Documentación
SkiRent

Autor:
Christopher Bolocan

Indice

1. Arquitectura del Proyecto : MVC

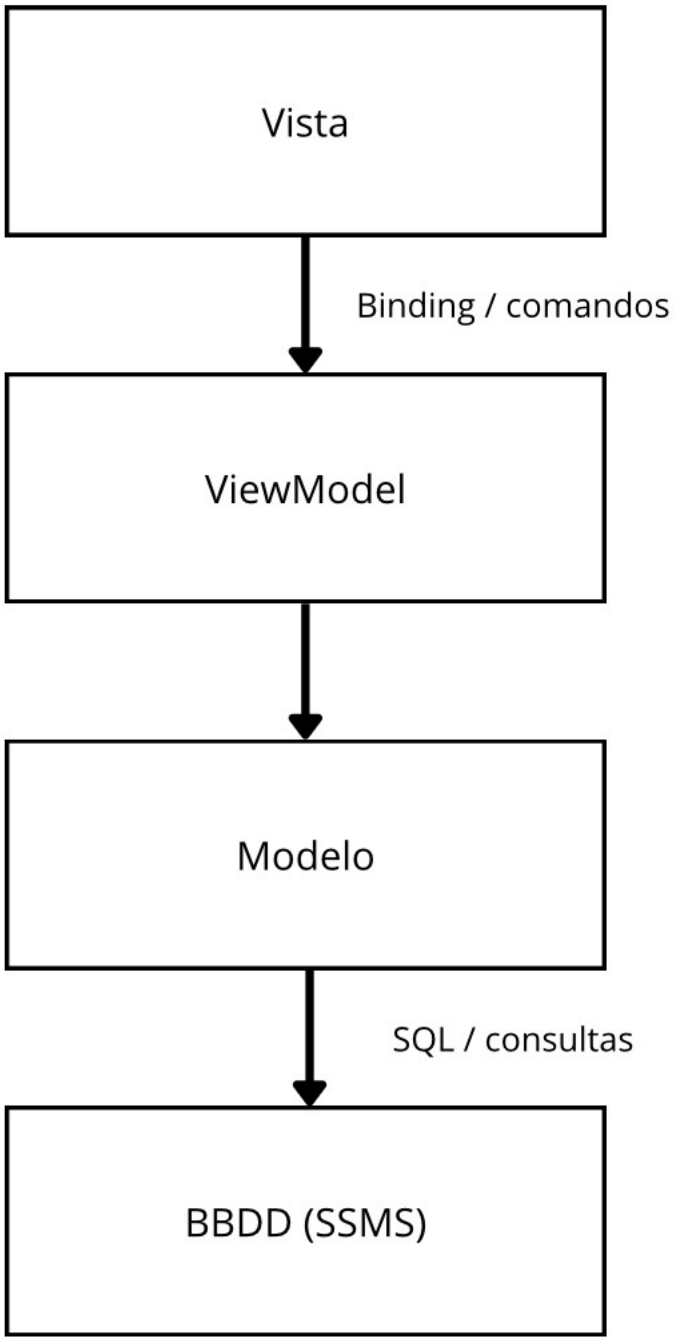
En este proyecto se ha utilizado el patrón MVC (Model – View – Controller), con el objetivo de mantener una estructura organizada, estructurada y fácil de ampliar en el futuro.

El patrón MVC permite separar claramente las responsabilidades de cada parte de la aplicación, evitando mezclar lógica de negocio con la interfaz gráfica o el acceso a datos.

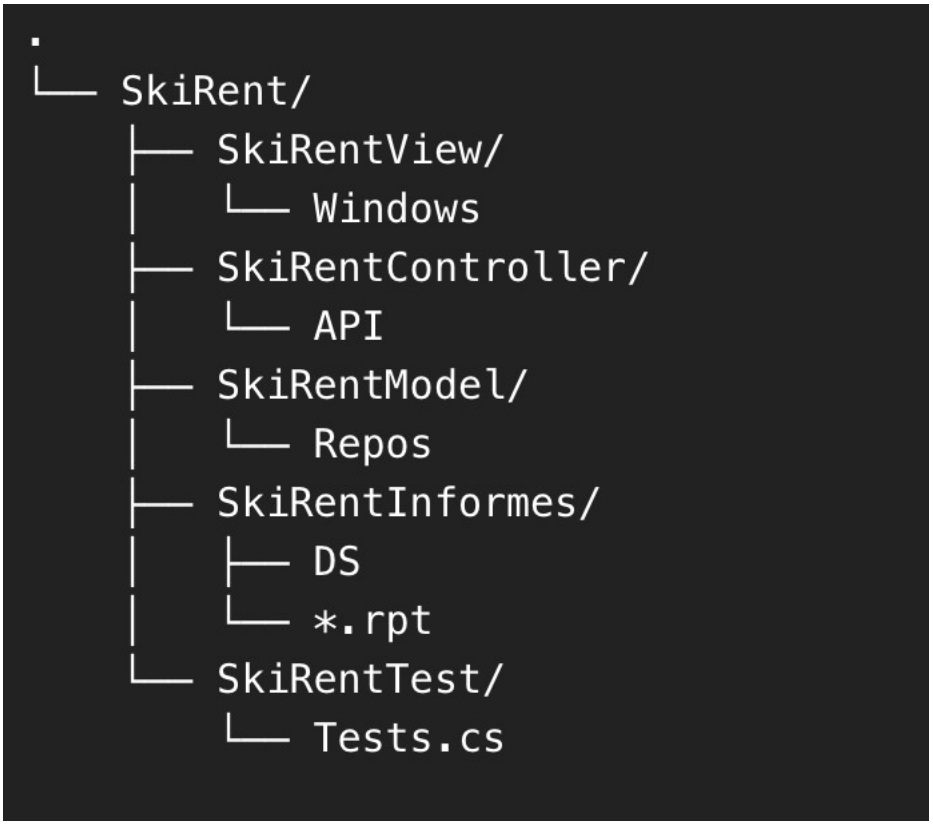
La arquitectura se divide en tres capas:

- **Capa de Presentación (View - WPF):**
  - **•Se ha utilizado Xaml para el diseño y C# para el code behind**
  - **•A traves de las ventanas se recoge la informacion del usuario**
  - **•Envia los datos al controlador.**
- **Capa de Lógica de Negocio (Controller / API):**
  - **•Es el centro de la aplicacion.**
  - **•Contiene las clases dentro de la carpeta API.**
  - **•Gestiona validaciones, reglas del negocio y control de stock.**
  - **•Actúa como intermediario entre la Vista y el Modelo.**
- **Capa de Datos (Model / Repositorios):**
  - **•Contiene los repositorios encargados de realizar las operaciones CRUD.**
  - **•Se comunica con SQL Server usando Entity Framework.**
  - **•No sabe nada de la de pantalla / UI ni datos obitene datos de alli directamente.**
- **Base de Datos (SQL Server):**
  - **•Almacena la información del sistema.**
  - **•Contiene las tablas Cliente, CategoriaMaterial, Material, Alquiler y LineaAlquiler.**

2. Diagrama de capas



3. Estructura de la solucion



En la solución no se han utilizado “carpetas” como tal , se han creado proyectos separados para tenerlo todo ordenado. Cada proyecto tiene su funcion.

- **SkiRentView**

  Aqui estan las ventanas que el usuario vera. Es el proyecto WPF. Se usa XAML para el diseño y el code behind para manejar eventos basicos de la interfaz como clicks y cargar datos en pantalla.

- **SkiRentController**

  Aqui esta la logica de negocio de la aplicacion. Contiene la carpeta API, donde estan las clases que actuan como controladores. Esta capa recibe las acciones de la vista, valida datos, aplica reglas como control de stock y llama al Model para acceder a la base de datos.

- **SkiRentModel**

  Aqui esta el acceso a datos. Contiene la carpeta Repos, con los repositorios que hacen las operaciones CRUD usando Entity Framework. Esta capa gestiona las consultas y devolucion de datos a los controllers.

- **SkiRentInformes**

  En este proyecto he creado los Datasets que son las listas internas del proyexto que alimentaran a los informes. Tambien se crean aquí los informes que se pueden ver desde la view.

- **SkiRentTest**

  Proyecto aparte para los tests con MSTest, para probar que ciertas cosas funcionan bien.Se han creado dos test unitarios y otro de integracion.

4. Explicacion de la Base de datos

La aplicacion SkiRent utiliza una base de datos en SQL Server llamada SkiRent.

Esta base de datos esta formada por 5 tablas principales:

Cliente
Guarda la informacion de los clientes que realizan alquileres.

CategoriaMaterial
Guarda las categorias del material (esquis, snowboard, botas, etc.).

Material
Guarda el inventario disponible, incluyendo precio por dia, stock y estado.

Alquiler
Representa cada alquiler realizado por un cliente, incluyendo fechas y estado.

LineaAlquiler
Guarda los materiales incluidos en cada alquiler, junto con los dias, cantidad y subtotal.

Relaciones

- **Un cliente puede tener varios alquileres.**
- **Un alquiler puede tener varias lineas.**
- **Un material puede aparecer en varias lineas de alquiler.**
- **Una categoria puede tener varios materiales.**

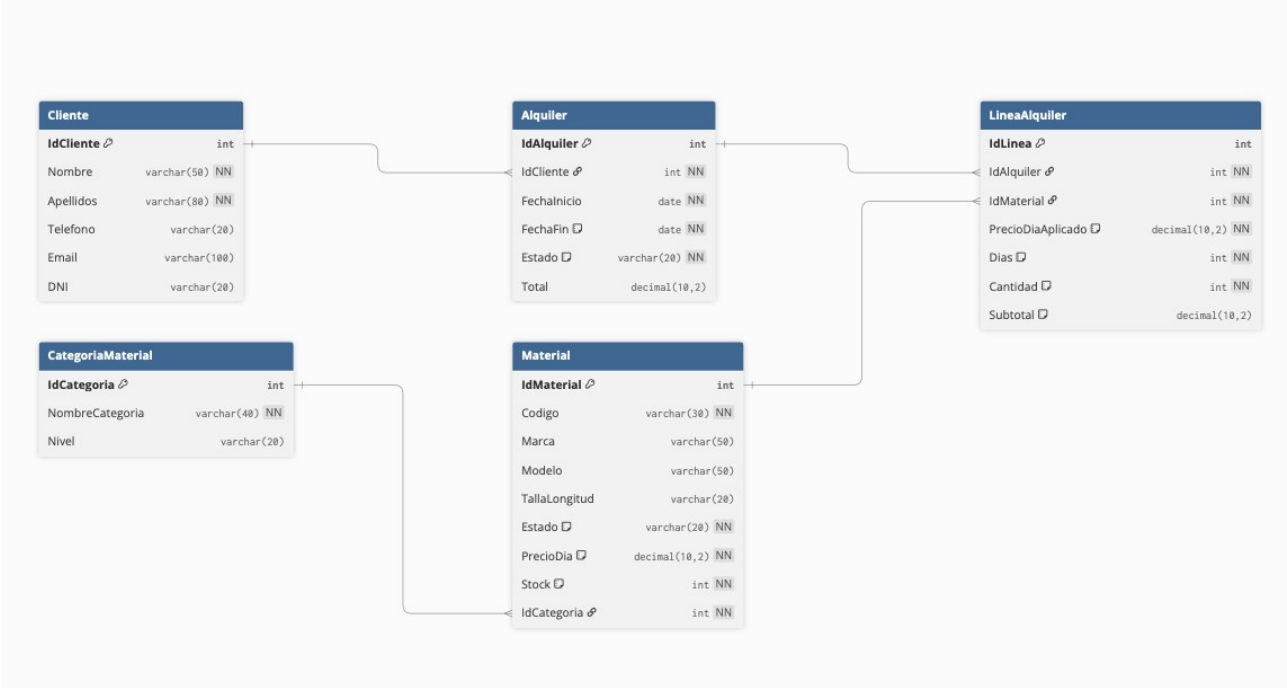
Restricciones

Se han aplicado restricciones para asegurar la integridad de los datos:

- **El precio y el stock no pueden ser negativos.**
- **La fecha fin no puede ser anterior a la fecha inicio.**
- **Los estados solo permiten valores definidos (Abierto, Cerrado, Cancelado, etc.).**
- **La cantidad y los dias deben ser mayores que cero.**

Con esta estructura se garantiza un correcto control del inventario y de los alquileres realizados.

5. Diagrama de la base de datos



El script de la BBDD se encuentra en el proyecto en:
github:https://github.com/Christopher-Blc/Proyecto_Final_WPF_SkiRent.git

6. Como ejecutar

Para ejecutar la aplicacion SkiRent correctamente, es necesario tener configurada la base de datos y las dependencias del proyecto en Visual Studio.

Requisitos previos

- **Visual Studio 2022 (o superior) con desarrollo de escritorio .NET**
- **SQL Server (LocalDB o instancia normal) y SSMS recomendado**
- **.NET Framework o .NET (segun el target del proyecto)**
- **Crystal Reports instalado si se van a visualizar los informes**

1. Crear la base de datos

1. **Abrir SQL Server Management Studio**
2. **Abrir una nueva query**
3. **Pegar y ejecutar el script de creacion de la BBDD (SkiRent.sql)**
4. **Verificar que se ha creado la base de datos SkiRent y sus tablas: Cliente, CategoriaMaterial, Material, Alquiler, LineaAlquiler**

2. Configurar la cadena de conexion

1. **Abrir el proyecto en Visual Studio**
2. **Localizar el archivo de configuracion donde este la connection string (normalmente App.config o el archivo de configuracion del Model)**
3. **Ajustar el nombre del servidor SQL para que coincida con el del equipo, por ejemplo:**
   - **(local)**
   - **.\SQLEXPRESS**
   - **(localdb)\MSSQLLocalDB**
4. **Guardar cambios**

3. Restaurar y compilar la solucion

1. **Abrir la solucion completa (WPF_SkiRent.sln)**
2. **Compilar la solucion con Build > Rebuild Solution**
3. **Comprobar que no hay errores de compilacion**

4. Establecer el proyecto de inicio

1. **En el Explorador de soluciones, hacer clic derecho sobre SkiRentView**
2. **Seleccionar Set as Startup Project**
3. **Ejecutar con F5 o el boton Start**

5. Ejecutar informes
Para visualizar los informes, abrir la ventana de informes desde el menu de la aplicacion.
Si Crystal Reports no esta instalado o falta alguna referencia, los informes pueden no cargarse, por lo que se recomienda tener Crystal Reports correctamente instalado en el entorno de desarrollo.

7. Herramientas utilizadas

Visual Studio 2022
Entorno de desarrollo integrado (IDE) utilizado para la programación de la aplicación en C#. Ha permitido la gestión de la solución con varios proyectos separados por capas (Model, Controller, View, Testing), facilitando la aplicación del patrón MVVM.

C# (.NET Framework)
Lenguaje principal del proyecto. Se ha utilizado para implementar la lógica de negocio, la gestión de datos y la interacción con la base de datos.

WPF (Windows Presentation Foundation)
Tecnología utilizada para el desarrollo de la interfaz gráfica. Permite la separación entre diseño (XAML) y lógica, así como el uso de layouts flexibles como Grid, StackPanel y DockPanel.

SQL Server
Sistema gestor de base de datos relacional utilizado para almacenar la información de clientes, materiales, alquileres y líneas de alquiler.

Entity Framework 6
ORM empleado para mapear las tablas de la base de datos a clases del modelo, facilitando las operaciones CRUD y reduciendo la necesidad de consultas SQL manuales.

Crystal Reports
Herramienta utilizada para la generación de informes. Se han desarrollado informes simples y de agrupamiento con totalización, cumpliendo los requisitos del proyecto.

Git y GitHub
Sistema de control de versiones utilizado para el seguimiento del desarrollo del proyecto. El código fuente se encuentra alojado en un repositorio privado.

Figma
Herramienta utilizada para el prototipado previo de las pantallas de la aplicación, permitiendo definir la estructura visual antes de la implementación en WPF.

MSTest / Proyecto de Testing
Se ha creado un proyecto específico de pruebas dentro de la solución para realizar tests unitarios y de integración sobre la lógica del sistema.