

**Nombre de los estudiantes: Jim Harry Ampudia Garza.**

**Axel Augusto Luna Cruz**

**Matriculas: 440203849, 870168120**

**Carrera: Ingeniería en Sistemas Computacionales.**

**Actividad: Validación y documentación.**

**Profesor: Julio Cesar Villa López.**

**Materia: Modelo y métodos para el desarrollo de software.**

**Fecha: 23 de junio del 2024**

**Campus: Cumbres.**

**Monterrey N.L**

Investiga y contesta:

Planeación.

En tu proyecto como explicas los siguientes elementos y como los llevas a cabo, como justificas su uso en tu proyecto.

------------------------------------

· Requerimientos Claros y Bien Definidos.

Tener requerimientos claros y bien definidos asegurándonos que todos los miembros del equipo tengan una comprensión común de lo que se espera lograr. Esto nos ayuda a evitar malentendidos.

* Documentar los requerimientos del sistema, como la funcionalidad de registrar, buscar, actualizar, eliminar y limpiar productos.
* Definir claramente los campos necesarios para cada operación (código, nombre, descripción, precio, existencias, categoría).
* Los comportamientos esperados para los botones de la interfaz.

· Planificación Eficaz.

La planificación eficaz permite establecer plazos realistas y prever posibles obstáculos.

* Creamos un cronograma detallado que incluye la creación de la base de datos, desarrollo de la interfaz y implementación de funcionalidades.
* Asignamos tareas específicas a cada miembro del equipo, definiendo claramente sus responsabilidades.

· Gestión de Proyectos y Metodologías Ágiles.

La gestión de proyectos y el uso de metodologías ágiles, como Scrum, permiten un desarrollo iterativo e incremental, facilitando la adaptación a cambios.

* Implementamos sprints semanales con reuniones diarias para revisar el progreso y ajustar el plan según si era necesario.
* Utilizamos herramientas como Jira para gestionar las tareas y el progreso del proyecto.

· Equipo de Trabajo Competente y Colaborativo.

Un equipo competente y colaborativo garantizamos que cada tarea se realice con la calidad y eficiencia necesarias.

* Seleccionamos a miembros del equipo con habilidades complementarias en desarrollo de software, diseño de bases de datos y gestión de proyectos.
* Fomentamos un ambiente de colaboración y comunicación.

· Comunicación Efectiva.

La comunicación efectiva todos los involucrados estaban al tanto del progreso, desafíos y cambios en el proyecto.

* Establecimos medios de comunicación claros y regulares, como reuniones diarias y reportes semanales.
* Utilizamos la herramienta de comunicación Microsoft Teams.

· Gestión de Calidad.

La gestión de calidad nos asegura que el producto final cumpla con los estándares y expectativas establecidos.

* Implementamos revisiones de código y pruebas unitarias para asegurar la calidad del software.
* Realizamos pruebas de aceptación del usuario para validar que el sistema cumple con los requerimientos.

· Gestión de Riesgos.

La gestión de riesgos nos permitió identificar y mitigar posibles problemas antes de que se conviertan en obstáculos mayores.

* Identificar riesgos potenciales y desarrollar planes de contingencia.
* Revisar regularmente los riesgos y ajustar las estrategias de mitigación según sea necesario.

· Enfoque en el Usuario Final.

El enfoque en el usuario final nos asegura que el producto sea útil y satisfactorio para quienes lo utilizarán.

* Recoger requisitos y feedback de los usuarios finales durante todo el proceso.
* Realizar pruebas para garantizar que la interfaz sea intuitiva y fácil de usar.

· Adaptabilidad y Flexibilidad.

La adaptabilidad y flexibilidad nos permitió ajustar el proyecto en respuesta a cambios en los requerimientos o el entorno.

* Mantuvimos una mentalidad abierta a cambios y mejoras.
* Implementamos prácticas ágiles que nos permiten iteraciones rápidas y ajustes continuos.

[PlantillaAgil11.xlsx](https://myuvmedu-my.sharepoint.com/:x:/g/personal/a440203849_my_uvm_edu_mx/ER8bSMR3kTlFugj1zlsxK9IBokNSfVEDXL-NgsDAKGRYzA?e=AAdyqC)

Practica.

draw

Código.

Ser requiere que se genere lo siguiente:

· Es necesario que todos los elementos del proyecto estén validados.

· El cliente ha solicitado:

o Que los campos código, precio y existencia sean solo números.

Al definir la estructura de tu base de datos los campos código, precio y existencia estarán declarados como solo números. Y solo se acepten valores numéricos en los comentarios proporcionados al código. Es importante dar instrucciones claras y específicas sobre cómo ingresar los datos en el código, precio y existencia, ya que deben utilizarse exclusivamente para almacenar valores numéricos.



o Al momento de borrar un elemento de su BD los campos de las cajas de texto se limpien automáticamente.

o Al momento de borrar aparezca un mensaje indicando que se borraran los registros con la opción si/no, si se presiona si se borre y si se presiona no se regrese al menú anterior.

Son funciones que se encargan de seleccionar el elemento a borrar de la base de datos y de realizar la operación de borrado, respectivamente. Una vez completada la operación de borrado, se llama a la función, que se encarga de limpiar los campos de las cajas de texto. Es importante asegurarse de que la operación de borrado en la base de datos se haya completado satisfactoriamente antes de limpiar los campos de las cajas de texto para evitar posibles errores.

Es por eso por lo que, al momento de borrar registros, es importante mostrar un mensaje de confirmación para que el usuario confirme su acción. Este mensaje debe indicar que se borrarán los registros al ofrecer las opciones de "si" o "no" para que el usuario pueda tomar una decisión. Si el usuario selecciona "si", se procederá a borrar los registros. Si selecciona "no", se cancelará la acción de borrado y se regresará al menú anterior. De esta manera, se garantiza que el usuario tenga la oportunidad de confirmar su acción antes de borrar los registros.



o Para que los botones de eliminar y editar estén habilitados es forzoso que se realice una búsqueda de otra manera estos deben estar deshabilitados.

Es importante que los botones de eliminar y editar estén deshabilitados hasta que el usuario seleccione un elemento específico de la lista de resultados. Una vez que se ha seleccionado un elemento, los botones se habilitarán y el usuario podrá realizar las acciones deseadas. Esto ayuda a prevenir errores y asegura que el usuario tenga claridad sobre qué elemento desea editar o eliminando.

Estas acciones solo pueden realizarse en un elemento seleccionado, y no en la lista completa de resultados. Esta funcionalidad se implementa comúnmente en interfaces de usuario para evitar que los usuarios eliminen o editen información de manera inadvertida, especialmente si no están seguros de qué elemento específico desean eliminar y editar.



o Al momento de realizar una edición aparezca un mensaje indicando que se editara el registro con la opción si/no, si se presiona si aplicara la edición, de no ser así mantendrá los registros sin modificación.

Al realizar una edición en un registro, aparecerá un mensaje indicando que se editará el registro y ofreciendo las opciones "sí" o "no". Si se selecciona "sí", se aplicará la edición realizada. En caso contrario, los registros se mantendrán sin modificación.





o Al realizar una eliminación o modificación estos se tienen que deshabilitar al realizar la operación.

Es importante mantener un registro detallado de todas las eliminaciones o modificaciones realizadas, ya que esto ayuda a mantener un historial claro de los cambios realizados en el sistema y facilita la identificación de posibles errores o problemas en el futuro. Además, es recomendable realizar una copia de seguridad de los datos antes de llevar a cabo cualquier eliminación o modificación importante, para evitar la pérdida de información importante.





o Cada operación limpiara las cajas de texto una vez hecha la operación.

Además de limpiar las cajas de texto después de cada operación, es importante documentar los pasos realizados y los resultados obtenidos. Esto facilitará la comprensión de las acciones realizadas y permitirá hacer un seguimiento adecuado de las tareas realizadas. La documentación adecuada puede servir de referencia en el futuro si hay que volver a realizar la misma tarea.



o Si se ingresa un código que ya exista indica que ese código no se pueda usar.

Se debe documentar en el sistema que el código ingresado ya existe y por lo tanto no se puede utilizar. Esto garantiza que se evite la duplicación de códigos en el sistema y se mantenga la integridad de la información. 

o Opcional desea que todos los datos de la BD se puedan consultar en una interfaz.

Se recomienda que todos los datos de la base de datos se puedan consultar mediante una interfaz para facilitar su acceso y visualización. Esto permitirá a los usuarios obtener la información necesaria de forma rápida y sencilla, lo cual mejorará la eficiencia en la utilización de la base de datos. Además, es recomendable mantener la documentación actualizada y dar soporte técnico si los usuarios tienen alguna duda o problema al utilizar la interfaz de consulta de datos. De esta manera, se garantiza que la base de datos sea una herramienta útil y eficaz para los usuarios.

o No desea que ninguno de los campos se quede vacío.

Es importante completar todos los campos para asegurarse de que la información proporcionada sea precisa y útil. Si algún campo se queda vacío, puede afectar la calidad de la comunicación o la comprensión de la información antes de continuar.



Documentación:

· Para el código es necesario documentarlo.

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace GestionProductos

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

btneliminar.Enabled = false;

btnactualizar.Enabled = false;

}

**Conexión a la Base de Datos**

// Clase que maneja la conexión a la base de datos MySQL

public static class Conexion

{

// Método para establecer y devolver la conexión a la base de datos

public static MySqlConnection conexion()

{

string servidor = "localhost";

string bd = "nombre\_de\_la\_base\_de\_datos";

string usuario = "tu\_usuario";

string password = "tu\_contraseña";

string cadenaConexion = "Database=" + bd + "; Data Source=" + servidor + "; User Id=" + usuario + "; Password=" + password + "";

try

{

MySqlConnection conexionBD = new MySqlConnection(cadenaConexion);

return conexionBD;

}

catch (MySqlException ex)

{

throw ex;

}

}

}

**Botón Agregar**

// Evento del botón "Agregar" para insertar un nuevo producto en la base de datos

private void btnagregar\_Click(object sender, EventArgs e)

{

// Capturar los valores de los campos de texto

string codigo = txtcodigo.Text;

string nombre = txtnombre.Text;

string descripcion = txtdescripcion.Text;

double precio = double.Parse(txtpreciopub.Text);

int existencias = int.Parse(txtexistencias.Text);

// Comprobar que el código es único

if (ExisteCodigo(codigo))

{

MessageBox.Show("El código ingresado ya existe. Ingrese un código único.");

return;

}

// Construir la consulta SQL para insertar un nuevo producto

string sql = "INSERT INTO productos (codigo, nombre, descripcion, precio\_publico, existencias) VALUES (@codigo, @nombre, @descripcion, @precio, @existencias)";

// Establecer la conexión a la base de datos

MySqlConnection con = Conexion.conexion();

con.Open();

try

{

// Crear y ejecutar el comando SQL

MySqlCommand comando = new MySqlCommand(sql, con);

comando.Parameters.AddWithValue("@codigo", codigo);

comando.Parameters.AddWithValue("@nombre", nombre);

comando.Parameters.AddWithValue("@descripcion", descripcion);

comando.Parameters.AddWithValue("@precio", precio);

comando.Parameters.AddWithValue("@existencias", existencias);

comando.ExecuteNonQuery();

MessageBox.Show("Producto agregado exitosamente.");

LimpiarCampos();

}

catch (MySqlException ex)

{

MessageBox.Show("Error al agregar el producto: " + ex.Message);

}

finally

{

con.Close();

}

}

**Botón Buscar**

// Evento del botón "Buscar" para buscar un producto por código

private void btnbuscar\_Click(object sender, EventArgs e)

{

string codigo = txtcodigo.Text;

MySqlDataReader reader = null;

string sql = "SELECT id, codigo, nombre, descripcion, precio\_publico, existencias FROM productos WHERE codigo = @codigo LIMIT 1";

MySqlConnection conexionBD = Conexion.conexion();

conexionBD.Open();

try

{

MySqlCommand comando = new MySqlCommand(sql, conexionBD);

comando.Parameters.AddWithValue("@codigo", codigo);

reader = comando.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

txtid.Text = reader.GetString(0);

txtcodigo.Text = reader.GetString(1);

txtnombre.Text = reader.GetString(2);

txtdescripcion.Text = reader.GetString(3);

txtpreciopub.Text = reader.GetString(4);

txtexistencias.Text = reader.GetString(5);

// Habilitar los botones de eliminar y editar

btneliminar.Enabled = true;

btnactualizar.Enabled = true;

}

}

else

{

MessageBox.Show("No se encontraron registros con ese código.");

}

}

catch (MySqlException ex)

{

MessageBox.Show("Error al buscar: " + ex.Message);

}

finally

{

conexionBD.Close();

}

}

**Botón Actualizar**

// Evento del botón "Actualizar" para modificar un producto existente

private void btnactualizar\_Click(object sender, EventArgs e)

{

// Capturar los valores de los campos de texto

string id = txtid.Text;

string codigo = txtcodigo.Text;

string nombre = txtnombre.Text;

string descripcion = txtdescripcion.Text;

double precio = double.Parse(txtpreciopub.Text);

int existencia = int.Parse(txtexistencias.Text);

// Mostrar un mensaje de confirmación

DialogResult result = MessageBox.Show("¿Está seguro que desea editar este registro?", "Confirmar edición", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

return;

}

// Construir la consulta SQL para actualizar el producto

string sql = "UPDATE productos SET codigo = @codigo, nombre = @nombre, descripcion = @descripcion, precio\_publico = @precio, existencias = @existencia WHERE id = @id";

// Establecer la conexión a la base de datos

MySqlConnection con = Conexion.conexion();

con.Open();

try

{

// Crear y ejecutar el comando SQL

MySqlCommand comando = new MySqlCommand(sql, con);

comando.Parameters.AddWithValue("@codigo", codigo);

comando.Parameters.AddWithValue("@nombre", nombre);

comando.Parameters.AddWithValue("@descripcion", descripcion);

comando.Parameters.AddWithValue("@precio", precio);

comando.Parameters.AddWithValue("@existencia", existencia);

comando.Parameters.AddWithValue("@id", id);

comando.ExecuteNonQuery();

MessageBox.Show("Registro modificado exitosamente.");

LimpiarCampos();

}

catch (MySqlException ex)

{

MessageBox.Show("Error al modificar el registro: " + ex.Message);

}

finally

{

con.Close();

}

}

**Botón Eliminar**

// Evento del botón "Eliminar" para eliminar un producto existente

private void btneliminar\_Click(object sender, EventArgs e)

{

// Capturar el id del producto a eliminar

string id = txtid.Text;

// Mostrar un mensaje de confirmación

DialogResult result = MessageBox.Show("¿Está seguro que desea eliminar este registro?", "Confirmar eliminación", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

return;

}

// Construir la consulta SQL para eliminar el producto

string sql = "DELETE FROM productos WHERE id = @id";

// Establecer la conexión a la base de datos

MySqlConnection con = Conexion.conexion();

con.Open();

try

{

// Crear y ejecutar el comando SQL

MySqlCommand comando = new MySqlCommand(sql, con);

comando.Parameters.AddWithValue("@id", id);

comando.ExecuteNonQuery();

MessageBox.Show("Registro eliminado exitosamente.");

LimpiarCampos();

}

catch (MySqlException ex)

{

MessageBox.Show("Error al eliminar el registro: " + ex.Message);

}

finally

{

con.Close();

}

}

**Botón Limpiar**

// Evento del botón "Limpiar" para limpiar todos los campos de texto

private void btnlimpiar\_Click(object sender, EventArgs e)

{

LimpiarCampos();

}

// Método para limpiar todos los campos de texto

private void LimpiarCampos()

{

txtid.Clear();

txtcodigo.Clear();

txtnombre.Clear();

txtdescripcion.Clear();

txtpreciopub.Clear();

txtexistencias.Clear();

// Deshabilitar los botones de eliminar y editar

btneliminar.Enabled = false;

btnactualizar.Enabled = false;

}

**Método para Verificar la Unicidad del Código**

// Método para verificar si un código ya existe en la base de datos

private bool ExisteCodigo(string codigo)

{

bool existe = false;

string sql = "SELECT COUNT(\*) FROM productos WHERE codigo = @codigo";

MySqlConnection con = Conexion.conexion();

con.Open();

try

{

MySqlCommand comando = new MySqlCommand(sql, con);

comando.Parameters.AddWithValue("@codigo", codigo);

int count = Convert.ToInt32(comando.ExecuteScalar());

if (count > 0)

{

existe = true;

}

}

catch (MySqlException ex)

{

MessageBox.Show("Error al verificar el código: " + ex.Message);

}

finally

{

con.Close();

}

return existe;

}

· Deberás crear un manual de usuario donde expliques cada elemento que compone tu sistema y para que funcione.

El sistema de gestión de productos permite registrar, buscar, actualizar, eliminar y limpiar registros de productos en una base de datos MySQL. La interfaz gráfica proporciona botones para cada una de estas funciones, facilitando la gestión del inventario de productos.

#### **Elementos del Sistema.**

**1-Campos de Texto:**

* + **Código:** Campo para ingresar el código único del producto.
  + **Nombre:** Campo para ingresar el nombre del producto.
  + **Descripción:** Campo para ingresar una breve descripción del producto.
  + **Precio:** Campo para ingresar el precio del producto.
  + **Existencias:** Campo para ingresar la cantidad en existencia del producto.

**2-Botones:**

* + **Guardar:**
    - Función: Guarda un nuevo producto en la base de datos.
    - Uso: Ingresar los detalles del producto en los campos de texto y hacer clic en el botón "Guardar".
  + **Buscar:**
    - Función: Busca un producto en la base de datos por su código.
    - Uso: Ingresar el código del producto en el campo "Código" y hacer clic en el botón "Buscar".
  + **Actualizar:**
    - Función: Actualiza los detalles de un producto existente, excepto su ID.
    - Uso: Ingresar el código del producto a actualizar, buscarlo, modificar los detalles necesarios y hacer clic en el botón "Actualizar".
  + **Eliminar:**
    - Función: Elimina un producto de la base de datos por su código.
    - Uso: Ingresar el código del producto en el campo "Código" y hacer clic en el botón "Eliminar".
  + **Limpiar:**
    - Función: Limpia todos los campos de texto de la interfaz.
    - Uso: Hacer clic en el botón "Limpiar".

**Funcionalidades**

* **Agregar Producto**
  + **Descripción**: Permite agregar un nuevo producto a la base de datos.
  + **Campos**: Código, Nombre, Descripción, Precio, Existencias.
  + **Acción**: Llena los campos requeridos y presiona el botón "Agregar". Un mensaje confirmará la adición exitosa del producto.
* **Buscar Producto**
  + **Descripción**: Permite buscar un producto en la base de datos por su código.
  + **Campos**: Código.
  + **Acción**: Ingresa el código del producto y presiona el botón "Buscar". Los campos se llenarán con la información del producto si se encuentra, de lo contrario, mostrará un mensaje de error.
* **Actualizar Producto**
  + **Descripción**: Permite actualizar los datos de un producto existente.
  + **Campos**: Código, Nombre, Descripción, Precio, Existencias.
  + **Acción**: Busca primero el producto a actualizar, modifica los campos necesarios y presiona el botón "Actualizar". Un mensaje confirmará la actualización exitosa del producto.
* **Eliminar Producto**
  + **Descripción**: Permite eliminar un producto de la base de datos por su código.
  + **Campos**: Código.
  + **Acción**: Ingresa el código del producto y presiona el botón "Eliminar". Un mensaje confirmará la eliminación exitosa del producto si se encuentra, de lo contrario, mostrará un mensaje de error.
* **Limpiar Campos**
  + **Descripción**: Limpia todos los campos de texto en la interfaz.
  + **Acción**: Presiona el botón "Limpiar". Todos los campos de texto se vaciarán.

**Requisitos finales del cliente**

* **Validaciones:**
  + Los campos "Código", "Precio" y "Existencias" deben ser numéricos.
  + Ningún campo debe quedar vacío al guardar o actualizar un producto.
* **Mensajes de Confirmación:**
  + Al eliminar un producto, aparecerá un mensaje de confirmación con opciones "Sí/No".
  + Al actualizar un producto, aparecerá un mensaje de confirmación con opciones "Sí/No".
* **Deshabilitar Botones:**
  + Los botones "Eliminar" y "Actualizar" estarán deshabilitados hasta que se realice una búsqueda exitosa.

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 1** | **Usuario: Administrador del Sistema** | |
| **Nombre historia: Validación de Campos Numéricos** | | |
| **Prioridad en negocio: Alta** | | **Riesgo en desarrollo: Baja** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Administrador del Sistema, necesito que los campos de código, precio y existencia acepten solo números para garantizar la integridad de los datos y prevenir errores de formato.** | | |
| **Observaciones: Esto asegurará que los datos ingresados sean consistentes y válidos, facilitando el procesamiento posterior.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 2** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Limpieza Automática de Campos al Eliminar** | | |
| **Prioridad en negocio: Media** | | **Riesgo en desarrollo: Baja** |
| **Puntos estimados: 2** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que los campos de las cajas de texto se limpien automáticamente al momento de borrar un elemento de la base de datos para facilitar el ingreso de nuevos datos.** | | |
| **Observaciones: Esto mejora la usabilidad del sistema al reducir la necesidad de limpieza manual de los campos.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 3** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Confirmación de Eliminación con Opción Sí/No** | | |
| **Prioridad en negocio: Alta** | | **Riesgo en desarrollo: Media** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que aparezca un mensaje de confirmación con la opción de sí o no al momento de borrar un registro, para evitar eliminaciones accidentales.** | | |
| **Observaciones: Esto proporciona una capa adicional de seguridad, asegurando que los registros no se eliminen accidentalmente.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 4** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Habilitar Botones de Eliminar y Editar solo después de Búsqueda** | | |
| **Prioridad en negocio: Media** | | **Riesgo en desarrollo:**  **Media** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que los botones de eliminar y editar se habiliten solo después de realizar una búsqueda para evitar operaciones en registros inexistentes.** | | |
| **Observaciones: Esto mejora la seguridad y la precisión de las operaciones de edición y eliminación.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 5** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Confirmación de Edición con Opción Sí/No** | | |
| **Prioridad en negocio:**  **alta** | | **Riesgo en desarrollo:**  **Media** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que aparezca un mensaje de confirmación con la opción de sí o no al momento de editar un registro, para evitar modificaciones accidentales.** | | |
| **Observaciones: Esto proporciona una capa adicional de seguridad, asegurando que los registros no se modifiquen accidentalmente.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 6** | **Usuario: Administrador del Sistema** | |
| **Nombre historia: Deshabilitar Botones después de Eliminación o Modificación** | | |
| **Prioridad en negocio:**  **Media** | | **Riesgo en desarrollo:**  **Baja** |
| **Puntos estimados: 2** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Administrador del Sistema, quiero que los botones de eliminar y modificar se deshabiliten después de realizar la operación para evitar operaciones redundantes.** | | |
| **Observaciones: Esto mejora la usabilidad del sistema al prevenir acciones repetitivas no deseadas.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 7** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Limpieza Automática de Campos después de Operaciones** | | |
| **Prioridad en negocio:**  **Media** | | **Riesgo en desarrollo:**  **Baja** |
| **Puntos estimados: 2** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que los campos de texto se limpien automáticamente después de cada operación (agregar, editar, eliminar) para preparar el formulario para la próxima entrada.** | | |
| **Observaciones: Esto mejora la eficiencia del usuario al preparar automáticamente el formulario para nuevas entradas.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 8** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Validación de Unicidad del Código** | | |
| **Prioridad en negocio:**  **Alta** | | **Riesgo en desarrollo:**  **Media** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, quiero que se valide que el código del producto sea único al momento de agregar un nuevo producto para evitar duplicados en la base de datos.** | | |
| **Observaciones: Esto garantiza la integridad de los datos y evita la duplicación de registros.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 9** | **Usuario: Administrador del Sistema** | |
| **Nombre historia: Consulta de Todos los Datos de la BD en la Interfaz** | | |
| **Prioridad en negocio:**  **Media** | | **Riesgo en desarrollo:**  **Media** |
| **Puntos estimados: 5** | | **Iteración asignada: 2** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Administrador del Sistema, quiero que se puedan consultar todos los datos de la base de datos en una interfaz para facilitar la revisión y el análisis de los productos.** | | |
| **Observaciones: Esto proporciona una visión completa del catálogo de productos y facilita la gestión de los mismos.** | | |

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | | |
| **Número: 10** | **Usuario: Usuario del Sistema** | |
| **Nombre historia: Validación de Campos Vacíos** | | |
| **Prioridad en negocio:**  **Alta** | | **Riesgo en desarrollo:**  **Baja** |
| **Puntos estimados: 3** | | **Iteración asignada: 1** |
| **Programador responsable: Jim y Axel** | | |
| **Descripción: Como Usuario del Sistema, no quiero que ningún campo quede vacío al momento de agregar o actualizar un producto para asegurar la completitud de la información.** | | |
| **Observaciones: Esto garantiza que todos los datos necesarios para los productos estén presentes y completos.**  **- Todas las historias deben ser probadas exhaustivamente para asegurar que cumplen con los criterios de aceptación especificados.**  **- La interfaz de usuario debe ser intuitiva y fácil de usar para maximizar la eficiencia del usuario.**  **- Las operaciones deben ser eficientes y rápidas para manejar grandes volúmenes de datos sin demoras significativas.** | | |

Entregar:

· Código y aplicación valida.

· Documentación.

· Planeación.

· Historias de usuario.

· BD. draw