PRUEBA TÉCNICA DESARROLLADOR WEB

Este documento tiene por objetivo dar los lineamientos principales para completar una serie de requerimientos sobre una solución de VISUAL STUDIO con SQL SERVER.

Para completar los requerimientos se contará con una Solución de Visual Studio pre-elaborada y que servirá como plantilla para completar los requerimientos y una Base de Datos con las estructuras iniciales. No debe preocuparse por cadenas de conexión.

TEMAS DE VALIDACIÓN

- Conocimientos Técnicos:
  - ASPNET MVC Y WEB API
  - ENTITY FRAMEWORK
  - SQL SERVER (PROGRAMACIÓN)
  - LENGUAJE DE PROGRAMACIÓN C#
  - HTML, JAVASCRIPT, AJAX, JQUERY
- Idea de Solución
- Cumplimiento de los Requerimientos
- Aplicación de Buenas Prácticas

REQUERIMIENTOS FUNCIONALES

Se desea crear una bitácora de Ingresos y Gastos para un grupo de personas con el fin de llevar un control de las finanzas.

Cada registro de Movimiento estará asociado a una persona y debe contar con los siguientes campos:

- Tipo de Movimiento (INGRESOS / GASTOS)
- Categoría (Alimentación, Transporte, Servicio Básico, etc.)
- Descripción Corta (Breve descripción del movimiento)
- Notas (Campo opcional para observaciones adicionales)
- Fecha del Movimiento
- Valor del Movimiento

Los Tipos de Movimientos son:

- INGRESOS: Representa un movimiento a favor
- GASTOS: Representa un movimiento negativo

Las Categorías agrupan a los movimientos con el fin de organizarlos y evaluarlos por categorías. Ejemplos de Categoría son:

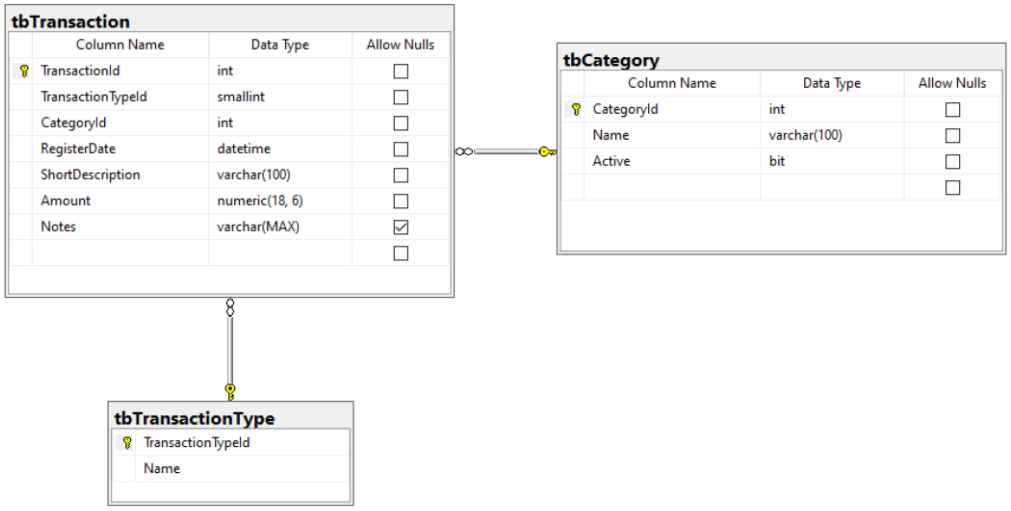
- Alimentación
- Transporte
- Educación

Se debe contar con un Resumen del Balance donde se incluyan los Saldos por Categoría.

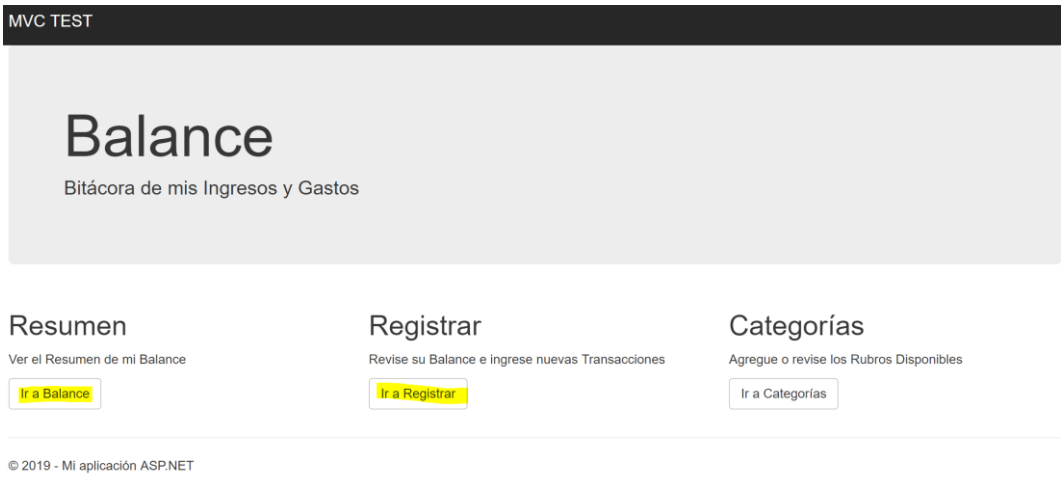
REQUERIMIENTOS ESPECIFICOS

Se contará con un diseño previo de la Base de Datos, este diseño no considera a la persona o la cuenta sobre la cual se registran los movimientos. Se deben realizar los cambios al diseño que considere necesario para cumplir con los requerimientos.

Diseño de Partida de Base de Datos:



Sistema de Partida:

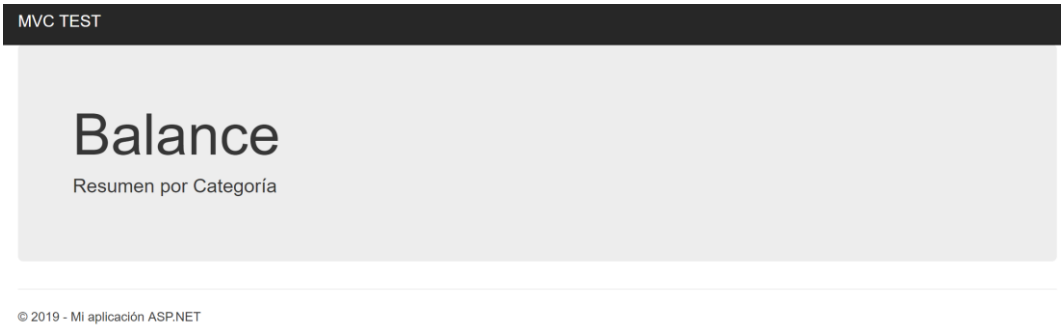


Se deben programar las páginas que son derivadas por los botones **Ir a Balance** para el reporte Resumen por Categorías, y el botón **Ir a Registrar** para el registro y edición de los Movimientos. El botón **Ir a Categorías** no requiere cambios y sirve para validar el flujo de comunicación entre los componentes de la aplicación.

El botón Ir a Registrar lo llevará a la página siguiente:

Complete el código para agregar y editar los movimientos

El botón **Ir a Balance** lo llevará a una página vacía donde deberá agregar el reporte del Resumen de Balance:



Requerimientos Funcionales Específicos:

1. El Sistema debe manejar como un dato de sesión una de las personas disponibles para referirse a los registros de movimientos y las consultas.
    a. Si no se ha elegido una persona, no se debe permitir la creación/edición/eliminación de movimientos o las consultas al resumen del Balance.
    b. Si ya se ha elegido una cuenta (mientras dure la sesión), el nombre de la cuenta debe estar siempre visible en cualquiera de las páginas del Sistema.
    c. Se debe ofrecer un mecanismo para poder cambiar la persona del Sistema en cualquier momento.
    d. Los Movimientos deben estar implícitamente filtrados para la persona que se eligió Dato de la Sesión, por lo que la Consulta de Movimientos y el Reporte de Resumen de Balance son afectados por este dato.
2. Se debe contar con una página para el registro de movimientos. Para crear/editar un movimiento se debe considerar lo siguiente:
    a. El editor debe mostrarse sobre la misma página de Registro de Movimientos, por lo que se debe mostrar usando un dialogo o popup manejado por javascript con el apoyo de cualquier librería como jquery u otros.

b. Grabar el Movimiento: Validar Fecha actual y no menor a 30 días, Valor del movimiento mayor a 0. Luego de Grabar el Movimiento, refrescar el Listado de Movimientos.

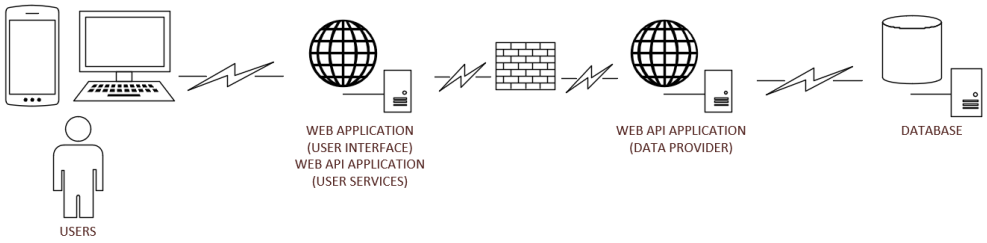
3. Página para el resumen del Balance:
   a. Deben mostrarse los resultados al cargar la página.
   b. El reporte es un detalle por categoría, ordenando de mayor a menor cuando el saldo por categoría es positivo, y de menor a mayor cuando el saldo de la categoría es negativo. Aquí se deben considerar los valores positivos y negativos de acuerdo con la categoría.
   Ejemplo (considere formato para los valores):

Categoría	**Saldo**
Remuneración	$ 1,000.00
Bonos	$ 200.00
Alimentación	($ 200.00)
Transporte	($ 100.00)
Total	**$ 900.00**

REQUERIMIENTOS TECNICOS

La solución debe ser desarrollada usando ASPNET MVC / WEB API junto con SQL SERVER.
Los requerimientos deben ser completados a partir de una solución pre-elaborada de Visual Studio y una base de datos existente. Donde se debe completar el diseño de la base de datos y las funciones faltantes de la aplicación WEB.

RESUMEN DE LA ARQUITECTURA

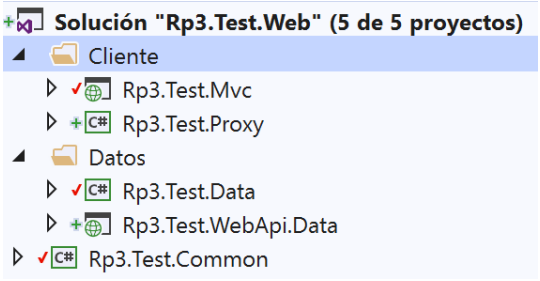


BASE DE DATOS

La Base de Datos tiene por nombre Rp3Test, para conectarse utilice el **usuario rp3test y clave rp3test**.

SOLUCIÓN VISUAL STUDIO

Proyectos:



Rp3.Test.Data (Acceso a Base de Datos)
Proyecto dedicado para el acceso a Datos mediante EntityFramework con SQL SERVER

Rp3.Test.WebApi.Data (Api para acceso a Datos)
Proyecto ASPNET WEB API, que se encarga de acceder a la Base de Datos utilizando la librería Rp3.Test.Data

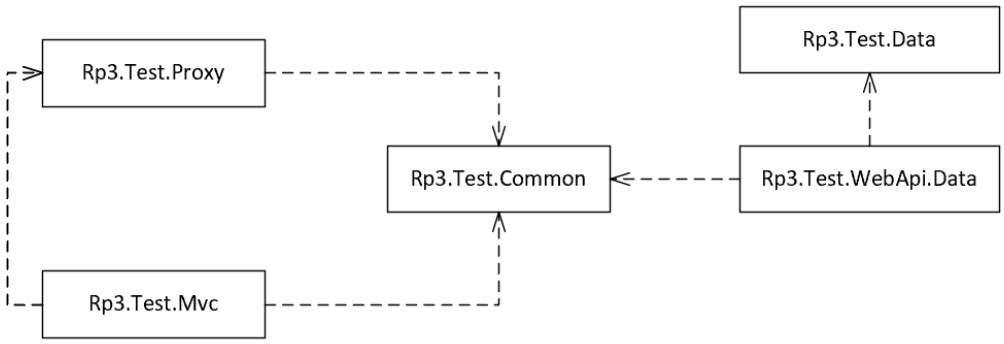
Rp3.Test.Common (Modelos para Transferencias de Datos)
Proyecto que define las clases(modelos) para que las aplicaciones Clientes (como Rp3.Test.Proxy) transmitan y reciban las respuestas del Api de Acceso a Datos(Rp3.Test.WebApi.Data).

Rp3.Test.Proxy
Proyecto responsable de armar el paquete para invocar al API de Acceso a datos(Rp3.Test.WebApi.Data).

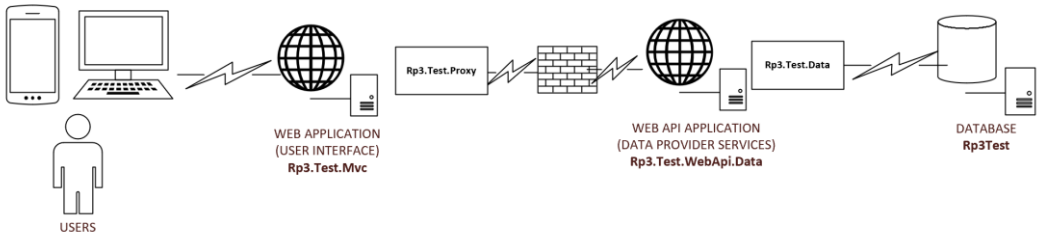
Rp3.Test.Mvc
Proyecto Cliente MVC para desarrollar la interfaz de usuario, y mediante Rp3.Test.Proxy lograr comunicarse con la API de Acceso a Datos Rp3.Test.WebApi.Data.

Relación de Dependencias entre Proyectos

La solución de Visual Studio ya mantiene estas relaciones de dependencia, las mismas que no deben ser modificadas.



ARQUITECTURA INCORPORANDO LOS PROYECTOS DE VISUAL STUDIO

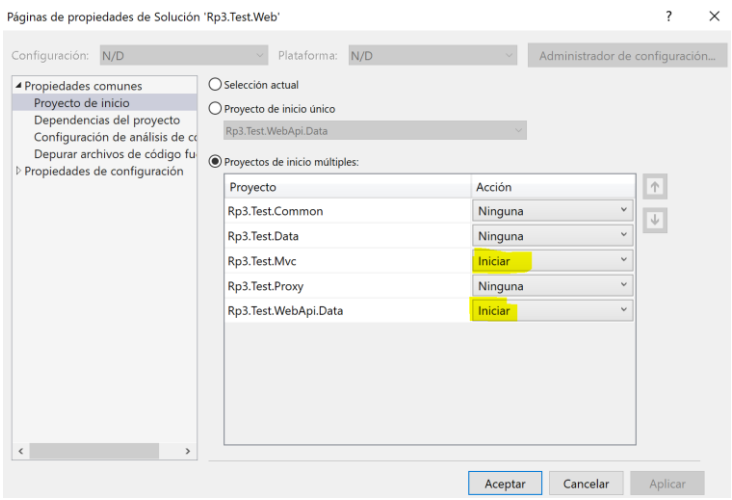


PLANTILLA DE LA SOLUCIÓN

La plantilla ya incluye conexión a la Base de Datos, los modelos para EntityFramework ya se encuentran definidos. Se deben incluir los modelos y propiedades que hagan falta a partir del diseño inicial.

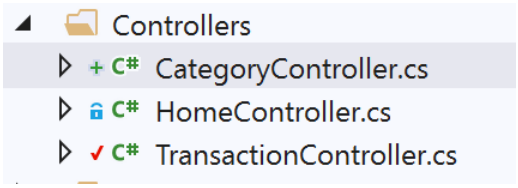
El programa puede ser ejecutado y se ejecuta con las funcionalidades básicas.

Considere que la solución de Visual Studio debe iniciar simultáneamente 2 proyectos:



Además de la comunicación con la base de datos, la Plantilla ya tiene definida la comunicación entre **Rp3.Test.Proxy** y **Rp3.Test.WebApi.Data.**

Controladores en **Rp3.Test.Mvc**



TransactionController esta destinado para desarrollar las funcionalidades del registro de Movimientos y el Reporte del Resumen del Balance.
CategoryController esta destinado para ofrecer los servicios para la funcionalidad de Categorías.
HomeController esta destinado para la carga de la página principal
Considere agregar las funcionalidades sobre los controladores existentes o cree nuevos Controladores de considerarlo necesario.

Importante

La plantilla existente (donde existen algunas funcionalidades creadas), le permitirá tener una referencia sobre el patrón utilizado para el desarrollo, los servicios que se utilizan y la forma en que se comunican los proyectos de la solución.

Ejemplo para Recuperar Datos a partir de un procedimiento Almacenado

El siguiente código puede utilizarlo como ayuda para ejecutar un procedimiento almacenado y guardar los datos en una Lista de objetos.

```
public List < Clase a retornar > GetBalance(int accountId, DateTime dateStart, DateTime dateEnd)
        {
            return this.DataBase.SqlQuery< Clase a retornar >
                ("EXEC dbo.spGetBalance @AccountId = {0}, @DateStart = {1}, @DateEnd = {2}", accountId,
dateStart, dateEnd).ToList();
        }
```