Juan de Colonia
CENTRO INTEGRADO DE FORMACIÓN PROFESIONAL

1-2-2024

PROYECTO 2EVA-RJT

2DAM-DESARROLLO DE INTERFACES

ROBERTO JOAO TIRÓN
Profesora: DIANA GARCIA TEJERINA

INDICE

MANUAL DE INSTALACION 3
MANUAL DE USUARIO 3
INTRODUCCION AL PROYECTO 3
Características Destacadas 3
1. Interfaz Atractiva e Intuitiva 3
2. Gestión Integral 3
3. Generación de Informes 3
4. Tecnologías de Vanguardia 3
5. Persistencia de Datos 4
6. Potentes Reportes 4
7. Patrón de Diseño MVVM 4
8. Alta Resistencia a Fallos 4
9. Facilidad de Conexión de Datos 4
OBJETIVOS DEL DESARROLLO 4
Creación de una Aplicación Completa y Estable 4
Diseño Espectacular y Atractivo 5
ESTRUCTURA DEL PROYECTO 5
EJEMPLO DE IMPLEMENTACION MVVM 7
Model (Modelo) 7
View (Vista) 8
View-Model (Vista-Modelo) 8
WORKFLOW (Flujo de Trabajo) 10
PROCESO DE DESARROLLO 10
Planificación 10
Diseño 10
Implementación 10
Pruebas 11
DESAFÍOS ENFRENTADOS 11
Integración de Tecnologías 11
Diseño de Interfaz de Usuario 11
Optimización 11
Aprendizaje Continuo 11
CONCLUSIONES 11

Aprendizaje Continuo 12
Importancia de la Estructura y Diseño 12
El Desarrollo como un Viaje 12
Logro de Objetivos.....12

MANUAL DE INSTALACION

Para acceder al manual de instalación puede seguir este enlace

Haga clic aquí

MANUAL DE USUARIO

Para acceder al manual de usuario puedes seguir este enlace

Haga clic aquí

INTRODUCCION AL PROYECTO

Descripción General

CRUD-APP es mucho más que una simple aplicación de gestión móvil; es tu aliado perfecto para administrar tu almacén de una manera atractiva y altamente intuitiva. Con la capacidad de realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en marcas, procesadores, almacenamientos y móviles, CRUD-APP consolida toda la gestión en una única y potente aplicación.

Características Destacadas

1. Interfaz Atractiva e Intuitiva

La interfaz de CRUD-APP está diseñada para cautivarte con una gama de colores uniforme y continua. La experiencia de usuario es nuestra prioridad, proporcionando una navegación sencilla y una presentación visualmente atractiva.

2. Gestión Integral

Realiza operaciones CRUD en todas las áreas esenciales de tu almacén: marcas, procesadores, almacenamientos y móviles, todo en un solo lugar. Simplifica tu trabajo y ahorra tiempo al tener toda la información centralizada.

3. Generación de Informes

No te pierdas nada de información crucial. CRUD-APP te permite generar informes detallados para tener una visión global de tu almacén en todo momento. Toma decisiones informadas y estratégicas basadas en datos concretos.

4. Tecnologías de Vanguardia

Desarrollada con las mejores tecnologías del mercado, CRUD-APP utiliza C# con el framework WPF basado en .NET 8.0 para ofrecer un rendimiento excepcional y una interfaz de usuario moderna.

5. Persistencia de Datos

La aplicación utiliza MySQL 8 para garantizar una sólida persistencia de datos. Tus datos estarán seguros y accesibles en todo momento, proporcionando una base robusta para la gestión de tu almacén.

6. Potentes Reportes

Los reportes son una parte esencial de la toma de decisiones. CRUD-APP utiliza Microsoft Reporting Services para ofrecer informes detallados y visualmente atractivos que te proporcionan información clave.

7. Patrón de Diseño MVVM

Con un sólido patrón de diseño MVVM (Modelo Vista Vista-Modelo), CRUD-APP ofrece una implementación estructurada que permite una rápida y fácil escalabilidad según tus necesidades específicas.

8. Alta Resistencia a Fallos

La aplicación cuenta con una sólida estructura que garantiza una alta resistencia a fallos. Además, se han implementado patrones como Observer y Singleton para mejorar aún más la estabilidad.

9. Facilidad de Conexión de Datos

Gracias a WPF y su sólida implementación del patrón MVVM, la conexión de datos desde el modelo hasta la vista es fácil y eficiente. La vista-modelo actúa como un puente eficaz para una gestión de datos fluida.

¡CRUD-APP es tu compañero ideal para la gestión eficaz y eficiente de tu almacén! Descarga la aplicación ahora y descubre una nueva forma de simplificar tu trabajo diario.

OBJETIVOS DEL DESARROLLO

Creación de una Aplicación Completa y Estable

El objetivo principal al iniciar el desarrollo de CRUD-APP fue diseñar y construir una aplicación completa que abarcara todos los aspectos clave de la gestión de un almacén. La estabilidad se consideró un pilar fundamental, asegurando que la aplicación funcione sin problemas y ofrezca una experiencia de usuario consistente.

Diseño Espectacular y Atractivo

La estética y el diseño visual fueron considerados elementos cruciales desde el principio. El objetivo era crear una interfaz atractiva que no solo sea funcional, sino que también proporcione una experiencia visualmente agradable para el usuario. Se buscó una gama de colores uniforme y continua para mejorar la coherencia en todo el diseño.

ESTRUCTURA DEL PROYECTO

El proyecto se ha estructurado en carpetas para su fácil comprensión y mantenimiento

- CORE
  - COMMANDS
    - Commands.cs
    - ExitCommands.cs
  - CONSTANTS
    - AnimationConstants.cs
    - DBConstans.cs
    - LoginConstans.cs
    - ModelsConstants.cs
  - DBCONNECTION
    - DBConnection.cs
    - Robertodb.dsn
  - ENUMS
    - Operation.cs
  - IMAGES
    - CARPETAS DE IMÁGENES
  - INTEFACES
    - CrudInterface.cs
    - IFilter.cs
    - IViewModelBase.cs
    - IWindowBase.cs
  - UTILS
    - PasswordHelper.cs
    - Utils.cs
  - Dbroberto.sql
  - MysqlConnector ODBC exe
- MODEL
  - M_Brand.cs
  - M_Phone.cs
  - M_PhoneStorage.cs
  - M_Processor.cs
  - M_Storage.cs
  - User.cs
- REPORTS
  - BrandsReport.rdl
  - PhoneReport.rdl

- ProcessorReport.rdl
- StorageReport.rdl
- Reports.xaml
  - Reports.xaml.cs

❖ THEME
- ModernBorder.xaml
- ModernComboBox.xaml
- ModernDataGrid.xaml
- ModernListBox.xaml
- ModernTabItem.xaml
- ModernMenuButton.xaml
- RoundedButton.xaml
- RoundedTextBox.xaml

❖ VIEW
- PAGES
  - V_Exit.xaml
    - V_Exit.xaml.cs
  - V_Help.xaml
    - V_Help.xaml.cs
  - V_Home.xaml
    - V_Home.xaml.cs
  - V_Informs.xaml
    - V_Informs.xaml.cs
  - V_Warehouse.xaml
    - V_Warehouse.xaml.cs
- WINDOW
  - V_BrandWindow.xaml
    - V_BrandWindow.xaml.cs
  - V_ErrorWindow.xaml
    - V_ErrorWindow.xaml.cs
  - V_Login.xaml
    - V_Login.xaml.cs
  - V_MainWindow.xaml
    - V_MainWindow.xaml.cs
  - V_PhoneStorageWindow.xaml
    - V_PhoneStorageWindow.xaml.cs
  - V_PhoneWindow.xaml
    - V_PhoneWindow.xaml.cs
  - V_ProcessorWindow.xaml
    - V_ProcessorWindow.xaml.cs
  - V_StorageWindow.xaml
    - V_StorageWindow.xaml.cs

❖ VIEWMODEL

- VM_Brand.cs
- VM_LoginUser.cs

- VM_Phone.cs
- VM_Storage.cs
- VM_Processor.cs
- VM_PhoneStorage.cs

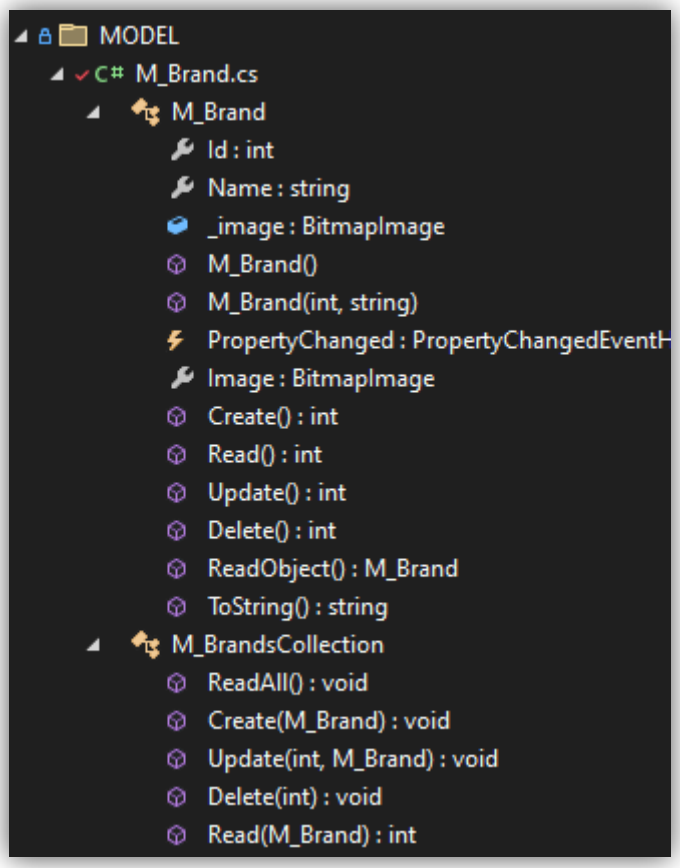
- App.xaml
  - App.xaml.cs
- CRUD-SETUP (Instalador)

EJEMPLO DE IMPLEMENTACION MVVM

CRUD-APP utiliza el patrón de diseño MVVM (Modelo-Vista-VistaModelo) para lograr una organización eficiente y una gestión clara de la lógica de la aplicación. A continuación, se proporciona un ejemplo simplificado para comprender cómo funciona este patrón en el contexto de la aplicación:

Model (Modelo)

En MVVM, el Modelo representa la capa de datos y la lógica de negocio. En CRUD-APP, el Modelo incluye clases como **Brand**, **Processor**, **Phone**, y **Storage** que definen las entidades y su lógica asociada. Además, se encarga de la persistencia de datos en la base de datos.



View (Vista)

La Vista en MVVM es la interfaz de usuario que presenta los datos y recibe la interacción del usuario. Las páginas XAML como V_Brands.xaml representan la Vista, definiendo la apariencia y disposición de los elementos visuales.

```
<DataGrid.Columns>

    <DataGridTemplateColumn Header="Imagen" Width="auto" MinWidth="100"...>

    <DataGridTextColumn Header="Id" Binding="{Binding Id}" IsReadOnly="True" Width="auto"/>

    <DataGridTextColumn Header="Marca" Binding="{Binding Name}" IsReadOnly="True"/>

</DataGrid.Columns>
```

DataGrid que tiene asociado datos del modelo que la vista-modelo ofrece a la vista

View-Model (Vista-Modelo)

La Vista-Modelo en MVVM actúa como intermediario entre la Vista y el Modelo. En CRUD-APP, las clases en el espacio de nombres VM_*, como VM_Brand, se encargan de proporcionar los datos a la Vista y manejar la lógica de presentación.

```
public VM_Brand()
{
    Brand = new M_Brand();
    BrandsCollection = [];
    BrandsCollection.ReadAll();
```

Inicialización del modelo en el vistamodelo, para unirlo a la vista

```
public bool Delete(int index)
{
    int result = Brand.Delete(); // guardamos el resultado de la operación de borrado
    if (result == DBConstants.REGISTER_DELETED)
    {
        InfoSuccessMessage?.Invoke("Success", "Marca eliminada correctamente");
        BrandsCollection.Delete(index);
        View?.Refresh();
        return true;
    }
    else
    {
        DBUtils.CheckStatusOperation(InfoErrorMessage, InfoSuccessMessage, InfoWarningMessage, result, "Marca");
        return false;
    }
}
```

Método de eliminación de una marca, se llama cuando el usuario hace clic en eliminar (vista llama a vistamodelo)

```
private void CreateUpdateDeleteBrand_Executed(object sender, ExecutedRoutedEventArgs e)
{

    if (ViewModel != null && v_Warehouse != null)
    {
        // si la validacion es correcta, ejecutamos la operacion
        if (ViewModel.ValidateInput())
        {
            // dependiendo de la operacion, ejecutamos una accion u otra
            switch (operation)
            {
                case Operation.CREATE:...

                case Operation.UPDATE:...

                case Operation.DELETE:

                    int i = v_Warehouse.BrandsGrid.SelectedIndex;

                    if (ViewModel.Delete(i))
                    {
                        Utils.SuccessMessage(v_Warehouse.infoTextBrand, "Marca " + ViewModel.Brand.ToString() + " eliminada correctamente");
                        Utils.UpdateDataGridToNextPosition(v_Warehouse.BrandsGrid, i);
                        _ = WindowAnimationUtils.FadeOutAndClose(this);

                    }
                    break;

                default:...
            }
```

*Desde la **vista**, al pulsar el botón (en este caso eliminar) se llama al método ViewModel.Delete de la vistamodelo*

En CRUD-APP, la lógica de validación y restricciones de datos se implementa principalmente en las clases del Vista-Modelo (VM_*). Aquí, estas clases aseguran que los datos ingresados sean válidos antes de interactuar con el Modelo o presentarse en la Vista.

```
private void CreateUpdateDeleteBrand_Executed(object sender, Execut
{

    if (ViewModel != null && v_Warehouse != null)
    {
        // si la validacion es correcta, ejecutamos la operacion
        if (ViewModel.ValidateInput())
        {
```

Como hemos visto antes, el ViewModel tiene un método de validación de datos, que llama la vista

```
public bool ValidateInput()
{
    if (string.IsNullOrEmpty(_brand.Name)) // si el nombre está vacío
    {
        InfoErrorMessage?.Invoke("Error", "El nombre no puede estar vacio");
        return false;
    }
    if (_brand.Name.Length < 2) // si el nombre tiene menos de 2 caracteres
    {
        InfoWarningMessage?.Invoke("Error", "El nombre debe tener al menos 2 caracteres");
        return false;
    }

    return true;
}
```

Método de validación de la vista modelo

WORKFLOW (Flujo de Trabajo)

1. **Inicio de la Aplicación:** La aplicación se inicia y carga las clases del Vista-Modelo, que a su vez inicializan y presentan los datos en la Vista.
2. **Interacción del Usuario:** Cuando un usuario interactúa con la Vista (agregar, modificar, eliminar), la acción se maneja en las clases del Vista-Modelo, que validan y procesan la solicitud.
3. **Actualización de la Vista y el Modelo:** Después de las interacciones, los cambios en los datos se reflejan en la Vista y se actualizan en el Modelo y la base de datos.

PROCESO DE DESARROLLO

El proceso de desarrollo de CRUD-APP se llevó a cabo en varias etapas clave, cada una de las cuales desempeñó un papel crucial en la creación de una aplicación robusta y eficiente.

Planificación

Antes de comenzar con la codificación, se realizó una planificación detallada para identificar los objetivos específicos de la aplicación. Esto incluyó la definición de las características esenciales, la arquitectura general y la asignación de tareas.

Diseño

Se desarrolló una arquitectura de software sólida utilizando el patrón de diseño MVVM (Modelo Vista Vista-Modelo). El diseño de la interfaz de usuario se centró en lograr una experiencia atractiva y fácil de usar para los usuarios finales.

Implementación

La fase de implementación implicó la escritura de código utilizando C# con el framework WPF basado en .NET 8.0. Se siguieron las mejores prácticas de codificación y se aprovecharon las características específicas de WPF, como el patrón MVVM para facilitar la conexión de datos entre el modelo y la vista.

Pruebas

Se llevaron a cabo pruebas exhaustivas en diferentes niveles: pruebas unitarias para funciones específicas, pruebas de integración para garantizar la interoperabilidad de los componentes y pruebas de sistema para evaluar la funcionalidad general de la aplicación. La detección y corrección de errores fue una parte integral de este proceso.

DESAFÍOS ENFRENTADOS

Durante el desarrollo de CRUD-APP, varios desafíos significativos surgieron y se abordaron de manera efectiva.

Integración de Tecnologías

La integración de tecnologías como C#, WPF, MySQL 8 y Microsoft Reporting Services presentó desafíos en la gestión de la comunicación entre estas plataformas. Se implementaron soluciones para garantizar una integración sin problemas.

Diseño de Interfaz de Usuario

La creación de una interfaz de usuario atractiva y coherente presentó desafíos en términos de diseño y experiencia del usuario. Se realizaron iteraciones y pruebas para refinar y optimizar la interfaz.

Optimización

Se implementó una estrategia de cargar datos en RAM, lo que permitió realizar operaciones rápidas hasta que el usuario cambiara de sección en la aplicación. Esta optimización redujo la carga en la base de datos y mejoró la velocidad de respuesta.

Aprendizaje Continuo

El uso de nuevas tecnologías y patrones de diseño requirió una curva de aprendizaje constante. La investigación activa y la consulta de recursos externos fueron esenciales para superar los desafíos relacionados con el aprendizaje continuo.

Estos desafíos fueron abordados con éxito gracias a un enfoque metódico, la disposición para aprender y adaptarse a medida que avanzaba el proyecto.

CONCLUSIONES

Tras dedicar un total de 81 horas (y más, considerando las horas no registradas para investigación y aprendizaje), se han extraído conclusiones valiosas del proceso de desarrollo de CRUD-APP.

Aprendizaje Continuo

La investigación de nuevas tecnologías y la implementación de patrones de diseño no solo fueron desafiantes, sino también enriquecedoras. Cada hora invertida en aprendizaje resultó ser una inversión en la mejora constante de habilidades y conocimientos.

Importancia de la Estructura y Diseño

La estructura sólida y el diseño cuidadoso son fundamentales para el éxito de una aplicación. La implementación de patrones como MVVM, Observer y Singleton contribuyó significativamente a la estabilidad y mantenimiento de CRUD-APP.

El Desarrollo como un Viaje

Desarrollar CRUD-APP no solo fue un proceso técnico, sino un viaje de autodescubrimiento y crecimiento profesional. Cada desafío superado ha añadido una capa de experiencia valiosa.

Logro de Objetivos

El logro de los objetivos establecidos al inicio del proyecto es evidente en la aplicación resultante. CRUD-APP se erige como una herramienta completa, estable y visualmente impresionante para la gestión de almacenes.

En resumen, las horas invertidas no solo fueron en el desarrollo de una aplicación, sino en la construcción de habilidades, la exploración de nuevas tecnologías y la creación de una solución práctica y atractiva. CRUD-APP es el resultado tangible de un esfuerzo significativo y continuará siendo una fuente de aprendizaje y satisfacción.