

# Hito Individual Programación



Adriel Sadia Mora 1 DAM

## **Definir un algoritmo y esbozar el proceso de construcción de una aplicación.**

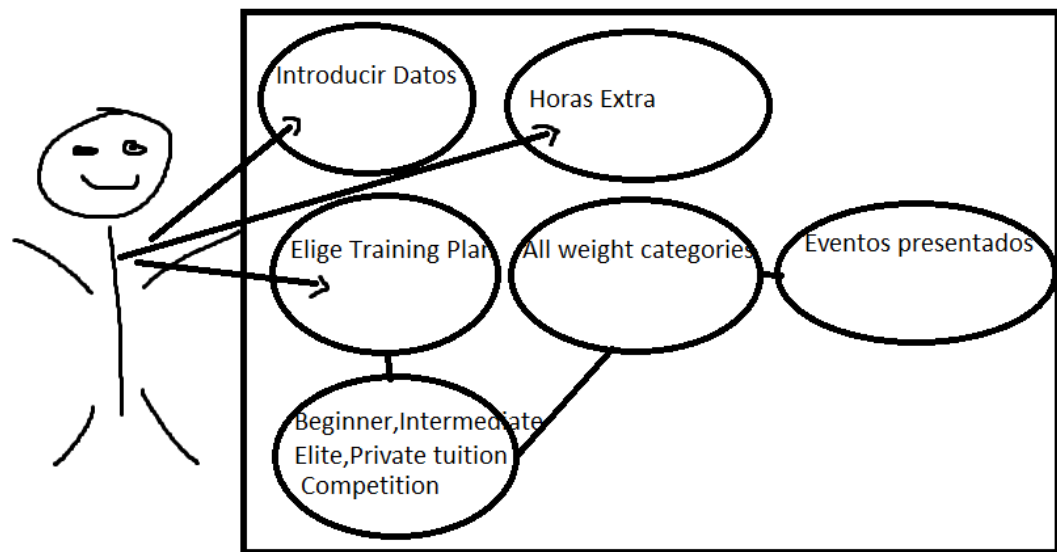
Algoritmo:

Un algoritmo es un conjunto ordenado y finito de pasos definidos que conducen a la solución de un problema o la realización de una tarea. Es una secuencia lógica de instrucciones que se ejecutan en un orden determinado para lograr un objetivo específico.

Proceso de construcción de una aplicación:

1. Definir los requisitos y objetivos: Comienza por comprender claramente los requisitos y objetivos de la aplicación. Identifica qué problema resolverá la aplicación y qué funcionalidades debe tener para lograrlo.
2. Diseñar la arquitectura: Crea un diseño de alto nivel de la aplicación, definiendo las diferentes capas, componentes y su interacción. Determina qué tecnologías utilizamos y cómo se estructurará la aplicación.
3. Desarrollar el backend: Comienza por construir el backend de la aplicación. Esto implica crear la lógica de negocio, diseñar y desarrollar la base de datos, y establecer las API y servicios necesarios. Utiliza el lenguaje de programación y el framework adecuados para tu aplicación.
4. Construir el frontend: Desarrolla la interfaz de usuario de la aplicación. Utiliza tecnologías web como HTML, CSS y JavaScript para crear las páginas y la interacción con el usuario. Asegúrate de que la interfaz sea intuitiva y fácil de usar.
5. Integrar el frontend y el backend: Conecta el frontend y el backend de la aplicación para que puedan comunicarse entre sí. Define cómo se intercambiarán los datos y cómo se gestionan las solicitudes del usuario.
6. Realizar pruebas: Realiza pruebas exhaustivas para garantizar que la aplicación funcione correctamente. Prueba cada funcionalidad, detecta y corrige los errores y verifica que la aplicación cumpla con los requisitos definidos.
7. Desplegar la aplicación: Prepara la aplicación para su implementación en un entorno de producción. Configura el servidor, optimiza el rendimiento y asegúrate de que la aplicación esté lista para ser utilizada por los usuarios.
8. Mantenimiento y actualización: Una vez que la aplicación está en funcionamiento, es importante realizar un mantenimiento regular para corregir errores, mejorar el rendimiento y agregar nuevas funcionalidades. Escucha los comentarios de los usuarios y mantén la aplicación actualizada con los avances tecnológicos.

Caso de uso de mi aplicación:



**Determinar los pasos que se dan desde la escritura del código hasta su ejecución.**

Esta sería la encuesta que tiene que rellenar el usuario:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Encuesta SoloCrossfit</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
</head>
<body>
<div class="container">
<h1>Encuesta SoloCrossfit</h1>
<form action="respuesta" method="post">
<div class="form-group">
<label for="nombre">Nombre:</label>
<input type="text" class="form-control" name="nombre" required>
</div>

<div class="form-group">
<label for="telefono">Teléfono:</label>
<input type="tel" class="form-control" name="telefono" required>
</div>
```

```
<div class="form-group">
  <label for="correo">Correo:</label>
  <input type="email" class="form-control" name="correo" required>
</div>
```

```
<div class="form-group">
  <label for="plan">Plan de trabajo:</label>
  <select class="form-control" name="plan">
    <option value="principiante">Principiante $25</option>
    <option value="intermedio">Intermedio $30</option>
    <option value="elite">Elite $35</option>
    <option value="privado">Sesion privada $9.5/hora</option>
    <option value="competicion">Competicion $22</option>
  </select>
</div>
```

```
<div class="form-group">
  <label for="peso">Peso actual:</label>
  <select class="form-control" name="peso">
    <option value="66">66 kg</option>
    <option value="73">73 kg</option>
    <option value="81">81 kg</option>
    <option value="90">90 kg</option>
    <option value="100">100 kg</option>
    <option value="mas">Más de 100 kg</option>
  </select>
</div>
```

```
<div class="form-group">
  <label for="eventos">Eventos presentados este mes:</label>
  <input type="number" class="form-control" name="eventos" required>
</div>
```

```
<div class="form-group">
  <label for="horasExtra">Horas extra de entrenamiento:</label>
  <select class="form-control" name="horasExtra">
    <option value="1">1 horas extra</option>
    <option value="2">2 hora extra ($10)</option>
    <option value="3">3 horas extra ($12)</option>
    <option value="4">4 horas extra ($14)</option>
    <option value="5">5 horas extra ($16)</option>
    <option value="6">6 horas extra ($18)</option>
  </select>
```

```

        </div>

        <button type="submit" class="btn btn-primary">Calcular</button>
    </form>
</div>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>

```

### **Este sería el servlet:**

```

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/")
public class respuesta extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String nombre = request.getParameter("nombre");
        int telefono = Integer.parseInt(request.getParameter("telefono"));
        String correo = request.getParameter("correo");
        String costoEntrenamientos = request.getParameter("plan");
        String peso = request.getParameter("peso");
        String eventos = request.getParameter("eventos");
        int horasExtra = Integer.parseInt(request.getParameter("horasExtra"));

        //Calculo de costos

        double costoEntrenamientosEuros = 0.0;
        if(costoEntrenamientos.equals("principiante")) {
            costoEntrenamientosEuros = 25.0;

```

```
} else if(costoEntrenamientos.equals("intermedio")) {  
    costoEntrenamientosEuros = 30.0;  
} else if(costoEntrenamientos.equals("elite")) {  
    costoEntrenamientosEuros = 35.0;  
} else if(costoEntrenamientos.equals("privado")) {  
    costoEntrenamientosEuros = 9.50;  
} else if(costoEntrenamientos.equals("competicion")) {  
    costoEntrenamientosEuros = 22.0;  
}
```

```
double costoHorasExtra = 0.0;
```

```
if(horasExtra == 1) {  
    costoHorasExtra = 5.0;  
} else if(horasExtra == 2) {  
    costoHorasExtra = 10.0;  
} else if(horasExtra == 3) {  
    costoHorasExtra = 15.0;  
} else if(horasExtra == 4) {  
    costoHorasExtra = 20.0;  
} else if(horasExtra == 5) {  
    costoHorasExtra = 25.0;  
}
```

```
// Calcular costo total
```

```
double costoTotal = costoEntrenamientosEuros;
```

```
switch (horasExtra) {  
    case 1:  
        costoTotal += 5.0;  
        break;  
    case 2:  
        costoTotal += 10.0;  
        break;  
    case 3:  
        costoTotal += 15.0;  
        break;  
    case 4:  
        costoTotal += 20.0;  
        break;  
    case 5:  
        costoTotal += 25.0;  
        break;  
    default:
```

```

        break;
    }

//Impresion de resultados

    out.println("<html><head><title>Datos del formulario</title>");
    out.println("<style>");
    out.println("table {border-collapse: collapse; width: 100%;}");
    out.println("th, td {text-align: left; padding: 12px;}");
    out.println("th {background-color: #0000ff; color: white; font-weight: bold;
text-transform: uppercase;}");
    out.println("tr:nth-child(even) {background-color: #f2f2f2;}");
    out.println(".container {max-width: 700px; margin: 0 auto; padding: 40px;
font-family: Arial, sans-serif;}");
    out.println(".title {font-size: 24px; margin-bottom: 24px;}");
    out.println(".label {width: 30%;}");
    out.println(".table {background-color: #e0e0e0;}");
    out.println("</style>");
    out.println("</head><body>");
    out.println("<div class='container'>");
    out.println("<h1 class='title'>Datos del formulario:</h1>");
    out.println("<table class='table'>");
    out.println("<tr><th class='label'>Nombre:</th><td>" + nombre + "</td></tr>");
    out.println("<tr><th class='label'>Teléfono:</th><td>" + telefono + "</td></tr>");
    out.println("<tr><th class='label'>Correo:</th><td>" + correo + "</td></tr>");
    out.println("<tr><th class='label'>Costo de entrenamientos:</th><td>" +
costoEntrenamientosEuros + " euros</td></tr>");
    out.println("<tr><th class='label'>Peso:</th><td>" + peso + "</td></tr>");
    out.println("<tr><th class='label'>Eventos a los que ha asistido:</th><td>" + eventos
+ "</td></tr>");
    out.println("<tr><th class='label'>Costo de horas extra al mes:</th><td>" +
costoHorasExtra + " euros</td></tr>");
    out.println("<tr><th class='label'>Costo total:</th><td>" + costoTotal + "
euros</td></tr>");
    out.println("</table>");
    out.println("</div>");
    out.println("</body></html>");

}
}

```

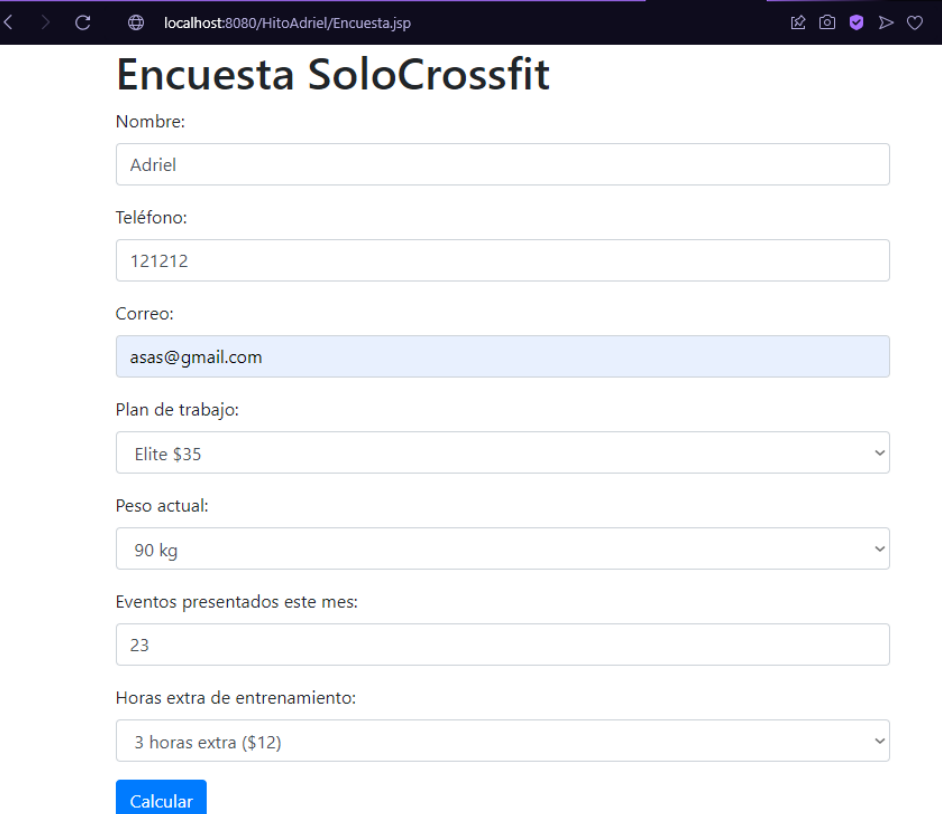
**Los pasos que se dan desde la escritura del código hasta su ejecución son los siguientes:**

1. Escritura del código: El código HTML y JavaScript mostrado en la encuesta es escrito por un desarrollador utilizando un editor de texto o un entorno de desarrollo integrado (IDE).
2. Guardar el archivo: Una vez escrito el código, se guarda en un archivo con una extensión adecuada, en este caso, podría ser un archivo con extensión ".html".
3. Configuración del entorno: Si se requiere, se configura el entorno de desarrollo para asegurarse de que tenga los recursos y las dependencias necesarias para ejecutar el código. En este caso, se utiliza el framework Bootstrap, por lo que se incluyen las referencias correspondientes en el código.
4. Transferencia del código: El archivo con el código se transfiere a un servidor web para que esté disponible en línea. Esto implica subir el archivo a un servidor a través de FTP u otros medios de transferencia de archivos.
5. Petición del usuario: Un usuario accede al formulario de la encuesta a través de un navegador web al ingresar la URL donde se encuentra alojado el archivo HTML.
6. Envío de la respuesta: Cuando el usuario completa el formulario y hace clic en el botón "Calcular", se envía una petición HTTP al servidor web que contiene el formulario con los datos ingresados.
7. Procesamiento del formulario: El servidor web recibe la petición y busca el código asociado a la acción definida en el atributo `action` del formulario. En este caso, la acción es "respuesta", por lo que el servidor buscará un controlador o script que maneje esta acción.
8. Procesamiento del código del servidor: El servidor ejecuta el código del controlador o script asociado a la acción. Este código puede estar escrito en lenguajes como Java, PHP, Python, entre otros. El código del servidor puede realizar validaciones, procesar los datos enviados en la petición y generar una respuesta.
9. Generación de la respuesta: El código del servidor genera una respuesta en función de la lógica definida en el controlador o script. Esta respuesta puede ser una nueva página HTML generada dinámicamente, una redirección a otra página, un mensaje de error, entre otros.
10. Envío de la respuesta al cliente: El servidor web envía la respuesta generada de vuelta al navegador del usuario que realizó la petición.
11. Renderización en el navegador: El navegador web recibe la respuesta del servidor y renderiza el contenido HTML, CSS y JavaScript correspondiente. En este caso, el navegador mostrará la página generada con los resultados de la encuesta o cualquier otra acción definida en el código del servidor.



12. Interacción con la página generada: El usuario puede interactuar con la página generada, ver los resultados de la encuesta, enviar más peticiones, realizar acciones adicionales, etc.

**Aquí los resultados de que el código funciona:**



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/HitoAdriel/Encuesta.jsp'. The page title is 'Encuesta SoloCrossfit'. The form contains the following fields:

- Nombre:
- Teléfono:
- Correo:
- Plan de trabajo:
- Peso actual:
- Eventos presentados este mes:
- Horas extra de entrenamiento:

At the bottom of the form is a blue button labeled 'Calcular'.



#### Datos del formulario:

NOMBRE:	Adriel
TELÉFONO:	121212
CORREO:	asas@gmail.com
COSTO DE ENTRENAMIENTOS:	35.0 euros
PESO:	90
EVENTOS A LOS QUE HA ASISTIDO:	23
COSTO DE HORAS EXTRA AL MES:	15.0 euros
COSTO TOTAL:	50.0 euros

### **En qué consisten los métodos procedimentales, orientados a objetos y son los paradigmas basados en eventos: sus características y la relación entre ellos.**

Los métodos procedimentales, orientados a objetos y basados en eventos son tres paradigmas de programación diferentes, cada uno con sus propias características y enfoques. Veamos en qué consiste cada uno y cómo se relacionan entre sí:

#### Métodos procedimentales:

El paradigma procedimental se basa en una secuencia de instrucciones que se ejecutan en orden para resolver un problema. Los programas están compuestos por funciones o procedimientos que manipulan variables y realizan operaciones. Este enfoque se centra en los algoritmos y en cómo se ejecutan paso a paso.

#### Características:

El programa se divide en una serie de procedimientos o funciones.

Las variables son globales o locales y pueden ser modificadas por cualquier función.

Se enfoca en la lógica del problema y en cómo realizar las operaciones.

#### Relación con los otros paradigmas:

El paradigma procedimental es el más antiguo y básico. Sirve como base para los otros paradigmas y se puede considerar como el nivel más bajo de abstracción en la programación.

#### Orientados a objetos:

El paradigma orientado a objetos se centra en la creación de objetos que contienen tanto datos (variables) como funciones (métodos) que operan en esos datos. Los objetos se crean a partir de clases, que definen la estructura y el comportamiento de los objetos.

Características:

El programa se estructura en clases y objetos.

Los objetos encapsulan datos y comportamiento en un solo lugar.

Se enfoca en la interacción entre objetos y la modificación de sus propiedades.

Relación con los otros paradigmas:

El paradigma orientado a objetos amplía el paradigma procedimental al agregar conceptos como encapsulación, herencia y polimorfismo. Puede considerarse una evolución del paradigma procedimental.

Basados en eventos:

El paradigma basado en eventos se centra en la programación reactiva, donde los programas responden a eventos generados por el sistema o por el usuario. Se basa en la idea de que los programas son impulsados por eventos y se ejecutan en respuesta a ellos.

Características:

El programa define eventos y cómo responder a ellos.

Los eventos pueden ser generados por el sistema, el usuario o el propio programa.

Se enfoca en la interacción y la sincronización de eventos.

Relación con los otros paradigmas:

El paradigma basado en eventos es diferente de los anteriores, ya que se centra en la reactividad y la interacción con eventos externos. Puede utilizarse en combinación con los paradigmas procedimentales u orientados a objetos para definir cómo responder a eventos específicos en el programa.

## **Explicar el proceso de depuración y las facilidades de depuración disponibles en el IDE**

La depuración es el proceso de identificar y corregir errores o defectos en un programa informático. Es una etapa crucial durante el desarrollo de software, ya que permite detectar y solucionar problemas que puedan afectar el funcionamiento correcto del programa. Los IDEs (Entornos de Desarrollo Integrados) proporcionan diversas facilidades para facilitar el proceso de depuración. A continuación, se explican ambos aspectos con más detalle:

Proceso de depuración:

1. Identificación del problema: En primer lugar, es importante identificar el comportamiento no deseado o los errores en el programa. Esto puede incluir bloqueos, excepciones, resultados incorrectos o cualquier otro comportamiento inesperado.

2. Colocación de puntos de interrupción: Un punto de interrupción es una instrucción o línea de código en la que se pausa la ejecución del programa para examinar su estado. Los puntos de interrupción se colocan en las secciones del código donde se sospecha que puede estar

ocurriendo un error. Al detener la ejecución en estos puntos, se puede examinar el estado de las variables y ejecutar el programa paso a paso para encontrar el origen del problema.

3. Inspección del estado del programa: Una vez que el programa se detiene en un punto de interrupción, se puede examinar el estado actual de las variables, objetos y estructuras de datos relevantes. Esto permite verificar si los valores son los esperados y si se están realizando las operaciones correctas.

4. Ejecución paso a paso: El depurador del IDE permite ejecutar el programa línea por línea o instrucción por instrucción después de un punto de interrupción. Esto facilita el seguimiento del flujo de ejecución y ayuda a identificar el momento exacto en el que ocurre un error.

5. Observación y corrección de errores: Durante la ejecución paso a paso, se pueden observar los cambios en el estado del programa y determinar si hay errores lógicos, valores incorrectos o malas asignaciones. Una vez que se encuentra el error, se realiza la corrección correspondiente y se continúa con la depuración.

Facilidades de depuración en el IDE:

Los IDEs proporcionan una serie de herramientas y funcionalidades para facilitar la depuración de programas. Algunas de las facilidades comunes incluyen:

1. Puntos de interrupción: Permiten pausar la ejecución del programa en puntos específicos para examinar el estado de las variables y realizar un seguimiento detallado del flujo de ejecución.

2. Inspección de variables: Permite ver los valores actuales de las variables en tiempo de ejecución y realizar un seguimiento de cómo cambian a medida que el programa se ejecuta.

3. Ejecución paso a paso: Permite ejecutar el programa línea por línea o instrucción por instrucción para analizar el comportamiento detallado del programa.

4. Consola de depuración: Proporciona una consola interactiva en la que se pueden ingresar comandos y evaluar expresiones en el contexto de depuración del programa.

5. Visualización de pila de llamadas: Muestra la secuencia de llamadas a funciones o métodos en curso, lo que facilita el seguimiento de la ejecución y la identificación de problemas relacionados con la pila de llamadas.

**Explique la norma de codificación que ha utilizado en su código.**

Mi código está escrito en lenguaje JSP (JavaServer Pages). En cuanto a la norma de codificación utilizada, se pueden observar las siguientes prácticas:

Uso de comillas: Se utilizan comillas dobles (") para delimitar los valores de los atributos en las etiquetas HTML.

Uso de etiquetas y atributos HTML: Se utilizan etiquetas HTML y atributos para estructurar el formulario de la encuesta.

Organización de elementos: Las etiquetas HTML están organizadas jerárquicamente dentro de la estructura básica de un documento HTML, con las etiquetas <head> y <body>.

Uso de enlaces externos: Se utilizan enlaces externos a bibliotecas, como Bootstrap, para importar estilos y funcionalidades adicionales al formulario.

### **Examina cómo se puede utilizar el proceso de depuración para ayudar a desarrollar aplicaciones más seguras y robustas**

Identificación de vulnerabilidades: Durante el proceso de depuración, es posible detectar y corregir errores y comportamientos inesperados que podrían ser explotados por atacantes. Estos errores pueden incluir condiciones de carrera, fugas de memoria, manipulación incorrecta de datos de entrada y errores de lógica que podrían llevar a problemas de seguridad.

Análisis de datos de entrada: Durante la depuración, es posible examinar los datos de entrada y salida del programa. Esto permite identificar y corregir problemas de validación de datos, como entradas incorrectas, datos maliciosos o inyecciones de código, que podrían llevar a vulnerabilidades de seguridad.

Seguimiento de errores y excepciones: La depuración proporciona información detallada sobre los errores y excepciones que se producen durante la ejecución de la aplicación. Esto permite identificar y corregir problemas que podrían afectar la seguridad y la estabilidad del sistema.

Prueba de límites y situaciones extremas: Durante la depuración, es posible probar la aplicación utilizando diferentes escenarios y situaciones extremas. Esto ayuda a identificar y corregir problemas relacionados con límites de memoria, desbordamiento de búfer, condiciones extremas y errores en el manejo de errores, lo que contribuye a mejorar la robustez y la seguridad del sistema.

Evaluación de algoritmos de cifrado y autenticación: La depuración permite examinar detalladamente los algoritmos y las técnicas de cifrado y autenticación utilizadas en la

aplicación. Esto ayuda a identificar posibles debilidades en la implementación de estos mecanismos y a corregirlos para mejorar la seguridad de la aplicación.

Pruebas de penetración: La depuración puede ser utilizada como parte de las pruebas de penetración, permitiendo a los desarrolladores identificar y solucionar vulnerabilidades antes de que sean aprovechadas por atacantes reales.