UNIVERSIDAD PRIVADA DE TACNA

INGENIERIA DE SISTEMAS

TITULO:

TRABAJO FINAL UNIDAD I - BIBLIOTECA

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Salamanca Contreras, Fiorella Rosmery	(2015053237)
Escalante Maron,Nelia	(2014049551)
Condori Gutierrez,Flor de Maria	(2015053227)
Coaquira Calizaya,Yerson	(2015053225)
Espinoza Caso,Lizbeth	(2011040667)

TRABAJO FINAL

SISTEMA BIBLIOTECA

1. PROBLEMA

Nos pide sistematizar una biblioteca, para una determinada reservación, o prestamos de algún libro y atención a los usuarios de la universidad de manera satisfactoria y darle una solución a través del visual studio.

1.1. Titulo Descriptivo del Proyecto

Para este proyecto, tiene como nombre BIBLIOTECA que esto forma parte de la sistematización de la biblioteca de la upt. podría ser en alguna otra biblioteca que lo necesiten este tipo de modelo, que será desarrollado de manera grupal y sería más que todo en nuestras casas o en la universidad misma donde lo desarrollaremos.

1.2. Formulación del Problema

Para la solución del problema de este dicho proyecto sería más que todo para que los usuarios se sientan conforme con la atención que le damos con este software y así no tenga ninguna queja al momento de ingresar a la biblioteca virtual.

2. MARCO TEÓRICO

2.1. MVC

En este diseño de biblioteca de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, Model, Views & Controllers, si lo prefieres en inglés.

MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

2.2. Entity Framework

Es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones orientadas a datos se han enfrentado a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos. Los datos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos subyacente donde se almacenan estos datos. Con Entity Framework, los desarrolladores pueden trabajar en un nivel mayor de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Dado que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework se pueden ejecutar en cualquier equipo en el que esté instalado .NET Framework a partir de la versión 3.5 SP1.

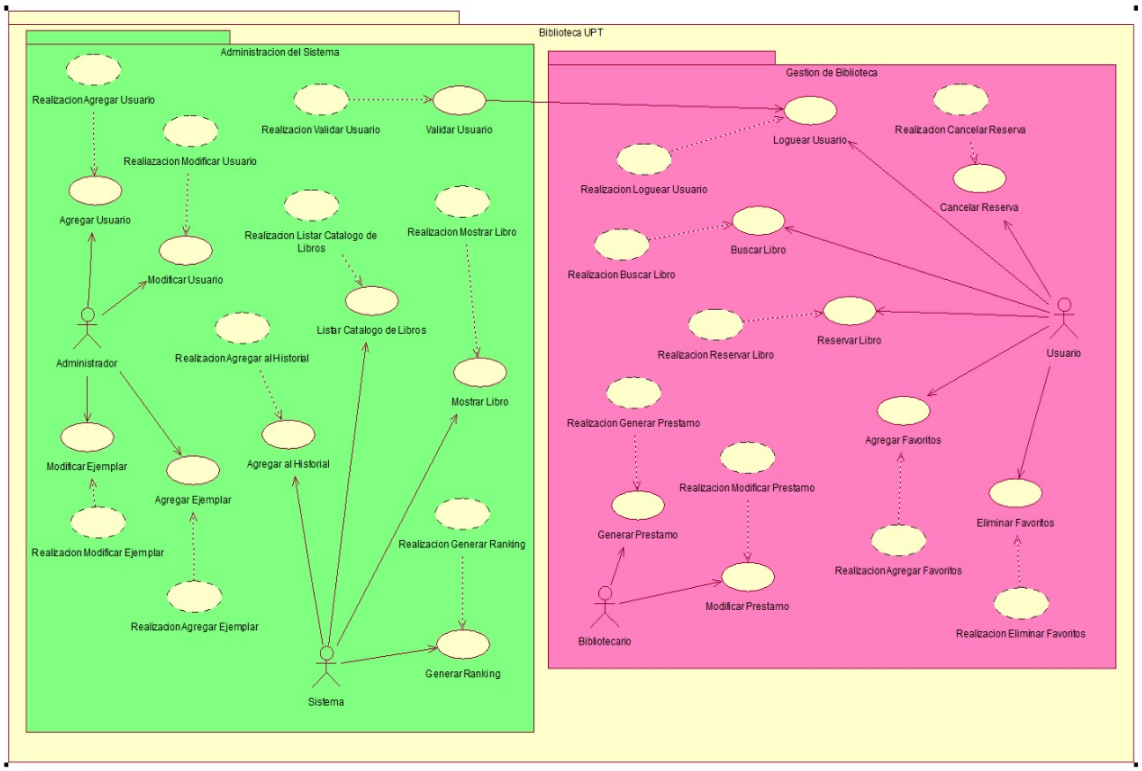
3. DESARROLLO

3.1. Análisis

- Requerimientos funcionales

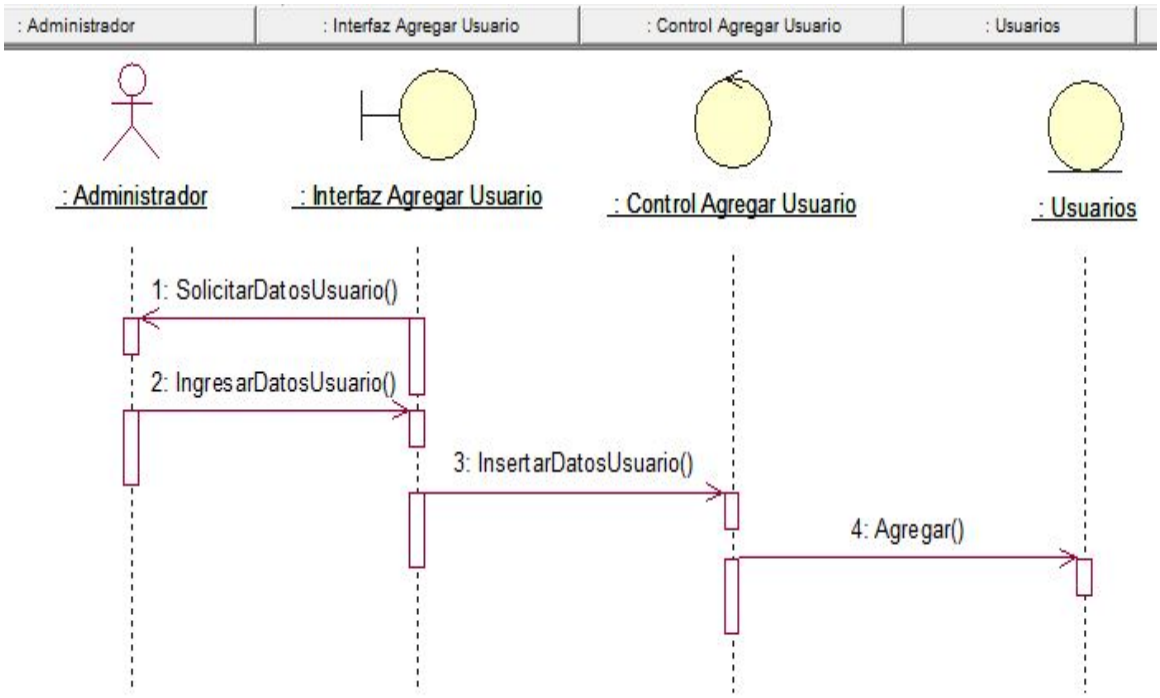
Módulos	Código	Requerimientos	Descripción	Prioridad
MANTENIMIENTO	RF-01	Buscar Libro	El sistema le permitirá al usuario realizar la búsqueda del libro, ya sea por el nombre del libro o alguna característica que el sistema tendrá en su interfaz.	ALTA
	RF-02	Reservar Libro	El sistema le permitirá al usuario realizar la reserva de un libro.	ALTA
	RF-03	Cancelar Libro	El sistema le permitirá al usuario cancelar el libro que ya reservo.	ALTA
	RF-04	Agregar Favorito	El sistema le permitirá al usuario poner en su lista los libros favoritos, según criterio de cada usuario.	ALTA
	RF-05	Eliminar Favoritos	El sistema le permitirá al usuario eliminar de los favoritos los libros que no sean de su agrado.	ALTA
	RF-06	Gestionar Libro	El sistema le permitirá al administrador realizar la parte de gestionar (actualizar, guardar, eliminar, etc).	ALTA
	RF-07	Registrar entrega libro	El sistema le permitirá al empleado registrar el préstamo del libro, el cual se le entregara al usuario.	ALTA
	RF-08	Registrar devolución libro	El sistema le permitirá al empleado registrar la devolución del libro del usuario.	ALTA
SEGURIDAD	RF-24	Autenticación	El usuario, administrador y empleado ingresar al sistema mediante la identificación de.	ALTA

- Caso de uso
  Este diagrama demuestra todos los requerimientos o casos de usos que existen en el proyecto de Bilioteca.

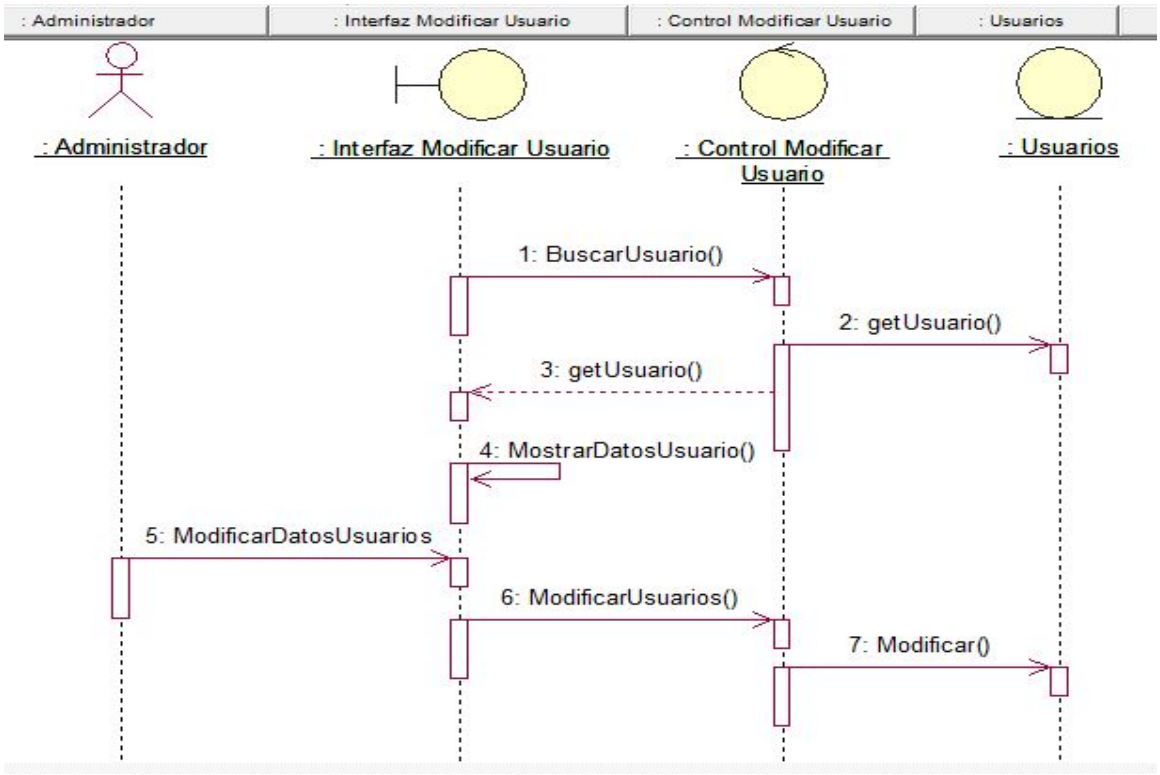


- Diagrama de secuencia

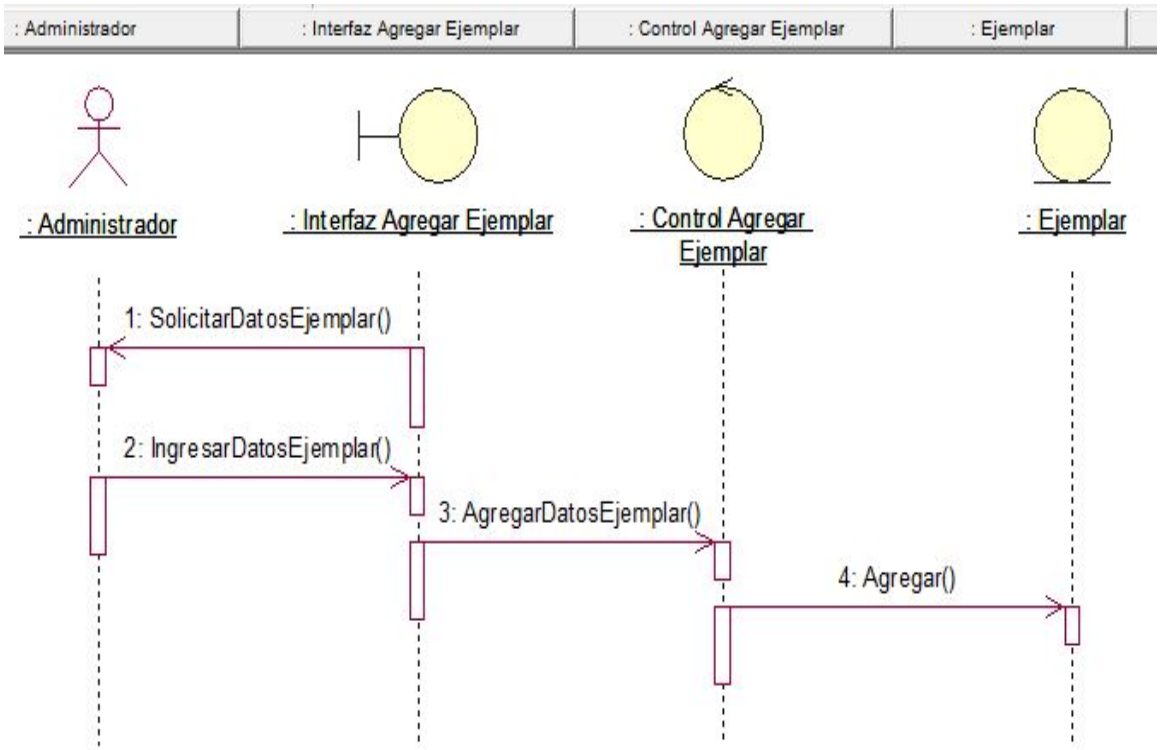
  Caso de Uso AGREGAR USUARIO



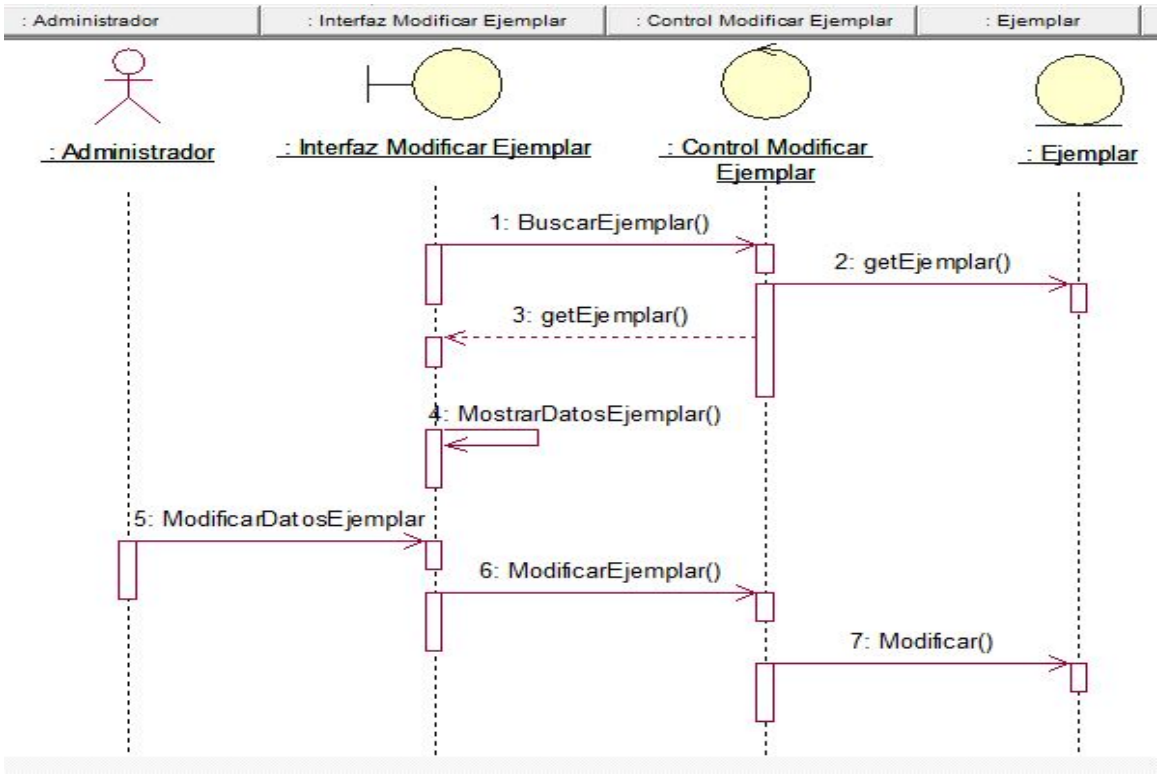
Caso de Uso MODIFICAR USUARIO



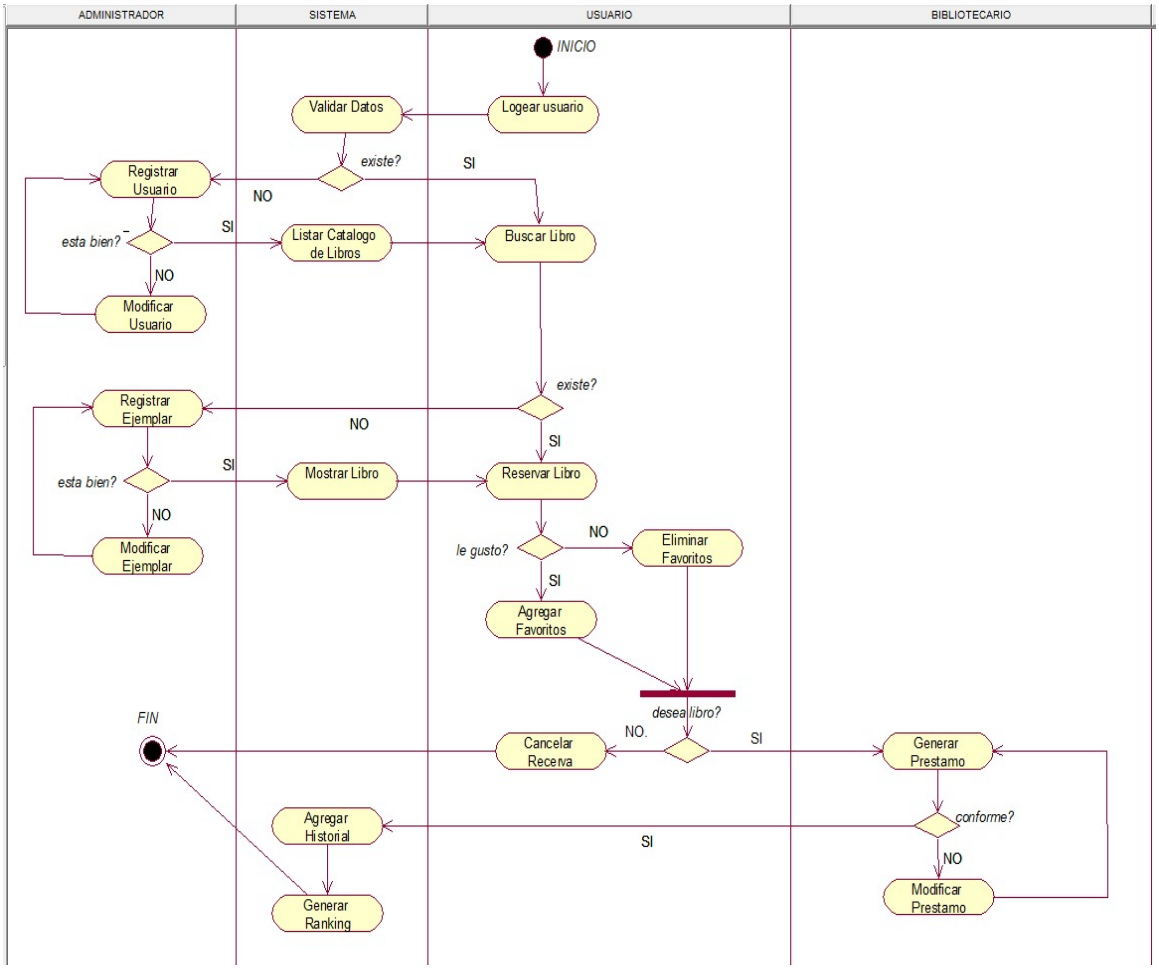
Caso de Uso AGREGAR EJEMPLAR



Caso de Uso MODIFICAR EJEMPLAR

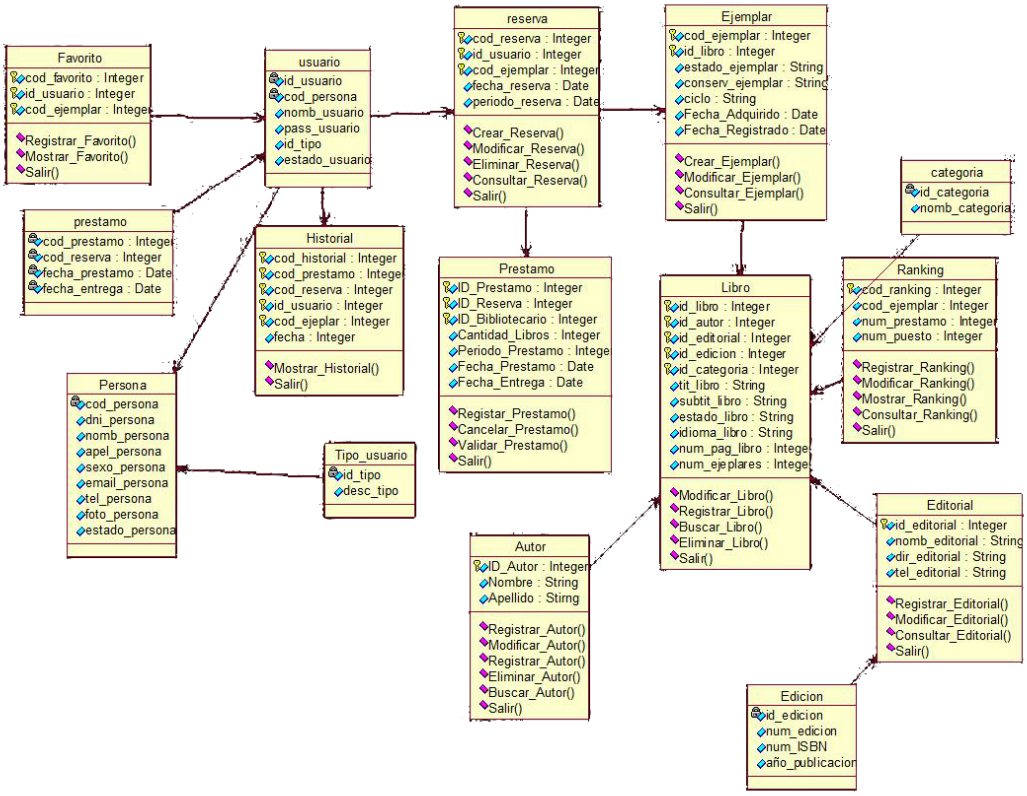


- Diagrama de Actividades
  Este diagrama de actividades propuesto, grafica de manera técnica las actividades que realizara el sistema de acuerdo con los casos de uso.

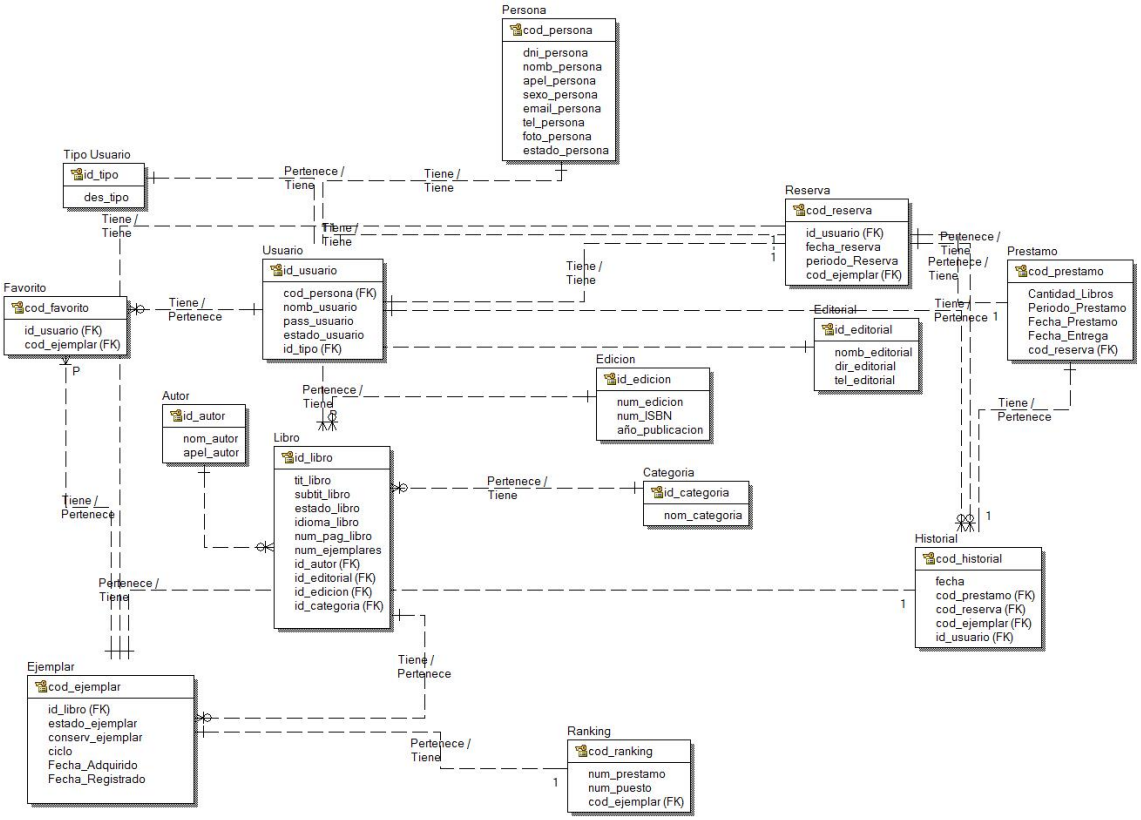


3.2. Diseño

- Diagrama de clases
  Este diagrama de clases de clases muestra todos las clases u objetos que estan presentes en el proyecto de Bilioteca.



- Modelo entidad relación
  Este diagrama es muestra la cardinalidad, relaciones y atributos de las Clases existentes en el proyecto. Este diagrama fue hecho en Erwin Data Modeler.



3.3. PRUEBAS UNITARIAS

Esta Prueba Unitaria corresponde a la clase Autor. Esta prueba unitaria pone a prueba el metodo "CrearAutor()" declarado en la clase.

```
[TestClass]
0 referencias
public class Test
{
    [TestMethod]
    0 referencias
    public void CrearAutor()
    {
        var autorcito = ClsAutor.RegistrarAutor("upt", "Perez");
        var reposit = new Repositorio_Infraestructura();
        reposit.Adicionar<ClsAutor>(autorcito);
        reposit.GuardarCambios();
        Assert.IsNotNull(autorcito);
        Assert.IsTrue(autorcito.nom_autor == "upt");
    }
```

Esta Prueba Unitaria corresponde a la clase Libro. Esta prueba unitaria pone a prueba el metodo "CrearLibro()" declarado en la clase.

```
[TestMethod]
0 referencias
public void CrearLibro()
{
    var librito = ClsLibro.RegistrarLibro(1, 1, 1, 1, "Harry Potter", "Caliza de Fuego", "Ingles", 520, 3);
    var reposit = new Repositorio_Infraestructura();
    reposit.Adicionar<ClsLibro>(librito);
    reposit.GuardarCambios();
    Assert.IsNotNull(librito);
    Assert.IsTrue(librito.id_autor == 1);
}
```

3.4. MAPEO

Clase Autor:

```
public class AutorMapeo : EntityTypeConfiguration<ClsAutor>
{
    1 referencia
    public AutorMapeo()
    {
        ToTable("TBL_AUTOR");
        HasKey(p => p.id_autor);
        Property(p => p.id_autor).HasColumnName("ID_AUTOR");
        Property(p => p.nom_autor).HasColumnName("NOM_AUTOR").HasMaxLength(250);
        Property(p => p.apel_autor).HasColumnName("APEL_AUTOR").HasMaxLength(250);
    }
}
```

Clase Categoria:

```
public class CategoriaMapeo : EntityTypeConfiguration<ClsCategoria>
{
    1 referencia
    public CategoriaMapeo()
    {
        ToTable("TBL_CATEGORIA");
        HasKey(p => p.id_categoria);
        Property(p => p.id_categoria).HasColumnName("ID_CATEGORIA");
        Property(p => p.nomb_categoria).HasColumnName("NOMB_CATEGORIA").HasMaxLength(250);
    }
}
```

Clase Edición:

```
public class EdicionMapeo : EntityTypeConfiguration<ClsEdicion>
{
    1 referencia
    public EdicionMapeo()
    {
        ToTable("TBL_EDICION");
        HasKey(p => p.id_edicion);
        Property(p => p.id_edicion).HasColumnName("ID_EDICION");
        Property(p => p.num_edicion).HasColumnName("NUM_EDICION").HasMaxLength(250);
        Property(p => p.num_ISBN).HasColumnName("NUM_ISBN").HasMaxLength(350);
        Property(p => p.anio_publicacion).HasColumnName("AÑO_PUBLICACION").HasMaxLength(5);
    }
}
```

Clase Editorial:

```
public class EditorialMapeo : EntityTypeConfiguration<ClsEditorial>
{
    1 referencia
    public EditorialMapeo()
    {
        ToTable("TBL_EDITORIAL");
        HasKey(p => p.id_editorial);
        Property(p => p.id_editorial).HasColumnName("ID_EDITORIAL");
        Property(p => p.nomb_editorial).HasColumnName("NOMB_EDITORIAL").HasMaxLength(250);
        Property(p => p.dir_editorial).HasColumnName("DIR_EDITORIAL").HasMaxLength(550);
        Property(p => p.tel_editorial).HasColumnName("TEL_EDITORIAL").HasMaxLength(15);
    }
}
```

Clase Ejemplar:

```
public class EjemplarMapeo : EntityTypeConfiguration<ClsEjemplar>
{
    1 referencia
    public EjemplarMapeo()
    {
        ToTable("TBL_EJEMPLAR");
        HasKey(p => p.cod_ejemplar);
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");
        Property(p => p.id_libro).HasColumnName("ID_LIBRO");
        Property(p => p.estado_ejemplar).HasColumnName("ESTADO_EJEMPLAR").HasMaxLength(100);
        Property(p => p.conserv_ejemplar).HasColumnName("CONSERV_EJEMPLAR").HasMaxLength(100);
        Property(p => p.ciclo).HasColumnName("CICLO").HasMaxLength(20);
        Property(p => p.Fecha_Adquirido).HasColumnName("FECHA_ADQUIRIDO");
        Property(p => p.Fecha_Registrado).HasColumnName("FECHA_REGISTRADO");

        HasRequired(m => m.libro).WithMany().HasForeignKey(f => f.id_libro);
    }
}
```

Clase Favorito:

```
public class FavoritoMapeo : EntityTypeConfiguration<ClsFavorito>
{
    1 referencia
    public FavoritoMapeo()
    {
        ToTable("TBL_FAVORITO");
        HasKey(p => p.cod_favoritos);
        Property(p => p.cod_favoritos).HasColumnName("COD_FAVORITOS");
        Property(p => p.id_usuario).HasColumnName("ID_USUARIO");
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");

        HasRequired(m => m.usuario).WithMany().HasForeignKey(f => f.id_usuario);
        HasRequired(m => m.ejemplar).WithMany().HasForeignKey(f => f.cod_ejemplar);
    }
}
```

Clase Historial:

```
class HistorialMapeo : EntityTypeConfiguration<ClsHistorial>
{
    1 referencia
    public HistorialMapeo()
    {
        ToTable("TBL_HISTORIAL");
        HasKey(p => p.cod_historial);
        Property(p => p.cod_historial).HasColumnName("COD_HISTORIAL");
        Property(p => p.cod_prestamo).HasColumnName("COD_PRESTAMO");
        Property(p => p.cod_reserva).HasColumnName("COD_RESERVA");
        Property(p => p.id_usuario).HasColumnName("ID_USUARIO");
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");
        Property(p => p.fecha).HasColumnName("FECHA");

        HasRequired(m => m.prestamo).WithMany().HasForeignKey(f => f.cod_prestamo);
        HasRequired(m => m.reserva).WithMany().HasForeignKey(f => f.cod_reserva);
        HasRequired(m => m.usuario).WithMany().HasForeignKey(f => f.id_usuario);
        HasRequired(m => m.ejemplar).WithMany().HasForeignKey(f => f.cod_ejemplar);
    }
}
```

Clase Libro:

```
public class LibroMapeo : EntityTypeConfiguration<ClsLibro>
{
    1 referencia
    public LibroMapeo()
    {
        ToTable("TBL_LIBRO");
        HasKey(p => p.id_libro);
        Property(p => p.id_libro).HasColumnName("ID_LIBRO");
        Property(p => p.id_autor).HasColumnName("ID_AUTOR");
        Property(p => p.id_editorial).HasColumnName("ID_EDITORIAL");
        Property(p => p.id_edicion).HasColumnName("ID_EDICION");
        Property(p => p.id_categoria).HasColumnName("ID_CATEGORIA");
        Property(p => p.tit_libro).HasColumnName("TIT_LIBRO").HasMaxLength(350);
        Property(p => p.subtit_libro).HasColumnName("SUBTIT_LIBRO").HasMaxLength(250);
        Property(p => p.estado_libro).HasColumnName("ESTADO_LIBRO").HasMaxLength(15);
        Property(p => p.idioma_libro).HasColumnName("IDIOMA_LIBRO").HasMaxLength(50);
        Property(p => p.num_pag_libro).HasColumnName("NUM_PAG_LIBRO");
        Property(p => p.num_ejemplares).HasColumnName("NUM_EJEMPLARES");

        HasRequired(m => m.autor).WithMany().HasForeignKey(f => f.id_autor);
        HasRequired(m => m.editorial).WithMany().HasForeignKey(f => f.id_editorial);
        HasRequired(m => m.edicion).WithMany().HasForeignKey(f => f.id_edicion);
        HasRequired(m => m.categoria).WithMany().HasForeignKey(f => f.id_categoria);
    }
}
```

Clase Personal:

```
public class PersonaMapeo : EntityTypeConfiguration<ClsPersona>
{
    1 referencia
    public PersonaMapeo()
    {
        ToTable("TBL_PERSONA");
        HasKey(p => p.cod_persona);
        Property(p => p.cod_persona).HasColumnName("COD_PERSONA");
        Property(p => p.dni_persona).HasColumnName("DNI_PERSONA").HasMaxLength(8);
        Property(p => p.nomb_persona).HasColumnName("NOMB_PERSONA").HasMaxLength(250);
        Property(p => p.apel_persona).HasColumnName("APEL_PERSONA").HasMaxLength(250);
        Property(p => p.sexo_persona).HasColumnName("SEXO_PERSONA").HasMaxLength(1);
        Property(p => p.email_persona).HasColumnName("EMAIL_PERSONA").HasMaxLength(250);
        Property(p => p.tel_persona).HasColumnName("TEL_PERSONA").HasMaxLength(15);
        Property(p => p.foto_persona).HasColumnName("FOTO_PERSONA").HasMaxLength(250);
        Property(p => p.estado_persona).HasColumnName("ESTADO_PERSONA").HasMaxLength(10);
    }
}
```

Clase Préstamo:

```
class PrestamoMapeo : EntityTypeConfiguration<ClsPrestamo>
{
    1 referencia
    public PrestamoMapeo()
    {
        ToTable("TBL_PRESTAMO");
        HasKey(p => p.cod_prestamo);
        Property(p => p.cod_prestamo).HasColumnName("COD_PRESTAMO");
        Property(p => p.cod_reserva).HasColumnName("COD_RESERVA");
        Property(p => p.fecha_prestamo).HasColumnName("FECHA_PRESTAMO");
        Property(p => p.fecha_entrega).HasColumnName("FECHA_ENTREGA");

        HasRequired(m => m.reserva).WithMany().HasForeignKey(f => f.cod_reserva);
    }
}
```

Clase Ranking:

```
class RankingMapeo : EntityTypeConfiguration<ClsRanking>
{
    1 referencia
    public RankingMapeo()
    {
        ToTable("TBL_RANKING");
        HasKey(p => p.cod_ranking);
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");
        Property(p => p.num_prestamo).HasColumnName("NUM_PRESTAMO");
        Property(p => p.num_puesto).HasColumnName("NUM_PUESTO");

        HasRequired(m => m.ejemplar).WithMany().HasForeignKey(f => f.cod_ejemplar);
    }
}
```

Clase Reserva:

```
public class ReservaMapeo : EntityTypeConfiguration<ClsReserva>
{
    1 referencia
    public ReservaMapeo()
    {
        ToTable("TBL_RESERVA");
        HasKey(p => p.cod_reserva);
        Property(p => p.cod_reserva).HasColumnName("COD_RESERVA");
        Property(p => p.id_usuario).HasColumnName("ID_USUARIO");
        Property(p => p.cod_ejemplar).HasColumnName("COD_EJEMPLAR");
        Property(p => p.fecha_reserva).HasColumnName("FECHA_RESERVA");
        Property(p => p.periodo).HasColumnName("PERIODO");

        HasRequired(m => m.usuario).WithMany().HasForeignKey(f => f.id_usuario);
        HasRequired(m => m.ejemplar).WithMany().HasForeignKey(f => f.cod_ejemplar);
    }
}
```

Clase TipoUsuario:

```
public class TipoUsuarioMapeo : EntityTypeConfiguration<ClsTipoUsuario>
{
    1 referencia
    public TipoUsuarioMapeo()
    {
        ToTable("TBL_TIPO_USUARIO");
        HasKey(p => p.id_tipo);
        Property(p => p.id_tipo).HasColumnName("ID_TIPO");
        Property(p => p.desc_tipo).HasColumnName("DESC_TIPO").HasMaxLength(20);
    }
}
```

Clase Usuario:

```
public class UsuarioMapeo : EntityTypeConfiguration<ClsUsuario>
{
    1 referencia
    public UsuarioMapeo()
    {
        ToTable("TBL_USUARIO");
        HasKey(p => p.id_usuario);
        Property(p => p.id_usuario).HasColumnName("ID_USUARIO");
        Property(p => p.cod_persona).HasColumnName("COD_PERSONA");
        Property(p => p.nomb_usuario).HasColumnName("NOMB_USUARIO").HasMaxLength(50);
        Property(p => p.pass_usuario).HasColumnName("PASS_USUARIO").HasMaxLength(100);
        Property(p => p.id_tipo).HasColumnName("ID_TIPO");
        Property(p => p.estado_usuario).HasColumnName("ESTADO_USUARIO").HasMaxLength(10);

        HasRequired(m => m.persona).WithMany().HasForeignKey(f => f.cod_persona);
        HasRequired(m => m.tipo).WithMany().HasForeignKey(f => f.id_tipo);
    }
}
```