UNIVERSIDAD CASTRO CARAZO

BACHILLERATO EN INGENIERÍA INFORMÁTICA

MANUAL TÉCNICO DEL SISTEMA DE FARMACIA DEL PUEBLO

ANGIE PAMELA MUÑOZ JIMÉNEZ
CÉDULA DE IDENTIDAD: 1-1853-0642

IN3114 PROGRAMACIÓN DE COMPUTADORAS VI

PROFESORA: KARLA TATIANA BRENES CAMPOS

SAN ISIDRO, PEREZ ZELEDON
SAN JOSÉ, COSTA RICA
2023

Contenido

Frameworks.

Programación C# en:

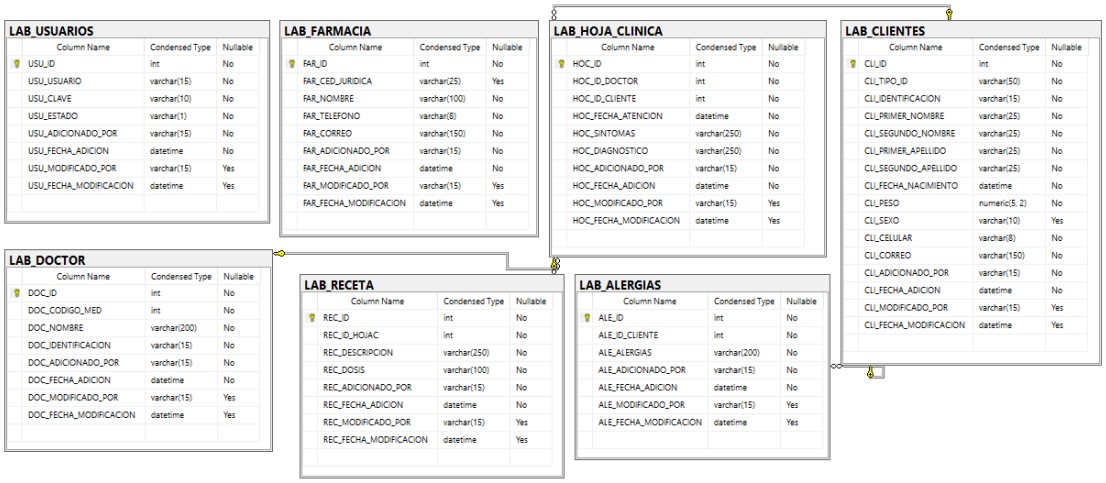
Visual Studio 2022. NETFramework, versión 4.7.2.

Base de Datos:

Microsoft SQL Server Management Studio 18.

Repositorio privado en GitHub.

Diagrama Entidad Relación.



Estándares de Base de Datos.

DESCRIPCIÓN	ESTÁNDAR	EJEMPLO
Nombre de la base de datos.	LABORATORIO	LABORATORIO
Nombre de las tablas.	LAB_nombre_tabla	LAB_USUARIOS
Nombre de los campos. 3 primeras letras referentes a la propia tabla.	USU_atributo	USU_ID
Nombre de las llaves primarias de las tablas. PK + Nombre de la tabla	PK_LAB_nombre_tabla	PK_LAB_USUARIOS
Nombre de las llaves foráneas de las tablas. Nombre de la tabla1 X Nombre de la tabla2.	tabla1_X_tabla2	CLI_X_HOC
Nombre de los checks constrain. CK + nombre de la tabla + atributo.	CK_LAB_nombre_tabla_atributo	CK_LAB_USUARIOS_ESTADO

Diccionario de Datos.

Reporte	Nombre del Servidor	Veterinaria	Nombre de la Tabla	Nombre del Esquema	Nombre de la Columna	Tipo de Dato	Caracteres Maximos	Permite Nulos	Es Auto-Incrementable	Descripcion
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_ID_CLIENTE	int	4	NO	NO	Funciona como llave fóranea para conectar la tabla de Cliente a las Alergias por medio del identificador.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_ALERGIAS	varchar	200	NO	NO	Funciona para almacenar la información de las alergias de los clientes.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_ALERGIAS	dbo	ALE_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_TIPO_ID	varchar	50	NO	NO	Funciona para almacenar la información del tipo de identificacion.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_IDENTIFICACION	varchar	15	NO	NO	Funciona para almacenar la información de la identificacion.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_PRIMER_NOMBRE	varchar	25	NO	NO	Funciona para almacenar la información del primer nombre.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_SEGUNDO_NOMBRE	varchar	25	NO	NO	Funciona para almacenar la información del segundo nombre.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_PRIMER_APELLIDO	varchar	25	NO	NO	Funciona para almacenar la información del primer apellido.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_SEGUNDO_APELLIDO	varchar	25	NO	NO	Funciona para almacenar la información del segundo apellido.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_FECHA_NACIMIENTO	datetime	8	NO	NO	Funciona para almacenar la información de la fecha de nacimiento.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_PESO	numeric	5	NO	NO	Funciona para almacenar la información del peso.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_SEXO	varchar	10	YES	NO	Funciona para almacenar la información del sexo.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_CELULAR	varchar	8	NO	NO	Funciona para almacenar la información del celular.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_CORREO	varchar	150	NO	NO	Funciona para almacenar la información del correo.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_CLIENTES	dbo	CLI_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_CODIGO_MED	int	4	NO	NO	Funciona para almacenar la información del código médico del doctor.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_NOMBRE	varchar	200	NO	NO	Funciona para almacenar la información del nombre del doctor.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_IDENTIFICACION	varchar	15	NO	NO	Funciona para almacenar la información de la identificación del doctor.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_DOCTOR	dbo	DOC_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_CED_JURIDICA	varchar	25	YES	NO	Funciona para almacenar la cédula jurídica de la farmacia.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_NOMBRE	varchar	100	NO	NO	Funciona para almacenar el nombre de la farmacia.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_TELEFONO	varchar	8	NO	NO	Funciona para almacenar el telefono de la farmacia.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_CORREO	varchar	150	NO	NO	Funciona para almacenar el correo de la farmacia.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_FARMACIA	dbo	FAR_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_ID_DOCTOR	int	4	NO	NO	Funciona como llave fóranea para conectar la tabla de Doctor a la Hoja Clínica por medio del identificador.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_ID_CLIENTE	int	4	NO	NO	Funciona como llave fóranea para conectar la tabla de Cliente a la Hoja Clínica por medio del identificador.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_FECHA_ATENCION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se atendio el cliente.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_SINTOMAS	varchar	250	NO	NO	Funciona para almacenar la información de los sintomas del cliente.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_DIAGNOSTICO	varchar	250	NO	NO	Funciona para almacenar la información el diagnostico del cliente.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_HOJA_CLINICA	dbo	HOC_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_ID_HOJAC	int	4	NO	NO	Funciona como llave fóranea para conectar la tabla de Hoja Clínica a la de Receta por medio del identificador.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_DESCRIPCION	varchar	250	NO	NO	Funciona para almacenar la información de la descripcion de la receta.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_DOSIS	varchar	100	NO	NO	Funciona para almacenar la información de la dosis de la receta.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_RECETA	dbo	REC_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_ID	int	4	NO	YES	Funciona como identificador único de la tabla.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_USUARIO	varchar	15	NO	NO	Funciona para almacenar la información del nombre de usuario.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_CLAVE	varchar	10	NO	NO	Funciona para almacenar la información de la clave de usuario.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_ESTADO	varchar	1	NO	NO	Funciona para almacenar la información del estado de usuario.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_ADICIONADO_POR	varchar	15	NO	NO	Funciona para almacenar la información del usuario que agregó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_FECHA_ADICION	datetime	8	NO	NO	Funciona para almacenar la fecha y hora en que se adicionaron los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_MODIFICADO_POR	varchar	15	YES	NO	Funciona para almacenar la información del usuario que modificó los datos.
DICCIONARIO DE DATOS	PAMELA_JIMENEZ\UMCA	LABORATORIO	LAB_USUARIOS	dbo	USU_FECHA_MODIFICACION	datetime	8	YES	NO	Funciona para almacenar la fecha y hora en que se modificaron los datos.

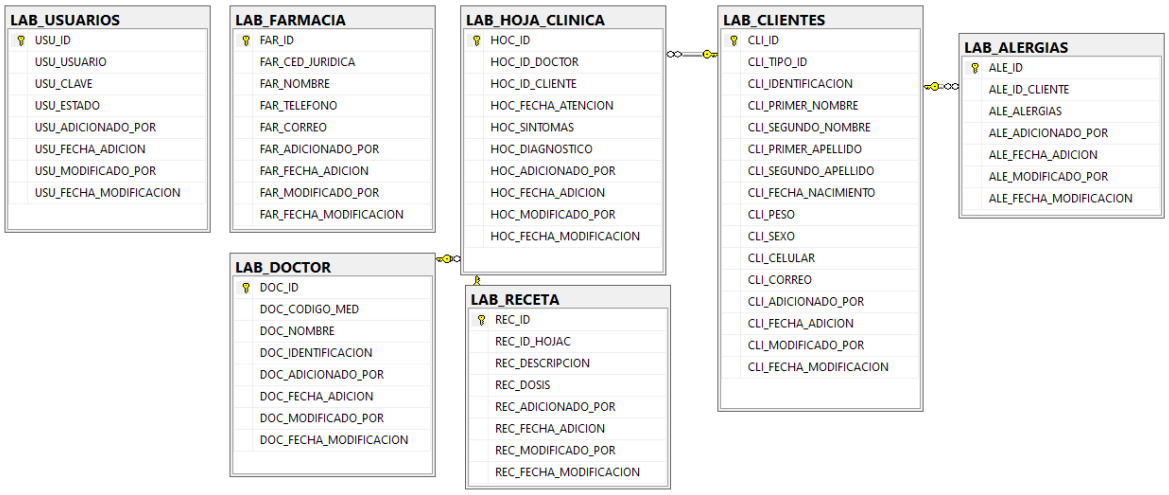
Scrips de Instalación.

```
-- CREACIÓN DE LA BASE DE DATOS.
CREATE DATABASE LABORATORIO;
USE LABORATORIO;

-- CREACIÓN DE LAS TABLAS.

-- TABLA CLIENTES.
CREATE TABLE LAB_CLIENTES
(
 CLI_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 CLI_TIPO_ID Varchar(50) NOT NULL,
 CLI_IDENTIFICACION Varchar(15) NOT NULL,
 CLI_PRIMER_NOMBRE Varchar(25) NOT NULL,
 CLI_SEGUNDO_NOMBRE Varchar(25) NOT NULL,
 CLI_PRIMER_APELLIDO Varchar(25) NOT NULL,
 CLI_SEGUNDO_APELLIDO Varchar(25) NOT NULL,
 CLI_FECHA_NACIMIENTO DATETIME NOT NULL,
 CLI_PESO Numeric(5,2) NOT NULL,
 CLI_SEXO Varchar(1) NOT NULL,
 CLI_CELULAR Varchar(8) NOT NULL,
 CLI_CORREO Varchar(150) NOT NULL,
 CLI_ADICIONADO_POR Varchar(15) NOT NULL,
 CLI_FECHA_ADICION Datetime NOT NULL,
 CLI_MODIFICADO_POR Varchar(15) NULL,
 CLI_FECHA_MODIFICACION Datetime NULL
)
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_CLIENTES ADD CONSTRAINT PK_LAB_CLIENTES PRIMARY KEY (CLI_ID)
GO

-- TABLA ALERGIAS.
CREATE TABLE LAB_ALERGIAS(
 ALE_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 ALE_ID_CLIENTE Int NOT NULL,
 ALE_ALERGIAS Varchar(200) NOT NULL,
 ALE_ADICIONADO_POR Varchar(15) NOT NULL,
 ALE_FECHA_ADICION Datetime NOT NULL,
 ALE_MODIFICADO_POR Varchar(15) NULL,
 ALE_FECHA_MODIFICACION Datetime NULL
)
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_ALERGIAS ADD CONSTRAINT PK_LAB_ALERGIAS PRIMARY KEY (ALE_ID)
GO

-- TABLA DOCTOR.
CREATE TABLE LAB_DOCTOR(
 DOC_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 DOC_CODIGO_MED Int NOT NULL,
 DOC_NOMBRE Varchar(200) NOT NULL,
 DOC_IDENTIFICACION VARCHAR(15) NOT NULL,
 DOC_ADICIONADO_POR Varchar(15) NOT NULL,
```

```
 DOC_FECHA_ADICION Datetime NOT NULL,
 DOC_MODIFICADO_POR Varchar(15) NULL,
 DOC_FECHA_MODIFICACION Datetime NULL
)
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_DOCTOR ADD CONSTRAINT PK_LAB_DOCTOR PRIMARY KEY (DOC_ID)
GO

-- TABLA HOJA CLINICA.
CREATE TABLE LAB_HOJA_CLINICA(
 HOC_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 HOC_ID_DOCTOR Int NOT NULL,
 HOC_ID_CLIENTE Int NOT NULL,
 HOC_FECHA_ATENCION DATETIME NOT NULL,
 HOC_SINTOMAS Varchar(250) NOT NULL,
 HOC_DIAGNOSTICO Varchar(250) NOT NULL,
 HOC_ADICIONADO_POR Varchar(15) NOT NULL,
 HOC_FECHA_ADICION Datetime NOT NULL,
 HOC_MODIFICADO_POR Varchar(15) NULL,
 HOC_FECHA_MODIFICACION Datetime NULL
)
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_HOJA_CLINICA ADD CONSTRAINT PK_LAB_HOJA_CLINICA PRIMARY KEY (HOC_ID)
GO

-- TABLA RECETA.
CREATE TABLE LAB_RECETA(
 REC_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 REC_ID_HOJAC Int NOT NULL,
 REC_DESCRIPCION Varchar(250) NOT NULL,
 REC_DOSIS Varchar(100) NOT NULL,
 REC_ADICIONADO_POR Varchar(15) NOT NULL,
 REC_FECHA_ADICION Datetime NOT NULL,
 REC_MODIFICADO_POR Varchar(15) NULL,
 REC_FECHA_MODIFICACION Datetime NULL
)
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_RECETA ADD CONSTRAINT PK_LAB_RECETA PRIMARY KEY (REC_ID)
GO

-- TABLA FARMACIA.
CREATE TABLE LAB_FARMACIA(
 FAR_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 FAR_CED_JURIDICA Int NOT NULL,
 FAR_NOMBRE Varchar(100) NOT NULL,
 FAR_TELEFONO Varchar(8) NOT NULL,
 FAR_CORREO Varchar(150) NOT NULL,
 FAR_ADICIONADO_POR Varchar(15) NOT NULL,
 FAR_FECHA_ADICION Datetime NOT NULL,
 FAR_MODIFICADO_POR Varchar(15) NULL,
 FAR_FECHA_MODIFICACION Datetime NULL
)
```

```
GO

-- LLAVE PRIMARIA.
ALTER TABLE LAB_FARMACIA ADD CONSTRAINT PK_LAB_FARMACIA PRIMARY KEY (FAR_ID)
GO

-- TABLA DE USUARIO.
CREATE TABLE LAB_USUARIOS
(
 USU_ID Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL,
 USU_USUARIO Varchar(15) NOT NULL,
 USU_CLAVE Varchar(10) NOT NULL,
 USU_ESTADO Varchar(1) NOT NULL,
 USU_ADICIONADO_POR Varchar(15) NOT NULL,
 USU_FECHA_ADICION Datetime NOT NULL,
 USU_MODIFICADO_POR Varchar(15) NULL,
 USU_FECHA_MODIFICACION Datetime NULL,
 CONSTRAINT CK_CR_USUARIOS_ESTADO CHECK (USU_ESTADO in ('A','I'))
)
GO

-- Add keys for table VT_USUARIOS

ALTER TABLE LAB_USUARIOS ADD CONSTRAINT PK_LAB_USUARIOS PRIMARY KEY (USU_ID)
GO

-- LLAVES FORANEAS

-- LAB_ALERGIAS - LAB_CLIENTES
ALTER TABLE LAB_ALERGIAS ADD CONSTRAINT CLI_X_ALER FOREIGN KEY (ALE_ID_CLIENTE)
REFERENCES LAB_CLIENTES (CLI_ID) ON UPDATE NO ACTION ON DELETE NO ACTION
GO

-- LAB_HOJA_CLINICA - LAB_DOCTOR
ALTER TABLE LAB_HOJA_CLINICA ADD CONSTRAINT DOC_X_HOC FOREIGN KEY (HOC_ID_DOCTOR)
REFERENCES LAB_DOCTOR (DOC_ID) ON UPDATE NO ACTION ON DELETE NO ACTION
GO

-- LAB_HOJA_CLINICA - LAB_CLIENTES
ALTER TABLE LAB_HOJA_CLINICA ADD CONSTRAINT CLI_X_HOC FOREIGN KEY (HOC_ID_CLIENTE)
REFERENCES LAB_CLIENTES (CLI_ID) ON UPDATE NO ACTION ON DELETE NO ACTION
GO

-- LAB_RECETA - LAB_HOJA_CLINICA
ALTER TABLE LAB_RECETA ADD CONSTRAINT HOC_X_REC FOREIGN KEY (REC_ID_HOJAC)
REFERENCES LAB_HOJA_CLINICA (HOC_ID) ON UPDATE NO ACTION ON DELETE NO ACTION

GO
```

Diagrama de Clases.



Descripción de Métodos.

Usuarios.

- Base de datos:

Se utiliza la tabla: LAB_USUARIOS.

- clsUsuario:

Contiene una cantidad de 8 atributos.

Contiene métodos set y get por cada atributo.

Contiene un método de contingencia para imprimir datos.

- dtoUsuario:

Contiene 4 métodos que se conectan por SQL:

validarIngreso: Recibe por parámetros un objeto del tipo de la clase usuario, se define un string con la consulta select, para validar que el usuario y clave ingresados existan en el sistema o estén correctos y finalmente se manda a ejecutar ante el SQL la consulta.

guardarUsuario: Recibe por parámetros un objeto del tipo de la clase usuario, se define un string con la consulta insert, para insertar todos los datos de un usuario y finalmente se manda a ejecutar ante el SQL la consulta.

consultarUsuario: Recibe por parámetros los campos de la pantalla usuarios y un objeto del tipo de la clase usuario y el usuario para saber cuál consultar, se define un string con la consulta select, para consultar todos los datos de un usuario y finalmente se manda a ejecutar ante el SQL la consulta.

modificarUsuario: Recibe por parámetros un objeto del tipo de la clase usuario y el usuario anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de un usuario y finalmente se manda a ejecutar ante el SQL la consulta.

- Usuario.xaml.cs:

Contiene un procedimiento que realiza el llamado de un método:

El procedimiento se encuentra en el botón de ingresar.

- Procedimiento validarIngreso:

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se crea un objeto del tipo de la clase usuarios.

Se invoca el data transfer object, en este caso sería el dtoUsuario.

Se realiza un if invocando el método de validarIngreso, en caso de que el método sea true se guardará el usuario ingresado en una variable global y se mostrará la siguiente pantalla, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Contiene un procedimiento para validar que solo se digiten letras en el campo usuario.

Contiene un procedimiento para validar que solo se digiten números y letras en el campo clave.

- Usuarios.xaml.cs:

Contiene 3 procedimientos que realizan llamado a los métodos.

2 procedimientos se encuentran en el botón aceptar.

Se crea una variable string para el estado del usuario.

Por medio de un if se valida que si está con el check activo el estado la variable valga “A” y si el check está inactivo la variable valga “I”.

Se validan que los campos no estén vacíos y en caso de que algún campo esté vació se mostrará un mensaje de advertencia.

Se invoca el data transfer object en este caso sería el dtoUsuario.

- Procedimiento guardarUsuario:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase usuario.

Se realiza un if invocando el método de guardarUsuario, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false mostrará un mensaje de advertencia.

Se limpian todos los campos.

- Procedimiento modificarUsuario:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase usuario.

Se realiza un if invocando el método de modificarUsuario, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false mostrará un mensaje de advertencia.

Se limpian todos los campos.

- Procedimiento consultarUsuario:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoUsuario.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el usuario consultado, se crea un objeto del tipo de la clase de usuario, se le asigna al usuario que su valor es el escrito en el campo de usuario y se invoca al método consultarUsuario.

Contiene un procedimiento para validar que solo se digiten letras en el campo usuario.

Contiene un procedimiento para validar que solo se digiten números y letras en el campo clave.

Clientes.

- Base de datos:

Se utiliza la tabla: LAB_CLIENTES.

- clsCliente:

Contiene una cantidad de 17 atributos.

Contiene métodos set y get por cada atributo.

Contiene un método de contingencia para imprimir datos.

- dtoCliente:

Contiene 3 métodos que se conectan por SQL:

insertarClientes: Recibe por parámetros un objeto del tipo de la clase cliente, se define un string con la consulta insert, para insertar todos los datos de un cliente y finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla cliente y un objeto del tipo de la clase cliente, se define un string con la consulta select, para consultar todos los datos de un cliente por medio del número de identificación y finalmente se manda a ejecutar ante el SQL la consulta.

modificarCliente: Recibe por parámetros un objeto del tipo de la clase cliente y el número de identificación anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de un cliente y finalmente se manda a ejecutar ante el SQL la consulta.

- Cliente.xaml.cs:

Contiene 3 procedimientos que realizan el llamado de los métodos:

2 procedimientos se encuentran el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoCliente.

- Procedimiento insertarClientes:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase cliente.

Se realiza un if invocando el método de insertarClientes, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarCliente:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase cliente.

Se realiza un if invocando el método de modificarCliente, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoCliente.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el número de identificación del cliente consultado, se crea un objeto del tipo de la clase de cliente, se le asigna al número de identificación del cliente que su valor es el escrito en el campo de número de identificación y se invoca al método consultaPorId.

Doctores.

- Base de datos:

Se utiliza la tabla: LAB_DOCTOR.

- clsDoctor:

Contiene una cantidad de 8 atributos.

Contiene métodos set y get por cada atributo.
Contiene un método de contingencia para imprimir datos.

- dtoDoctor:

Contiene 3 métodos que se conectan por SQL:

insertarDoctor: Recibe por parámetros un objeto del tipo de la clase doctor, se define un string con la consulta insert, para insertar todos los datos de un doctor y finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla doctor y un objeto del tipo de la clase doctor, se define un string con la consulta select, para consultar todos los datos de un doctor por medio del número de identificación y finalmente se manda a ejecutar ante el SQL la consulta.

modificarDoctor: Recibe por parámetros un objeto del tipo de la clase doctor y el número de identificación anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de un doctor y finalmente se manda a ejecutar ante el SQL la consulta.

- Doctor.xaml.cs:

Contiene 3 procedimientos que realizan el llamado de los métodos:

2 procedimientos se encuentran el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoDoctor.

- Procedimiento insertarDoctor:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase doctor.

Se realiza un if invocando el método de insertarDoctor, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarDoctor:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los

permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase doctor.

Se realiza un if invocando el método de modificarDoctor, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoDoctor.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el número de identificación del doctor consultado, se crea un objeto del tipo de la clase de doctor, se le asigna al número de identificación del doctor que su valor es el escrito en el campo de identificación y se invoca al método consultaPorId.

Alergias.

- Base de datos:

Se utiliza la tabla: LAB_ALERGIAS.

- clsAlergias:

Contiene una cantidad de 7 atributos.

Contiene métodos set y get por cada atributo.
Contiene un método de contingencia para imprimir datos.

- dtoAlergias:

Contiene 5 métodos que se conectan por SQL:

insertarAlergias: Recibe por parámetros un objeto del tipo de la clase alergias, se define un string con la consulta insert, para insertar todos los datos de una alergia de un cliente y finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla alergias y un objeto del tipo de la clase alergias, se define un string con la consulta select, para consultar todos los datos de una alergia por medio del identificador del cliente y finalmente se manda a ejecutar ante el SQL la consulta.

modificarAlergia: Recibe por parámetros un objeto del tipo de la clase alergias y el identificador anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de una alergia de un cliente y finalmente se manda a ejecutar ante el SQL la consulta.

consultarClientes: Recibe por parámetro el campo del combobox del cliente de la pantalla alergias, se define un string con la consulta select, para consultar todos los

identificadores de los clientes y finalmente se manda a ejecutar ante el SQL la consulta.

consultarCedCliente: Recibe por parámetro el identificador seleccionado y el campo del número de identificación del cliente en la pantalla alergias, se define un string con la consulta select, para consultar el número de identificación del cliente según el identificador seleccionado y finalmente se manda a ejecutar ante el SQL la consulta.

- Alergias.xaml.cs:

Contiene 5 procedimientos que realizan el llamado de los métodos:

- Procedimiento consultarClientes:

Contiene una función tipo void llamada cargarClientes.

Se invoca el data transfer object, en este caso sería el dtoAlergias.

Se llama el método consultarClientes.

Se inicializa la función para que se carguen los identificadores en el combobox de clientes.

2 procedimientos se encuentran el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoAlergias.

- Procedimiento insertarAlergias:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los

permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase alergias.

Se realiza un if invocando el método de insertarAlergias, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarAlergia:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase alergias.

Se realiza un if invocando el método de modificarAlergia, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoAlergias.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el identificador del cliente consultado, se crea un objeto del tipo de la clase de alergias, se le asigna al

identificador del cliente que su valor es el seleccionado en el combobox de clientes y se invoca al método consultaPorId.

- Procedimiento consultarCedCliente:

El procedimiento se encuentra en el evento SelectionChenged del combobox del cliente.

Guarda en una variable global el identificador del cliente seleccionado.

Se invoca el data transfer object en este caso sería el dtoAlergias.

Se invoca el método consultarCedCliente.

Hojas Clinicas.

- Base de datos:

Se utiliza la tabla: LAB_HOJA_CLINICA.

- clsHojaClinica:

Contiene una cantidad de 10 atributos.

Contiene métodos set y get por cada atributo.
Contiene un método de contingencia para imprimir datos.

- dtoHojaClinica:

Contiene 7 métodos que se conectan por SQL:

insertarHojaClinica: Recibe por parámetros un objeto del tipo de la clase hoja clínica, se define un string con la consulta insert, para insertar todos los datos de una hoja clínica y finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla hoja clínica y un objeto del tipo de la clase hoja clínica, se define un string con la consulta select, para consultar todos los datos de una hoja clínica por medio del identificador del cliente y finalmente se manda a ejecutar ante el SQL la consulta.

modificarHojaClinica: Recibe por parámetros un objeto del tipo de la clase hoja clínica y el identificador anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de una hoja clínica y finalmente se manda a ejecutar ante el SQL la consulta.

consultarDoctores: Recibe por parámetro el campo del combobox del doctor de la pantalla hoja clínica, se define un string con la consulta select, para consultar todos los identificadores de los doctores y finalmente se manda a ejecutar ante el SQL la consulta.

consultarCedDoctores: Recibe por parámetro el identificador seleccionado y el campo del número de identificación del doctor en la pantalla hoja clínica, se define un string con la consulta select, para consultar el número de identificación del doctor según el identificador seleccionado y finalmente se manda a ejecutar ante el SQL la consulta.

consultarClientes: Recibe por parámetro el campo del combobox del cliente de la pantalla hoja clínica, se define un string con la consulta select, para consultar todos los identificadores de los clientes y finalmente se manda a ejecutar ante el SQL la consulta.

consultarCedCliente: Recibe por parámetro el identificador seleccionado y el campo del número de identificación del cliente en la pantalla hoja clínica, se define un string con la consulta select, para consultar el número de identificación del cliente

según el identificador seleccionado y finalmente se manda a ejecutar ante el SQL la consulta.

- HojaClinica.xaml.cs:

Contiene 7 procedimientos que realizan el llamado de los métodos:

- Procedimiento consultarDoctores:

Contiene una función tipo void llamada consultarDoctores.

Se invoca el data transfer object, en este caso sería el dtoHojaClinica.

Se llama el método consultarDoctores.

Se inicializa la función para que se carguen los identificadores en el combobox de doctores.

- Procedimiento consultarClientes:

Contiene una función tipo void llamada cargarClientes.

Se invoca el data transfer object, en este caso sería el dtoHojaClinica.

Se llama el método consultarClientes.

Se inicializa la función para que se carguen los identificadores en el combobox de clientes.

2 procedimientos se encuentran el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoHojaClinica.

- Procedimiento insertarHojaClinica:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los

permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase hoja clínica.

Se realiza un if invocando el método de insertarHojaClinica, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarHojaClinica:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase hoja clínica.

Se realiza un if invocando el método de modificarHojaClinica, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoHojaClinica.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el identificador del cliente consultado, se crea un objeto del tipo de la clase de hoja clínica, se le asigna al

identificador del cliente que su valor es el seleccionado en el combobox de clientes y se invoca al método consultaPorId.

- Procedimiento consultarCedDoctores:

El procedimiento se encuentra en el evento SelectionChenged del combobox del doctor.

Guarda en una variable global el identificador del doctor seleccionado.

Se invoca el data transfer object en este caso sería el dtoHojaClinica.

Se invoca el método consultarCedDoctores.

- Procedimiento consultarCedCliente:

El procedimiento se encuentra en el evento SelectionChenged del combobox del cliente.

Guarda en una variable global el identificador del cliente seleccionado.

Se invoca el data transfer object en este caso sería el dtoHojaClinica.

Se invoca el método consultarCedCliente.

Recetas.

- Base de datos:

Se utiliza la tabla: LAB_RECETA.

- clsReceta:

Contiene una cantidad de 8 atributos.

Contiene métodos set y get por cada atributo.
Contiene un método de contingencia para imprimir datos.

- dtoReceta:

Contiene 4 métodos que se conectan por SQL:

insertarRecetas: Recibe por parámetros un objeto del tipo de la clase receta, se define un string con la consulta insert, para insertar todos los datos de una receta finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla receta y un objeto del tipo de la clase receta, se define un string con la consulta select, para consultar todos los datos de una receta por medio del identificador de la hoja clínica y finalmente se manda a ejecutar ante el SQL la consulta.

modificarReceta: Recibe por parámetros un objeto del tipo de la clase receta y el identificador anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de una receta y finalmente se manda a ejecutar ante el SQL la consulta.

consultarHojasClinicas: Recibe por parámetro el campo del combobox de las hojas clínicas de la pantalla receta, se define un string con la consulta select, para

consultar todos los identificadores de las hojas clínicas y finalmente se manda a ejecutar ante el SQL la consulta.

- Receta.xaml.cs:

Contiene 4 procedimientos que realizan el llamado de los métodos:

- Procedimiento consultarHojasClinicas:

Contiene una función tipo void llamada consultarHojasClinicas.

Se invoca el data transfer object, en este caso sería el dtoReceta.

Se llama el método consultarHojasClinicas.

Se inicializa la función para que se carguen los identificadores en el combobox de hojas clínicas.

2 procedimientos se encuentran en el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoReceta.

- Procedimiento insertarRecetas:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase receta.

Se realiza un if invocando el método de insertarRecetas, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarReceta:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase receta.

Se realiza un if invocando el método de modificarReceta, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoReceta.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global el identificador de la hoja clínica consultada, se crea un objeto del tipo de la clase de receta, se le asigna al identificador de la hoja clínica que su valor es el seleccionado en el combobox de hojas clínicas y se invoca al método consultaPorId.

Farmacias.

- Base de datos:

Se utiliza la tabla: LAB_FARMACIA.

- clsFarmacia:

Contiene una cantidad de 9 atributos.

Contiene métodos set y get por cada atributo.

Contiene un método de contingencia para imprimir datos.

- dtoFarmacia:

Contiene 3 métodos que se conectan por SQL:

insertarFarmacia: Recibe por parámetros un objeto del tipo de la clase farmacia, se define un string con la consulta insert, para insertar todos los datos de una farmacia y finalmente se manda a ejecutar ante el SQL la consulta.

consultaPorId: Recibe por parámetros los campos de la pantalla farmacia y un objeto del tipo de la clase farmacia, se define un string con la consulta select, para consultar todos los datos de la farmacia por medio de la cédula jurídica y finalmente se manda a ejecutar ante el SQL la consulta.

modificarFarmacia: Recibe por parámetros un objeto del tipo de la clase farmacia y la cédula jurídica anterior para saber cuál es el que se debe modificar, se define un string con la consulta update, para actualizar todos los datos de una farmacia y finalmente se manda a ejecutar ante el SQL la consulta.

- Farmacia.xaml.cs:

Contiene 3 procedimientos que realizan el llamado de los métodos:

2 procedimientos se encuentran el botón de aceptar.

Se validan que los campos se encuentren llenos, en caso de que alguno se encuentre vació se muestra un mensaje de advertencia.

Se invoca el data transfer object, en este caso el dtoFarmacia.

- Procedimiento insertarFarmacia:

Por medio de un if se valida que el radioButton “Insertar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase farmacia.

Se realiza un if invocando el método de insertarFarmacia, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento modificarFarmacia:

Por medio de un if se valida que el radioButton “Modificar” se encuentre chequeado, si este es true entonces valida que los caracteres digitados no sean mayores a los permitidos y si esto pasa entonces muestra un mensaje de advertencia, además se crea un objeto del tipo de la clase farmacia.

Se realiza un if invocando el método de modificarFarmacia, en caso de que el método sea true se mostrará un mensaje de éxito, pero en caso de que el método sea false se mostrará un mensaje de advertencia.

Se limpian los campos.

- Procedimiento consultaPorId:

El procedimiento se encuentra en el botón buscar.

Se invoca el data transfer object en este caso sería el dtoFarmacia.

Por medio de un if se valida que el radioButton “Consultar” se encuentre chequeado, si este es true entonces guarda en una variable global la cédula jurídica consultada, se crea un objeto del tipo de la clase de farmacia, se le asigna a la cédula jurídica de la farmacia que su valor es el escrito en el campo de cédula jurídica y se invoca al método consultaPorId.

Variables Globales.

Se encuentran en la carpeta llamada Utilidades.

Se encuentran diferentes clases.

- variablesGlobales:

Encontramos 4 funciones estáticas.

2 tipo string para almacenar los datos de guardarUsuario, guardarI.

1 tipo int para almacenar los datos de guardarId.

Y una de tipo escribirLog para almacenar la información del Log.

- util:

Contiene 2 funciones estáticas tipo void, una para validar que solamente ingresen letras y otra para que solamente ingresen números y letras.

- usuarioLogin:

Contiene una función estática tipo string para almacenar el usuarioLogueado.

- guardarUsuario:

Contiene una función estática tipo string para almacenar el guardarUsu.

- guardarId:

Contiene una función estática tipo string para almacenar el guardarIdentificacion.

- escribirLog:

Contiene las funciones necesarias para almacenar el log.