INSTITUTO TECNOLOGICO ®

DE JIQUILPAN

“CUESTIONARIO”

INGENIERIA DE SOFTWARE

INGENIERIA EN SISTEMAS COMPUTACIONALES

PRESENTA:

EDGAR CORTÉS RESÉNDIZ

M

JIQUILPAN, MICHOACÁN, MARZO DE 2025

Actividad 2: Lenguajes para el Desarrollo de Software en Ingeniería de Software

1. ¿Qué factores influyen en la elección de un lenguaje de programación para un proyecto?

El uso al que va dirigido, su escalabilidad y su portabilidad, si es el caso de que la aplicación o el uso del lenguaje, con esto si el proyecto es extenso se usa un lenguaje mucho muy robusto que pueda soportar toda la carga del programa, además de que podría ser hacia que propósito, porque puede ser un lenguaje puramente grafico o un lenguaje que sea dedicado a la parte de la conexión del proyecto.

2. ¿Cómo afecta el paradigma de programación (orientado a objetos, funcional, imperativo) en la selección de un lenguaje?

Afecta significativamente, ya que, su propósito es relevante porque los programas son creados en base a un propósito en específico, esto convierte que los paradigmas sean esenciales para la selección del lenguaje, unos convirtiéndolos exclusivamente para un propósito y otros que incluyen muchos paradigmas en uno solo.

3. ¿Cómo influye la comunidad y el soporte de un lenguaje en su adopción dentro de la industria?

Influye de manera significativa, ya que, muchos de los errores son por el feedback que tiene por parte de la comunidad, esto hace que los creadores de los lenguajes sean mas robustos y esto hace que la industria pueda invertir mas en estos tipos de lenguajes.

4. ¿Cuáles son las ventajas y desventajas de utilizar lenguajes de propósito específico frente a los de propósito general?

Mientras que los de uso especifico son eficientes y optimizados para tareas concretas, pero tiene un uso limitado y con un menor soporte. Por otra parte, los de uso general son más versátiles, que cuentan con una amplia documentación y facilitan la integración con diversas herramientas, aunque son menos eficientes en tareas específicas.

5. ¿Cómo influye la compatibilidad de un lenguaje con librerías y frameworks en su selección?

Mientras que en algunos de los lenguajes pueden ser utilizados para un cierto propósito, ya que en ocasiones ciertos lenguajes soportan mas lo gráfico que por ejemplo una conexión entre servidores, esto diversificando y que a su vez especializa a algunos lenguajes para el soporte de una sola tarea.

6. ¿Por qué algunos lenguajes son más utilizados en ciertos sectores de la industria (ej. Python en IA, Java en aplicaciones empresariales)?

Esto es debido a su versatilidad en el uso, por ejemplo, en Python se usa mayormente en IA, ya que esto es gracias a su código simple, que no requiere una gran codificación, lo que lo convierte en mas entendible para el programador, que también sirve para la codificación de una IA. Por otro lado, Java es usado mas en entornos empresariales por su gran robustez y que también por su amplio campo de uso del lenguaje

7. ¿Qué criterios se deben considerar para evaluar la seguridad de un lenguaje de programación?

Para evaluar la seguridad de un lenguaje, se debe considerar su gestión de memoria, el manejo de tipos, los mecanismos de control de acceso y la protección contra inyecciones de código. También es clave el soporte para cifrado, la frecuencia de actualizaciones y la ejecución en entornos seguros. Estos factores ayudan a minimizar vulnerabilidades y fortalecer la seguridad del software.

8. ¿De qué manera la escalabilidad de un software puede depender del lenguaje en el que está desarrollado?

La escalabilidad de un software depende del lenguaje en aspectos como el manejo de concurrencia, la eficiencia en la gestión de memoria y el soporte para arquitecturas distribuidas. Lenguajes como Java, Go y Erlang ofrecen herramientas para alta concurrencia y escalabilidad horizontal, mientras que otros, como Python o PHP, pueden requerir optimizaciones adicionales. Además, la disponibilidad de frameworks y bibliotecas influyen en la facilidad de escalamiento, así como el ecosistema de soporte y la integración con infraestructuras en la nube.

9. ¿Cómo se mide el rendimiento de un lenguaje de programación en diferentes escenarios de desarrollo?

El rendimiento de un lenguaje de programación se mide en diferentes escenarios a través de diversos factores, como el tiempo de ejecución, el uso de memoria, la escalabilidad y la eficiencia en la gestión de recursos. En pruebas de tiempo de ejecución, se evalúa cuán rápido se ejecutan los algoritmos en situaciones específicas. El uso de memoria mide cuánta memoria consume el programa durante su ejecución. La escalabilidad se refiere a cómo un lenguaje maneja el aumento en la carga de trabajo, mientras que la eficiencia en el manejo de recursos implica la optimización de procesos y la reducción de la latencia en aplicaciones concurrentes o de alto rendimiento. Las pruebas suelen incluir benchmarks específicos y comparaciones con otros lenguajes para evaluar el desempeño en situaciones reales de desarrollo.

10. ¿Qué papel juegan las herramientas de desarrollo y los entornos integrados

(IDE) en la selección de un lenguaje?

Las herramientas de desarrollo y los IDE son clave al elegir un lenguaje de programación, ya que facilitan tareas como la depuración, el autocompletado y la integración con control de versiones. Un

buen IDE mejora la productividad, mientras que herramientas como compiladores y bibliotecas optimizan el proceso de desarrollo y despliegue. Estas facilidades hacen que un lenguaje sea más atractivo, especialmente en proyectos grandes.

11. Explica con un ejemplo cómo la sintaxis de un lenguaje puede facilitar o dificultar su aprendizaje.

```
for i in range(1, 6):
    print(i)
```

Aquí, la sintaxis es clara y fácil de entender: usamos un bucle for, la función range() para generar los números, y print() para mostrarlos. La estructura es intuitiva, lo que permite a los principiantes concentrarse en la lógica del programa sin preocuparse demasiado por detalles complejos.

En comparación, en C++, el mismo código podría ser más complicado:

```
#include <iostream>

using namespace std;


int main() {
    for(int i = 1; i <= 5; i++) {
        cout << i << endl;
    }
    return 0;
}
```

Aquí, hay más elementos: se necesita incluir una biblioteca (#include <iostream>), definir una función principal (int main()), y manejar la salida con cout en lugar de una función simple como print(). Estos detalles adicionales pueden hacer que los nuevos programadores se sientan abrumados y distraídos de la tarea principal, lo que hace que C++ sea más difícil de aprender al principio en comparación con Python.

12. ¿Cómo se implementa una función simple en tres lenguajes diferentes y qué diferencias observas?

```
def sumar(a, b):
    return a + b
```

En Python, la sintaxis es simple y clara. No se requiere especificar el tipo de los parámetros ni el tipo de retorno. Además, el uso de def para declarar la función y la ausencia de llaves {} hace que el código sea muy legible.

```
function sumar(a, b) {
   return a + b;
}
```

En JavaScript, la sintaxis es similar a Python en cuanto a la estructura de la función. Se utiliza la palabra clave function para declarar la función. Al igual que en Python, no es necesario especificar los tipos de los parámetros ni el tipo de retorno.

```
#include <iostream>
using namespace std;

int sumar(int a, int b) {
   return a + b;
}

int main() {
   cout << sumar(3, 4) << endl;
   return 0;
}
```

En C++, es necesario especificar el tipo de los parámetros y el tipo de retorno (int en este caso). Además, se requiere una función main() para ejecutar el programa, y el uso de #include <iostream> para manejar la salida. La sintaxis es más compleja en comparación con Python y JavaScript, ya que involucra más detalles.

Las diferencias entre los tres lenguajes son principalmente en la sintaxis y la estructura. Python es el más simple, sin necesidad de declarar tipos y con una sintaxis clara usando def. JavaScript es similar a Python, pero usa la palabra clave function. C++ requiere declarar los tipos de parámetros y del retorno, además de incluir bibliotecas como #include <iostream> y tener una función main(), lo que lo hace más complejo y estructurado.

13. ¿Qué problemas podrías encontrar al migrar un proyecto de un lenguaje a otro?

Al migrar un proyecto entre lenguajes, los problemas comunes incluyen diferencias de sintaxis, incompatibilidad en los tipos de datos, falta de equivalentes para bibliotecas, posibles problemas de rendimiento y portabilidad, y la curva de aprendizaje para los desarrolladores al adaptarse al nuevo lenguaje.

14. ¿Qué consideraciones se deben tomar al desarrollar software multiplataforma con diferentes lenguajes?

Al desarrollar software multiplataforma, se deben considerar varios aspectos clave. Primero, la compatibilidad entre plataformas, asegurando que el código funcione correctamente en distintos sistemas operativos como Windows, macOS y Linux. Luego, la interoperabilidad entre los lenguajes utilizados, para que los diferentes componentes del software se comuniquen de manera eficiente. Es crucial también gestionar bien las dependencias, ya que las bibliotecas y herramientas deben ser compatibles con todas las plataformas objetivo. Además, se debe evaluar el rendimiento de cada lenguaje en las diferentes plataformas, ya que algunos pueden tener un mejor desempeño en determinados sistemas. Finalmente, se debe pensar en la facilidad de mantenimiento y actualización, ya que trabajar con varios lenguajes puede hacer que la gestión y evolución del proyecto sea más compleja.

15. ¿Cómo podrías justificar la elección de un lenguaje de programación para un proyecto específico basado en los datos obtenidos en tu análisis?

La elección de un lenguaje de programación para un proyecto específico se justifica según criterios como el rendimiento requerido, la facilidad de desarrollo (tiempo de implementación), la compatibilidad con plataformas y tecnologías existentes, la disponibilidad de bibliotecas o frameworks adecuados, y la experiencia del equipo en el lenguaje elegido.