PROYECTO AGENCIA DE TURISMO

TALLER DE PROGRAMACIÓN

OLGA LUCÍA ROA BOHÓRQUEZ

KAROL STEFANY ORDOÑEZ PEÑALOZA

YEFERSON PIÑEROS PEÑA

09/08/2024

**REQUERIMIENTOS FUNCIONALES**

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

|  |  |
| --- | --- |
| IDENTIFICADOR: | NOMBRE: |
|  |  |
| REQUERIMIENTO QUE LO UTILIZA O ESPECIALIZA: | DOCUMENTOS DE VISUALIZACIÓN ASOCIADOS: |
|  |  |
| ENTRADA: | SALIDA: |
|  |  |
| DESCRIPCIÓN: | |
|  | |
| MANEJO DE SITUACIONES ANORMALES: | |
|  | |
| CRITERIOS DE ACEPTACIÓN: | |
|  | |

Enuncie y explique brevemente los paradigmas de programación propuestos en dicho artículo, investigando la información solicitada en el siguiente cuadro comparativo.

|  |  |  |  |
| --- | --- | --- | --- |
| Paradigma de Programación. | Ventajas. | Desventajas. | Lenguaje de programación. |
| Programación Imperativa. | 1. Es fácil de entender ya que sigue una secuencia de instrucciones claras. 2. Permite un control preciso en la ejecución y manipulación de las cosas. 3. Es más eficiente en rendimiento. | 1. Se llega a complicar a medida que el programa crece.  2. Los errores suelen ser difíciles de rastrear.  3. No siempre se puede implementar la recursividad. | C, Pascal. |
| Paradigma Declarativa. | 1. Se puede ir directo al objetivo sin necesidad de describir el proceso.  2. Reduce los errores de manejo del estado (variables que se modifican en el proceso).  3. Los compiladores pueden trabajar de manera óptima. | 1. No es Intuitivo para su aprendizaje.  2. Puede ser menos eficientes al hacer uso de abstracciones  3. suele ser usado para problemas en específico. | Haskel para programación funcional y Prolog para programación lógica. |
| Programación Orientada a Objetos (OOP). | 1. Facilita la organización del código y a su vez la recursividad.  2. Pilares como herencia y polimorfismo permiten flexibilidad y reutilización.  3. Puede mantenerse el código. | 1. Puede hacerse muy complejo para proyectos simples.  2. Pilares como la abstracción pueden afectar el rendimiento en la compilación.  3. Un diseño mal hecho o implementado puede afectar a futuro el mantenimiento. | Python, Java, C++. |
| Programación Orientada a Aspectos (AOP). | 1. Maneja de mejor manera aspectos como la seguridad y los registros al separarlos.  2. Los aspectos pueden usarse de manera recursiva.  3. Fácil de mantener y actualizar ya que los cambios tienden a realizarse en un mismo lugar donde se agrupan las actividades de gestión. | 1. Incrementa la complejidad del código y su entendimiento.  2. Tiene muy pocas herramientas en comparación con los otros paradigmas.  3.puede sobrecargarse en la ejecución y ser lento. | AspectJ. |
| Programación Funcional. | 1. Facilita el razonamiento del código.  2. Sus funciones pueden ser editadas de manera sencilla.  3. Maneja variables inmutables (no cambian en el tiempo) lo que facilita la ejecución de tareas en paralelo. | 1. Es uno de los paradigmas más difíciles de aprender en especial para quienes manejan “programación imperativa”  2. A su vez, la inmutabilidad afecta el rendimiento.  3. Tiene pocas herramientas y menos bibliotecas. | Lisp, Haskell, Erlang. |
| Programación Lógica. | 1. Especifica lo que se desea sin detallar en el proceso.  2. Es fácil solucionar cualquier tipo de errores que se produzcan sobre la marcha.  3. Maneja bases sólidas y simples. | 1. No es muy eficiente a comparación de otros paradigmas.  2. Tiene una curva de aprendizaje baja.  3. Tiende a solucionar problemas muy específicos. | Prolog. |
| Programación Reactiva. | 1. Responde a eventos en tiempo real, lo que hace que reaccione rápido a cambios y actualizaciones inmediatas.  2. Maneja de manera adecuada la concurrencia lo que le permite responder bien a tareas simultaneas.  3. Facilita el mantenimiento y mantiene un flujo de datos claros. | 1. Suele ser difícil de entender y a su vez difícil de depurar.  2. La gestión de todos los eventos puede ser costosa.  3. A comparación de otros paradigmas este es el de menos soportes. | Java (RxJava, ReactiveX Bibliotecas). |
| Programación Basada en Eventos. | 1. El mejor paradigma en cuanto a respuestas de eventos.  2. tiene una gran escalabilidad para los usuarios.  3. En la programación basada en eventos, sus componentes pueden funcionar de manera independiente. | 1. Es complejo de mantener.  2. Al no tener un flujo lineal es más difícil de depurar.  3. el mal manejo de este paradigma conduce directamente a problemas de rendimiento. | 1“JavaScript en el contexto de desarrollo web es un ejemplo común de programación basada en eventos.” |

Los paradigmas son formas diferentes de realizar códigos para resolver determinados problemas. Cada paradigma tiene su manera de organizar y manejar la información, lo que afecta la estructura de un código y la manera en que enfrenta el problema. Como lo vimos anteriormente cuentan con características únicas lo que hace que enfrentar un problema pueda ser más fácil con uno u otro.  
De manera popular, hacemos uso del paradigma de “Programación Orientada a Objetos (POO)” ya que usa muchos lenguajes de programación modernos, maneja de manera ordenada el código, es flexible y facilita el manejo de proyectos grandes.  
Aun así, el paradigma más eficiente es el de “Programación Imperativa” ya que permite un control preciso sobre la manipulación del estado y flujo de ejecución. Lo anterior permite optimizar y usar mejor los recursos del sistema.  
Ya por último tenemos al paradigma menos eficiente que es el de “Programación Lógica” y esto se debe a su constante sobrecarga y la necesidad de soluciones mediante retrocesos automáticos, además, es, computacionalmente, costoso y no es muy rápido resolviendo problemas.

Bibliografía

1. 1<https://keepcoding.io/blog/paradigmas-de-programacion/> - Keepcoding Tech School. (2024). ¿Qué son los paradigmas de programación y qué tipos hay?