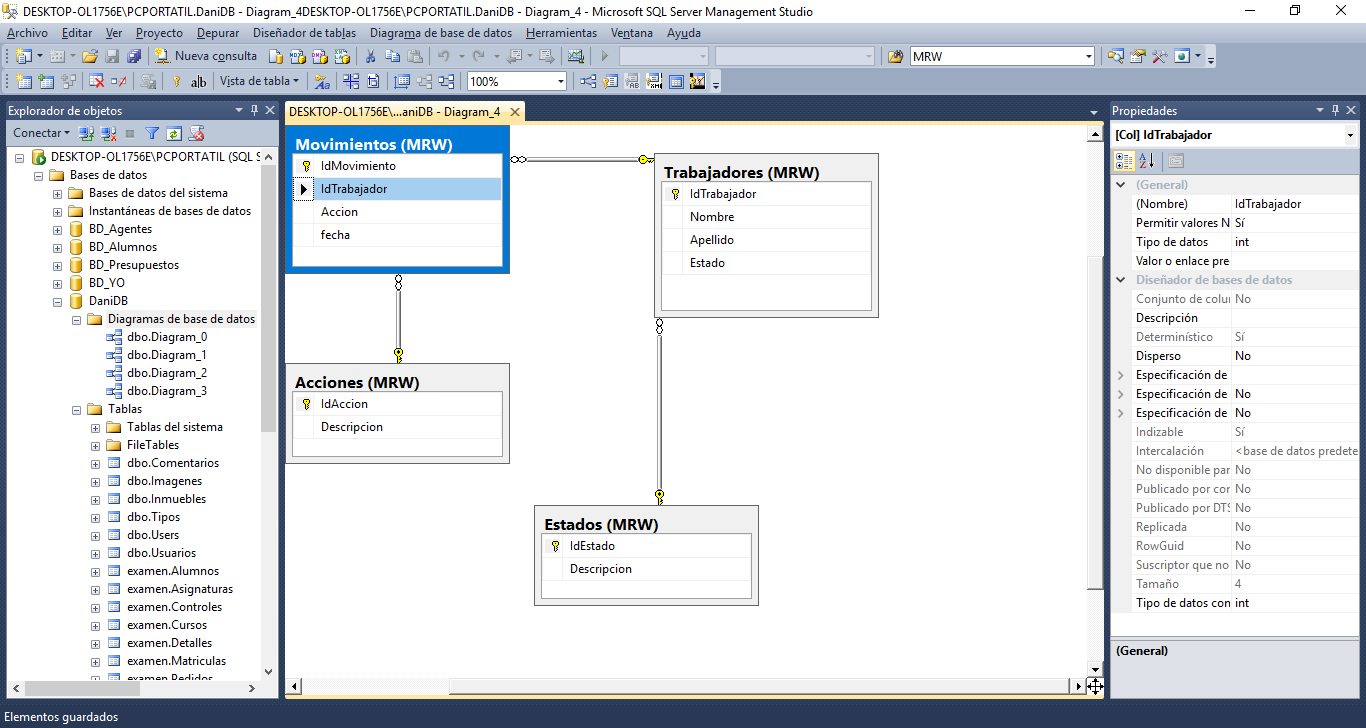
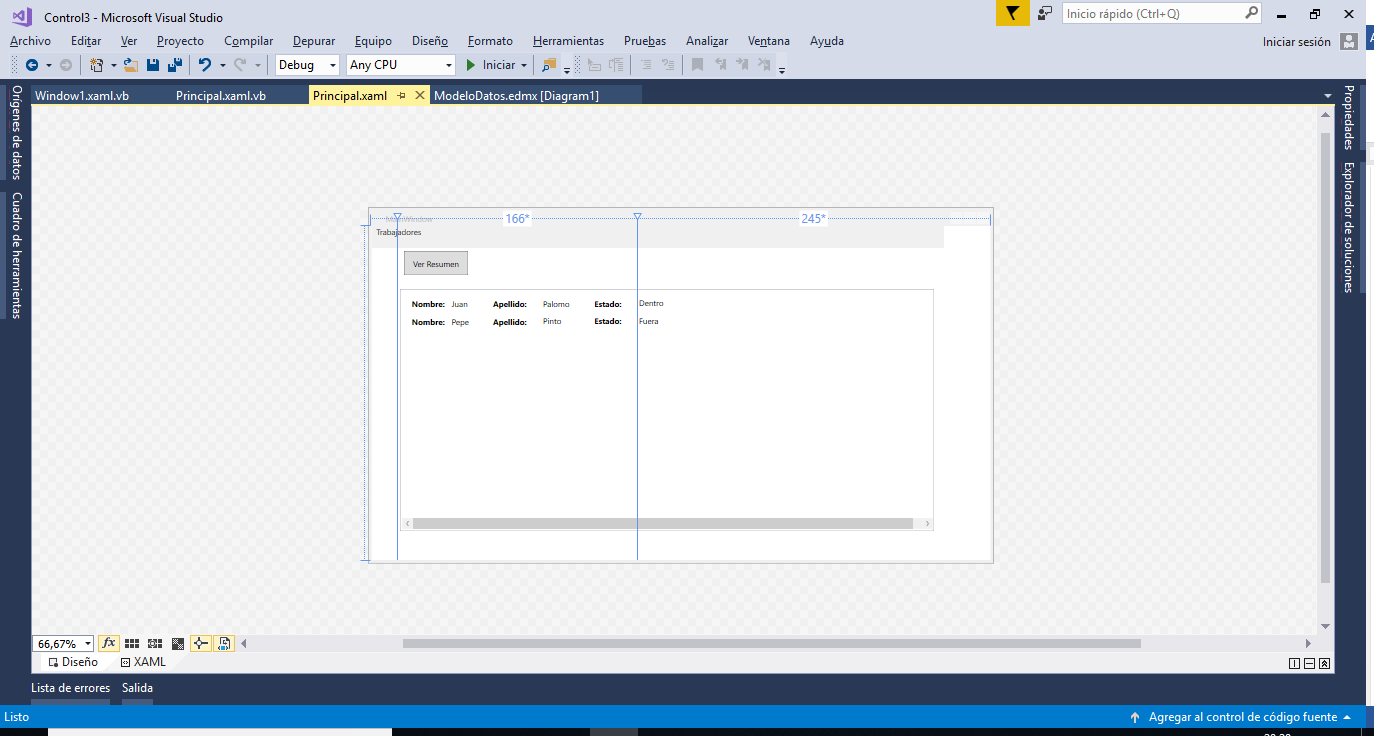
Queremos realizar una aplicación como la siguiente para controlar los trabajadores que entran y salen de una fábrica.

Para ello contamos con la siguiente base de datos.



* **Trabajadores** guarda los datos de los trabajadores.
* **Estados** solo tiene 2 filas, una para “Dentro” y otra para “Fuera”
* **Movimientos:** almacena las entradas y salidas de los usuarios.
* **Acciones:** Almacena dos filas, una para “Entrar” y otra para “Salir”

1. Crear la clase MainViewModel para que tenga las siguientes características.
   * 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 ListaEmpleados que devolverá todos Empleados del modelo.
2. **La pantalla principal tendrá el siguiente aspecto**:

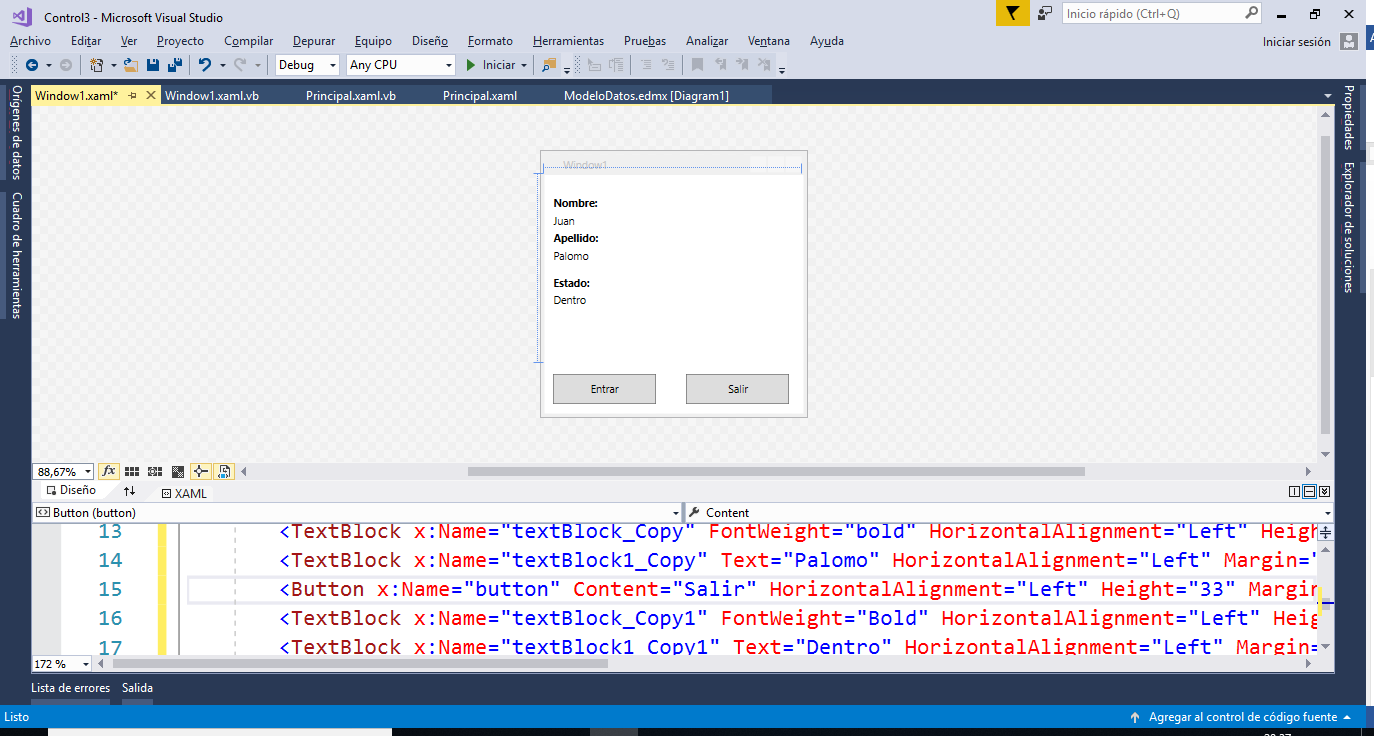


* + Al pulsar ver resumen Creará un objeto de la clase MainViewModel creada en 1 y lo establecerá como data context del grid.
  + El otro control de la pantalla será un listbox . Mediante enlace de datos deberá visualizar los nombres y apellidos de todos los trabajadores, mostrando además la descripción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

1. **Programar el Objeto Window1View model:**

* Tendrá una variable privada para el modelo de datos.
* Una variable privada que recibe el empleado Actual.
* Tendrá un constructor público que recibe el modelo y el usuario Actual inicializando ambas variables. Lanzará el evento de cambio de las propiedades que se comentan a continuación.
* Propiedad pública de solo lectura con el **nombre** del empleado
* Propiedad pública de solo lectura con el **apellido** del empleado.
* Propiedad pública de solo lectura con la **descripción** del estado del empleado.
* Propiedad pública booleana **dentro** que devuelve true si el estado es dentro, false en caso contrario
* Propiedad pública booleana **fuera** que devuelve true si el estado es fuera, false en caso contrario.
* Método público Entrar que:
  + - Pondrá el estado del trabajador a 1 (Dentro).
    - 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. IdTrabajador: será el id del trabajador actual (usar la propiedad).
      3. Accion: Será el código correspondiente a “Entrar”.
      4. Fecha: Será la de hoy.
    - Actualizará el modelo.
* Método público Salir que:
  + - pondrá el estado del trabajador a 2(Out).
    - 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. IdTrabajador: será el id del trabajador actual (usar la propiedad).
      3. Accion: Será el código correspondiente a “Salir”.
      4. Fecha: Será la de hoy.
      5. Actualizará el modelo.

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



* + Programar la propiedad “TrabajadorActual” para recibir un objeto Trabajador, que será el que se use para instanciar el Window1ViewModel.
  + Los controles Label mostrarán, mediante enlace de datos el nombre y el apellido del trabajador actual, así como la descripción del Estado.
  + Al cargarse el formulario, la propiedad enabled del botón entrar estará enlazada a la propiedad **fuera** del viewModel y la propiedad enabled del botón salir estará enlazada a la propiedad **dentro** del view model.
  + Al pulsar el botón “Entrar” invocará al método Entrar del viewModel.
  + Al pulsar el botón “Salir” invocará el método Salir del viewModel.

En ambos casos guardará cambios y se descargará el formulario.