**Prueba Teórica Ingeniero de Desarrollo II, III**

**Nombre aspirante: Victor Danilo Castañeda Pinzon**

**Fecha 11/29/2021 Calificación: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PARTE 1 – CONCEPTOS GENERALES**

* De ejemplos de al menos 2 requerimientos no funcionales que siempre tomaría en cuenta

| **Modulariadad**: En cada uno de los desarrollos que hago intento que las funcionalidades y los componentes del software se puedan dividir en pequeños modulos que puedan ser reemplazados facilmente sin alterar la totalidad del còdigo.  **Escabilidad**: Siempre que hago cualquier tipo de desarrollo, lo hago pensando que un proceso se pueda replicar para un solo movimiento o para millones de movimientos y que puedan operar sobre diferentes escenarios. Por ejemplo, hago un desarrollo para transacciones bancarias, hago que ese desarrollo se aplique para una solo transacción o para millones de transacciones que pueden ser diferentes, por ejemplo, que las transacciones tengan campos faltantes, que la información no llegue siempre de la misma manera y contemplando todos estos escenarios. |
| --- |

* ¿Cuál es la diferencia entre un Servicio Web SOAP y un servicio REST?

| Los servicios WEB son la manera de conectar o comunicar dos componentes de software, por ejemplo, un front-end escrito en JavaScript con un backend construido en Python. Estas formas de conectar estos servicios pueden ser construidos utilizando diferentes arquitecturas, en esta caso las arquitecturas presentadas son SOAP y REST. Estas dos arquitecturas funcionan bajo el mismo protocolo de comunicación HTTP(Hipertext transfer protocol). Se diferencian principalmente en que la arquitectura SOAP esta basada en el lenguaje XML mientras que REST puede soportar diferentes lenguajes , pero es comun y casi que necesario utilizar el lenguaje JSON. Estos dos lenguajes son los que se utilizan para enviar peticiones y recibir respuestas de parte del servicio web. |
| --- |

* ¿Qué es un servidor de aplicaciones ( IIS ) ?

| Internet Information Services, es lo que se conoce como un servidor web especialmente utiliza para windows, haciendo que una maquina local (Puede ser el computador personal) se convierta en servidor en donde se pueden ejecutar multiples servicios o se pueda utilizar de diferentes maneras. Una de las formas más comúnes para utilizarlo es para alojar una apliación web y que esta pueda ser ejecutada y vista a través de un navegador web. |
| --- |

* ¿Qué es *Inversion of Control*? de ejemplo de algún framework

| Es un forma de programación que va en contra de la manera tradicional de programar, pues utilizando este tipo de programación un framework toma controla el flujo de funcionamiento del programa, en python no se utiliza muy comunmente, es una manera de programar que es màs utilizada cuando se trabaja con Java o C#. Cuando se trabaja con .NET se utiliza implicitamente esta teoria. |
| --- |

* ¿Qué es WCF?

| Se refiere a un servicio, aplicación o framework que permite desarrollar aplicaciones orientadas a servicios, estas aplicaciones se comunican por medio de mensajes utilizando la lógica de un arquitectura cliente servidor, de estos servicios hay diferentes tipos. |
| --- |

* ¿Qué es un binding en WCF, nombre algunos de los más utilizados?

| En un servicio de aplicaciones orientada a servicios un binding es un elemento importante de los endpoints de comunicación. Un binding establece los parametros necesarios para que la comunicación se pueda establecer. Comunmente los bindings establecen información de los protocolos de comunicación, la forma de codificar la información y bajo que protocolo se pueden transmitir estas comunicaciones. El más utilizado es el BasicHttpBinding. |
| --- |

* ¿Qué es un Branch, Merge y Commit? ¿en qué se usan en común?

| Este conjunto de elementos son conceptos de un servicio de control de versiones, uno de los más utilizados es GIT. A un folder que es administrado bajo un sistema de control de versiones se le conoce como un repositorio, este repositorio puede estar localmente en la maquina del editor o remotamente que es donde se juntan los cambios de todos los editores. Un repositorio gestionado por un sistema de control de versiones funciona como un arbol, en donde hay un punto de partida que es el comienzo del seguimiento y en la lógica del árbol ese inicio es la la raiz. Este árbol puede tener diferentes ramas, por ejemplo la rama de desarrollo o la rama de producción, alternativamente pueden existir otras ramas, cuando se trabaja con ramas estas se pueden fusionar para completar por ejemplo el un requerimiento, esto de fusionar dos ramas es lo que se conoce como un merge, que es unir dos ramas para que los cambios de dos ramas ahora se vean en una sola. Por otro lado, el proceso de control de versiones se hace mediante etiquetas, por ejemplo un editor esta trabajando en una caracteristica de un software, lo común es que se cree una nueva rama para iniciar este desarrallo y se empiece con el desarrollo, por ejemplo el primer día el editor cambia dos archivos del proyecto entonces por medio de una etiqueta que se le conoce como commit, se deja la evidencia de que archivos fueron modificados y que parte de los archivos fueron modificados, lo que hace pensar que una rama pueda estar compuestas de muchos commits. Una vez terminado el desarrollo el editor lo que haria seria unir su rama con la rama principal para aplicar el concepto de merge. |
| --- |

* En C# ¿Cuál es la diferencia entre una clase base y una interfaz?

| Una interfaz define los comportamientos y métodos que debe tener un objeto para utilizar un componente del software, por lo que se infiere que una interfaz en un intermediario que no contiene lógica de programación, mientras que las clases base son las plantillas para construir objetos y que tienen logica de programación, una clase base puede implementar una interfaz para que se valide que tipo de objeto va a utilizar la clase base. |
| --- |

* Se tiene una variable de tipo string que contiene una fecha y hora en formato desconocido, ¿cómo identificaría el formato? **(\*\* Python)**

| El modulo datetime de python tiene implementado un método para transformar variables de tipo string a variables de tipo datetime. Para utilizar este método es necesario conocer de algunas clausulas o expresiones regulares para saber que parte y como extraerla desde la variable string y en que formato representar el objeto datetime, estas expresiones regulares pueden ser: (%d, para identificar el dia de una fecha, %m, para indetificar el mes de una fecha, %y para identificar el año, %H para encontrar la hora en formato 24 horas, %M para encontrar los minutos de una hora y %S para encontrar los segundos de una hora). Visto desde el codigo esto funcionaria de la siguiente manera, primero se debe importar el modulo datetime desde la libreria datetime, segundo conocer la variable que se quiere estudiar, y por ultimo realizar la transformación del objeto string a objeto datetime.  ################################################################################  from datetime import datetime  str = '18/09/19 01:55:19  date\_time\_obj = datetime. strptime(date\_time\_str, '%d/%m/%y %H:%M:%S')  ################################################################################# |
| --- |

* Explique la diferencia de las clausulas *is* y *as* en C#

| En algunos lenguajes de programación existen clausulas para identificar si una variable es correspondiente a un tipo, por ejemplo en C# se puede evaluar si una variable pertenece a un tipo en especifico utilizando la clausula "is" o clausula "as". |
| --- |

**PARTE 2 – CONCEPTOS APLICADOS**

De un ejemplo de código, en C++ o C#, ejemplificando los siguientes Conceptos:

* HERENCIA

| ################################  class Father():  def \_\_init\_\_(self, a, b):  self.a = a  self.b = b    def speak(self) - > str:  return f'Father speak {self.a}'  class Child(Father):  pass  ################################## |
| --- |
| ////////////////////////////////////////  class Father  {  public Father(int a, string b)  {  Console.Write(a);  }  }  class Child: Father  {  public void Greet()  {  Console.WriteLine(b);  }  }  ///////////////////////////////////////////// |

* POLIMORFISMO

| ######################################  class Father():  def \_\_init\_\_(self, a):  self.a = a    def greeting(self):  print("Father greeting")  class Child(Father):  def \_\_init\_\_(self):  self.child\_var = 1  def greeting(self):  print("Child greeting")  ######################################## |
| --- |
| /////////////////////////////////////////////////////  class Father  {  public Father(int a)  {  Console.WriteLine("");  }  public virtual void Greet()  {  Console.WriteLine("Father greeting");  }  }  class Child : Father  {    public Child(int b)  {  Console.WriteLine("");  }  public override void Greet()  {  Console.WriteLine("Child greeting");  }  }  //////////////////////////////////////////////////////////// |

* ENCAPSULAMIENTO

| ############################################################  class MyClass:  nv = "normal\_var"  \_\_nb = "private\_var"  def \_\_private\_method(self):  pass  def normal\_method(self):  pass  ############################################################# |
| --- |
| ////////////////////////////////////////////////////////////////  private class myClass  {  private int b = 0;  private void greet()  {  Console.WriteLine("Father greeting");  }  }  //////////////////////////////////////////////////////////////// |

* SOBRECARGA DE METODOS

| ######################################  class Father():  def \_\_init\_\_(self, a):  self.a = a    def greeting(self):  print("Father greeting")  class Child(Father):  def \_\_init\_\_(self):  self.child\_var = 1  def greeting(self):  print("Child greeting")  ######################################## |
| --- |
| /////////////////////////////////////////////////////  public class Father  {  public Father(int a)  {  Console.WriteLine("");  }  public virtual void Greet()  {  Console.WriteLine("Father greeting");  }  }  public class Child : Father  {    public Child(int b)  {  Console.WriteLine("");  }  public override void Greet()  {  Console.WriteLine("Child greeting");  }  }  //////////////////////////////////////////////////////////// |

* CONSTRUCTOR

| ##############################  class MyClass():  def \_\_init\_\_(self, a, b):  self.a = a  self.b = b  ############################### |
| --- |
| /////////////////////////////////////////  public class MyClass  {  public Father(int a, string b)  {  Console.Write(a);  }// end constructor  }//end MyClass  ////////////////////////////////////////// |

* GENERICS

| ######################################################  from typing import TypeVar, Generic  T = TypeVar('T')  class MyClass(Generic[T]):  def \_\_init\_\_(self) -> None:  self.items: List[T] = []  def push(self, item: T) -> None:  self.items.append(item)  def pop(self) -> T:  return self.items.pop()  def empty(self) -> bool:  return not self.items  ################################################# |
| --- |
| using System;  public class MyClass<T> {    private T data;    public T value  {    get  {  return this.data;  }  set  {  this.data = value;  }  }  } |

**PARTE 3 - SQL**

**Con base en la siguiente tabla:**



**Realice un script con una consulta que permita conocer la siguiente información:**

* Que devuelva las personas que no tienen segundo nombre, pero si tienen segundo apellido.

| SELECT \*  FROM Personas AS P  WHERE NOT EXIST P.SegundoNombre AND EXIST P.SegundoApellido; |
| --- |

2. Consulta de Personas cuyo padre se llama Pedro.

| SELECT \*  FROM Personas P, Padre Pd  WHERE P.IdPadre = Pd.Identificacion AND Pd.PrimerNombre = 'Pedro'; |
| --- |

3. Número de personas por año de nacimiento

| SELECT COUNT(P.Identificacion)  FROM Personas P  GROUP BY (P.) |
| --- |

4. Consulta de toda la tabla que muestre además la edad en meses de la persona en una nueva columna.

| SELECT P.Identificacion, P.PrimerNombre, P.PrimerApellido, ((CURRENT\_DATE - P.FechaNacimiento)/30) as DeltaMeses  FROM Personas P; |
| --- |

5. La cantidad de madres que tiene más de 5 hijos

| SELECT COUNT(P.IdMadre) > 5 as NumeroMadres  FROM Personas P  INNER JOIN Madre M On P.IdMadre = M.Identifacion; |
| --- |