

BankApp

[Tecnologías usadas](#)

[Implementación](#)

[.Net 6.0](#)

[Base de datos](#)

[Preparación](#)

[Restaurar la base de datos](#)

[Crear la base de datos mediante scripts](#)

[Código](#)

[Descripción del programa](#)

[Pantalla - Control](#)

[Pantalla - Customer](#)

[Parte superior - Búsqueda de cliente](#)

[Parte intermedia - Información del cliente](#)

[Parte inferior - Acciones sobre un cliente](#)

[Pantalla - Bank Account](#)

[Base de datos](#)

[Tabla - Customer](#)

[Tabla - Account](#)

[Diagrama](#)

Tecnologías usadas

- .Net 6.0
- C#
- WPF (Windows Presentation Foundation)
- SQL server 2019
- T-SQL

Implementación

.Net 6.0

Este proyecto fue creado en .Net 6.0 por lo que es importante tener descargada e instalada la última versión que se puede encontrar de forma oficial en este [enlace](#).

Base de datos

Preparación

Los datos de la aplicación provienen de una base de datos alojada en SQL Server 2019, que se puede descargar desde este [enlace](#).

Una vez descargado es necesario instalarlo, para este proyecto se han utilizado las opciones por defecto que ofrece el asistente de instalación salvo en el apartado **“Configuración del Motor de base de datos- Configuración del servidor”**, en el cual se ha seleccionado el modo **“Autenticación de modo mixto”** lo que permite usar el usuario “sa”, usado en el proyecto para mayor comodidad.

Es importante recordar los datos introducidos y la contraseña asignada al usuario “sa”, pues serán usados en la cadena de conexión del programa.

Es recomendable seguir la [guía de instalación oficial de SQL Server](#) de Microsoft.

Tras instalar SQL Server 2019, debemos descargar e instalar [SQL Server Management Studio](#), que nos permitirá restaurar la copia de seguridad de la base de datos de ejemplo (archivo con extensión .bak), o usar los scripts proporcionados para crear la estructura de la base de datos y rellenarla con los datos de ejemplo (archivos con extensión .sql).

Restaurar la base de datos

Si elegimos esta opción, debemos abrir **SQL Server Management Studio** e iniciar sesión con el usuario “sa”.

Para restaurar la base de datos, seguiremos los pasos indicados en la [guía oficial de Microsoft](#).

Podremos encontrar el archivo con extensión .bak en la carpeta Script.

Crear la base de datos mediante scripts

La alternativa al método anterior es usar los scripts con extensión .sql que se encuentran en la carpeta Scripts.

Para ejecutar los scripts, abrimos **SQL Server Management Studio** e iniciar sesión con el usuario “sa”, tras ello pulsamos File > Open > File ... > Seleccionamos el archivo “create_structure.sql” y pulsamos “Execute”.

Tras ello, realizamos la misma operación con el script “fill_tables.sql”.

Código

Para asegurar la conexión con la base de datos debemos modificar la cadena de conexión en el archivo “App.Config” y reemplazar los datos por los introducidos cuando se instaló la base de datos.

Es importante que la cadena de conexión se siga llamando “bankapp”.

Una vez realizado este cambio ya es posible compilar el código, e implementar el programa en el equipo del cliente.

Descripción del programa

El programa está formado por 3 capas:

- Los **Models**: Son clases usadas para representar la estructura de un cliente (Customer) o de las cuentas de banco (BankAccount).
- Los **Viewmodels**: Son clases que exponen la información de un Model, y que incluye toda la lógica que una View puede realizar.
- Las **vistas o Views**: Está formada por un archivo con extensión .xaml que define la interfaz que verá el usuario y su correspondiente clase con extensión .cs, usado principalmente para definir que *viewmodel* usará la *view*.

Las 3 pantallas visibles en este programa son:

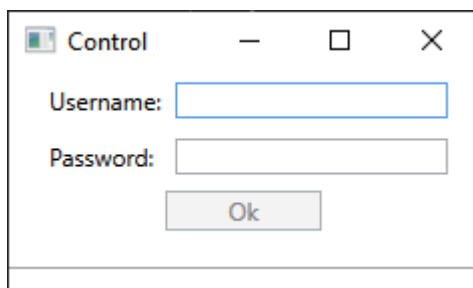
Control: Una pantalla de login que permite al cliente identificarse con su usuario y contraseña.

Customer: Pantalla principal que permite crear, leer, actualizar y borrar clientes, así como acceder a la pantalla Bank Account.

Bank Account: Muestra las cuentas bancarias del cliente seleccionado en la pantalla *Customer*.

Pantalla - Control

Esta pantalla está formada por dos label, con un TextBox para el campo del nombre de usuario y un PasswordBox para la contraseña.

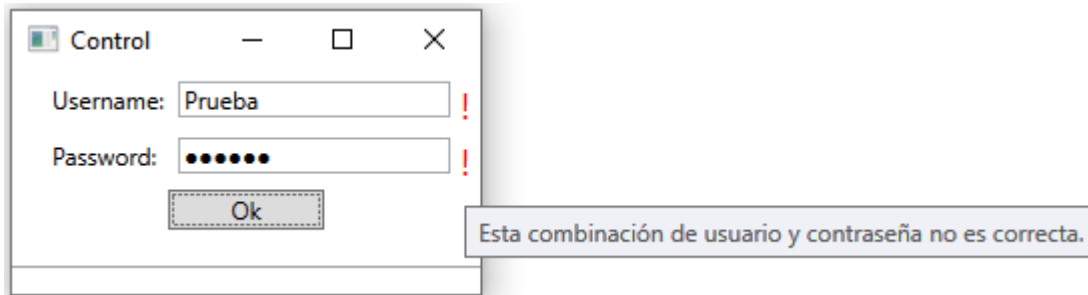


Tanto el campo TextBox como el PasswordBox están ligados a sus correspondientes propiedades en la ViewModel *LoginViewModel*.

Cabe destacar que el campo PasswordBox no permite ligar el campo a una propiedad de forma directa, por lo que se a usado una [propiedad adjunta](#) que simula el comportamiento del campo "Text" de las TextBox. La implementación para conseguirlo puede encontrarse en la clase *PasswordHelper* que se encuentra en la carpeta *Utils*.

También se incluye un control de tipo Button, ligado a un ICommand en el mismo *ViewModel*, que se habilita cuando ambos campos no están vacíos.

Al ser presionado comprobará si la combinación introducida existe en base de datos, devolviendo un error de no ser así en ambos campos, o una confirmación y abriendo la pantalla **Customer** si son correctos.



Pantalla - Customer

Esta pantalla está dividida en 3 partes:

- La parte superior contiene la búsqueda del cliente.
- La parte intermedia contiene los campos que conforman un cliente.
- La parte inferior con iconos que permiten realizar acciones sobre un cliente.

A screenshot of a Windows-style window titled "Customer". At the top, there is a search bar labeled "Code:" with a magnifying glass icon. Below the search bar is a horizontal line with three dots "...". Underneath this line is a form with several input fields, each with a label to its left: "First Name:", "Last Name:", "Username:", "Password:", "Country:", "Region:", "City:", and "Address:". At the bottom of the window, there is a row of four icons: a floppy disk (save), a document with a plus sign (add), a trash can (delete), and a hamburger menu (options).

Parte superior - Búsqueda de cliente

Esta parte está formada por una Label y una Textbox que permite al usuario introducir un código de cliente.

Junto a ello aparece un control Button, que al ser presionado buscará el código de cliente en la tabla Customer de la base de datos, de existir un cliente con ese ID, devuelve un MessageBox informando al usuario de que se ha cargado correctamente el cliente, y se Este botón sólo puede ser presionado si pasa todas las validaciones que se encuentran en la clase CustomerIDValidationRule.

De no cumplir mostrará un mensaje de error al usuario mostrando cual es el problema.

Parte intermedia - Información del cliente

En esta parte se muestra la información que conforman a un cliente.

Puede ser cubierta manualmente para crear un cliente, cubierta automáticamente al buscar un cliente, o modificada tras buscar un cliente.

Cada parte de la información del cliente está formada por una Label y un Textbox, salvo el campo "Password", que está formado por una Label y un PasswordBox, evitando así que se vea la contraseña en claro.

Todos los campos están ligados a una propiedad del ViewMododel *CustomerViewmodel*, y que a su vez contiene validaciones en sus correspondientes clases (que se pueden encontrar en la carpeta "Validations").

En caso de que un campo no cumpla las validaciones, se informará al usuario mediante una exclamación roja al lado del campo, que mostrará los errores al poner el ratón encima de ella.

Parte inferior - Acciones sobre un cliente

Esta parte está dedicada a lanzar acciones sobre un cliente.

Para ello está formada de 4 controles Button ligados a un ICommand en el ViewModel.

Partiendo de la izquierda a la derecha son:

- **Nuevo usuario:** Crea un cliente nuevo en base de datos.
Para que el botón se habilite es necesario que los campos de la parte intermedia no esten vacios y que esos campos pasen sus correspondientes validaciones.

Tras pulsar el botón se comprueba si el nombre de usuario ya está en uso, de estar en uso se levanta un error en el campo *Username* mostrando el error, si no existe se crea el cliente.

Tras crear el usuario se limpian los campos de la parte intermedia y se muestra un *MessageBox* confirmando la operación.

- **Guardar cambios:** Guarda la información del cliente que se haya introducido en los campos.

Para que el botón se habilite es necesario que los campos de la parte intermedia no estén vacíos y que esos campos pasen sus correspondientes validaciones.

Tras habilitarse el botón, se comprueba que el campo *Username* no exista ya en base de datos por otro usuario, si ya existe se muestra un aviso en el campo correspondiente, de no existir se actualiza la información en base de datos si la fecha en la que se recuperó la información del cliente es inferior a cuando se actualizó por última vez su información en la tabla *Customer* de la base de datos, mostrando un *MessageBox* con el resultado de la operación para informar al usuario y limpiando los campos de la parte intermedia.

- **Eliminar usuario:** Permite eliminar un cliente.

El botón se habilita si se ha cargado un cliente previamente.

De tener un cliente cargado, se intenta borrar de la tabla *Customer* de base de datos el cliente que tenga el mismo id que el cliente cargado, mostrando un *MessageBox* con el resultado de la operación para informar al usuario y limpiando los campos de la parte intermedia.

Importante: Esta operación produce un efecto en cadena y también borra las cuentas bancarias asociadas con el cliente si las tiene.

- **Mostrar cuentas:** Permite mostrar las cuenta/s bancaria/s relacionada/s con el cliente.

El botón se habilita si se ha cargado un usuario previamente y si el cliente tiene cuentas que mostrar, de no haberlo hecho o no tener cuentas muestra un *MessageBox* informando de ello al usuario.

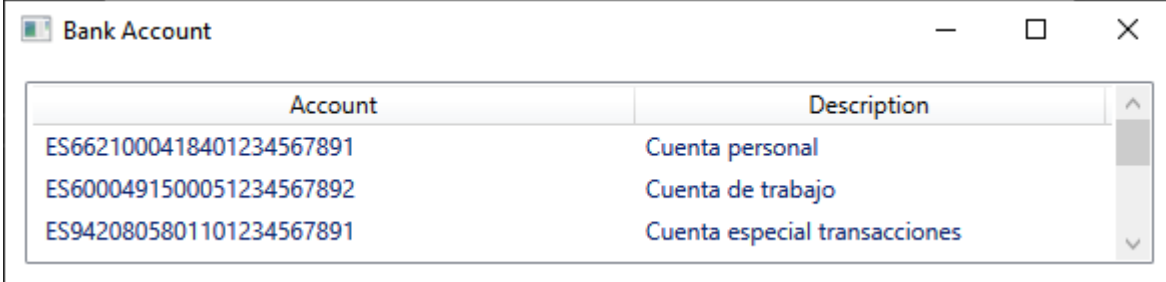
Abre la pantalla **Bank Account**.

Pantalla - Bank Account

Esta pantalla muestra las cuentas del cliente y su descripción.

Está formada por una ListView, con dos GridViewColumn para representar el número de cuenta y su descripción.

No es posible modificar los datos, tampoco es posible usar la pantalla "Customer" mientras esta ventana no se cierre.



The screenshot shows a window titled "Bank Account" with a standard Windows title bar (minimize, maximize, close buttons). Inside the window is a table with two columns: "Account" and "Description". The table contains three rows of data, each with a blue hyperlink-style account number and a corresponding description.

Account	Description
ES6621000418401234567891	Cuenta personal
ES6000491500051234567892	Cuenta de trabajo
ES9420805801101234567891	Cuenta especial transacciones

Base de datos

La base de datos está formada por 2 tablas, una almacena la información que define al cliente (Customer) y la otra las cuentas bancarias y descripciones.

Ambas tablas están relacionadas para permitir el cruce de datos.

Tabla - Customer

Contiene toda la información que define a un cliente, como su nombre, apellido, etc.

A mayores incluye el campo id, que aumenta cada vez que se crea un cliente, y el campo last_update, que nos permitirá tener un control de cuando se modifica la información, y permitir concurrencia e impedir que los usuarios actualicen datos de un mismo cliente sin haber obtenido la última versión de los datos antes.

Estas son las características de las columnas en base de datos

- id: int NOT NULL Autoincremental Primary Key
- first_name: nvarchar(50)
- last_name: nvarchar(50)
- username: nvarchar(50)
- password: nvarchar(100)
- country: nvarchar(100)
- region: nvarchar(100)
- city: nvarchar(250)
- last_update: datetime

Tabla - Account

Contiene la información de las cuentas bancarias, su descripción y cómo relacionarla con los clientes.

La columna `customer_id` está relacionada con la columna `id` de la tabla *Customer* para permitir identificar su propietario.

Estas son las características de las columnas en base de datos

- `id`: int NOT NULL Autoincremental Primary Key
- `customer_id`: int NOT NULL Foreign key (Customer.id)
- `account_number`: nvarchar(50)
- `description`: nvarchar(max)

Diagrama

