMundo.Text = "El usuario es menor de edad"; //esto es lo que se pone por defecto

if (esMayorDeEdad)

{

label\_HolaMundo.Text = "El usuario es mayor de edad";

} // Aclara luego que podría poner el else, y que es mas claro.

Escribe de esa manera el bool. Es interesante, porque queda todo muy implícito.

Cuando uno modifica el diseño, el nombre dentro del código se cambia solo.

int edad = Int32.Parse(textoEdad);

Eso agarra texto y convierte a número.

Se agregó un área de texto donde se puede ingresar un número y luego al tocar cambiar te devuelve si es mayor o menor de edad:

private void Btn\_Cambiar\_Click(object sender, EventArgs e)

{

string textoEdad = txtEdad.Text;

int edad = Int32.Parse(textoEdad);// Eso agarra texto y convierte a número.

bool esMayorDeEdad = edad >= 18;

lblResultado.Text = "El usuario es menor de edad"; //esto es lo que se pone por defecto

if (esMayorDeEdad)

{

lblResultado.Text = "El usuario es mayor de edad";

}}

Si pones un texto, se genera un error y te lleva a la parte del código que no funciona.

Ejercicio IMC

Arrays

Repite que un texto es también un array.

Array multidimensional. Es un array dentro de otro array.

Funciones

Para facilitar la lectura de código se puede ponerle muchas anotaciones. Otra forma es con funciones, con nombres descriptivos, lo cual el recomienda.

List<int> lista = new List<int>();

lista.Add(2);

lista.Add(20);

lista.Add(15);

lista.Add(77);

mostrarListado(lista);

}

private void mostrarListado(List<int> lista)

{

for (int i = 0; i < lista.Count; i++)

{

int numero = lista[i];

MessageBox.Show("El número es: "+numero);

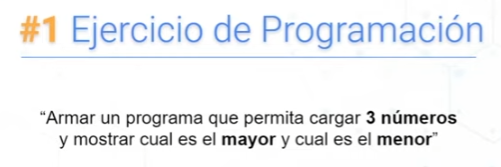
}

Así crea esta función. Pasando como argumento los valores de una lista.

Bugs es errores de programación. Significa insectos, eso es porque en un inicio entraban insectos en el hardware.

Debuggear es seguir paso a paso, para ver dónde está el error.  
Hay que hacer click a la izquierda de los números, y va a aparecer un círculo rojo. Luego start debug.

Ejercicio mayor/menor

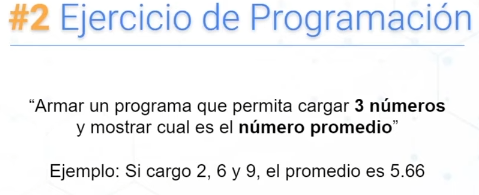




string txtNum1 = txtPrimero.Text;

int num1 = Int32.Parse(txtNum1);

Son 2 formas de recoger lo que el usuario escribe en un textarea y convertirlo a número para poder usarlo.



private void btnPromedio\_Click(object sender, EventArgs e)

{

string textNum1 = txtPrimero.Text;

double Num1 = Int32.Parse(textNum1);

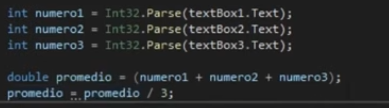
double Num2 = Int32.Parse(txtSegundo.Text);

double Num3 = Int32.Parse(txtTercero.Text);

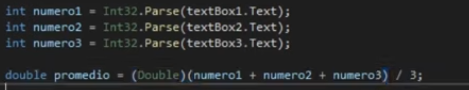
double promedio = (Num1 + Num2 + Num3) / 3;

MessageBox.Show("El promedio de los 3 números es: "+ promedio);

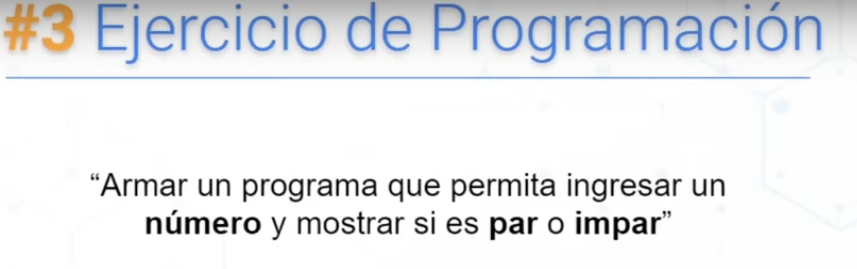
}

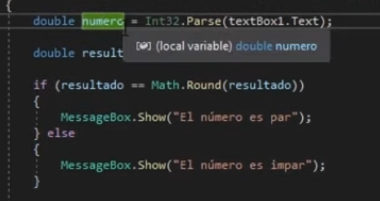


También se podría hacer así. Si se hubiera puesto el dividido 3, en la línea de arriba, el resultado hubiese ninguneado los decimales.



También así. Con casting. Se la ve pro.

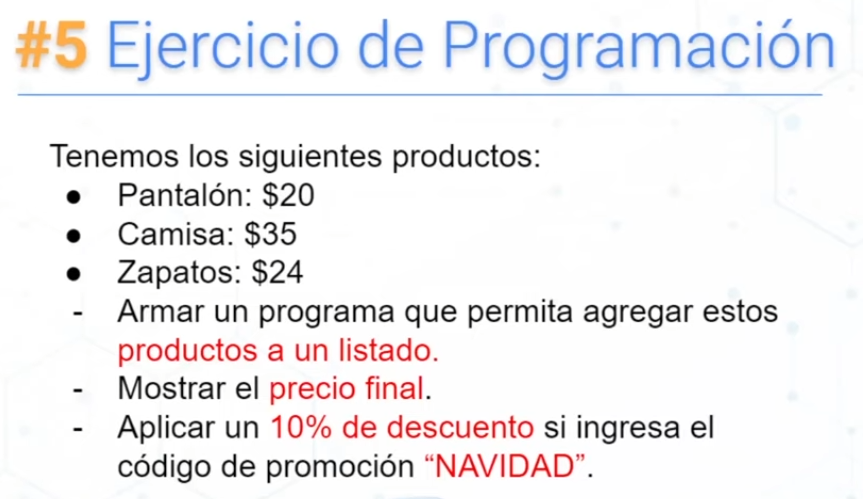




El lo había hecho así. Math.Round elimina los decímales.

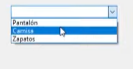
Los números pares divididos por 2 no dan decimales como resultado.

Por eso iguala y lo hace así.



Arranca poniendo un listbox. También un ComboBox, donde se van a agregar los productos para que el usuario elija.

Hay que hacer 2 click sobre la parte grisácea del form y se crea la función form1load. Apenas se cargue el formulario se va a ejecutar.



private void Form1\_Load(object sender, EventArgs e)

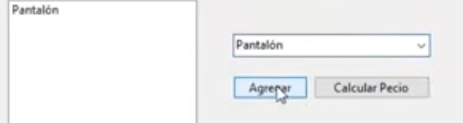
{

cboProctos.Items.Add("Pantalón");

cboProctos.Items.Add("Camisa");

cboProctos.Items.Add("Zapatos");

}



private void btnAgregar\_Click(object sender, EventArgs e)

{

string productoSeleccionado = cboProctos.SelectedItem.ToString(); //Para obtener el producto seleccionado

listCarritoDeCompras.Items.Add(productoSeleccionado); //Con esta linea sea agrega al textarea

}

private void btnCalcular\_Click(object sender, EventArgs e)

{

double precioFinal = 0;

int totalProductosCargados = listCarritoDeCompras.Items.Count; //Este count nos dice cuantos ítems hay

for (int i = 0; i < totalProductosCargados; i++)

{

string productoElegido = listCarritoDeCompras.Items[i].ToString();

MessageBox.Show(productoElegido) ;

}

Hasta acá, al tocar el botón calcular, recorre los objetos que fueron agregados al textare y devuelve su nombre, no hace ningún cálculo.

private void btnCalcular\_Click(object sender, EventArgs e)

{

double precioFinal = 0;

int totalProductosCargados = listCarritoDeCompras.Items.Count; //Este count nos dice cuantos ítems hay

for (int i = 0; i < totalProductosCargados; i++)

{

string productoElegido = listCarritoDeCompras.Items[i].ToString();

if (productoElegido== "Pantalón")

{

precioFinal += 20;

}

if (productoElegido == "Camisa")

{

precioFinal += 35;

}

if (productoElegido == "Zapatos")

{

precioFinal += 24;

}

}

MessageBox.Show("El precio final es: $" + precioFinal);

Ahí suma los valores y te da un resultado final.

Luego agrega label y textarea para el código de descuento.

if (txtPromo.Text == "NAVIDAD") //Acá recoge la info ingresada en el textarea de la promoción. Luego establece el if.

{

precioFinal = precioFinal \* 0.9;

}

MessageBox.Show("El precio final es: $" + precioFinal);

Ejercicio de mostrar números primos

//Ejercicio de números primos

double numero = 22;

bool esNumeroPrimo = true;

for (int i = 2; i < numero; i++)

{

if(numero % i == 0)

{

esNumeroPrimo = false;

}

}

if (esNumeroPrimo)

{ MessageBox.Show("El número es primo");

}

else

{

MessageBox.Show("No se trata de un número primo");

}

Hasta ahí. Le pasas por el código un número y te devuelve si es o no primo.

private void btnCalcular\_Click(object sender, EventArgs e)

{

string resultado = "";

double numero = 2;

int contador = 0;

while (contador<100)

{

if (esNumeroPrimo(numero)==true)

{

resultado = resultado + numero + ", " ; //Acá va agregando strings, la coma es la forma de separar los elementos.

contador++;

}

numero++;

}

MessageBox.Show(resultado);

}

private bool esNumeroPrimo (double Num )

{

bool esPrimo = true;

for (int i = 2; i < Num; i++)

{

if (Num % i == 0)

{

esPrimo = false;

}

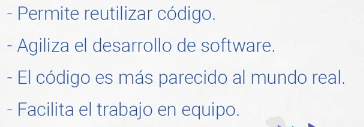
}

return esPrimo;

Ahí si te devuelve un mbox con los 100 primeros números primos.

Programación orientada a objetos

Es la heredera de la programación estructurada.



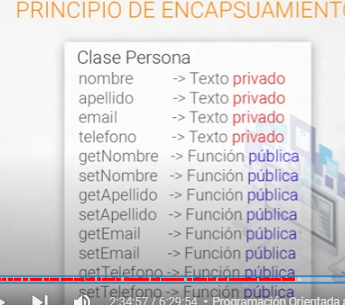
Aquí hay clases. Las cuales permiten guardar variables y funciones. A las variables, aquí se las llama propiedades y a sus funciones métodos.

Encapsulamiento

Privado. Solo se puede usar dentro de la misma clase.

Public. Dentro de otros lugares de código se puede acceder.

No es obligatorio hacerlo, pero es una buena práctica. Se debe ocultar el estado interno del objeto. Obligar a que toda interacción se realice con los métodos del objeto.



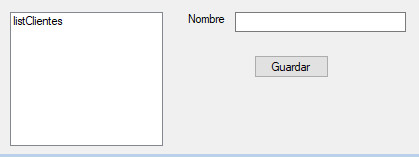
Y así, si queremos acceder al nombre, no deberíamos poner el nombre de nombre, sino crear un método público que nos permita acceder al nombre y otro para modificarlo: get y set.

Ejercicio Gestión Clientes

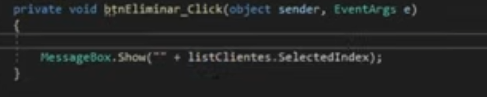
GestionClientes ventanaGestionClientes = new GestionClientes();.

Ese new GestionClientes () es el nombre de la clase, si o si debe coincidir. VentanaGestionClientes sería el objeto.

Tira un listbox y luego hace doble click sobre la parte gris del diseño.



Luego agrega el botón Eliminar:



Con eso de ahí, al tocar eliminar, se abre un mbox con el número del índice del nombre que se escribió..

private void button1\_Click(object sender, EventArgs e)

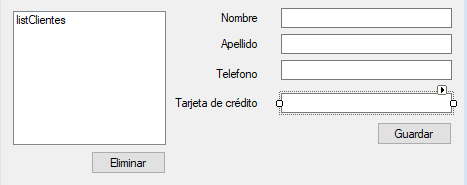
{

int indice = listClientes.SelectedIndex;

listClientes.Items.RemoveAt(indice);

}

Con eso, se toca en uno de los nombres de la lista (y ahí se captura el índice) y en la línea de abajo se lo elimina.



Para poder incorporar el resto de los ítems, hay que incorporar clases. En la clase, mete todas las variables.

Se debe aplicar el encapsulamiento:Por eso hace los 4 strings de los ítems, privados, para que no se pueda acceder a ellos desde afuera.

Cliente cliente = new Cliente(); // es la variable cliente, del tipo Cliente. y se crea un nuevo objeto Cliente.

Si en otra clase pusiera eso. Poniendo cliente. podría acceder a sus variables, si fuesen públicas.

Por eso se hace el getter y el setter.

Getter/Setter

private string nombre;

private string apellido;

private string telefono;

private string tarjetaDeCredito;

public string Nombre //Con mayúscula

{

get { return nombre; }

set { nombre = value; } //El value lo genera automáticamente el set. Es lo que pone el usuario.

}

Dice después que en las nuevas versiones de c#, con eso de abajo ya alcanza.



Podría, desprolijamente, quedar así:

{

private string nombre;

//Si se lo hace de la segunda manera, en una sola línea,estas 3 variables siguientes hay que borrarlas, ya que no son necesarias

private string apellido;

private string telefono;

private string tarjetaDeCredito;

public string Nombre //Con mayúscula

{

get { return nombre; }

set { nombre = value; } //El value lo genera automáticamente el set. Es lo que pone el usuario.

}

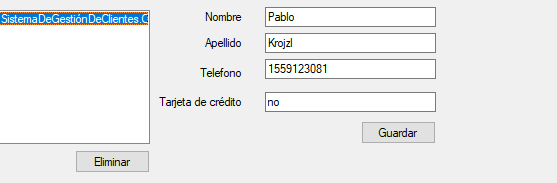
public string Apellido { get; set; }

public string Telefono { get; set; }

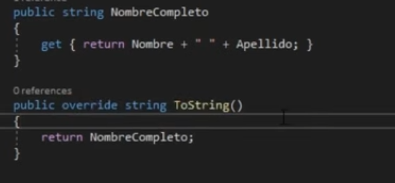
public string TarjetaDeCredito { get; set; }

Ahora. Cuando vas a otra clase y tocas cliente. van a aparecer las 4 opciones creadas con el get. Es decir, lo mismo pero en mayúsculas.

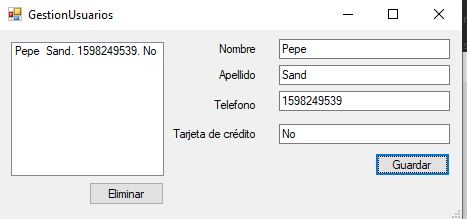
Por defecto todas las clases tienen una función “tostring” que se ejecuta.



Lo que está en azul es lo dicho, cualquier cosa.



Para que no pase eso, hay que usar override. Para que sobreescriba el tostring por default.



public string Nombre //Con mayúscula

{

get { return nombre; }

set { nombre = value; } //El value lo genera automáticamente el set. Es lo que pone el usuario.

}

public string Apellido { get; set; }

public string Telefono { get; set; }

public string TarjetaDeCredito { get; set; }

public override string ToString()

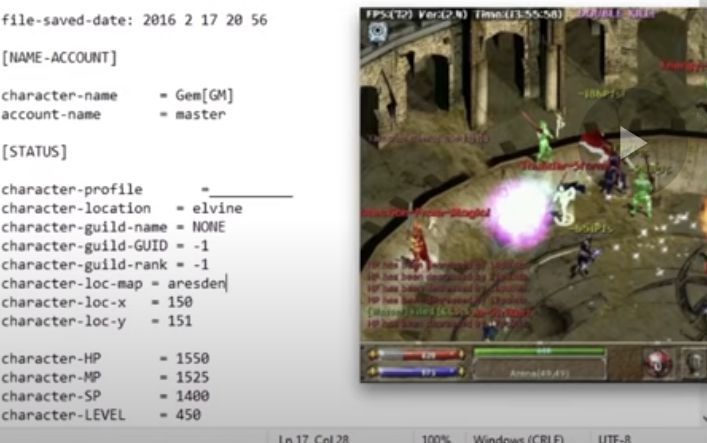
{

return nombre + " " + Apellido + ". " + Telefono + ". " + TarjetaDeCredito;

}

Yo lo agregué así, y quedó piola piola.

Base de datos



Antes las bases de datos eran blocs de notas. Cada personaje, en este caso, tenía el suyo. El procesamiento de info se hace un poco lento.

Aparecen sistemas de gestión de datos. También los datacenters.

Blockchain es una base de datos descentralizada. Varias computadores en bases de datos por todo el mundo. Antes, estaban en un solo lugar específico.

Hay bases relacionales y otras no relacionales.

Relacionales:

Se basa en la relación que tienen los datos entre sí.

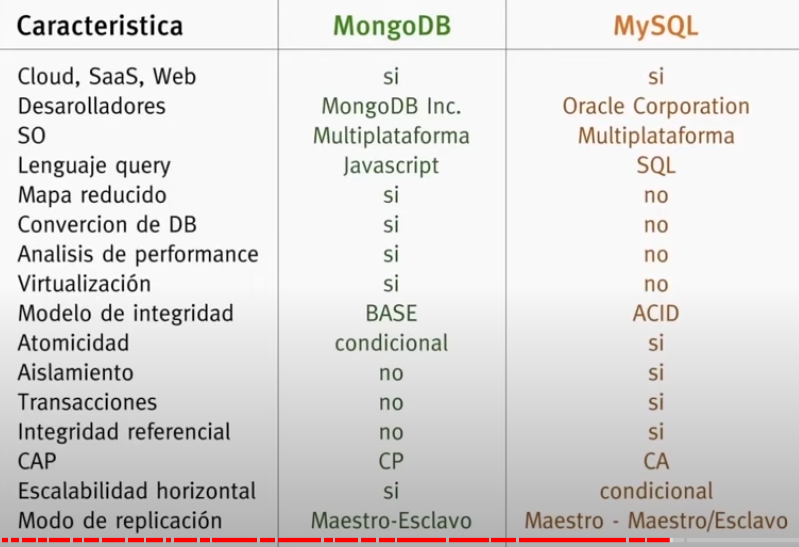


María DB es MySQL

No relacionales (o no SQL)

Son mas flexibles y fáciles de desarrollar.

MongoDB; CouchDB; FIREBASE; Hadoop; Cassandra; Voldemort.





El ID permite escribir mucho menos. Lo cual hace que el archivo pese menos, y que se procese mas rápido.

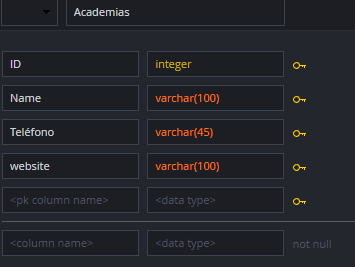


Lo mismo, Char es menos pesado que text.

Recomienda usar app.sqlDBM.com para hacer el diseño de base de datos.

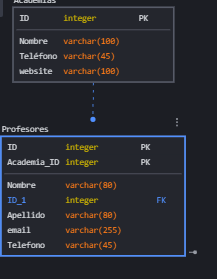
New table

PK es primary key. Ahí pone ID. A la derecha integer.



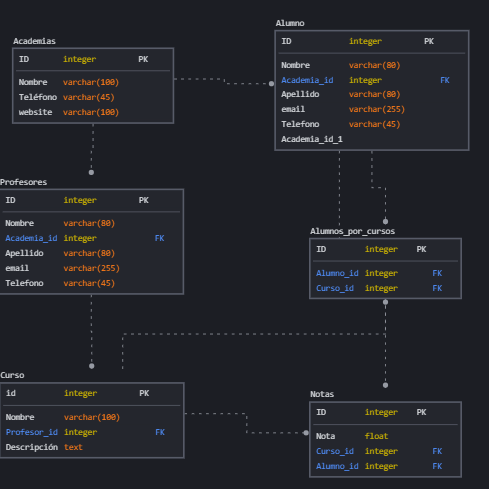
Luego, la tabla de profesores.

Hay una fila que es AcademiaID que se vincula con la tabla de academia. Se edita academia y se va al ID, ahí se establece Relationship y se pincha luego ACADEMIAID.



Se ve así. Se agrega ID. Luego se le puede cambiar el nombre. Le agrega FK foreign Key.

Tablas intermedias: Alumnos\_por\_cursos. Con alumnos\_id y cursos\_id



Esta página permite generar el código sql para utilizar esas tablas. Yendo a forward engineer. Generate SQL.

CREATE TABLE `Academias`

(

`ID` integer NOT NULL ,

`Nombre` varchar(100) NOT NULL ,

`Teléfono` varchar(45) NOT NULL ,

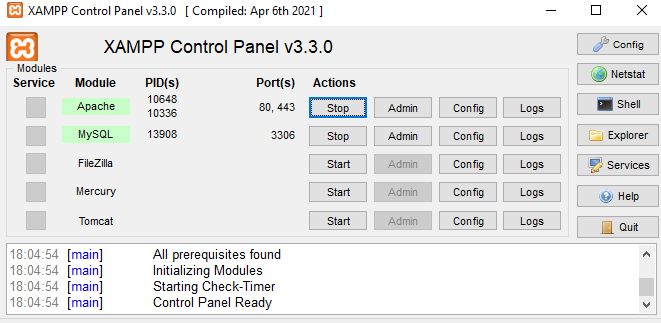
`website` varchar(100) NOT NULL ,

PRIMARY KEY (`ID`)

);  
De a una table por vez (porque es la versión gratuita).

XAMPP instalar. Con eso instala MySQL.

Con Apache (que viene con XAMPP) permite conectar a la base de datos de MySQL.



Primero se levanta esos dos APACHE y MySQL

En Chrome busca localhost/phpmyadmin

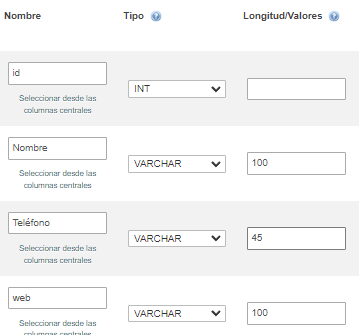
Podría ser: localhost/phpMyAdmin

Se entra a un servidor local. Este gestor permite crear base de datos, modificarlas, ejecutar código SQL.

New: Academias. Tipo UTF8bin

Las tablas se crean en un orden especial.

A\_I es autoincrement.



Al resto de los botonoes mucha bola no le da.

En structure de la tabla a la que se le hace la relación, hay una pestaña que dice relation view. En constrain name se le pone nombre a esa relación. Columna que se utiliza Academia\_id/academias/academias/id

Se podría hacer todo con SQL.

Luego se puede exportar, en SQL, normalmente. Eso luego se puede abrir en VS.