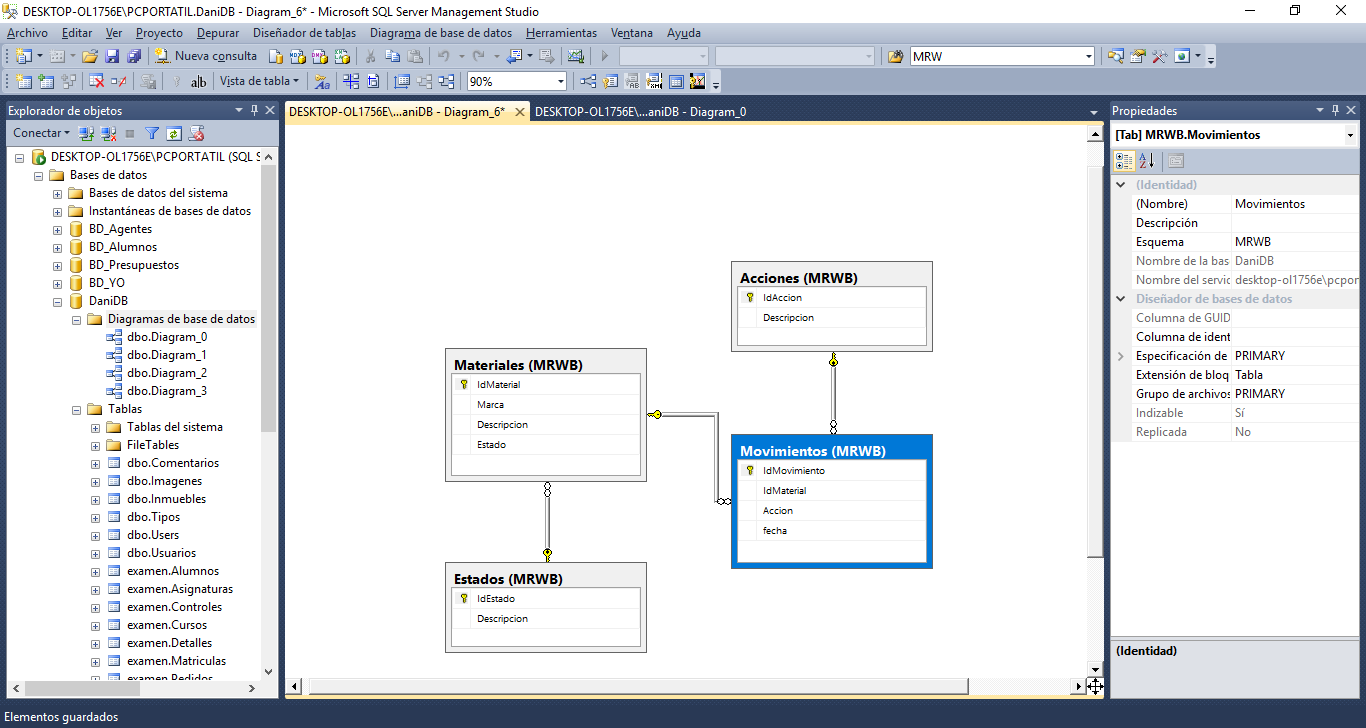
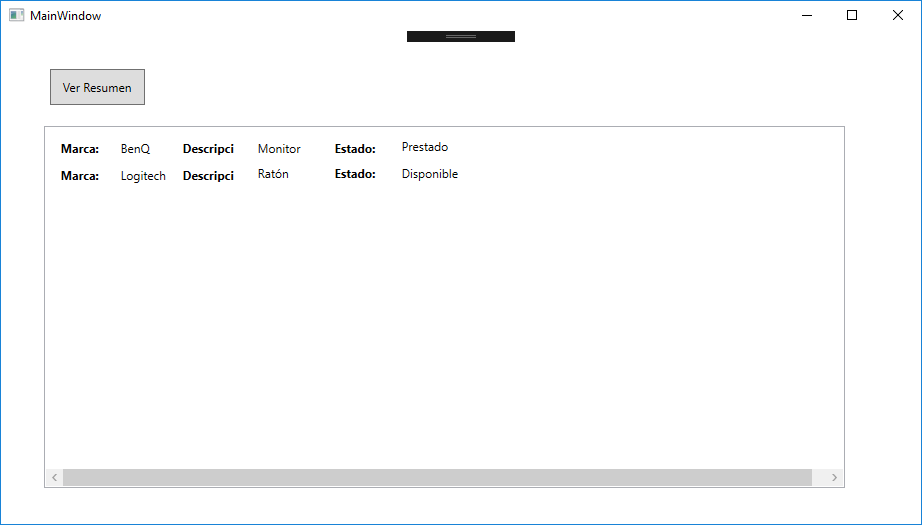
Queremos realizar una aplicación como la siguiente para controlar las entradas y salidas del material disponible en una empresa.

Para ello contamos con la siguiente base de datos.



* **Materiales** guarda los datos de los distintos materiales.
* **Estados** solo tiene 2 filas, una para “Prestado” y otra para “Disponible”
* **Movimientos** guarda la información de las retiradas y devoluciones de material.
* **Acciones** almacena 2 filas, una para “Retirar” y otra para “Devolver”.

1. Crear la clase MainViewModel para que tenga las siguientes características. (2.5 Puntos)
   * Tendrá una variable privada que será el modelo
   * Tendrá un constructor público que recibe el modelo para inicializar la variable modelo.
   * Tendrá una propiedad de solo lectura ListaMateriales, que devolverá la lista de todos los materiales del modelo.
2. La pantalla principal tendrá el siguiente aspecto: (2.5 Puntos)

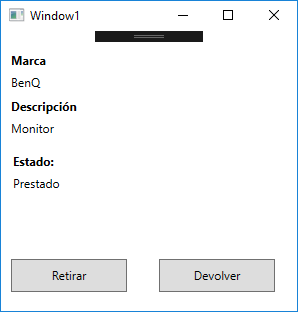


* + Al pulsar el botón ver resumen Creará un objeto de la clase MainViewModel creada en 1 y lo establecerá como datacontext del grid.
  + El otro control de la pantalla será un listbox. Mediante enlace de datos deberá visualizar la descripción y la marca de todos los materiales, mostrando además la descripción del estado correspondiente.
  + Al hacer doble click en la lista mostrará el formulario Window1 que se explica en el ejercicio 4, pasando el elemento elegido a dicho formulario.

1. **Programar el Objeto Window1View model: (2.5 Puntos)**

* Tendrá una variable privada para el modelo de datos.
* Una variable privada que almacenará el material Actual.
* Tendrá un constructor público que recibe el modelo y el material para inicializar las variables privadas. Lanzará el evento de cambio de las propiedades que se comentan a continuación.
* Propiedad pública de solo lectura con la descripción del material actual.
* Propiedad pública de solo lectura con la Marca del material actual.
* Propiedad pública de solo lectura con la **descripción** del estado del material actual.
* Propiedad pública booleana **prestado** que devuelve true si el estado es “prestado”, false en caso contrario
* Propiedad pública booleana **disponible** que devuelve true si el estado es “disponible”, false en caso contrario.
* Método público Devolver que:
  + - Pondrá el estado del Material a 1 (Disponible).
    - Añadirá un Movimiento con los siguientes datos:
      1. IdMovimiento: lo calculará como el máximo id de los existentes +1. Si no hay movimientos pondrá un 1.
      2. IdMaterial: será el id del material actual (usar la propiedad).
      3. Accion: Será el código correspondiente a “Devolver”.
      4. Fecha: Será la de hoy.
    - Actualizará el modelo.
* Método público Retirar que:
  + - pondrá el estado del trabajador a 2(“Prestado”).
    - Añadirá un Movimiento con los siguientes datos:
      1. IdMovimiento: lo calculará como el máximo id de los existentes +1. Si no hay movimientos pondrá un 1.
      2. IdMaterial: será el id del trabajador actual (usar la propiedad).
      3. Accion: Será el código correspondiente a “Retirar”.
      4. Fecha: Será la de hoy.
      5. Actualizará el modelo de datos.

1. **El formulario Window1 tendrá el siguiente aspecto:**



* + Programar la propiedad “MaterialActual” para recibir un objeto Material, que será el que se use para instanciar el Window1ViewModel.
  + Los controles Label mostrarán, mediante enlace de datos la marca y la descripción del material actual, así como la descripción del Estado.
  + Al cargarse el formulario, la propiedad enabled del botón Retirar estará enlazada a la propiedad **disponible** del viewModel y la propiedad enabled del botón Devolver estará enlazada a la propiedad **Prestado** del view model, por lo que solo permitirá devolver si está prestado, o retirar si está disponible.
  + Al pulsar el botón “Devolver” invocará al método Entrar del viewModel.
  + Al pulsar el botón “Retirar” invocará el método Salir del viewModel.

En ambos casos guardará cambios y se descargará el formulario.