El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

CICLO III:

Desarrollo de software

MisionTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

Sesión 12:

Desarrollo Software

Diseño de bases de datos

Integración de bases de datos

Mision TIC 2022

Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Definir y diseñar una base de datos relacional
2. Construir una base de datos en SQLite con varias tablas
3. Manipular la gestión de información en la base de datos construida.
4. Explicar y aplicar el concepto de conexión a una base de datos relacional.
5. Construir una conexión entre la aplicación web y la base de datos.

Diseño de Base de Datos

Para recordar:

- “Una base de datos es una colección de información organizada de forma que un software pueda seleccionar rápidamente los fragmentos de datos que necesite”.
- “Una base de datos es un sistema de archivos electrónico”.
- “Las bases de datos tradicionales se organizan por campos, registros y archivos”.
- “Una base de datos es un conjunto de datos almacenados y organizados con el fin de facilitar su acceso y recuperación mediante el uso de un ordenador”.
- “Una base de datos es una serie de datos organizados y relacionados entre sí los cuales son recolectados y explotados por un software o sistema de información”.

Diseño de Base de Datos
Modelos

Bases de Datos relacionales

- Colección de datos organizados en un conjunto de tablas formalmente definidas, desde las cuales se puede acceder a los datos o ingresarlos de muchas formas diferentes sin necesidad de reorganizar las tablas de la base de datos. El Lenguaje de Consultas Estructuradas (SQL) fue diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
- Las bases de datos relacionales se fundamentan en la organización de la información en pequeñas secciones unidas o integradas mediante unos identificadores.
- Como características principales se encuentra que:
  - Son robustas, debido a la gran capacidad de almacenamiento.
  - Son menos vulnerables ante fallas.

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Diseño de Base de Datos
Modelos

Elementos de Bases de Datos relacionales (SQL)

- Relaciones base y derivadas.
- Restricciones y dominios.
- Clave única.
- Clave primaria.
- Clave foránea.

Mision TIC 2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Diseño de Base de Datos
Modelos

Bases de datos no relacionales (NoSQL)

- Diseñadas especialmente para modelos de datos específicos, poseen esquemas flexibles para crear aplicaciones modernas.
- Son fáciles de desarrollar, por lo que son altamente reconocidas, tanto en funcionalidad como en rendimiento a escala.
- Las bases de datos no relacionales (NoSQL), no tienen un identificador que se pueda usar para relacionar un conjunto de datos y otros. La información se organiza normalmente mediante documentos y es muy útil cuando no se necesita de un esquema exacto de lo que se va a almacenar.

MisionTIC2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Diseño de Base de Datos
Modelos

Características de Bases de datos no relacionales

- Consistencia Eventual.
- Estructura distribuida.
- Ausencia de esquema en los registros de datos.
- Escalabilidad horizontal sencilla.
- Es útil cuando el volumen de los datos crece rápidamente.

MisionTIC2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Diseño de Base de Datos
Modelos

Tipos de Bases de datos no Relacionales

- **Clave-valor:** Modelo de base de datos no relacional muy similar a los diccionarios. Utiliza el método de clave-valor para almacenar datos como un conjunto de pares donde la clave representa un valor único.
- **Documentos:** Consiste en archivos donde se puede almacenar de forma eficiente e intuitiva los datos facilitando el almacenamiento y consulta de estos.
- **Gráficos:** También son bases de datos que permiten expresar las relaciones y diferencias entre diferentes datos y tipos de datos.
- Existen otros tipos como en-memoria y búsqueda que son no son tan comunes.

MisionTIC2022

El futuro digital es de todos MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Diseño de Base de Datos
Modelos

Cuando utilizar SQL o NOSQL

SQL

- Cuando el crecimiento del volumen de los datos es mínimo o se evidencia crecimiento alguno.
- Cuando solo es necesario un solo servidor para asumir las necesidades del proceso.
- Cuando los usuarios solo utilizan el sistemas lo estrictamente necesario.

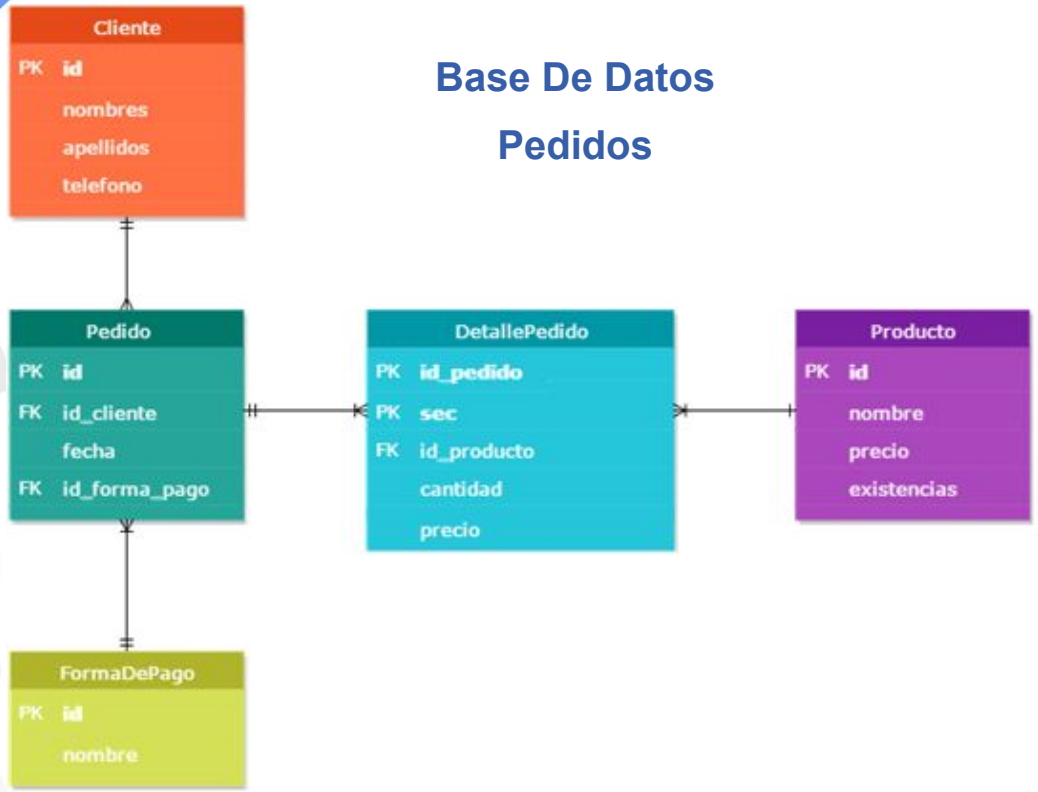
NOSQL

- Cuando hay crecimiento rápido en algunos momentos específicos.
- Cuando no se pueden prever las necesidades del proceso.
- Cuando hay picos de uso del sistema por parte del cliente en muchas ocasiones.

Mision TIC 2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación



Base De Datos

Pedidos

Diseño de Base de Datos con múltiples tablas

Mision TIC 2022

Diseño de Base de Datos con múltiples tablas

Creación de tablas

```
create table Cliente(
    id integer primary key,
    nombre text,
    apellidos text,
    teléfono text
);

create table FormaDePago(
    id integer primary key,
    nombre text
);
```

Diseño de Base de Datos con múltiples tablas

Creación de tablas

```
create table Pedido(
    id integer primary key,
    fecha text,
        id_cliente integer references Cliente(id),
        id_forma_pago integer references FormaDePago(id)
);
create table Producto(
    id integer primary key,
    nombre text,
    precio real,
    existencia integer
);
```

Diseño de Base de Datos con múltiples tablas

Creación de tablas

```
create table DetallePedido(
    sec integer primary key,
        cantidad integer,
        precio real,
        id_pedido integer references Pedido(id),
        id_producto integer references Producto(id)
);
```

Restricción "foreign key" para establecer los campos “id_pedido" e “id_producto” como claves externas que hagan referencia a los campos “id" de “Pedido” y “Producto“ respectivamente.

Diseño de Base de Datos con múltiples tablas

Inserción de datos

```
insert into Clientes (id, nombre, apellidos, telefono)
  values('109899', 'Juan', 'Pérez Gómez', 3023345664);
insert into Clientes (id, nombre, apellidos, telefono)
  values('104695', 'José', 'López Ariza', 3123675774);

insert into FormaDePago (id, nombre)
  values(1, 'Tarjeta de crédito');
insert into FormaDePago (id, nombre)
  values(2, 'Tarjeta dédito');
insert into FormaDePago (id, nombre)
  values(3, 'Efectivo');
```

Diseño de Base de Datos con múltiples tablas

Inserción de datos

```
insert into Producto (id, nombre, precio, existencia)
  values('99', 'Mouse', 25000, 64);
insert into Clientes (id, nombre, precio, existencia)
  values('104', 'Teclado', '48000', 77);
insert into Clientes (id, nombre, precio, existencia)
  values('47', 'Monitor', '368000', 23);

insert into Pedido (id, id_cliente, fecha, id_forma_pago)
  values(1008, 109899, '6/12/20', 2);
insert into Pedido (id, id_cliente, fecha, id_forma_pago)
  values(1009, 109899, '23/11/20', 3);
insert into Pedido (id, id_cliente, fecha, id_forma_pago)
  values(1018, 104695, '1/12/20', 2);
```

Diseño de Base de Datos con múltiples tablas

Inserción de datos

```
insert into DetallePedido (sec, id_pedido, id_producto, precio, cantidad)
  values(1, 1008, 104, 48000, 2);
insert into DetallePedido (sec, id_pedido, id_producto, precio, cantidad)
  values(2, 1018, 104, 48000, 4);
insert into DetallePedido (sec, id_pedido, id_producto, precio, cantidad)
  values(3, 1009, 47, 368000, 1);
```

Qué pasaría si se realizan las siguientes operaciones?

1. insert into DetallePedido (sec, id_pedido, id_producto, precio, cantidad)
   values(3, 1019, 47, 368000, 1);
2. delete from Pedido where id=1008;
3. update Pedido set id=1000 where codigo=1008;

Diseño de Base de Datos con múltiples tablas

Al Intentar ingresar una fila en la tabla DetallePedido con “id_pedido" que no existe en la tabla “Pedido“, se produce un error ya que no se encuentra el código.

La restricción "foreign key" actúa en eliminaciones y actualizaciones. Si se intenta eliminar un registro o modificar un valor de clave primaria de una tabla donde una clave foránea hace referencia a dicho registro, SQLite no lo permite.

Desarrollo Web - Frontend y Backend Flask – Base de Datos

- El cambio más importante en la introducción de la web 2.0 es la interacción del usuario con el aplicativo web, realizando consultas y/o manipulando datos. Permitiendo así que la web sea el medio para un trabajo colaborativo entre los usuarios, dejando atrás la web 1.0 en donde las páginas web eran solo contenedores de información.
- Los datos manipulados por el usuario son almacenados en una base de datos centralizada, en donde a través de un aplicativo web (frontend) los datos son capturados y enviados al servidor (backend) para su procesamiento y almacenamiento.

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Actores principales

- Archivo principal de la aplicación de Python.
- Archivo donde se establecen los diferentes formularios mediante Flask a nivel de servidor.
- Vistas html necesarias para visualizar e ingresar los datos.

Desarrollo Web - Frontend y Backend
Flask – Base de Datos

- Antes de utilizar SQLite3 en Python, se debe importar el módulo sqlite3, el cual permitirá la conexión y la manipulación con la base de datos a través de las sentencias sql.
- Para importar el módulo sqlite3 en el archivo principal del aplicativo web de extensión .py se deben incluir las líneas de la derecha.
- El módulo de error permite procesar los errores que pueda generar cualquier sentencia que utilice sqlite3.

Archivo principal aplicativo web

```
#importar el modulo sqlite3

import sqlite3

#importar modulo de error de sqlite3

from sqlite3 import Error
```

Desarrollo Web - Frontend y Backend
Flask – Base de Datos

- La función detallada a la derecha, realiza la conexión con la base de datos llamada *mydatabase.db.*
- *def sql_connection* devuelve un objeto de tipo conexión que permite la conexión con la base de datos sqlite3.
- La sentencia *con = sqlite3.connect('mydatabase.db')* está contenida en un bloque try con el fin de capturar los posibles errores que se puedan generar.

Archivo principal aplicativo web

```
def sql_connection():
    try:
        con = sqlite3.connect('mydatabase.db')
        return con;
    except Error:
        print(Error)
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

- En el archivo principal se definirán también 4 funciones las cuales realizarán las operaciones en la base de datos (inserción, consulta, edición, y eliminación, CRUD por sus siglas en inglés).
- La siguiente imagen muestra la función de inserción, llamada sql_insert_productos.

```
def sql_insert_producto(codigo, nombre, cantidad):
    strsql =
    "insert into productos (codigo, nombre, cantidad) values('"+codigo+"', '"+nombre+"', "+cantidad+");"
    con = sql_connection()
    cursorObj = con.cursor()
    cursorObj.execute(strsql)
    con.commit()
    con.close()
```

Desarrollo Web - Frontend y Backend
Flask – Base de Datos

```
#crear una variable que contenga la cadena sql que inserta los datos recibidos por parámetro
strsql = "insert into productos (codigo, nombre, cantidad) values('"+codigo+"', '"+nombre+"', "+cantidad+");“
#el objeto con obtiene la conexión con la base de datos llamando al método creado anteriormente
con = sql_connection()
#es necesario un objeto cursor, el cual se obtiene de la variable de conexión, para ejecutar las sentencias sql
cursorObj = con.cursor()
#ejecutar la sentencia sql enviada por parámetro
cursorObj.execute(strsql)
#actualizar los cambios realizados a la base de datos
con.commit()
#cerrar la conexión
con.close()
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

- La función sql_select_productos indicada a la derecha, ejecuta el query que selecciona todas las filas de la tabla productos y las devuelve con la variable productos.
- El método *fetchall,* ejecutado a través del objeto cursor, devuelve una lista con todas las filas de la tabla, cada elemento de la lista tiene el número de ítems correspondiente a la cantidad de columnas consultadas.

```
def sql_select_productos():
    strsql = "select * from productos;"
    con = sql_connection()
    cursorObj = con.cursor()
    cursorObj.execute(strsql)
    productos = cursorObj.fetchall()
    return productos
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

La función sql_edit_producto ejecuta el query que edita un registro en la tabla productos dado un identificador.

```
def sql_edit_producto(id, codigo, nombre, cantidad):
    strsql = "update productos set codigo = '"+codigo+"', nombre = '"+nombre+"', cantidad = "+cantidad+" where id = "+id+";"
    con = sql_connection()
    cursorObj = con.cursor()
    cursorObj.execute(strsql)
    con.commit()
    con.close()
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

La función sql_delete_producto, ejecuta el query que elimina un registro en la tabla productos dado un identificador.

```
def sql_delete_producto(id):
    strsql = "delete from productos where id = "+id+";"
    con = sql_connection()
    cursorObj = con.cursor()
    cursorObj.execute(strsql)
    con.commit()
    con.close()
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Es necesario de igual forma, crear las rutas que serán accedidas a través de las vistas html.

- Ruta para visualizar la lista de productos
- Ruta para agregar productos
- Ruta para editar productos
- Ruta para eliminar productos

En los métodos creados a continuación se utiliza el objeto request, este permite acceder a los valores de las variables enviadas por las vistas html. Antes de utilizarlo se debe importar la clase.

from flask import request

Desarrollo Web - Frontend y Backend Flask – Base de Datos

La ruta '/productos' mediante la función *productos,* realiza un llamado a la función *sql_select_productos* y envía el resultado a la vista productos.html para su visualización.

```
app.route('/productos')
def productos():
  productos = sql_select_productos()
  return render_template('productos.html', productos = productos)
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

A continuación se define la ruta '/nuevo', la cual se configura para ser accedida mediante los métodos get y post.

```
@app.route('/nuevo', methods=['GET', 'POST'])
def nuevo():
   if  request.method == "GET": #Si la ruta es accedida a través del método GET entonces
     form = Producto() #Crea un nuevo formulario de tipo producto
     return render_template('nuevo.html', form=form) #redirecciona vista nuevo.html enviando la variable form
   if  request.method == "POST": #Si la ruta es accedida a través del método POST entonces
     cod = request.form["codigo"] #asigna variable cod con valor enviado desde formulario  en la vista html
     nom = request.form["nombre"] #asigna variable nom con valor enviado desde formulario en la vista html
     cant = request.form["cantidad"] #asigna vble cant con valor enviado desde formulario en la vista html
     sql_insert_producto(cod, nom, cant) #llamado de la función para insertar el nuevo producto
     return "OK“
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Se define la ruta '/edit', la cual se configura para ser accedida mediante el método get.

```
@app.route('/edit', methods=['GET'])
def editar_producto():
    id = request.args.get('id') #captura de la variable id enviada a través de la URL
    codigo = request.args.get('codigo') #captura de la vble código enviada a través de la URL
    nombre = request.args.get('nombre') #captura de la vble nombre enviada a través de la URL
    cantidad = request.args.get('cantidad') #captura de la vble cantidad enviada a través de la URL
    sql_edit_producto(id, codigo, nombre, cantidad) #llamado de la función de edición de la base de datos
    return "OK"
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Luego, se define la ruta ‘/delete’, la cual se configura para ser accedida mediante el método get.

```
@app.route('/delete', methods=['GET'])
def borrar_producto():
    id = request.args.get('id') #captura de la variable id enviada a través de la URL
    sql_delete_producto(id) #llamado a la función de borrado de la base de datos
    return "OK“
```

Una vez creadas las funciones y rutas necesarias en el archivo principal del aplicativo web, se procede a crear el formulario en el que se digita los datos para su ingreso en la base de datos.

Desarrollo Web - Frontend y Backend Flask – Base de Datos

El segmento de código indicado a la derecha define un formulario mediante una clase llamada *Producto* en el archivo forms.py. Cada clase definirá un formulario en la aplicación web.

Es necesario importar las clases StringField, IntegerField, SubmitField de wtforms ya que estos serán los objetos necesarios en la vista html.

- StringField: campo de texto
- IntegerField: campo de tipo entero
- SubmitField: botón de tipo submit

```
from flask_wtf import FlaskForm

from wtforms import StringField, IntegerField, SubmitField

from wtforms.validators import DataRequired


class Producto(FlaskForm):

    codigo = StringField('Codigo',
validators=[DataRequired()])

    nombre = StringField('Nombre', validators=[DataRequired()])

    cantidad = IntegerField('Cantidad',
validators=[DataRequired()])

    enviar = SubmitField('Agregar Producto')
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Cada clase representa un formulario (a nivel de backend). En la clase Producto se crean los campos requeridos para este formulario. Si se desea otro formulario con otras características, se deberá crear otra clase en el mismo archivo forms.py

Con la sentencia *from wtforms.validators import DataRequired,* se importan las clases necesarias para validar que los campos sean requeridos, es decir que no se encuentren vacíos.

```
codigo = StringField('Codigo', validators=[DataRequired()])
```

- codigo: nombre de la variable
- StringField: tipo de campo que se mostrará en la vista
- Codigo: nombre del campo
- validators: vector que indica las validaciones que se realizarán en este campo, para el caso solo se realiza la validación DataRequired()

Desarrollo Web - Frontend y Backend Flask – Base de Datos

- En la vista, es decir, el archivo de extensión html, es donde se capturan o leen los datos que serán almacenados posteriormente en la base de datos.
- En la sección de la derecha se tiene el ejemplo de la implementación del formulario correspondiente a la clase *Producto*.

```
<form action="/nuevo" method="post">
        {{form.codigo.label}} {{form.codigo}}
        <br>
        {{form.nombre.label}} {{form.nombre}}
        <br>
        {{form.cantidad.label}} {{form.cantidad}}
        <br>
        {{form.enviar}}
        {{form.hidden_tag()}}
</form>
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

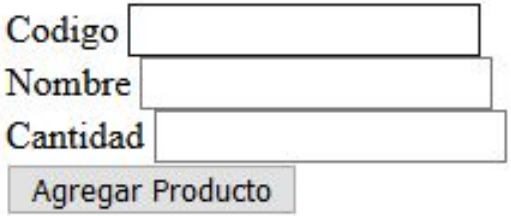
En el formulario html se establecen los parámetros action y method:

- action: indica la acción que ejecutará el formulario cuando se presione el botón enviar, el cual debe ser de tipo submit.
- method: establece el método por el cual se enviaran los datos al servidor, para el caso es *post*.

De igual manera se crean los labels e inputs para los campos definidos en el backend del formulario (archivo de extensión .py).

Desarrollo Web - Frontend y Backend Flask – Base de Datos

El siguiente es el resultado de acceder a la ruta /nuevo en la barra de direcciones:



Al presionar Agregar Producto, se envían los datos a través del método POST y es recibido por la ruta /nuevo, ruta definida en el campo action del formulario. Recordar que el código y nombre son campos de texto y la cantidad un campo numérico.

Desarrollo Web - Frontend y Backend Flask – Base de Datos

La página en la cual se visualizarán los datos de la tabla se llama productos.html. En esta vista se crea una tabla de 4 columnas, id, código, nombre, cantidad. En cuanto a las filas estas dependen de la cantidad de elementos que tenga el objeto *productos* recibido desde el back end luego de acceder a la ruta '/productos‘.

Todo aquello que se encuentre dentro de los símbolos {% %} corresponderá a código Python.

{% for pro in productos %}: ciclo para recorrer la lista de productos, cada producto se accede mediante la variable pro

{% endfor %}: fin del ciclo

pro[0]: primer ítem del objeto pro (corresponde al id)

```
<table id="tabel" border="1">
  <tbody>
    <tr>
      <td width="2">Id</td>
      <td width="2">Codigo</td>
      <td width="2">Nombre</td>
      <td width="2">Cantidad</td>
    </tr>
    {% for pro in productos %}
    <tr>
      <td width="2">{% print(pro[0]) %}</td>
      <td width="2">{% print(pro[1]) %}</td>
      <td width="2">{% print(pro[2]) %}</td>
      <td width="2">{% print(pro[3]) %}</td>
    </tr>
    {% endfor %}
  </tbody>
```

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Resultado de acceder a la ruta /productos en la barra de direcciones

Id	Codigo	Nombre	Cantidad
2	0002	Licuadora 10 velocidades	12

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Para editar un producto, es necesario enviar los valores de las variables a través de la barra de direcciones, las variables necesarias son las establecidas en la ruta '/edit' establecida anteriormente.

Ejemplo de la ruta en la barra de direcciones:

http://localhost:5000/edit?id=1&nombre=Secadora&cantidad=7&codigo=002

Al colocar el símbolo '?' en la dirección, se está indicando que a continuación se enviarán variables con sus respectivos valores, el símbolo & separa las variables enviadas.

Para el ejemplo se envían 4 variables, id con valor de 1, nombre con valor de Secadora, cantidad con valor de 7, y código con valor de 002.

Desarrollo Web - Frontend y Backend Flask – Base de Datos

Para eliminar un producto, es necesario enviar el valor del id a través de la barra de direcciones, la ruta '/delete' establecida anteriormente es la encargada de eliminar un producto dado un id.

Ejemplo de la ruta en la barra de direcciones:

http://localhost:5000/delete?id=1

Para el ejemplo se eliminará el producto de código 1.

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

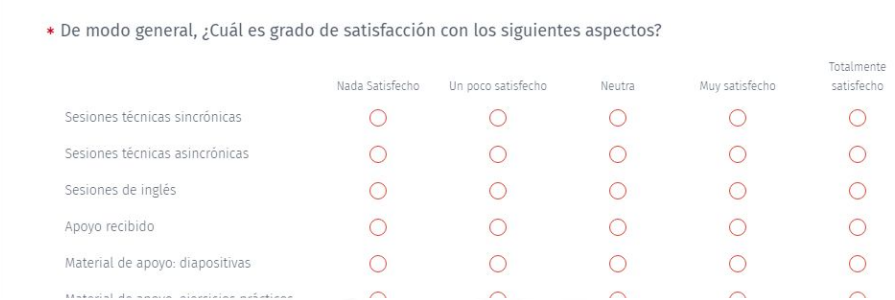
Ejercicios de práctica

Mision TIC 2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Seguimiento Habilidades Digitales en Programación



* De modo general, ¿Cuál es grado de satisfacción con los siguientes aspectos?

	Nada Satisfecho	Un poco satisfecho	Neutra	Muy satisfecho	Totalmente satisfecho
Sesiones técnicas sincrónicas	○	○	○	○	○
Sesiones técnicas asincrónicas	○	○	○	○	○
Sesiones de inglés	○	○	○	○	○
Apoyo recibido	○	○	○	○	○
Material de apoyo: diapositivas	○	○	○	○	○

Completa la siguiente encuesta para darnos retroalimentación sobre esta semana ▼▼▼

https://www.questionpro.com/t/ALw8TZlxOJ

MisionTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

¡GRACIAS

POR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!

Mision TIC 2022