2011

UNIVERSIDAD POLITECNICA DE VALENCIA

[GUÍA FARMACOTERAPEUTICA APLICACIÓN]

Escuela Técnica Superior de Ingeniería Informática etsinf

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

CONSORCI HOSPITAL GENERAL UNIVERSITARI VALÈNCIA

Centro de Sistemas de Información

Autor:
Jorge Rafael Soro Doménech

Carrera:
Ingeniería Técnica Informática Aplicada a la Gestión

Director del proyecto UPV:
Eliseo Jorge Marzal Calatayud

Director del proyecto CHGUV:
Raquel del Pino Marrero

Tabla de contenido

1 INTRODUCCIÓN

El abordaje multidisciplinar de problemas de salud complejos, sobre los que aumentan cada día su conocimiento y las posibilidades de intervención, plantea nuevos retos que exigen la puesta al día de las estrategias del sistema sanitario en materia de uso racional de medicamentos y productos sanitarios. Por eso se debe de buscar la mayor eficiencia para la gestión de los recursos del sistema sanitario.

Se ha realizado una guía fármaco-terapéutica la cual contiene los fármacos que existen en un hospital tanto los dispensables como aquellos que no lo son, creando un catálogo en el cual se está buscando una profundización en la eficiencia de la gestión de los recursos fármaco-terapéuticos. Esta guía permite una economía de escala en la elaboración y recuperación de información por cualquier centro sanitario.

La AVS[1] impulsará los consensos terapéuticos a nivel de cada departamento de salud que estén basados en la evidencia y formulados a partir de documentos de actuación clínica validados por organismos de ámbito autonómico, nacional o internacional, así como su inclusión, para interacción informatizada, en el sistema de información sanitario corporativo.

Para facilitar a los profesionales que opten por utilizar esta guía y protocolos como elementos de consulta, las recomendaciones son expresadas con niveles de evidencia. El acceso a esta información podrá realizarse por el profesional en el proceso de preinscripción y desde la historia clínica del paciente.

2 PRESENTACIÓN

El **Hospital General Universitario de Valencia,** al igual que cualquier otro centro sanitario, tiene como objetivo proporcionar una asistencia de calidad orientada hacia la búsqueda de la eficiencia, eficacia y la satisfacción de los usuarios. En el cumplimiento de este objetivo el fármaco juega un papel fundamental, así como el

[1] **AVS:** Agencia Valenciana de Salud.

profesional que lo maneja, que con su buen hacer ha de contribuir a un uso racional de éste.

Por todo ello, la **Guía Fármaco-terapéutica** es sin duda una herramienta fundamental para los profesionales al facilitarles la información básica de los medicamentos disponibles. Una de las funciones de la Comisión Hospitalaria de Farmacia y Terapéutica consiste en seleccionar los medicamentos susceptibles de utilización intrahospitalaria y recogerlos en la presente Guía Fármaco-terapéutica, con el objetivo de servir de ayuda ante la toma de decisiones terapéuticas basadas en el uso racional del medicamento.

La continua investigación de nuevos fármacos y de nuevas aplicaciones terapéuticas de los ya existentes hace que las guías fármaco-terapéuticas de los hospitales sean dinámicas y en constante cambio y evolución.

Prueba de ello es este manual o guía de medicamentos que nos complace presentar con gran satisfacción y que es fruto del trabajo continuo de nuestros profesionales, por lo que agradecemos muy sinceramente a todos su colaboración.

3 CONCEPTO GUÍA FÁRMACO-TERAPÉUTICA

Una Guía Fármaco-terapéutica es una aplicación que contiene una relación limitada de medicamentos recomendados para la prescripción en un ámbito determinado, seleccionada a partir de la oferta farmacéutica en función de unos criterios previamente establecidos, con la participación y el consenso de los profesionales a los que va destinada.

La guía establece las bases teóricas para orientar a los profesionales en la elección del medicamento más seguro, efectivo y eficiente para el tratamiento de un problema particular en un paciente determinado. Por este motivo, el listado de medicamentos se acompaña de otra información considerada de interés (indicaciones, interacciones, efectos adversos, presentaciones comerciales, etc.).

Entre los objetivos que persigue la aplicación de una guía podemos destacar:

- Facilitar al profesional la elección crítica de los medicamentos, al proporcionar una información objetiva y contrastada sobre los mismos y que se puede consultar rápidamente.
- Mejorar el perfil de prescripción farmacológico, a través de una selección racional de los medicamentos.
- Impulsar la formación continuada, tanto en la fase de elaboración de la guía como en la de consulta.
- Buscar la máxima eficiencia posible en el empleo de los recursos sanitarios, mediante la selección de fármacos con una buena relación coste / eficacia.
- Establecer un mecanismo rutinario de evaluación constante de la oferta de medicamentos.

4 OBJETIVOS

Con la elaboración de este documento pretendemos alcanzar los siguientes objetivos:

- Definir el concepto de guía fármaco-terapéutica.
- Establecer criterios y efectuar recomendaciones para la manipulación de la guía fármaco-terapéutica.
- Determinar los contenidos que tiene la guía fármaco-terapéutica.
- Proporcionar nociones sobre el funcionamiento de la guía fármaco-terapéutica.

5 ESPECIFICACIÓN DE REQUISITOS IEE

5.1 Introducción

5.1.1 Propósito

El propósito de la especificación de requisitos es definir cuáles son los requerimientos que debe tener el prototipo para la guía fármaco-terapéutica.

Cómo su nombre indica, la guía fármaco-terapéutica nos permitirá realizar consultas sobre presentaciones farmacéuticas con toda la información detallada.

Esta especificación de requerimientos está destinada a ser leída por los usuarios o cualquier sujeto que tenga interés en saber cómo funciona el producto.

5.1.2 Ámbito

El producto que se describe a continuación es un prototipo para la consulta de presentaciones farmacéuticas.

Va destinado a los profesionales, los cuales van a tener la posibilidad de poder activar o desactivar presentaciones farmacéuticas para que puedan ser visualizadas a la hora de realizar las consultas.

5.1.3 Referencias

Para la recaudación de este texto se han tenido encuentra los siguientes documentos:

- IEEE STD 830- IEEE Guide to Software Requirements' Specifications. IEEE Standard Board. (IEE STD 830)
- Guía del IEEE para la Especificación de Requerimientos Software, de ejemplo. Pagina Web de Ingeniería del Software. (Ingenieria del software)
- Roger S. Pressman. "Ingeniería del Software. Un enfoque práctico". 4-64 2014. (Pressman)

5.2 Descripción general

5.2.1 Perspectiva del producto

El prototipo debe funcionar bajo cualquiera de los sistemas operativos Windows 32/64 bits Windows 2000, Windows XP, Windows 2003, Windows 7.

El presente producto deberá ser capaz de funcionar en cualquier equipo informático que reside en el Consorcio del Hospital General de Valencia con una conexión al dominio CHGUV, ya que funcionará conectado a una base de datos que estará ubicada en un servidor del dominio CHGUV.

El producto una vez sea ejecutado por el usuario, antes de iniciarse deberá realizar una comprobación de versión para ver si hay algún cambio, y si lo hay, actualizarse.

5.2.2 Generalidades del producto

5.2.2.1 Funciones del producto

El producto deseado, debe de poder realizar con éxito las siguientes funcionalidades:

- Al ejecutar el acceso directo del producto se realizará una comprobación de versión del producto, si se ha quedado anticuada, se debe de actualizar.
- Autentificación: se solicitará un usuario y una contraseña.
- Inicialización del sistema.
- Se mostrará un menú en el cual se podrá visualizar todas las pantallas activas, a las cuales se van a poder acceder.
- Se va a poder consultar a través de la pestaña *Catalogo*, todas las presentaciones farmacéuticas. Se podrá buscar:
  - Por código (el código nacional de la presentación farmacéutica que hace referencia al código único de esta en todo el país).
  - Por descripción (nombre de la presentación farmacéutica).
- Posibilidad de activar o desactivar presentaciones farmacéuticas en la pestaña *Catálogo*.

- Posibilidad de activar o desactivar información que luego puede que contenga la presentación farmacéutica. Esto se podrá realizar a través de las pantallas de mantenimiento que son simples pantallas en las cuales se muestra un listado con la información referente a dicho mantenimiento (principio activo, tramo edad, tramo peso…).
- Realización de forma periódica una carga de datos que actualizará la base de datos del aplicativo.

5.3 Características del usuario

Los usuarios de este prototipo serán los médicos y farmacéuticos del Consorcio Hospital General de Valencia.

Los farmacéuticos serán los que podrán realizar activaciones o no de presentaciones farmacéuticas y además también podrán activar o no elementos que puedan contener las presentaciones farmacéuticas, tales como tramos de edad, vías de administración…

5.4 Restricciones generales

El prototipo será diseñado para que funcione sobre Windows XP/Windows 7, y en los ordenadores del Consorcio Hospital General de Valencia.

5.5 Supuestos y dependencias

El sistema depende de una base de datos que estará conectada vía un servidor que estará ubicado en el dominio CHGUV.

La carga de la información contenida en el catálogo se realizará por los administradores del prototipo.

El prototipo usará Silverlight y Microsoft .NET Framework 4.

5.6 Requerimientos específicos

5.6.1 Requerimientos funcionales

5.6.1.1 Inicialización del sistema

Al iniciarse el sistema se conectará con el servidor y comprobará la versión de la aplicación. Si la aplicación se ha quedado anticuada, el sistema se actualizará.

Si no es necesario actualizar la aplicación, el sistema se iniciará y solicitará los datos para realizar la autentificación: usuario y contraseña.

Una vez la autentificación del sistema es correcta, el sistema se iniciará.

Al iniciarse se establecerá la conexión a una base de datos que estará en un servidor.

Cuando todo haya funcionado correctamente, saldrá una pantalla inicial en la cual nos aparecerá una barra, en la parte superior, dónde aparecerán las diferentes secciones por las que podemos navegar. Dichas secciones son las siguientes: Catálogo, Mantenimiento simple, Mantenimiento compuesto, Ayuda y Cambiar Usuario. Todas estas secciones son visibles para todos los usuarios autentificados en la aplicación.

5.6.1.2 Catálogo

El catálogo será la pantalla más utilizada debido a que es la que reúne toda la información sobre las presentaciones farmacéuticas.

En esta pantalla se van a poder realizar búsquedas para obtener información detallada sobre presentaciones farmacéuticas.

Toda la información sobre una presentación farmacéutica seleccionada será mostrada en todas las pestañas existentes en el Catálogo (todo lo que se puede realizar en el Catálogo lo podremos ver con detalle en el manual en el punto 7.2).

5.6.1.3 Mantenimiento

Las pantallas de mantenimiento nos van a mostrar listados con la información existente a lo que haga referencia dicha pantalla (ejemplo: tramo edad, tramo peso…).

Se podrán realizar búsquedas por todos los campos que estén disponibles en la pantalla actual.

Cuando tengamos seleccionado un registro, podemos observar que el estado ha pasado a "Modificación" (ver etiqueta estado en la barra inferior). Con la cual cosa, se podrán realizar modificaciones. También podemos pasar a estado "Alta" y registrar un nuevo registro (todo lo que se puede realizar en una pantalla de mantenimiento lo podremos ver con detalle en el manual en el punto 7.2).

5.6.2 Requerimientos de interfaces externas

5.6.2.1 Interfaces de usuario

La interfaz será de tipo formulario manejándose con el teclado. La interfaz es intuitiva y con apariencia visual moderna.

5.6.2.2 Interfaces hardware

El prototipo inicialmente puede ser usado en todos los dispositivos admitidos por Windows.

5.6.2.3 Interfaces software

Se utilizaran las interfaces de Windows Presentation Foundation (WPF), por eso es necesario por parte del usuario de la aplicación de disponer de los paquetes Silverlight y Microsoft .NET Framework 4 para poder visualizar correctamente el prototipo.

5.6.3 Interfaces de comunicaciones

El prototipo está diseñado para ejecutarse en modo local (físico).

Se utiliza la red para actualizar la versión de la aplicación si es necesario.

El servidor de la base de datos estará ubicado en el Consorcio Hospital General Universitario de Valencia y por tanto, se necesitará disponibilidad de red para realizar las conexiones necesarias a este.

5.6.4 Requerimientos de eficiencia

El producto está pensado para ser usado por muchos usuarios. Por eso estará controlado mediante un proceso de autentificación antes de inicializar la aplicación.

También se han controlado los tiempos de respuesta para evitar futuros errores.

5.6.4.1 Limitaciones hardware

Las limitaciones del hardware son las que impone Windows.

5.6.5 Atributos

5.6.5.1 Seguridad

El prototipo debe de estar diseñado para que actúe ante situaciones de error avisando al usuario y recuperando la aplicación frente a la ocurrencia de errores del sistema.

Esto se realizará mediante comprobaciones en todos los aspectos del prototipo.

Para el acceso a la aplicación se va a utilizar autentificación de usuario para garantizar la seguridad en los accesos a la aplicación.

5.6.5.2 Mantenimiento

El producto al ser un prototipo no necesitara un mantenimiento especial.

5.6.6 Otros requerimientos

5.6.6.1 Base de datos

El producto utilizará el gestor de base de datos Microsoft SQL Server 2005. Dicho gestor se utilizará para crear una base de datos en dónde se almacenará toda la información necesaria de la aplicación.

5.6.6.2 Operaciones

Sólo existirá una instancia del prototipo en cada momento.

5.6.6.3 Requerimientos de adaptación a situaciones

El prototipo no tiene prevista su adaptación a situaciones especiales de funcionamiento.

6 ANÁLISIS FUNCIONAL

6.1 Presentación

Se presenta el resultado del trabajo de análisis realizado para identificar los requerimientos y el desarrollo funcional de la Guía Fármaco-Terapéutica del Consorcio Hospital General Universitario de Valencia. Como elemento adicional recoge la propuesta de estándares y metodología que se deberían seguir en las fases de diseño tecnológico, construcción e implantación de la solución.

6.2 Descripción de la solución propuesta

6.2.1 Planteamiento general

La Guía Fármaco-Terapéutica tiene la finalidad de facilitar la consulta de presentaciones farmacéuticas del Hospital General Universitario de Valencia. El sistema a implantar permitirá consultar y modificar por parte del usuario presentaciones farmacéuticas, así como sus elementos. Los requerimientos más remarcables de mejora son:

- Mejorar la eficiencia en la consulta de presentaciones farmacéuticas y garantizar una mejor calidad de información.

6.2.1.1 Requisitos generales del sistema de información

- IDENTIFICACIÓN
  - Nombre de usuario
  - Contraseña
- PERIODO FUNCIONAL
  - La consulta y modificación de presentaciones farmacéuticas se podrá realizar siempre y cuando la aplicación de la Guía Fármaco-Terapéutica este instalada en el sistema.

6.2.1.2 Requisitos técnicos generales del sistema

Las características técnicas generales que debe cumplir el sistema globalmente son las siguientes:

- **Interactividad:** Los usuarios interactuarán ante una interfaz gráfica intuitiva.
- **Seguridad:** Autentificación de usuario para poder acceder a la aplicación. Así como el control de quién realiza las altas y/o modificaciones de los registros.
- **Eficacia:** Esta aplicación permitirá la consulta de cualquier presentación farmacéutica en cualquier equipo del departamento de farmacia.
- **Orientación al usuario:** El programa será de fácil manejo, orientado a usuarios que posean un mínimo de conocimientos de informática.
- **Cumplimiento de estándares:** Se seguirán los estándares españoles y europeos de seguridad, ergonomía y calidad.

6.2.1.3 Requisitos de seguridad

Se deberá de garantizar la seguridad de la información desde el punto de vista de integridad, confidencialidad y disponibilidad y el cumplimiento de la legislación vigente en cada momento durante el plazo de ejecución del contrato relativa a la protección de datos de carácter personal.

Deberá garantizar que las herramientas que se proponen en la solución, Sistemas Operativos, Bases de Datos y Aplicación, cumplen con los distintos artículos definidos en el reglamento de medidas de seguridad (RD 994/1999) de los ficheros automatizados que contienen datos de carácter personal, en la propia Ley de Protección de Datos o cualquier otra normativa vigente a lo largo de la ejecución del contrato.

No se registrarán datos de carácter personal en ficheros que no reúnan las condiciones que se determinen por el Real Decreto 994/1999 respecto a su integridad y seguridad Análisis Funcional Mesías y a las de los centros de tratamiento, locales, equipos, sistemas y programas (artículo 9.2. LOPD). Para mantener la privacidad de la información, las tablas de datos sólo deben ser accesibles a través del Sistema Gestor de la Base de Datos, estableciéndose los mecanismos adecuados para restringir los accesos y copias de tablas por procedimientos no definidos en la aplicación.

6.2.2 Plan de formación, análisis, diseño y construcción de la aplicación

6.2.2.1 Fase de lanzamiento del proyecto

Objetivos	**Facilitar el inicio y la ejecución del Proyecto garantizando el compromiso de todos los participantes.**
Actividades	**Participantes**
➢ Identificación de interlocutores clave de todas las Áreas involucradas ➢ Constitución de los equipos del proyecto: o Comité de Dirección o Comité de Seguimiento o Equipos de trabajo: funcional y técnico ➢ Presentación del equipo de proyecto ➢ Análisis de la documentación previa Revisión del Plan de Proyecto ➢ Elaboración del plan detallado del proyecto ➢ Establecimiento de agendas ➢ Asignación de tareas	1 Responsable del proyecto / Jefe del proyecto / Responsable técnico
Resultados	
➢ Equipo de trabajo constituido y organizado ➢ Acta de la Reunión de lanzamiento del proyecto ➢ Plan de Comunicación inicial ➢ Conocimiento y compromiso por parte de las personas implicadas en el plan	

Tabla 1 : FASE DE LANZAMIENTO DEL PROYECTO

6.2.2.2 Fase de análisis y estudios de viabilidad

Objetivos	**Definir el Modelo identificando, analizando y describiendo los contenidos y servicios que compondrán este proyecto.**
Actividades	**Participantes**
➢ Definición del Modelo de Prestación Servicios ➢ Definición de contenidos ➢ Elaboración de un análisis detallado de requisitos: o Servicios básicos	1 Responsable Proyecto / Jefe de Proyecto / Responsable Técnico 1 Analista Funcional
Resultados	
➢ Análisis Funcional	

Tabla 2 : FASE DE DISEÑO DE ANÁLISIS Y ESTUDIOS DE VIABILIDAD

6.2.2.3 Fase de diseño del sistema

Objetivos	**Realizar el Diseño Técnico de la Arquitectura Orientada a Servicios y de la Solución considerados viables a partir del documento de Análisis funcional.**
Actividades	**Participantes**
➢ Servicios Básicos ➢ Instalación del Entorno de Desarrollo ➢ Diseño del modelo de datos ➢ Diseño de interfaz de usuario	1 Jefe de Proyecto / Responsable Técnico / Responsable Proyecto 1 Analista Funcional / Arquitecto de Software
Resultados	
➢ Entorno de Desarrollo ➢ Documento de Modelo de Datos ➢ Prototipo interfaz de usuario	

Tabla 3 : FASE DE DISEÑO DEL SISTEMA

6.2.2.4 Fase de desarrollo y pruebas unitarias

Objetivos	**Construir una solución que responda a las especificaciones realizadas, que cumpla con los criterios técnicos del Hospital y con las máximas garantías de calidad.**
Actividades	**Participantes**
➢ Construcción y parametrización del sistema ➢ Desarrollo de los Servicios de la solución ➢ Realización de Pruebas unitarias del sistema	Director de Proyecto Personas clave identificadas: o Responsables sistemas
Resultados	
➢ Versión 1.0	

Tabla 4 : FASE DE DESARROLLO Y PRUEBAS UNITARIAS

6.2.2.5 Fase de implantación y despliegue

Objetivos	**Despliegue y puesta en marcha de la solución en el entorno de producción.**
Actividades	**Participantes**
➢ Instalación del Entorno de Explotación ➢ Elaboración de los manuales de instalación y de ➢ contingencia ➢ Implantación (Despliegue)	1 Responsable Proyecto / Jefe de Proyecto / Responsable Técnico 1 Analista Funcional / Programador
Resultados	
➢ Manual de la aplicación ➢ Informe con las incidencias producidas durante esta fase	

Tabla 5 : FASE DE IMPLEMENTACIÓN Y DESPLIEGUE

6.2.2.6 Fase de pruebas generales

Objetivos	**Comprobar el funcionamiento del sistema desarrollado para determinar las posibles deficiencias y evaluar su comportamiento**
Actividades	**Participantes**
➢ Diseño, Ejecución y Validación del Plan de Pruebas: o Usuarios de la empresa o Usuarios finales o Usuarios técnicos ➢ Resolución de las incidencias detectadas ➢ Detección de mejoras funcionales	1 Responsable Proyecto / Jefe de Proyecto / Responsable Técnico 1 Analista Funcional / Programador
Resultados	
➢ Informe de incidencias detectadas y corregidas. ➢ Aprobación, Validación y Aceptación por parte del Hospital	

Tabla 6 : FASE DE PRUEBAS GENERALES

6.3 Organización y gestión del proyecto

6.3.1 Estructura organizativa

El modelo de relación define la manera en la que se propone llevar a cabo el control y seguimiento de la calidad del servicio que se proporcionará. Este modelo de relación está basado en una interlocución a tres niveles.

Las principales funciones de los Órganos de Representación considerados en el modelo descrito son los siguientes:

Comité de Dirección

El Comité de Dirección comprende a los representantes de la dirección del proyecto integrado por el Gerente del Hospital, el Director de Informática y el Director del Proyecto Final de Carrera.

El objetivo de este Comité será asegurar que todas las decisiones y acciones relacionadas con el servicio prestado se discuten y se acuerdan abiertamente.

El Comité de Dirección se reunirá periódicamente con el objetivo de revisar de forma continuada la adecuación del acuerdo contractual.

Las funciones de este comité serán:

- Aprobar la estrategia general del Proyecto.
- Aprobar cualquier cambio que se produzca en la planificación inicial del Proyecto.
- Establecer las líneas de la dirección estratégica de la prestación de los servicios.
- Gestionar y resolver de forma eficaz los asuntos y cuestiones surgidas en relación con el acuerdo y que no puedan resolverse por el Director del Proyecto.

La Dirección del Proyecto será asumida por el Director de Proyecto (Director Funcional) designado por el Hospital, y desarrollará su actividad de forma conjunta con todos los proyectos englobados dentro del Plan.

Las funciones del Director de Proyecto serán:

- Aprobar el Diseño del Sistema, que incluirá el diseño de la arquitectura, el catálogo de requisitos del sistema, y los procedimientos de seguridad y control de acceso del mismo.
- Realizar el seguimiento y control del contrato, exigiendo su cumplimiento en los términos acordados, y resolviendo los conflictos que puedan surgir entre ambas partes.

Comité de Seguimiento

El Comité de Seguimiento formado por los responsables ejecutivos del proyecto en las dos organizaciones, y que se reunirá según se determine en el lanzamiento del proyecto, es el órgano de coordinación del proyecto, encargado de velar por el cumplimiento de los objetivos del proyecto y de la toma de decisiones ante posibles incidencias o cambios.

Las funciones de este comité son las siguientes:

- Controlar el progreso de los diferentes servicios en el alcance del proyecto. Se incorporan en estas reuniones la documentación de los informes de seguimiento.
- Evaluar y aprobar las posibles variaciones del alcance, plazos, requisitos que puedan surgir.
- Resolución de las eventuales incidencias que puedan surgir durante la ejecución del proyecto.
- Informar al Comité de Dirección sobre el avance del proyecto.

El Comité de Seguimiento estará formado:

Por parte del Hospital:

- El Director de Proyecto, designado por el Hospital.
- Un responsable TIC, designado por el Hospital.
- Director del Proyecto de Final de Carrera.

6.3.2 Equipo de proyecto

El Equipo de Proyecto estará formado el Director de Proyecto, el responsable TIC del Hospital, el Jefe de Proyecto designados por la empresa, Director del Proyecto Final de Carrera, así como el personal asignado al proyecto y el grupo de expertos designado por el Director de Proyecto.

Podrá incorporarse a éste, en cualquier fase del proyecto, aquel personal técnico, funcional o usuario final del sistema, que se estime oportuno para la consecución final del proyecto.

Las funciones de este equipo son las siguientes:

- Ejecución de las actividades comprendidas en el Análisis, Diseño y Desarrollo del proyecto.
- Definir las necesidades a nivel de usuario final.

- Ejecutar las distintas actividades para realizar el (objeto del Proyecto), de acuerdo con las directrices marcadas por la Dirección del Proyecto, y por los comités de dirección y seguimiento del proyecto.
- Definición y ejecución del plan de pruebas diseñado para cada proceso, módulo y las de integración del sistema.
- Documentar y elaborar la aplicación.

7 MODELADO

En el apartado de modelado, se han realizado distintos tipos de diagramas que han sido necesarios para el diseño de la aplicación entre ellos:

- Casos de uso
- Diagrama secuencial
- Diagrama UML
- Diagrama entidad relación de la base de datos

7.1 Casos de uso

Un caso de uso (Garcia, 2007) es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

A continuación, se muestra un diagrama de caso de uso ilustrando los pasos y actividades necesarias para las acciones que se pueden realizar en una pantalla de la Guía Fármaco-Terapéutica. Ya sea de Mantenimiento o el Catálogo.

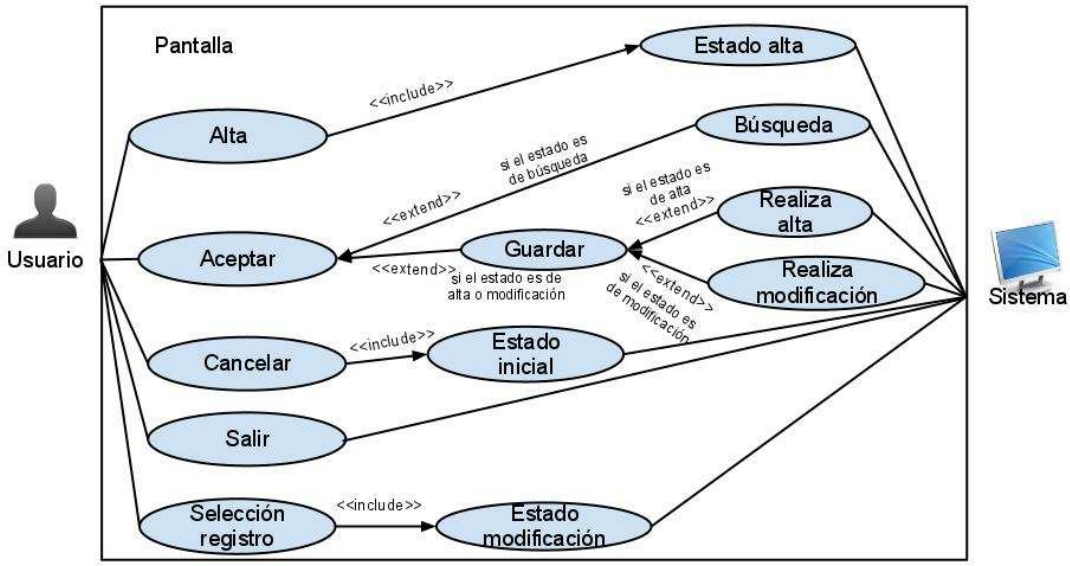

Ilustración 1 : CASO DE USO DE LA PANTALLA

Para aclarar un poco el caso de uso de Alta se detalla a continuación en la siguiente tabla:

CASO DE USO	ALTA
Actores	Usuario
Resumen	Un usuario podrá seleccionar en el menú de acciones la acción llamada Alta, que pondrá la pantalla en estado de alta.
Precondiciones	La pantalla pasará a estar en estado de alta.
Incluye	Estado alta.
Usuario	**Sistema**
Hace clic en el menú de acciones en el botón de Alta para pasar la pantalla a estado de alta.	
	El sistema una vez comprueba que el usuario ha hecho clic en el botón de acción Alta, pasa la pantalla a estado de alta.

Tabla 7 : CASO DE USO DE ALTA

Al hacer clic en aceptar obtenemos la siguiente tabla que representa el caso de uso de la acción Aceptar:

CASO DE USO	ACEPTAR
Actores	Usuario
Resumen	Un usuario podrá seleccionar en el menú de acciones la acción llamada Aceptar, que según en el estado que se encuentre la pantalla realizará una búsqueda, confirmará una alta o una modificación
Precondiciones	Si el estado es de búsqueda, el sistema realizará una búsqueda. Si el estado es de modificación o alta, pasará a la acción de guardado que depende de que estado tenga la pantalla realizará una cosa o otra.
Extiende	Si el estado es de búsqueda extiende a Búsqueda. Si el estado es de modificación o alta extiende a Guardar.
Usuario	**Sistema**
Hace clic en el menú de acciones en el botón de Aceptar.	
	El sistema una vez comprueba que el usuario ha hecho clic en el botón de acción Aceptar, comprueba en qué estado esta la pantalla. Si está en estado de búsqueda, realizará una búsqueda por el campo que este rellenado. Si está en estado de modificación o de alta, realizará la confirmación de estas en la acción de guardado.

Tabla 8 : CASO DE USO DE ACEPTAR

Para ver otro ejemplo ilustrativo, el siguiente diagrama de casos de uso que representa la acción de selección de un registro en el Catálogo de la guía.

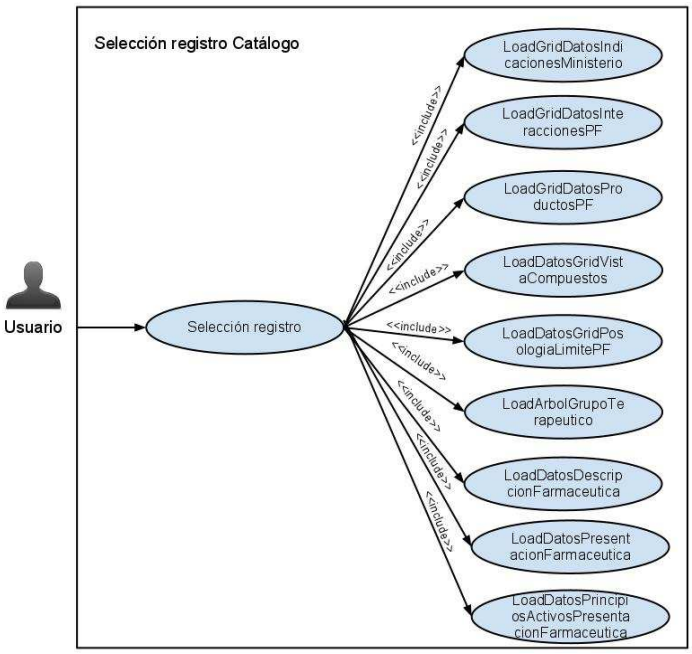

Ilustración 2 : CASO DE USO DE SELECCIÓN REGISTRO DEL CATÁLOGO

7.2 Diagrama de secuencia

El principal objetivo (Sparx System, 2007) de los diagramas de secuencia es identificar la comunicación dentro del sistema y las operaciones de las clases. También muestra la secuencia de mensajes entre objetos durante un escenario concreto.

Las características que tiene un diagrama se secuencia son las siguientes:

- **Un Escenario** es una secuencia de sucesos que se produce durante una ejecución concreta de un sistema.
- **Un suceso (o evento)** es algo que transcurre durante un período de tiempo. Un suceso es una transmisión de información de dirección única entre un objeto y otro. No es como una llamada a subrutina, que proporciona un valor. En el mundo real un objeto envía un suceso a otro objeto y puede esperar una respuesta, pero la respuesta es otro suceso distinto.

Para ilustrar los anteriores conceptos, el siguiente ejemplo:

Todas las pantallas de la guía están compuestas por 3 clases, más tarde se explica detalladamente en la documentación del proyecto: una interfaz, un presente (clase que actúa como intermediaria entre la vista y la clase de persistencia) y una vista.

El presente realiza conversaciones con una clase (llamada dtGuia.cs) que tiene los métodos que acceden a la base de datos para operar con información (recuperar, buscar, insertar, actualizar). Luego también realiza conversaciones con la vista, pasándole la información que va a ser mostrada y recuperando la información de esta (gracias a la comunicación con la clase que obtiene la información de la base de datos).

El siguiente diagrama de secuencia, detalla explícitamente las comunicaciones descritas anteriormente.

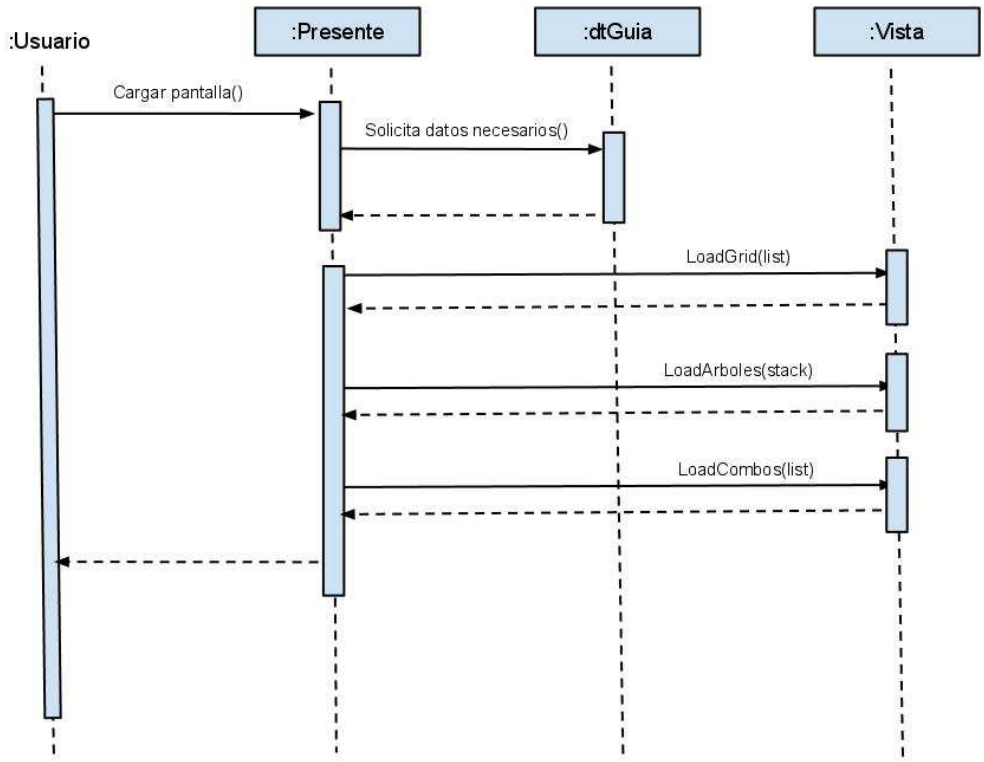

Ilustración 3 : DIAGRAMA DE SECUÉNCIA DE LAS PANTALLAS

7.3 Diagrama de clases UML

Lenguaje Unificado de Modelado (Wikipedia-UML, 2011) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Está respaldado por el OMG (Object Management Group).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

A continuación se ilustra el diagrama de clases pertinente a la guía fármaco-terapéutica.

7.4 Diagrama entidad relación de la base de datos

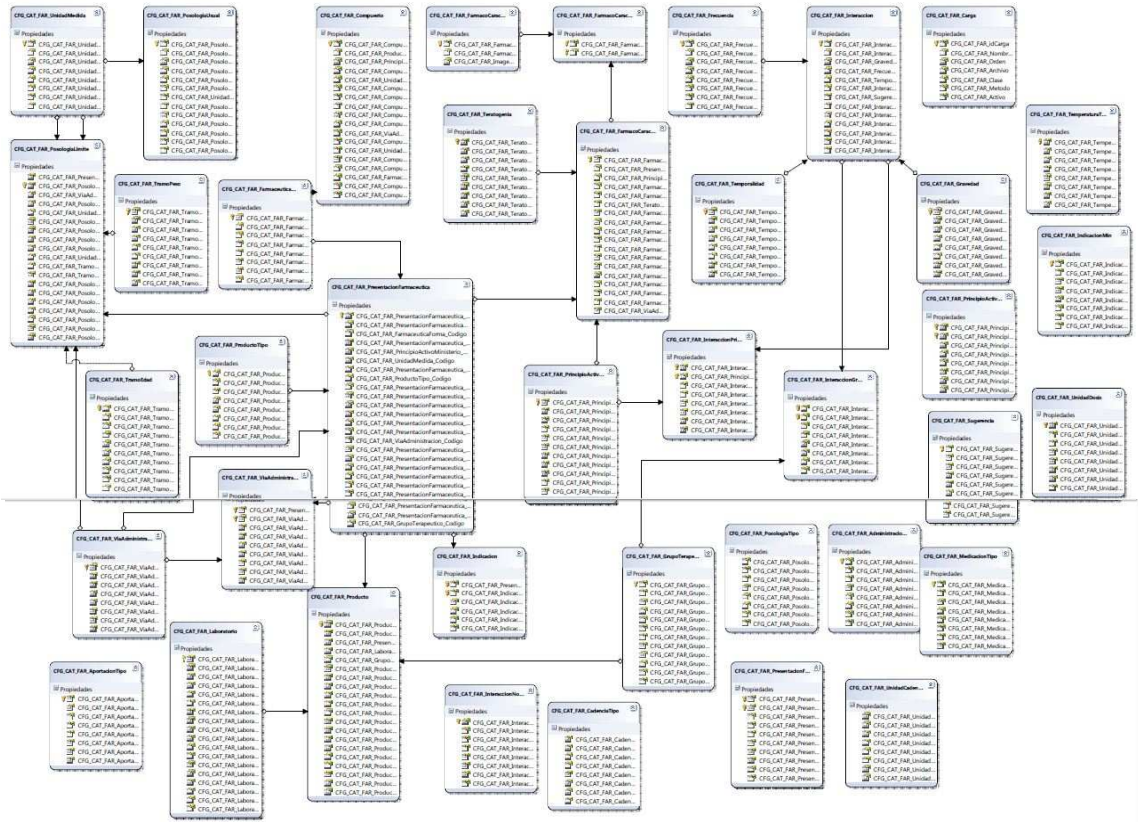

Ilustración 4 : DIAGRAMA DE CLASES DE LA GUÍA

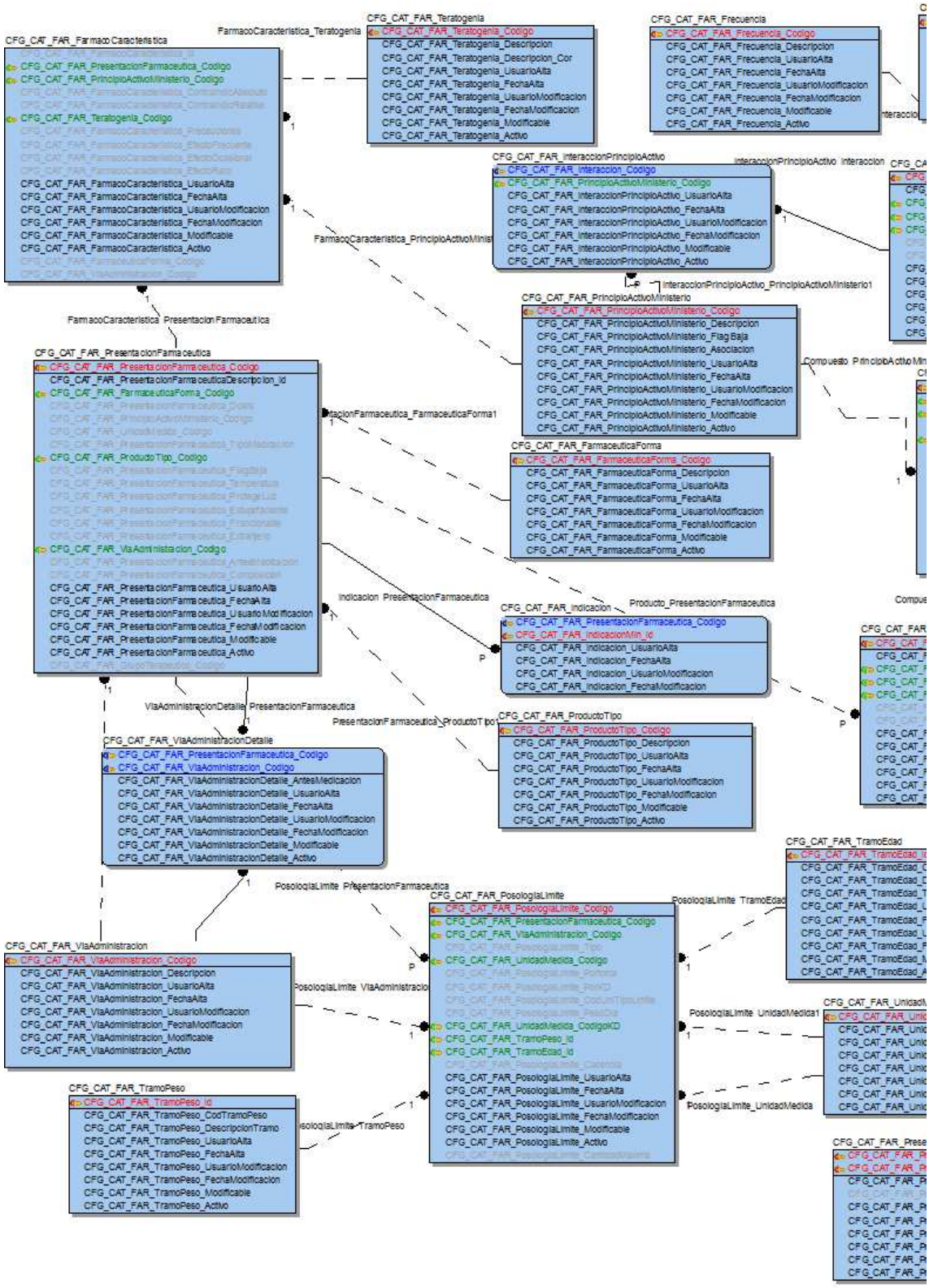

Ilustración 5: DIAGRAMA ENTIDAD RELACIÓN PARTE 1

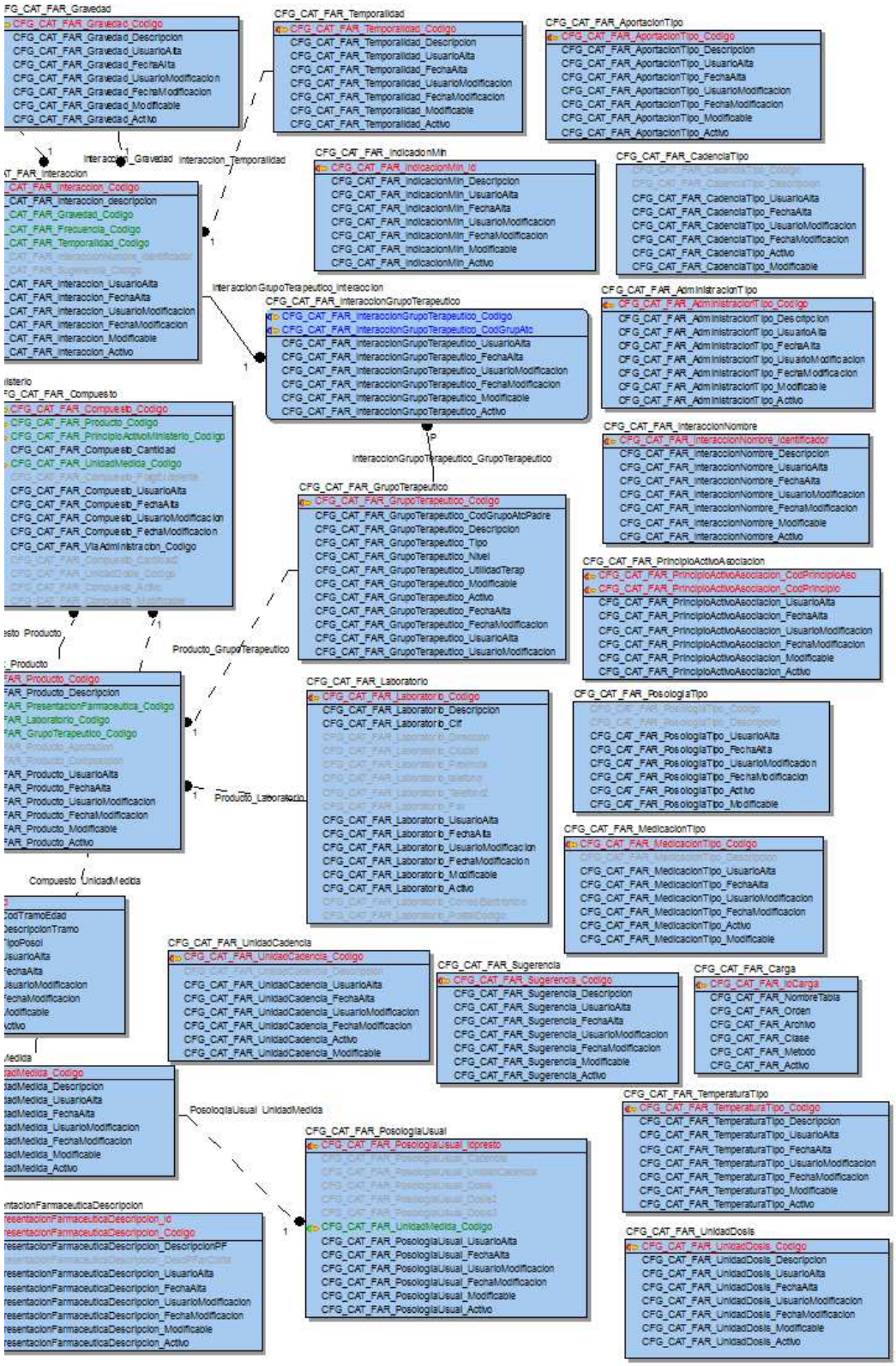

Ilustración 6: DIAGRAMA ENTIDAD RELACIÓN PARTE 2

La capa de persistencia de la aplicación ha sido implementada con el motor de base de datos Microsoft SQL Server 2005 [2]y con la tecnología LINQ[3] de Visual Studio 2010.

A partir de la creación de una instancia de Microsoft SQL Server 2005 en un servidor se procede a la inclusión en esta, de todas las tablas y vistas necesarias para la aplicación.

Una vez ya disponibles todas las tablas y vistas en el servidor, se procede con el IDE[4] Visual Studio 2010 y la tecnología LINQ a añadir a la solución de la Guía Fármaco-Terapéutica la base de datos completa para la correcta usabilidad en la aplicación. Esto se hace con un nuevo proyecto incluido dentro de la solución que representa la capa de persistencia.

Dentro del proyecto de persistencia, LINQ se encarga de crear automáticamente todos los objetos que representan cada tabla y vista. A partir de aquí hace falta configurar la conexión con la base de datos.

Ya incluida la base de datos en el programa de desarrollo y creada la conexión con el servidor, sólo falta crear el archivo "dtGuia.cs" en el cual se incluyen todos los métodos necesarios para recuperar, insertar y actualizar datos en el caso de la guía.

Con todo lo anterior ya hecho, ya está disponible para usar la base de datos que a continuación se detalla con un diagrama. Con sólo instanciar "dtGuia" en cualquier sitio, podremos acceder al contenido de la base de datos y realizar cualquier operación.

[2] **SQL SERVER 2005:** En el anexo 12.4 de forma detallada.
[3] **LINQ:** En el anexo 12.2 podemos ver detallada dicha tecnología.
[4] **IDE:** Un entorno de desarrollo integrado (también conocido como entorno de diseño integrado, entorno integrado de depuración o entorno de desarrollo interactivo) es una aplicación de software que proporciona servicios integrales a los programadores para el desarrollo de software.

8 IMPLEMENTACIÓN

Para la implementación de la aplicación se ha utilizado como modelo de desarrollo el Modelo Vista-Presente[5]. Se ha escogido este modelo debido a que todas las pantallas de la aplicación guardan similitudes respecto a las funcionalidades. Este modelo permite crear una independencia entre la capa de business y la capa de pages.

La distribución de las diferentes partes de la solución es la siguiente:

- **Proyecto de carga de datos:** este proyecto es independiente a la solución. Se utiliza para la realización de la carga de datos que periódicamente la Conselleria de Sanitat hace llegar al Hospital.
- **Proyecto de inicialización de la solución completa:** este proyecto se encarga de la autentificación del usuario y de la carga de toda la aplicación.
- **Proyecto referente a la Capa de Business:** en este proyecto se incluye todo lo esencial para el soporte de datos necesario para la solución final (descrito con detalle en el apartado 7).
- **Proyecto referente a la Capa de Pages:** este proyecto dispone de todas las pantallas existentes en la aplicación. Todas estas necesitarán usar la capa de Business anteriormente descrita.

8.1 Comentarios código del proyecto de carga de datos

La carga de información en la base de datos se hace a partir de unos ficheros .dat que proporciona conselleria de sanitat al hospital con toda la información que debe de contener las tablas. Por eso, se ha tenido que hacer un proyecto adicional para pasar toda esa información existente en ficheros .dat a la base de datos para poder ser utilizada por la aplicación.

[5] **Modelo Vista-Presente:** en el Anexo 12.1 esta detallado.

8.2 Comentarios código del proyecto de inicialización de la solución

El proyecto de inicialización utiliza una librería del propio Hospital que contiene métodos de autentificación que son usados en este proyecto para facilitar la autentificación en el dominio CHGUV.

Por otra parte, en este proyecto también se puede encontrar la pantalla principal de la aplicación que es la que inicializa todo el sistema.

Los ficheros que hacen referencia a lo anteriormente descrito son los siguientes:

- PrincipalWin.xaml[6] /PrincipalWin.cs
- Acceso

8.3 Comentarios código del proyecto de la capa de business

La capa de persistencia, como hemos descrito en el apartado 7, está compuesta principalmente por dos archivos:

- LINQGuiaFarmacoterapeutica.dbml
- dtGuia.cs[7]

Como estos dos archivos son muy grandes y largos, en los anexos indicados anteriormente en cada archivo podremos encontrar un fragmento de código para hacerse una idea de cómo funcionan.

8.4 Comentarios código del proyecto de la capa pages

La aplicación está compuesta por muchas pantallas las cuales tienen similares fines, por eso todas siguen la siguiente estructura:

[6] **XAML:** es el lenguaje de formato para la interfaz de usuario para la Base de Presentación de Windows (WPF por sus siglas en inglés) y Silverlight, el cual es uno de los "pilares" de la interfaz de programación de aplicaciones .NET en su versión 3.0 (conocida con anterioridad con el nombre clave WinFX).

[7] **Archivo dtGuia.cs:** Para ver con detalle el archivo, ver Anexo 12.8.

- **Interfaz**[8]**:** archivo dónde se declaran los métodos que van a ser compartidos entre la Vista y el Presente.
- **Presente**[9]**:** archivo dónde se ubican todas las funciones que describen el comportamiento de una determinada pantalla.
- **Vista**[10]**:** La vista está formada por dos archivos. El archivo "**xaml.cs**" es dónde se ubican las funciones que ayudan a cargar la información a la pantalla. El otro archivo es él "**.xaml**" en el cual se define todo el diseño de la pantalla de forma sencilla.

Cómo se ha indicado arriba, en los anexos podemos encontrar los archivos pertenecientes a la pantalla tramo de edad detallada al máximo. Todas las pantallas siguen la misma estructura y métodos similares.

9 MANUAL

9.1 Control de acceso

La aplicación utilizará una autentificación para confirmar que el usuario de la guía fármaco-terapéutica tiene los permisos necesarios para acceder a esta.

Ilustración 7 : AUTENTIFICACIÓN USUARIO

8 **Interfaz:** Para ver con detalle un archivo de interfaz, ver Anexo 12.5.
9 **Presente:** Para ver con detalle un archivo de presente, ver Anexo 12.6.
10 **Vista:** Para ver con detalle los archivos de la vista, ver Anexo 12.7.

El funcionamiento es simple, sólo tendrá que introducir el nombre de **usuario** y a continuación la **contraseña** que puede contener una clave alfanumérica[11].

Si usted no puede acceder (no tiene permisos) contacte con el administrador para que le proporcione los permisos necesarios.

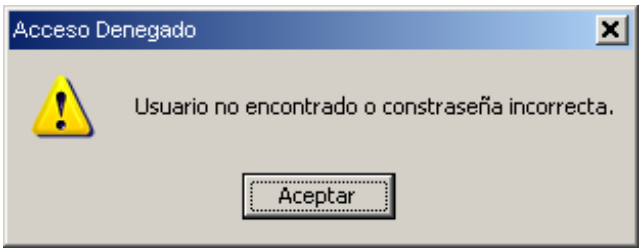

Ilustración 8 : USUARIO NO ENCONTRADO O CONTRASEÑA INCORRECTA

9.2 Funcionamiento general

La **Guía Fármaco-Terapéutica** como ya hemos comentado en los anteriores puntos, tiene el objetivo de facilitar la consulta de información por el profesional, por eso, la guía ha sido diseñada por y para este.

La aplicación ofrece la sección de **Mantenimiento** de la información del sistema en la cual se va a poder realizar cualquier tipo de mantenimiento de la guía.

Otra sección que ofrece la aplicación es la del **Catálogo**, en el cual se va a poder realizar búsquedas completas de presentaciones farmacéuticas y a partir de estas obtener información detallada. Si las presentaciones farmacéuticas son modificables será posible realizar modificaciones en estas.

Cuando hablamos de registros nos referimos a la información que aparece en los listados de las diferentes pantallas. Un registro correspondería a una fila del listado (un tramo de edad, una interacción…)

[11] **Alfanumérico:** Es un término colectivo para identificar letras del alfabeto latino y de números arábigos.

Todas las pantallas de Mantenimiento menos la del Catálogo, tienen disponibles todos los estados. **El Catálogo no tiene el estado de alta**, debido a que no se van a poder realizar altas de presentaciones farmacéuticas como tal.

A continuación, se van a detallar los **estados por los que puede pasar una pantalla**, ya sea de Mantenimiento o del Catálogo (recordar que el Catálogo no dispone del estado de alta).

ESTADO	DESCRIPCIÓN
Alta	Todos los campos necesarios para realizar un alta de un registro serán habilitados. Este estado sólo estará disponible en las pantallas de mantenimiento.
Modificación	Todos los campos que puedan ser modificados de un elemento serán activados para que puedan modificarse.
No modificación	El único campo que estará activo es el de “Modificable”.
Búsqueda	Los únicos campos habilitados para la búsqueda serán en los que se pueda realizar alguna búsqueda (código, descripción…)

Tabla 9 : ESTADOS DE UNA PANTALLA

9.2.1 Barra de estado

La **barra de estado** nos muestra información sobre la pantalla que tengamos abierta. En concreto, nos muestra el estado en el que se encuentra esta (ubicada en la parte inferior de cualquier pantalla de la aplicación).

Ilustración 9 : BARRA DE ESTADO INICIAL

Si cuando estamos en una pantalla, seleccionamos un registro del listado que aparece, en la barra de estado obtendremos información específica sobre dicho registro.

Usuario Alta	Fecha Alta	Usuario Modificación	Fecha Modificación	No Modificable	Estado

Ilustración 10 : BARRA DE ESTADO CON ELEMENTO SELECCIONADO

El significado de los campos de la barra de estado son los siguientes:

- **Usuario Alta:** Nombre del usuario que ha realizado el alta de un registro.
- **Fecha Alta:** Fecha en la que se realizó el alta.
- **Usuario Modificación:** Nombre de usuario que ha realizado la última modificación.
- **Fecha Modificación:** Fecha de la última modificación
- **No Modificable:** Si el registro es "No Modificable", en la barra de estado aparecerá.
- **Estado:** Aparecerá el estado correspondiente al estado de la pantalla en un momento determinado.

9.2.2 Acciones

Según el estado en el que se encuentre la pantalla, se podrán realizar unas acciones u otras.

Icono	Descripción
 **Nuevo**	Sirve para pasar el estado de cualquier pantalla de mantenimiento a estado "Nuevo". Todos los campos que sean necesarios para el alta de un nuevo registro serán habilitados y por tanto, será posible cumplimentarlos.
 **Aceptar**	Sirve para la siguientes opciones: ❖ Ejecutar búsquedas

	❖ Confirmar nuevos registros y darlos de alta ❖ Confirmar modificaciones
 **Cancelar**	Con este botón conseguimos volver al estado inicial limpiando todos los campos y dejándolos por defecto.
 **Cerrar**	Al hacer clic en este botón la pantalla que este mostrándose será cerrada. La aplicación la podremos cerrar mediante la ventana convencional de Windows.

Tabla 10 : ACCIONES EN UNA PANTALLA

9.2.3 Operaciones con registros

La guía fármaco-terapéutica permite realizar búsquedas y modificaciones en el **Catálogo**. En las pantallas de **Mantenimiento**, permite realizar altas, modificaciones y búsquedas.

Existen dos casillas en todas las pantallas de la guía que nos informan de si un registro esta activo o si es modificable. El significado de dichas casillas se describe en la siguiente tabla:

ICONO	DESCRIPCIÓN
Activo	El campo “Activo” sirve para indicar si un registro está activo. Se refiere a si va a ser mostrado en el catálogo.
Modificable	El campo “Modificable” sirve para indicar si un registro es modificable. Si esta activo, el registro será modificable. Si no está activo, no se podrá realizar ninguna modificación salvo el campo “Modificable”.

Tabla 11 : CASILLAS DE ACTIVO Y MODIFICABLE

El activar registros repercutirá luego en la pantalla del catálogo, ya que toda la información que se muestra se basa en la que está activa en las pantallas de mantenimiento.

9.2.3.1 Dar de alta un registro

Para llegar al estado de alta *(recordar que en el catálogo no se pueden realizar altas)*, hay que seguir los siguientes pasos:

1. Ir al menú acciones, ir hacer clic en "Nuevo". Esto hará que la pantalla pase a estado "Alta".

2. Se habilitaran los campos necesarios para el alta. A partir de ahí, hay que rellenar todos los campos necesarios.

3. Una vez todo rellenado, hay que ir al menú acciones y hacer clic en  "Aceptar". Si todo ha ido correctamente, aparecerá un mensaje con la confirmación del alta. Si hay algo incorrecto, aparecerá un mensaje advirtiéndolo

Para volver al estado de búsqueda, simplemente en pulsar la tecla "ESC" se volverá al estado inicial.

9.2.3.2 Realizar la modificación de un registro

Para llegar al estado de modificación se puede llegar a través de una búsqueda o de una selección de un registro, para eso hay que seguir los siguientes pasos:

1. Hay que realizar alguna búsqueda y/o seleccionar un registro, entonces podremos comprobar en la barra de estado que la pantalla ha pasado a estado de modificación.

2. Se habilitaran los campos necesarios para la modificación. A partir de ahí, hay que rellenar todos los campos que se desee modificar.

3. Una vez se haya modificado lo deseado, hay que ir al menú acciones y hacer clic en  "Aceptar". Si todo ha ido correctamente, se guardará la modificación. Si hay algo incorrecto, aparecerá un mensaje advirtiéndolo.

Para volver al estado de búsqueda, simplemente en pulsar la tecla “ESC” se volverá al estado inicial.

9.2.3.3 Realizar la búsqueda de un registro o varios

Al abrir cualquier pantalla de la aplicación, se pondrá en estado de búsqueda, con la cual cosa permitirá realizar búsquedas para consultar, modificar…

Una vez se compruebe en la barra de estado que la pantalla está en estado de búsqueda, podremos ver que se activan los campos que se pueden utilizar para realizar alguna búsqueda.

Se introduce lo que se desee buscar en los campos correspondientes. Luego se procede a ejecutar la búsqueda, para hacerlo hay dos formas:

- Ir al menú acciones y hacer clic en “Aceptar”.
- En el campo por el que queremos buscar, pulsar la tecla “Intro”.

Para volver al estado de búsqueda, simplemente en pulsar la tecla “ESC” se volverá al estado inicial.

9.2.4 Cancelar

La función de Cancelar es la de borrar todo lo que se haya podido hacer y vuelve al estado de búsqueda. Para realizar esta acción, hay dos formas de hacerlo:

- Se puede volver al estado inicial pulsando la tecla “ESC”.
- Se puede volver mediante el menú acciones, pulsando el botón “Cancelar”.

9.2.5 Cerrar

Para cerrar cualquier pantalla de la aplicación, hay que ir al menú de acciones y pulsar en el botón que pone “Cerrar”. También, en la pestaña de cualquier

pantalla podemos ver que está el icono de cerrar, con sólo pulsarlo la pantalla se cerrará.

9.3 Catálogo

En el catálogo se van a poder realizar búsquedas sobre presentaciones farmacéuticas por código (el código nacional de la presentación farmacéutica que hace referencia al código único de esta en todo el país) y por descripción. Una vez realizada la búsqueda, obtendremos un listado con todos los resultados encontrados.

Cuando ya tengamos en la pantalla el resultado de la búsqueda, si queremos obtener información detallada sobre una presentación farmacéutica en concreto sólo tendremos que seleccionarla en el listado y instantáneamente se cargaran todos sus datos en los campos correspondientes del Catálogo.

Para entender el catálogo, a continuación se va a explicar cómo funcionan las búsquedas y la información que puede ser consultada.

Ilustración 11 : DETALLES DEL CATÁLOGO

9.3.1 Búsquedas

9.3.1.1 Búsqueda por código

Para realizar una búsqueda por código (nacional), se introduce el código de la presentación farmacéutica que deseemos obtener su información. *Ejemplo: 1024.*

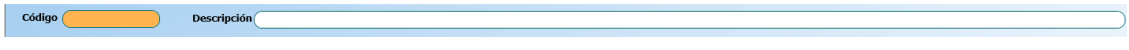

Ilustración 12 : BÚSQUEDA POR CÓDIGO

Una vez ya introducido, tenemos dos opciones para ejecutar la búsqueda:

- Ir al menú acciones y hacer clic al botón Aceptar.
- Pulsar la tecla Intro.

En caso de que el código sea correcto nos mostrará la información de la presentación farmacéutica que corresponda a dicho código. Si no es correcto, la aplicación nos lo advertirá.

9.3.1.2 Búsqueda por descripción

Para realizar una búsqueda por descripción, se introduce la descripción de la presentación farmacéutica que deseamos obtener en el campo correspondiente. *Ejemplo: paracetamol.*

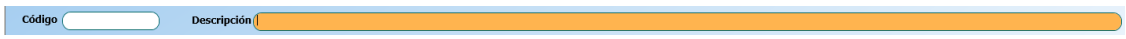

Ilustración 13 : BÚSQUEDA POR DESCRIPCIÓN

Una vez ya introducida, tenemos dos opciones para ejecutar la búsqueda:

- Ir al menú acciones y hacer clic al botón Aceptar.
- Pulsar la tecla Intro.

El funcionamiento de la búsqueda por descripción es el siguiente:

Se introduce una palabra a buscar, a partir de esta se obtiene un listado con los resultados. El listado puede ser visualizado por el usuario y si además quiere obtener información completa sobre una presentación farmacéutica, sólo tendrá que seleccionar una.

Si en la descripción introducimos una palabra que tenga menos de 4 caracteres, la aplicación nos advertirá con un mensaje de que la búsqueda puede tardar más tiempo en devolver los resultados.

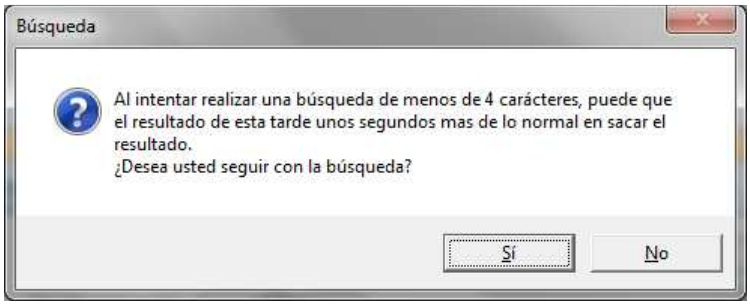

Ilustración 14 : MENSAJE ADVERTENCIA EN LA BÚSQUEDA EN EL CATÁLOGO

A continuación se van a describir las diferentes posibles consultas disponibles del catálogo que pueden ser visualizadas bajo la previa selección de una presentación farmacéutica.

9.3.2 Consultas

9.3.2.1 Datos Generales

Para obtener los datos generales de una presentación farmacéutica, es necesario que previamente se haya seleccionado una presentación farmacéutica del listado.

Una vez seleccionada, se cargarán todos los datos disponibles de esta, los cuales se podrán consultar.

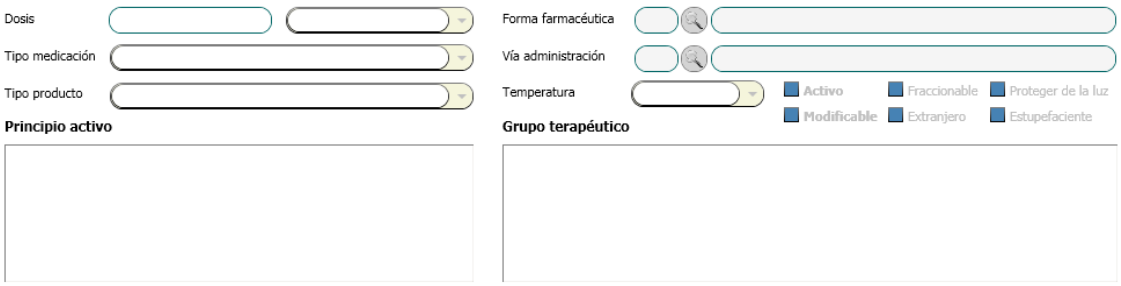

Ilustración 15 : PESTAÑA DATOS GENERALES

- **Dosis:** Cantidad de principio activo de un medicamento, expresado en unidades de volumen o peso por unidad de toma en función de la presentación.
- **Tipo medicación:** se mostrará de la lista de tipos de medicación el perteneciente a esta presentación.
- **Tipo producto:** se mostrará de la lista de tipos de producto el perteneciente a esta presentación.

- **Forma farmacéutica:** es la disposición individualizada a que se adaptan los fármacos (principios activos) y excipientes (materia farmacológicamente inactiva) para constituir un medicamento. Dicho de otra forma, la disposición externa que se da a las sustancias medicamentosas para facilitar su administración.
- **Vía administración:** forma que se elige para hacer llegar un fármaco hasta su destino.
- **Temperatura:** temperatura en la que se presenta la presentación.
- **Principios activos y asociados:** jerarquía de los principios activos y asociados de la presentación farmacéutica.
- **Grupo terapéutico:** organización de las sustancias farmacológicas y medicamentos agrupados.
- **Fraccionable:** muestra si es fraccionable.
- **Extranjero:** muestra si es extranjero o nacional.
- **Protege luz:** muestra si es necesario proteger de la luz.
- **Estupefaciente:** muestra si es un estupefaciente.
- **Activo:** indica si la presentación farmacéutica está de alta en el hospital, si es así se mostrará en la guía.
- **Modificable:** indica si dicha presentación farmacéutica es modificable por el personal farmacéutico.

9.3.2.2 Características

La pestaña de características se subdivide en Generales, Contraindicaciones y Efectos. En esta pestaña, una vez se haya seleccionado una presentación farmacéutica, obtendremos toda la información disponible sobre las características.

Ilustración 16 : PESTAÑA CARACTERÍSTICAS

Es posible que alguna presentación farmacéutica no contenga información en las características.

Si se quiere modificar las características se podrá hacer si el estado es modificable con lo cual todos los campos que se puedan modificar estarán activos.

Para confirmar la nueva información introducida y guardarla, hay que ir al menú de acciones y hacer clic en el botón Aceptar.

A continuación se van a detallar las sub pestañas de Características.

9.3.2.3 Características generales

Las características generales de una presentación farmacéutica son las siguientes:

- **Precauciones:** Actuar de forma cautelosa para evitar posibles efectos.
- **Antes medicación:** Información previa a la administración de una medicación.
- **Teratogénia:** La teratogénia es la capacidad de producir malformaciones. Se dice que una infección, droga o producto es teratogénico cuando es capaz de producir malformaciones.

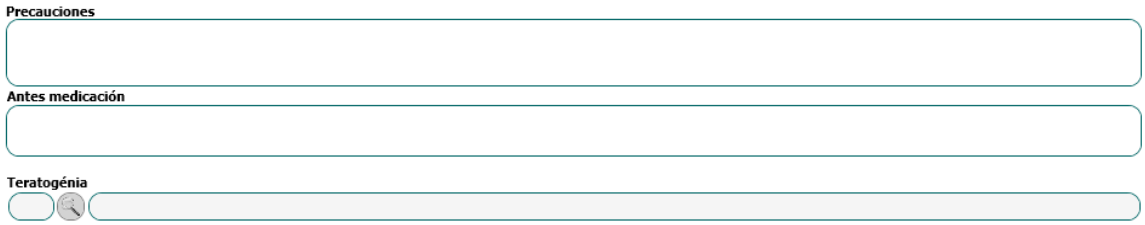

Ilustración 17 : PESTAÑA CARACTERÍSTICAS GENERALES

9.3.2.4 Contraindicaciones

Una contraindicación es una condición o un factor, que incrementa los riesgos involucrados al usar una particular medicación o droga, llevando a cabo un procedimiento médico o comprometiendo en una actividad particular.

Los campos que representan dichas contraindicaciones son:

- **Contraindicación absoluta:** es una condición que prohíbe tajantemente el uso de un tratamiento en conjunto. Por ejemplo, un neumotórax no tratado podría ser una "contraindicación absoluta" para oxigenoterapia.
- **Contraindicación relativa:** condición en contra del uso de un tratamiento que aumente la relación riesgo/beneficio. Por ejemplo, una historia de úlcera péptica es una contraindicación a tomar aspirina. Si, en cambio, el beneficio de emplear aspirina es visto como mayor que el riesgo de una recurrencia a úlcera, y no hay razonables alternativas disponible, el tratamiento aún se indica.

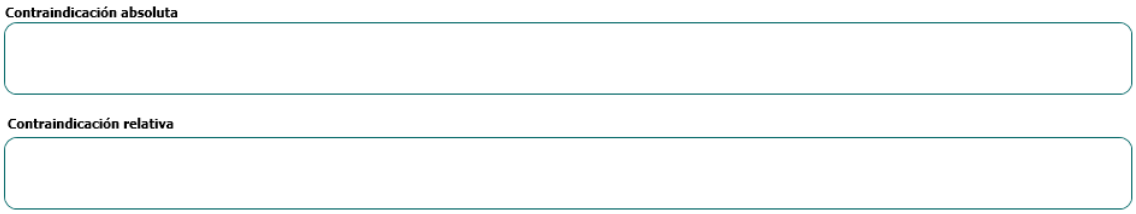

Ilustración 18 : PESTAÑA CONTRAINDICACIÓNES

9.3.2.5 Efectos

Los efectos se definen como cualquier reacción nociva no intencionada que aparece a dosis normalmente usadas en el ser humano para profilaxis, diagnóstico o tratamiento o para modificar funciones fisiológicas. A continuación los efectos que aparecen en dicha pestaña:

- **Efecto frecuente:** reacciones que se repiten a menudo.
- **Efecto ocasional:** reacciones que ocurren de forma ocasional.
- **Efecto raro:** reacciones que ocurren raramente.

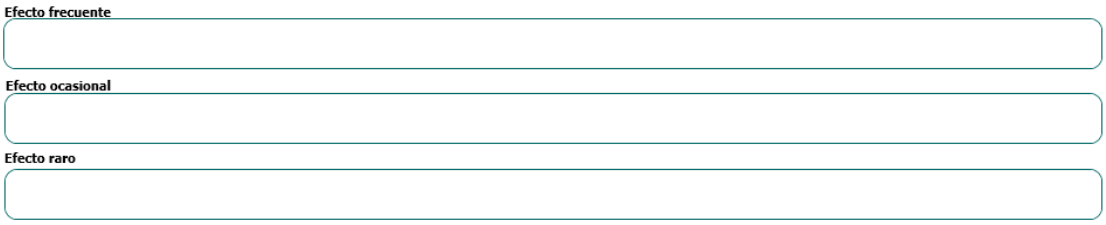

Ilustración 19 : PESTAÑA EFECTOS

9.3.2.6 Indicaciones

Una indicación es el término que describe una razón válida para emplear una prueba diagnóstica, un procedimiento médico, un determinado medicamento, o técnica quirúrgica.

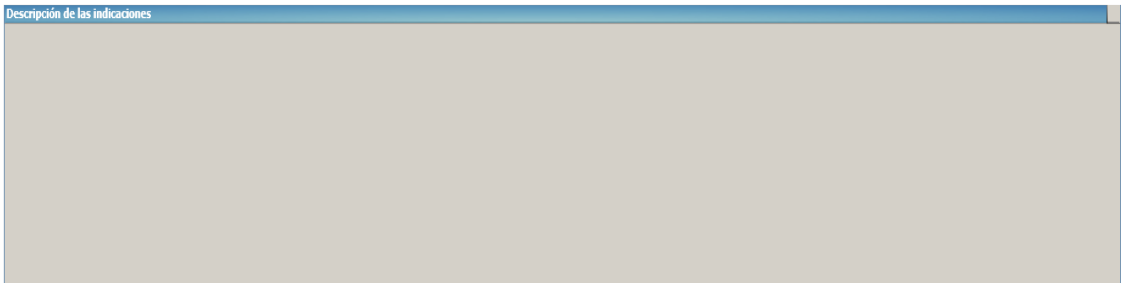

Ilustración 20 : PESTAÑA INDICACIONES

9.3.2.7 Interacciones

Se conoce como interacción farmacológica a la modificación del efecto de un fármaco por la acción de otro cuando se administran conjuntamente. Esta acción puede ser de tipo sinérgico (cuando el efecto aumenta) o antagonista (cuando el efecto disminuye).

Son ejemplos ilustrativos la asociación de la codeína al paracetamol para aumentar su efecto analgésico o la combinación de ácido clavulánico y la amoxicilina para evitar la resistencia de las bacterias al antibiótico.

En la pestaña de interacciones vamos a encontrar los siguientes tipos:

- Interacciones del principio activo de la presentación farmacéutica.
- Interacciones de los principios activos asociados al principio activo.

Si hay alguna interacción de la presentación farmacéutica, ya sea relacionada con el principio activo o con el principio activo asociado se podrá visualizar en alguna de las tres listas existentes en la pestaña de interacciones.

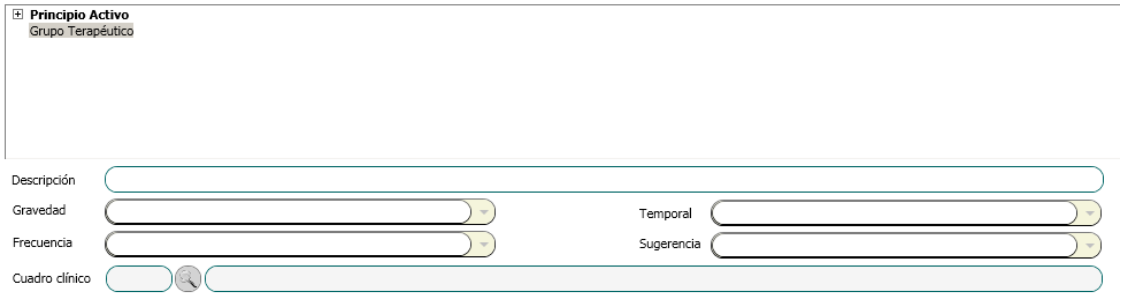

Ilustración 21 : PESTAÑA INTERACCIONES

Para poder ver la información disponible, vemos que en la izquierda aparece un icono como el siguiente ⊞ que permite desplegar un listado con las interacciones correspondientes.

Además, del listado anterior, podemos seleccionar una interacción en concreto para obtener una información más completa, en la cual encontraremos la información siguiente:

- **Descripción:** detalle sobre una interacción.
- **Gravedad:** representa importancia de una interacción
- **Frecuencia:** representa la cantidad de veces que se repite una interacción.
- **Sugerencia:** consejos que se recomiendan llevar a cabo.
- **Temporal:** cada cuanto tiempo ocurre una interacción.
- **Cuadro clínico:** representa las manifestaciones clínicas que se definen por la relación entre los signos y síntomas que presentan en una determinada enfermedad.

9.3.2.8 Posología límite

La posología es la rama de la farmacología que estudia la dosificación de los fármacos. Para hacer una adecuada posología de los diversos fármacos que existen, se debe de tomar en cuenta diversos factores. A continuación vamos a poder ver los campos que contiene la pantalla de posología límite (dosificación límite).

Además al seleccionar una de ellas, se podrá obtener información adicional, que aparece en los siguientes campos:

- **Tipo cadencia:** frecuencia en la que se repite la posología.
- **Tipo posología:** representa el tipo de posología (Pediatría, Adulto, Geriátrica).
- **Cantidad máxima:** dosis máxima de la posología.
- **Por día/por kg y día:** dosis por día o por kg y día.
- **Tramo edad:** intervalo de edad en el cual se va a poder dosificar.
- **Tramo peso:** intervalo de peso en el cual se va a poder dosificar.
- **Vía administración:** forma que se elige para hacer llegar un fármaco hasta su destino.
- **Por toma:** si la posología es por toma.

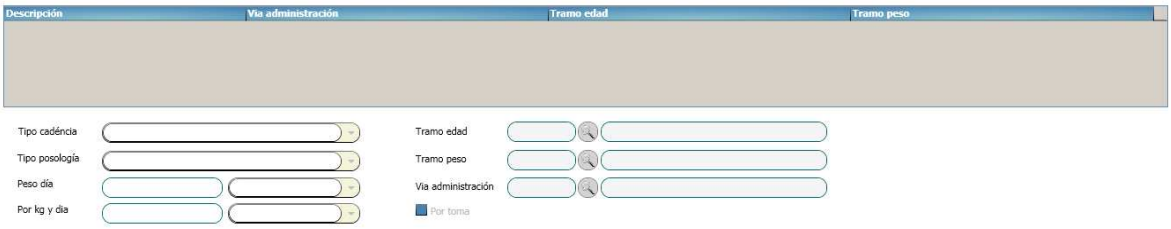

Ilustración 22 : PESTAÑA POSOLOGÍA LÍMITE

9.3.2.9 Posologías usuales

Una presentación farmacéutica puede tener diferentes posologías usuales del mismo tipo (Pediatría, Adulto, Geriátrica). En esta pestaña veremos un listado de las posibles. A continuación vamos a poder ver los campos que contiene la pantalla de posología límite (dosificación usual).

Además al seleccionar una de ellas, se podrá obtener información adicional.

- **Cadencia:** frecuencia en la que se debe de administrar la dosis (en horas o días).
- **Dosis:** cantidad de la dosis.
- **Dosis 2:** cantidad de la dosis 2.
- **Dosis 3:** cantidad de la dosis 3.

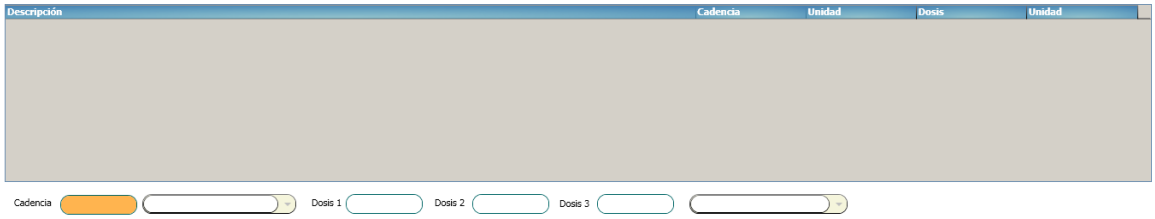

Ilustración 23 : POSOLOGÍA USUAL

9.3.2.10 Compuestos

Una presentación farmacéutica está compuesta por varias composiciones. En esta pestaña, se va a visualizar un listado de todas las composiciones que una presentación pueda tener.

Además al seleccionar una de ellas, se podrá obtener la siguiente información adicional:

- **Vía administración:** forma que se elige para hacer llegar un fármaco hasta su destino.
- **Producto :** Nombre del producto que pertenece a la composición
- **Principio activo :** principio activo del producto de la composición
- **Excipiente:** muestra si es excipiente.
- **Cantidad:** cantidad de la composición.
- **Cantidad máxima:** cantidad máxima de la composición.

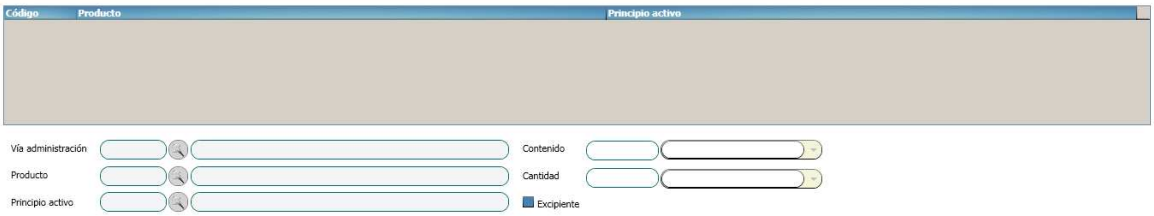

Ilustración 24 : COMPUESTOS

9.3.2.11 Productos

Una presentación farmacéutica puede tener varios productos. En esta pestaña, se va a visualizar un listado de todos los productos que una presentación pueda tener.

Además al seleccionar uno de ellos, se podrá obtener la siguiente información adicional:

- **Descripción:** Nombre comercial del producto.
- **Laboratorio:** Lugar dotado de los medios necesarios para fabricar productos.
- **Grupo terapéutico:** Recoge el sistema u órgano sobre el que actúa, el efecto farmacológico, las indicaciones terapéuticas y la estructura química del fármaco.
- **Tipo de aportación:** Si hay algún tipo de colaboración.

Ilustración 25 : PRODUCTOS

9.4 Mantenimiento

Definimos el mantenimiento de la aplicación como pantallas que se van a utilizar para visualizar, modificar información específica y además añadir nueva información.

El mantenimiento de la aplicación de divide en dos secciones: Mantenimiento simple y Mantenimiento compuesto.

Ilustración 26 : ICONOS MANTENIMIENTO SIMPLE

En resumen, las pantallas de mantenimiento se utilizan para gestionar la información que luego es mostrada en el catálogo a la hora de cargar cualquier presentación farmacéutica.

9.4.1 Mantenimiento simple

Para ilustrar una pantalla de mantenimiento simple, vamos a utilizar como ejemplo el tramo de edad, a continuación se puede ver la captura de este.

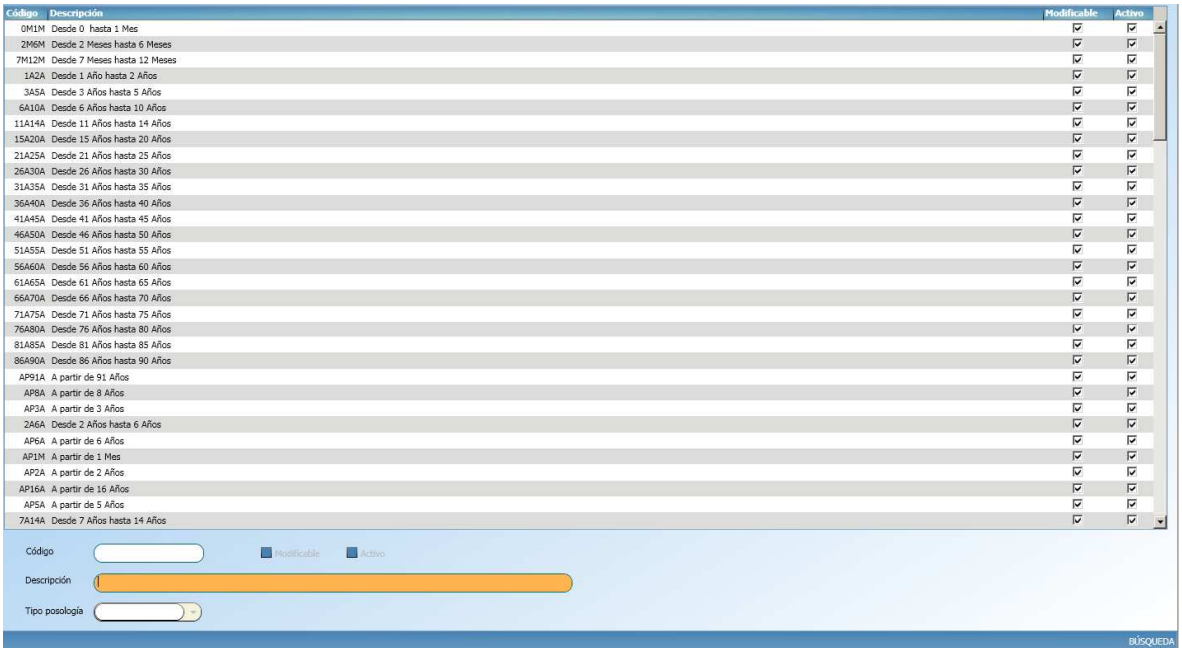

Ilustración 27 : MANTENIMIENTO SIMPLE

Cualquier pantalla de mantenimiento simple tiene los siguientes elementos:

- Menú acciones
- Listado
- Campos
- Barra de estado

En cada pantalla de mantenimiento simple podremos comprobar que arriba se activa un menú llamado **Acciones**. En este, vamos a encontrar las acciones Nuevo, Aceptar, Cancelar y Salir anteriormente ya descritas.

El **listado** contiene todos los registros existentes y los muestra de forma detallada.

Los **campos** que se visualizan en la pantalla de mantenimiento tienen varias funciones según el estado en el que se encuentre la pantalla. (En el punto 9.2 se explica el funcionamiento).

Podemos realizar búsquedas principalmente por Código y Descripción. Hay pantallas de mantenimiento que tienen muchos más campos y por tanto, permiten búsquedas por otros campos (producto, principio activo…). Para ejecutar la búsqueda, se puede realizar de dos formas distintas:

- Menú acciones
- Tecla Intro

Por regla general, ante cualquier estado, si queremos volver al estado inicial y deshacer todo lo que podamos haber realizado, con presionar la tecla "ESC" lo conseguiremos.

Si queremos visualizar todos los campos con detalle de un registro que hay en el listado, es necesario seleccionarlo del listado.

9.4.2 Mantenimiento compuesto

Para describir las pantallas de mantenimiento compuesto, vamos a ver la pantalla "Producto" como ejemplo. Cualquier pantalla de mantenimiento compuesto tiene los mismos elementos que las pantallas de mantenimiento simple. Los elementos son los siguientes:

- Menú acciones
- Listado
- Campos
- Barra de estado

La diferencia entre mantenimiento simple y compuesto reside en que en el mantenimiento compuesto obtenemos información compuesta de varios mantenimientos simples.

Es decir, en un mantenimiento simple sólo estamos visualizando o modificando una cosa, por ejemplo un tramo de edad. En un mantenimiento compuesto estamos visualizando o modificando varias cosas, por ejemplo un producto, y un producto se compone de una presentación farmacéutica, de un grupo terapéutico, de un

laboratorio y además una relación con los códigos SIVSA de los medicamentos disponibles en el hospital.

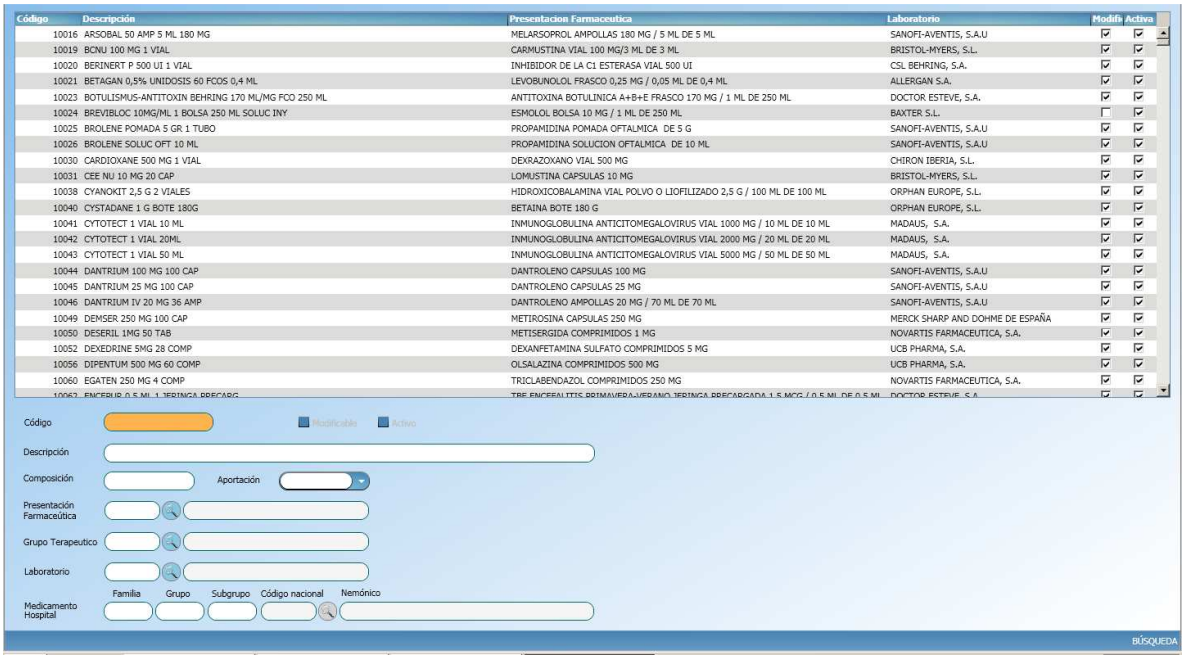

Ilustración 28 : MENTENIMIENTO COMPUESTO

10 PRUEBAS UNITARIAS

10.1 Introducción

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto.

10.2 Características

Para que una prueba unitaria sea buena se deben cumplir los siguientes requisitos:

- **Automatizable:** no debería requerirse una intervención manual. Esto es especialmente útil para integración continúa.
- **Completas:** deben cubrir la mayor cantidad de código.

- **Repetibles o Reutilizables:** no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez.
- **Independientes:** la ejecución de una prueba no debe afectar a la ejecución de otra.
- **Profesionales:** las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

10.3 Ventajas

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cuatro ventajas básicas:

- **Fomentan el cambio:** Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- **Documenta el código:** Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
- **Separación de la interfaz y la implementación:** Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro.
- **Los errores están más acotados y son más fáciles de localizar:** dado que tenemos pruebas unitarias que pueden desenmascararlos.

10.4 Limitaciones

Es importante darse cuenta de que las pruebas unitarias no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto. Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir en realidad la unidad de

programa bajo estudio. Las pruebas unitarias sólo son efectivas si se usan en conjunto con otras pruebas de software.

10.5 Pruebas por pantalla

Las pruebas se han realizado por usuarios ajenos a esta, con conocimientos mínimos de informática y sin ninguna relación con el desarrollo de la aplicación.

Cómo todas las pantallas son similares de apariencia y tienen las mismas funcionalidades se han realizado las mismas pruebas para todas.

A continuación una tabla con todas las pruebas realizadas a todas las pantallas.

PRUEBAS	SUPUESTO
Comparación contenido barra de estado.	La barra de estado tiene que tener todos los datos que deben de ser mostrados. Debe de actualizarlos correctamente.
Comprobación estado de búsqueda.	Los campos necesarios para realizar búsquedas deberán estar activos.
Comprobación estado de alta.	Los campos necesarios para realizar un alta deben de estar activos. Hay algunos que serán obligatorios.
Comprobación estado de modificación.	Los campos necesarios para realizar una modificación deben de estar activos.
Comprobar que la longitud de los campos	Todos los campos de la pantalla deben

este limitada.	de tener limitada su longitud. Esta longitud será la que se ha indicado en la base de datos.
Comprobar que los campos tengan un formato correcto.	Comprobar que si un campo es numérico no acepte texto y no sea negativo.
Comprobar la realización de un alta.	Ver que todos los campos necesarios para realizar el alta han sido recogidos por la aplicación. Comprobar que se ha insertado correctamente en la base de datos el nuevo registro.
Comprobación la realización de una modificación.	Ver que todos los campos necesarios para realizar la modificación han sido recogidos por la aplicación. Comprobar que se ha actualizado correctamente en la base de datos el registro.
Comprobación de la realización de búsquedas a partir de los campos disponibles.	Se pueden realizar búsquedas por distintos campos. Comprobar para cada campo disponible para la búsqueda que recoge bien la información y realiza la búsqueda correctamente. Después, ver si muestra correctamente en el listado el resultado.
Comprobar que los campos necesarios no	Comprobar que en los campos

admitan un campo en blanco para el alta o la modificación.	necesarios para realizar una inserción o modificación no se pueda introducir un campo en blanco.
Comprobar si hay registros duplicados.	Antes de realizar una nueva inserción, comprobar con el código introducido si ya existe en la base de datos. Si ya existe mostrar un mensaje informando de que el elemento esta duplicado y volver al estado inicial. Si no está duplicado, seguir con la inserción.
Comprobar que los mensajes de información de la pantalla sean homogéneos a todas las otras.	Los mensajes informativos de una nueva alta, elemento duplicado, errores, avisos, etc. todos deben de ser iguales en todas las pantallas.
Comprobar que se active el menú acciones en la pantalla y funcione correctamente.	Cada pantalla tiene su propio menú de acciones. Este menú aparentemente es el mismo para todas, pero funcionalmente es diferente para cada una. Comprobar que realiza las funciones correctamente.
Comprobar sintácticamente la pantalla.	Todos los títulos de las pantallas deben estar escritos correctamente. También los títulos de los iconos, columnas, nombres de campos, etc.
Comprobar que el ancho de cada columna	Respecto a la información que saca

de los listados es correcto.	cada listado, se ha establecido un ancho a cada columna.
Comprobación títulos de las pantallas.	Comprobar que los títulos de cada pantalla hacen referencia exactamente a esa pantalla.
Comprobación iconos de las pantallas.	Cada icono debe tener tamaño 64x64 y 128x128. Hay dos tamaños debido los diferentes tipos de resoluciones de pantalla.
Comprobación con diferentes resoluciones para visualizar el menú de mantenimiento que tiene muchos iconos.	Ver en distintas resoluciones que los iconos se muestran correctamente.
Comprobación icono aplicación.	Ver en la barra de tareas y en la barra de la aplicación que aparece el icono que hace referencia a la guía.
Comprobación que los fondos de cada pantalla tengan el mismo degradado.	El degradado en cada pantalla tiene que ser el mismo.
La situación de todos los campos es similar o igual en todas las pantallas.	Respecto a la cantidad de campos en cada pantalla, todos deben de estar situados de forma similar en todas las pantallas.

Tabla 12 : PRUEBAS POR PANTALLA

11 CONCLUSIÓN

El proyecto de la Guía Fármaco-Terapéutica nace gracias a la necesidad de incorporar, en el servicio de Farmacia del Hospital General Universitario de Valencia , de alguna forma un control de los medicamentos existentes en el hospital.

En su momento la guía fue propuesta cómo una aplicación que debería ser similar a una biblioteca digital de los medicamentos que el Hospital dispusiera para realizar un mantenimiento correcto y personalizado por parte del Hospital.

Así que se empezó con una simple especificación de requisitos, seguida de un análisis funcional. A esto le siguió el diseño de los diferentes diagramas necesarios para facilitar la implementación de las bases de datos y de las clases necesarias para obtener una solución de calidad.

Cuando la aplicación ya estaba en fase de pruebas, el servicio de Farmacia comunicó que si se podría relacionar de alguna forma los medicamentos existentes en el hospital (almacenados en las tablas de la base de datos de SIVSA) con los medicamentos que conselleria de sanitat ha proporcionado al Hospital (todos los medicamentos que están dados de alta en la Generalitat Valenciana y están almacenados en la base de datos de la Guía Fármaco-Terapéutica gracias a la carga de datos realizada a partir de los ficheros proporcionados por conselleria de sanitat).

Se comprobaron que los códigos de los medicamentos almacenados en la base de datos de la guía estaban identificados por un código que correspondía al Código Nacional (el código nacional de la presentación farmacéutica que hace referencia al código único de esta en todo el país) del medicamento. Luego en la base de datos del Hospital, concretamente en la base de datos SIVSA, estaban almacenados los medicamentos que el Hospital utiliza, que también están identificados por el Código Nacional. Se procedió a realizar un cruce de la información mediante un procedimiento y se actualizaron las tablas para que sólo mostrara “Activos” los medicamentos existentes en el Hospital.

En resumen, ha sido una gran experiencia haber creado la Guía Fármaco-Terapéutica tanto por los conocimientos adquiridos, como por el reconocimiento por los profesionales del servicio de Farmacia por el proyecto creado.

12 ANEXO

12.1 Modelo vista presente (MVP)

12.1.1 ¿Cuál es el concepto?

Las aplicaciones web tradicionales utilizan un framework[12] llamado ASP.NET[13] que contienen la interfaz de usuario (UI) en el código subyacente de la página. Esto hace que la lógica sea difícil o imposible de probar con frecuencia. Deja la interfaz de usuario en un estado frágil, donde los cambios a la interfaz de usuario a menudo rompen la lógica.

Ilustración 29 : MODELO VISTA PRESENTE

El patrón modelo Vista-Presente (Model View Presenter (MVP) Bundle, 2008) proporciona una forma a los desarrolladores para separar la lógica de la interfaz de usuario en un formulario donde es más fácil de probar.

[12] **Framework:** En el desarrollo de *software* es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, con base a la cual otro proyecto de *software* puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

[13] **ASP.NET:** Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

Además, la separación hace que la lógica de la capa de negocio esté centrada en la interfaz de usuario. Por tanto, menos propenso a fallos cuando se realizan cambios a la interfaz de usuario.

12.1.2 Problema planteado

Una página que tiene una aplicación web contiene controles para mostrar datos del dominio de aplicación. Un usuario puede modificar los datos y presentar los cambios. La página recupera los datos de dominio, se ocupa de los eventos de usuario, altera otros controles en la página en respuesta a los acontecimientos, y presenta los datos de dominio cambiado. Escribir el código en la página web (la clase de código subyacente) hace que una clase sea compleja, difícil de mantener, y difícil de probar. Además, es difícil tener código compartido entre las páginas Web que requieren el mismo comportamiento.

Hay unas necesidades:

- Se desea maximizar el código que se puede probar con la automatización.
- Se desea compartir el código entre las páginas que requieren el mismo comportamiento.
- Se quiere que la lógica de negocio sea independiente de la lógica de la interfaz de usuario para que sea más fácil de entender y mantener.

Una posible solución:

Separar las responsabilidades de la representación visual y el comportamiento de control de eventos en diferentes clases, la vista y el presentador. La clase que representa la vista (la página web) gestiona los controles en la página, y se reenvía eventos a una clase de presentador. El presentador contiene la lógica para responder a los eventos, actualizar el modelo (lógica de negocio y los datos de la aplicación) y, a su vez, manipular el estado de la vista.

Para facilitar las pruebas del presentador, el presentador tiene una referencia a la interfaz de vista en lugar de a la implementación de vista concreta. De esta manera, se puede fácilmente reemplazar la vista real con un simulacro de ejecución para ejecutar las pruebas.

12.1.3 Implementación

El proceso de implementación (Model-View-Presenter Pattern, 2007) típica del patrón Modelo-Vista-Presente incluye las siguientes tareas:

- **La implementación de una clase de presentador.** El presentador debe manejar los eventos de usuario, actualizar el modelo, y manipular el estado de la vista.
- **La implementación de una interfaz de vista.** La interfaz de la vista debe exponer los elementos de la vista del estado.
- **La implementación de una clase de vista.** La vista contiene objetos de interfaz de usuario. La vista debe enviar los eventos de usuario al presentador y exponer las propiedades para que el presentador pueda manipular el estado de la vista.

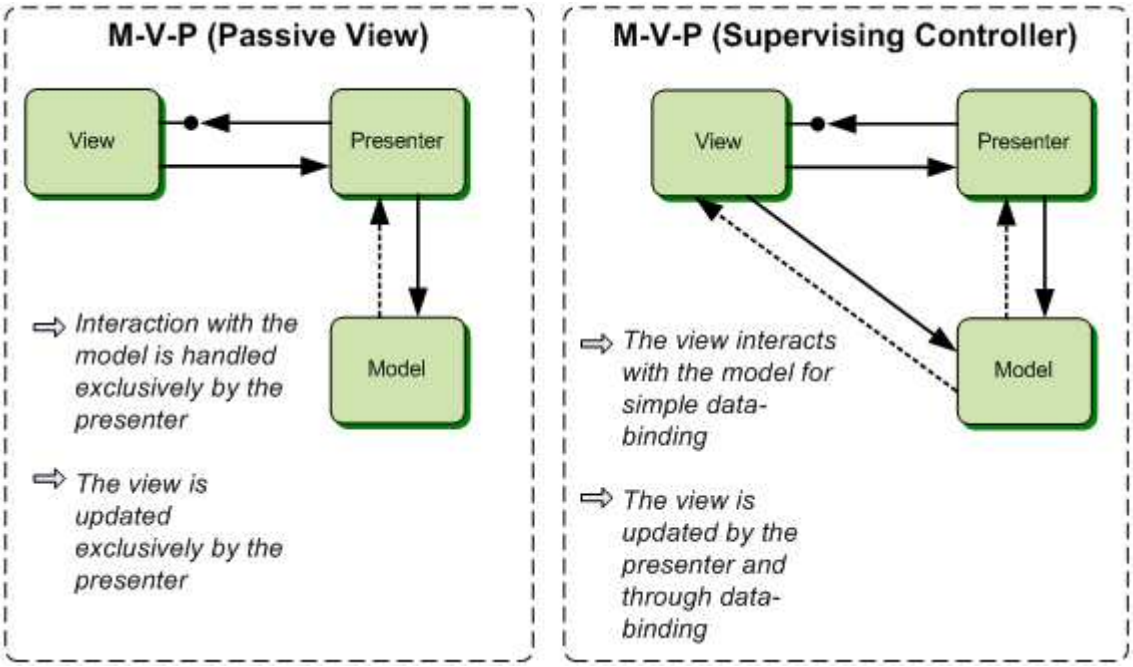

Ilustración 30 : MODELO VISTA PRESENTE 2

El presentador

Por lo general, el presentador se implementa mediante Test Driven Development (TDD)[14]. A partir del presentador, la aplicación le permite centrarse en los requerimientos del negocio y la funcionalidad de la aplicación de forma independiente de los detalles de implementación de la interfaz.

Al aplicar el presentador, tendrá que crear una vista simple y objetos de modelo para poder expresar la interacción entre estos y el presentador. Para poder probar el presentador de manera independiente, el presentador tiene la referencia de la interfaz de la vista en lugar de ver la aplicación concreta. De esta manera, se puede fácilmente reemplazar la vista con un simulacro de aplicación al escribir y ejecutar pruebas.

La vista

La comunicación con la vista generalmente se logra mediante la creación y consulta de propiedades del objeto para establecer y obtener el estado de la vista, respectivamente. Otro enfoque válido consiste en la invocación de métodos en la vista.

La interfaz

La interfaz de la vista debe exponer el estado de la vista. Por lo general, una interfaz que representa la vista contiene propiedades para que el presentador pueda establecer y consultar el estado de la vista. La exposición de las propiedades sobre los métodos en la vista por lo general mantiene el presentador más sencillo porque no necesita saber acerca de los detalles de implementación de vista, como cuando los datos se van a enlazar a los controles de interfaz de usuario.
En función de cómo interactúa con la vista del presentador, la interfaz de la vista tiene elementos adicionales.

[14] **Test Driven Development:** Desarrollo guiado por pruebas (Wikipedia, 2011). Es una práctica de programación que involucra otras dos prácticas: *Escribir las pruebas primero (Test First Development)* y *Refactorización (Refactoring).*

12.2 Language-Integrated Query (LINQ)

12.2.1 ¿Qué aporta LINQ?

Language-Integrated Query (LINQ) (LINQ : Language-Integrated Query) es un conjunto de características presentado en Visual Studio 2008 que agrega capacidades de consulta eficaces a la sintaxis de los lenguajes C# y Visual Basic. LINQ incluye patrones estándar y de fácil aprendizaje para consultar y actualizar datos, y su tecnología se puede extender para utilizar potencialmente cualquier tipo de almacén de datos. Visual Studio incluye ensamblados de proveedores para LINQ que habilitan el uso de LINQ con colecciones de .NET Framework, bases de datos SQL Server, conjuntos de datos de ADO.NET y documentos XML.

12.2.2 Consulta con LINQ

Una consulta (Introducción a las consultas LINQ) es una expresión que recupera datos de un origen de datos. Las consultas normalmente se expresan en un lenguaje de consultas especializado. A lo largo del tiempo se han ido desarrollando lenguajes diferentes para los distintos tipos de orígenes de datos, como SQL para las bases de datos relacionales y XQuery[15] para XML.

Por tanto, los desarrolladores han tenido que aprender un nuevo lenguaje de consulta para cada tipo de origen de datos o formato de datos que deben usar. LINQ simplifica esta situación al proporcionar un modelo coherente para trabajar con los datos de varios tipos de formatos y orígenes de datos. En una consulta LINQ, siempre se trabaja con objetos. Se utilizan los mismos modelos de codificación básicos para consultar y transformar datos de documentos XML, bases de datos SQL, conjuntos de datos ADO.NET[16], colecciones .NET [17]y cualquier otro formato para el que haya disponible un proveedor LINQ.

[15] **XQuery:** es un lenguaje de consulta diseñado para colecciones de datos XML. Es semánticamente similar a SQL, aunque incluye algunas capacidades de programación.
[16] **ADO.NET:** es un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos.
[17] **NET:** es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Todas las operaciones de consulta LINQ se componen de tres acciones distintas:

1. Obtención del origen de datos.
2. Creación de la consulta.
3. Ejecución de la consulta.

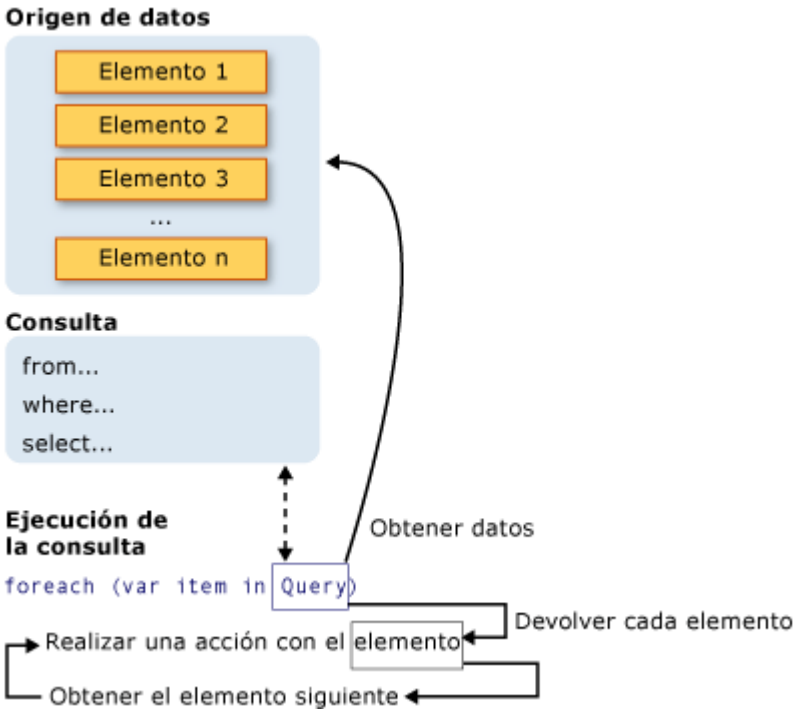

Ilustración 31 : LINQ

12.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) (Introducción a WPF) es un sistema de presentación de la próxima generación, para crear aplicaciones cliente de Windows que proporcionen una experiencia impactante para el usuario desde el punto de vista visual.

El núcleo de WPF es un motor de representación basado en vectores e independiente de la resolución que se crea para sacar partido del hardware de gráficos moderno. WPF extiende el núcleo con un conjunto completo de características de desarrollo de

aplicaciones que incluye Extensible Application Markup Language (XAML)[18], controles, enlace de datos, diseño, gráficos 2-D y 3-D, animación, estilos, plantillas, documentos, multimedia, texto y tipografía. WPF se incluye en Microsoft .NET Framework[19], de modo que es posible compilar aplicaciones que incorporen otros elementos de la biblioteca de clases de .NET Framework.

Ilustración 32 : WPF

WPF existe como un subconjunto de .NET Framework que se busca en la mayor parte en el espacio de nombres System.Windows. Si se ha compilado previamente las aplicaciones con .NET Framework utilizando las tecnologías administradas como ASP.NET y Windows Forms, el WPF fundamental que programa la experiencia debería estar familiarizado; crea instancias de las clases, establece las propiedades, llama a los métodos y eventos de controlador, todos utilizando su lenguaje de programación .NET Framework favorito, como C# o Visual Basic.

[18] **XAML:** es un lenguaje declarativo basado en XML, optimizado para describir gráficamente interfaces de usuarios visuales ricas desde el punto de vista gráfico.

[19] **.NET Framework:** es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma.

WPF proporciona mejoras de programación adicionales para el desarrollo de aplicaciones cliente de Windows. Una mejora evidente es la capacidad para programar una aplicación mediante código de lenguaje marcado y subyacente, una experiencia con la que resultará familiar a los programadores de ASP.NET.

En general, se utiliza el lenguaje marcado Extensible Application Markup Language (XAML) para implementar la apariencia de una aplicación, y los lenguajes de programación administrados (subyacentes) para implementar su comportamiento.

Esta separación entre la apariencia y el comportamiento aporta las ventajas siguientes:

- Se reducen los costes de programación y mantenimiento, al no estar el marcado específico de la apariencia estrechamente relacionado con el código específico del comportamiento.
- La programación es más eficaz porque los diseñadores pueden implementar la apariencia de una aplicación al mismo tiempo que los programadores implementan su comportamiento.
- Se pueden usar varias herramientas de diseño para implementar y compartir el marcado XAML, a fin de responder a los requisitos de quienes colaboran en la programación de aplicaciones.
- La globalización y localización de las aplicaciones WPF se ha simplificado en gran medida

12.4 Microsoft SQL Server 2005

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle[20], PostgreSQL[21] o MySQL[22].

[20] **Oracle:** es un sistema de gestión de base de datos objeto-relacional (o ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation.

SQL Server 2005 (Microsoft SQL SERVER 2005) ha sido diseñado para ayudar a las empresas a enfrentarse a retos. SQL Server 2005 es la solución de gestión de la información de Microsoft que procura mayor escalabilidad, disponibilidad y seguridad a la información empresarial y las aplicaciones de análisis al tiempo que simplifica su creación, implantación y gestión.

Esta solución, erigida sobre la fortaleza de SQL Server 2000, proporciona una plataforma de administración de datos que ayuda a empresas de cualquier tamaño a:

- Desarrollar e implantar aplicaciones empresariales más escalables, fiables y seguras.
- Optimizar la productividad del sector TI reduciendo la complejidad en la creación, implantación y administración de las aplicaciones de bases de datos.
- Aumentar las capacidades de los desarrolladores con un entorno de desarrollo valioso, flexible y actual para que creen bases de datos más seguras.
- Compartir datos a través de múltiples plataformas, aplicaciones y dispositivos para facilitar la interconexión entre sistemas internos y externos.
- Ofrecer soluciones de inteligencia empresarial que ayuden a tomar decisiones con fundamento y aumentar la productividad por toda la empresa
- Controlar los costos sin sacrificar el rendimiento, la disponibilidad ni la fiabilidad.

12.5 Archivo interfaz de la pantalla de tramo de edad

En la vista se implementaran métodos que tienen como fin cargar la información que va a ser mostrada en los objetos de los formularios.

En el presente es dónde se implementa todas las funcionalidades que pueda tener un formulario (búsquedas, guardar…). Es el que actúa como intermediario entre la capa de persistencia y la vista para gestionar toda la información.

[21] **PostgreSQL:** es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.
[22] **MySQL:** es un sistema de gestión de bases de datos relacional, multihilo y multiusuario

En la interfaz se declaran todos los métodos y atributos que han sido implementados en la vista y van a ser usados en el presente.

En resumen, en la interfaz se declaran los métodos y atributos de la vista que son necesarios para la implementación del presente. Por ejemplo: métodos los cuales se les pasa información que va a ser mostrada en la vista, atributos de la pantalla necesarios para realizar búsquedas, inserciones, actualizaciones…

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Controls;
using CHGUV.Guia.Farmacoterapeutica.Business.HGUVHis;
namespace CHGUV.Guia.Farmacoterapeutica.Pages.Maestros.TramoEdad
{
    public interface TramoEdadInterface
    {
        //Definimos los campos necesarios para la página de TramoEdad
        string Codigo {get;}
        string Descripcion{get;}
        bool Activo{get;}
        bool Modificable{get;}
        string TipoSol { get; }
        /// <summary>
        /// Declaración de la carga de una lista de tramos de edad
        /// </summary>
        /// <param name="lista">Listado de tramo de edad</param>
        void Load(List<CFG_CAT_FAR_TramoEdad> lista);
        /// <summary>
        /// Declaración de la carga de un objeto de tramo de edad
        /// </summary>
        /// <param name="via">Un objeto de tramo de edad</param>
        void LoadTramoCampos(CFG_CAT_FAR_TramoEdad tramo);
        /// <summary>
        /// Declaración de la carga de una lista de tipos de posología
        /// </summary>
        /// <param name="via">Una lista de tipos de posología</param>
        void LoadCombo(List<CFG_CAT_FAR_PosologiaTipo> lista);
        /// <summary>
        /// Estado de inicio
        /// </summary>
        void Inicio();
        /// <summary>
        /// Estado de modificación
        /// </summary>
        void Modificacion();
        /// <summary>
        /// Estado cerrar
        /// </summary>
        void Cerrar();
        /// <summary>
        /// Estado de búsqueda
        /// </summary>
        void Busqueda();
        /// <summary>
        /// Estado de alta
        /// </summary>
```

```
        void Alta();
        void NoModificable();}}
```

Código C# 1: INTERFAZ TRAMO DE EDAD

12.6 Archivo presente de la pantalla tramo de edad

El presente es el archivo en dónde están los métodos que hacen de intermediario entre la vista y la capa de persistencia. Estas funciones realizan la gestión de la información de la pantalla.

En el caso de las pantallas de la guía, se implementan métodos de búsqueda, de guardado (inserciones y modificaciones), de transferencia de información a la vista, etc. A continuación se puede ver el código del presente de la pantalla tramo de edad:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using WpfCCLibrary.Shared.Presenter; //incluir
using CHGUV.Guia.Farmacoterapeutica.Business.HGUVHis;

namespace CHGUV.Guia.Farmacoterapeutica.Pages.Maestros.TramoEdad
{
    public class TramoEdadPresenter : Presenter<TramoEdadInterface>
    {
        #region Atributes
        /// <summary>
        /// Enumeración de estados que puede tener el formulario
        /// </summary>
         public enum EstadosRegistroES { INICIO, ALTA, MODIFICACION,BUSQUEDA };
         //Atributos de estado

         private EstadosRegistroES _Estado = EstadosRegistroES.INICIO;
         //Inicializamos el estado a INICIO

         private dtGuia dtguia = new dtGuia(); //Creamos una referencia a la capa de
         negocio en particular a dtguia
        #endregion

        #region Properties
        /// <summary>
        /// Propiedades del Estado
        /// </summary>
        public EstadosRegistroES Estado
        {
            get { return _Estado; }
            set { _Estado = value; }
        }
        #endregion

        #region Methods
        #region LoadTramoEdad
        /// <summary>
        /// Cargar una lista de tramos de edad en el formulario de TramoEdad
        /// </summary>
```

```
public void LoadTramoEdad()
{
    try
    {
        View.Load(dtguia.GetAllTramoEdad());
    }
    catch (Exception e)
    {
        System.Windows.MessageBox.Show("Los datos no han sido cargados
        correctamente : "+ e.Message,
        "Carga",System.Windows.MessageBoxButton.OK);
    }
}
/// <summary>
/// Cargar una lista de tipos de posología
/// </summary>
public void LoadCombo() {
    View.LoadCombo(dtguia.GetAllPosologiaTipo());
    }
#endregion
#endregion

#region Busqueda
/// <summary>
/// Realiza una búsqueda por Código de tramo edad
/// </summary>
public bool BusquedaCodigo()
{
    bool res = false;
    try
    {
        if (View.Codigo == "")  //Si no hay código
        {
            _Estado = EstadosRegistroES.INICIO;
            View.Inicio(); // La pantalla pasa a estado Inicio
            res = false;
        }
        else
        {
            CFG_CAT_FAR_TramoEdad ter =
            dtguia.GetTramoEdadByCod(View.Codigo);
            if (ter != null)
            {
                res = true;
                _Estado = EstadosRegistroES.MODIFICACION;
                List<CFG_CAT_FAR_TramoEdad> lista = new
                List<CFG_CAT_FAR_TramoEdad>();
                lista.Add(ter);
                if (View.Modificable == false)
                    View.NoModificable();
                else View.Modificacion();
                View.Load(lista);
                View.LoadTramoCampos(ter); // Cargamos los datos del elemento
      en los campos
            }
            else { res = false; }
        }
    }
         catch (Exception busqueda) { System.Windows.MessageBox.Show("Error
        en la busqueda del codigo : " + busqueda.Message + " El objeto no
        existe o no se ha encontrado.", "Error en la busqueda",
        System.Windows.MessageBoxButton.OK,
        System.Windows.MessageBoxImage.Information); View.Inicio();}

    return res;
```

```
}
/// <summary>
/// Realiza una búsqueda por descripción de tramo edad
/// </summary>
public bool BusquedaDescripcion()
{
    bool res = false;
    try
    {
        if (View.Descripcion == "")
        {
            _Estado = EstadosRegistroES.INICIO;
            View.Inicio(); // La pantalla pasa a estado Inicio
            res = false;
        }
        else
        {
              List<CFG_CAT_FAR_TramoEdad> lista =
              dtguia.GetTramoEdadByDescripcion(View.Descripcion);

            if (lista.Count != 0 && lista.ElementAt(0) != null)
            {
                _Estado = EstadosRegistroES.MODIFICACION;
                View.Modificacion();
                View.Load(lista);
                res = true;
            }
            else { res = false; }
        }
                }catch (Exception exc) { System.Windows.MessageBox.Show("Error
   en la busqueda por descripción : " +exc.Message); }
    return res;
}
#endregion

#region Aceptar
/// <summary>
/// Acciones a realizar cuando se hace click en el botón Aceptar.
/// </summary>
public void Aceptar()
{
    try
    {
        switch (_Estado)
        {
            case EstadosRegistroES.INICIO:
                {
                    View.Busqueda();
                    if (View.Codigo != "")
                    {
                        if(!this.BusquedaCodigo()) {

                        System.Windows.MessageBox.Show("No se a encontrado el
                        tramo edad con código [" + View.Codigo + "]", "Error
                        en la búsqueda", System.Windows.MessageBoxButton.OK,
                        System.Windows.MessageBoxImage.Exclamation);

                        }
                    }
                    else if (View.Descripcion != "")
                    {
                         if (!this.BusquedaDescripcion()) {
                         System.Windows.MessageBox.Show("No se a encontrado
                         el tramo edad con la descripción [" +
                         View.Descripcion + "]", "Error en la búsqueda",
```

```
                                System.Windows.MessageBoxButton.OK,
                                System.Windows.MessageBoxImage.Exclamation); }
                        }
                    } break;
                case EstadosRegistroES.ALTA:
                    {
                        if (this.Guardar())
                        {
                            View.Inicio();
                            _Estado = EstadosRegistroES.INICIO;
                        }
                    } break;
                case EstadosRegistroES.MODIFICACION:
                    {
                        this.Modificar();

                        if (this.Guardar())
                        {
                            View.Inicio();
                            _Estado = EstadosRegistroES.INICIO;
                        }

                    } break;
            }
        }
catch (Exception aceptar) { System.Windows.MessageBox.Show("Error en el boton
de Aceptar : " + aceptar.Message + " El guardado no ha sido correcto",
"Guardar", System.Windows.MessageBoxButton.OK,
System.Windows.MessageBoxImage.Information); }
}

#endregion

#region Cancelar
/// <summary>
/// Acciónes a realizar cuando se hace clic en el botón Cancelar
/// </summary>
public void Cancelar()
{
    _Estado = EstadosRegistroES.INICIO;
    View.Inicio();
}
#endregion

#region Alta
/// <summary>
/// Acciones a realizar cuando se hace clic en el botón de Alta
/// </summary>
public void Alta()
{
    _Estado = EstadosRegistroES.ALTA;
    View.Alta();
}
#endregion

#region Guardar
/// <summary>
/// Acciones a realizar cuando se hace click en el botón Guardar
/// </summary>
public bool Guardar()
{
    bool res = false;

        if (Validar() == true)
        {
```

```
            CFG_CAT_FAR_TramoEdad t =dtguia.GetTramoEdadByCod(View.Codigo);

                    if (t != null)
                    {
                     t.CFG_CAT_FAR_TramoEdad_CodTramoEdad = View.Codigo;
                     t.CFG_CAT_FAR_TramoEdad_DescripcionTramo = View.Descripcion;
                     t.CFG_CAT_FAR_TramoEdad_TipoPosol = View.TipoSol;
                     t.CFG_CAT_FAR_TramoEdad_Activo = View.Activo;
                     t.CFG_CAT_FAR_TramoEdad_Modificable = View.Modificable;
                     t.CFG_CAT_FAR_TramoEdad_FechaModificacion = DateTime.Now;
                     t.CFG_CAT_FAR_TramoEdad_UsuarioModificacion = Usuario;
                    }
                List<CFG_CAT_FAR_TramoEdad> lista = new t<CFG_CAT_FAR_TramoEdad>();
                    switch (_Estado)
                    {
case EstadosRegistroES.MODIFICACION:
    {
              if (t != null){
                 dtguia.UpdateTramoEdad(t);
                 res = true;
              }else{
                 System.Windows.MessageBox.Show("No se han podido realizar las
                 modificaciones del tramo de edad [" + View.Descripcion + "] .Vuelva
                 a intentarlo porfavor.", "Modificación",
                 System.Windows.MessageBoxButton.OK,
                 System.Windows.MessageBoxImage.Information);
                   }
  } break;
  case EstadosRegistroES.ALTA:
                            {
  if (t == null) // si no se ha encontrado algun elemento con el mismo código
                    {
        try{
                    t = new CFG_CAT_FAR_TramoEdad();
                    t.CFG_CAT_FAR_TramoEdad_CodTramoEdad = View.Codigo;
                    t.CFG_CAT_FAR_TramoEdad_DescripcionTramo = View.Descripcion;
                    t.CFG_CAT_FAR_TramoEdad_TipoPosol = View.TipoSol;
                    t.CFG_CAT_FAR_TramoEdad_Activo = View.Activo;
                    t.CFG_CAT_FAR_TramoEdad_Modificable = View.Modificable;
                    t.CFG_CAT_FAR_TramoEdad_FechaModificacion = DateTime.Now;
                    t.CFG_CAT_FAR_TramoEdad_UsuarioModificacion = Usuario;
                    t.CFG_CAT_FAR_TramoEdad_UsuarioAlta = Usuario;
                    t.CFG_CAT_FAR_TramoEdad_FechaAlta = DateTime.Now;
                    dtguia.InsertTramoEdad(t);

                  System.Windows.MessageBox.Show("El tramo de edad [" + View.Descripcion
                  + "] ha sido dado de alta correctamente", "Alta registro",
                  System.Windows.MessageBoxButton.OK,
                  System.Windows.MessageBoxImage.Information);
                  res = true;
            } catch (Exception rt){
                      System.Windows.MessageBox.Show("No se ha podido registrar en
                      BD." + rt.Message, "Error de escritura",
                      System.Windows.MessageBoxButton.OK,
                      System.Windows.MessageBoxImage.Information);
                       return res;
                                  }
                  }

else { System.Windows.MessageBox.Show("No se ha podido realizar el alta del tramo de
edad [" + View.Descripcion + "]. Ya existe.", "Elemento duplicado",
System.Windows.MessageBoxButton.OK, System.Windows.MessageBoxImage.Information); }

       } break;
                    }
```

```
        }
        return res;
}
#endregion

#region Validar
/// <summary>
/// Validación de los datos existentes en el formulario
/// </summary>
/// <returns>Devuelve si son validos o no</returns>
private bool Validar()
{

    if (View.Codigo == "" || View.Codigo.Trim() == "")
    {
       System.Windows.MessageBox.Show("El campo Código no puede estar
       vacío", "Revise campos", System.Windows.MessageBoxButton.OK,
       System.Windows.MessageBoxImage.Information);
                    return false;
    }

     if (View.Descripcion == "" || View.Descripcion.Trim() == "")
    {
       System.Windows.MessageBox.Show("El campo Descripción no puede estar
       vacío", "Revise campos", System.Windows.MessageBoxButton.OK,
       System.Windows.MessageBoxImage.Information);
                    return false;
    }

     if (View.TipoSol == "" || View.TipoSol.Trim() == "")
    {
       System.Windows.MessageBox.Show("El campo tipo posología no puede estar
       vacío", "Revise campos", System.Windows.MessageBoxButton.OK,
       System.Windows.MessageBoxImage.Information);
        return false;
    }
    return true;
}
#endregion

#region Modificar
/// <summary>
/// Acciones a realizar cuando se hace una modificación
/// </summary>
public void Modificar()
{

   List<CFG_CAT_FAR_TramoEdad> lista = new List<CFG_CAT_FAR_TramoEdad>();
    lista.Add(dtguia.GetTramoEdadByCod(View.Codigo));
    _Estado = EstadosRegistroES.MODIFICACION;
    if (View.Modificable == false)
        View.NoModificable();
    else View.Modificacion();
}
#endregion

#region Iniciarestado
public void IniciarEstado()
{
    _Estado = EstadosRegistroES.INICIO;
}
#endregion

#region Comprobar si Codigo Existe
```

```
        public bool ComprobarCodigoExiste(string codig)
        {

            if (dtguia.GetTramoEdadByCod(codig) != null) return true;
            else return false;
        }
        #endregion

        internal void GridSeleccionado()
        {
            CFG_CAT_FAR_TramoEdad edad = dtguia.GetTramoEdadByCod(View.Codigo);
            View.LoadTramoCampos(edad);
            _Estado = EstadosRegistroES.MODIFICACION;
            if (View.Modificable == false)
                View.NoModificable();
            else View.Modificacion();
        }
    }
}
```

Código C# 2: PRESENTE TRAMO DE EDAD

12.7 Archivos de la vista de la pantalla de tramo de edad

La vista cómo ya se ha mencionado anteriormente se compone en 2 partes, la parte visual y la parte de gestión de la vista. La parte visual se codifica con código XAML el cual representa todos los objetos necesarios para visualizar correctamente la vista. A continuación un ejemplo (**archivo .xaml**) de la parte visual de la pantalla de tramo edad:

```
<Page x:Class="CHGUV.Guia.Farmacoterapeutica.Pages.Maestros.TramoEdad.TramoEdadPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:my1="clr-namespace:WpfCCLibrary;assembly=WpfCCLibrary"
  Title="Tramo de edad" xmlns:my="clr-
namespace:WpfCCLibrary.Control;assembly=WpfCCLibrary"
xmlns:mic="http://schemas.microsoft.com/wpf/2008/toolkit" mc:Ignorable="d"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
d:DesignHeight="720" d:DesignWidth="1050" KeyDown="Page_PreviewKeyDown"
Loaded="Page_Loaded">
    <Page.Resources>
        <DataTemplate x:Key="ItemTemplate">
            <StackPanel>
 <TextBlock Text="{Binding CFG_CAT_FAR_PosologiaTipo_Tipo}"/>
                <TextBlock Text="{Binding CFG_CAT_FAR_PosologiaTipo_Descripcion}"/>
            </StackPanel>
        </DataTemplate>
    </Page.Resources>
    <Grid>
<Grid.Background>
            <LinearGradientBrush EndPoint="0.5,1.5" StartPoint="0,0">
                <GradientStop Color="#FF87BDEB" Offset="0" />
                <GradientStop Color="White" Offset="1" />
            </LinearGradientBrush>
        </Grid.Background>
        <my:WpfLabel Content="Código" HorizontalAlignment="Left" Margin="39,0,0,107"
Name="wpfLabel1" Height="21" VerticalAlignment="Bottom" />
```

```
            <my:WpfTextBox Height="23" HorizontalAlignment="Left"
Margin="121,0,0,107" Name="TextBoxCodigo" VerticalAlignment="Bottom" Width="134"
MaxLength="10" KeyDown="TextBoxCodigo_PreviewKeyDown" />
            <my:WpfLabel Content="Descripción" Height="22" HorizontalAlignment="Left"
Margin="38,0,0,71" Name="wpfLabel2" VerticalAlignment="Bottom" Width="64" />
            <my:WpfTextBox Height="23" HorizontalAlignment="Left" Margin="121,0,0,71"
Name="TextBoxDescripcion" VerticalAlignment="Bottom" Width="581" MaxLength="150"
KeyDown="TextBoxDescripcion_PreviewKeyDown" />
            <my:WpfCheckBox Content="Modificable" Height="16"
HorizontalAlignment="Left" Margin="324,0,0,110" Name="CheckBoxModificable"
VerticalAlignment="Bottom" Estilo="StlWpfCheckBoxAzul" />
        <my:WpfCheckBox Content="Activo" Height="16" HorizontalAlignment="Left"
Margin="428,0,0,110" Name="CheckBoxActivo" VerticalAlignment="Bottom"
Estilo="StlWpfCheckBoxAzul" />
        <my:WpfImage Height="24" HorizontalAlignment="Left" Margin="261,0,0,106"
Name="ImageCodigoNoNume"  Stretch="Fill" VerticalAlignment="Bottom" Width="24"
Source="/CHGUV.Guia.Farmacoterapeutica.Pages;component/Images/Cancel_peq.png"
Visibility="Collapsed" >
                <my:WpfImage.ToolTip>
                    <ToolTip>
                        El código debe ser numérico
                    </ToolTip>
                </my:WpfImage.ToolTip>
            </my:WpfImage>
            <Label Content="Tipo posología" Height="25" HorizontalAlignment="Left"
Margin="33,0,0,36" Name="label1" VerticalAlignment="Bottom" />
        <my:WpfComboBox Height="23" HorizontalAlignment="Left" Margin="121,0,0,36"
Name="comboBoxPosologia" VerticalAlignment="Bottom" Width="131" ItemsSource="{Binding
CFG_CAT_FAR_PosologiaTipo}" Estilo="StlWpfComboBoxBlackAzul" />
        <my:WpfStatusBar  Visibility="Visible" Height="25" VerticalAlignment="Bottom"
BorderThickness="0.7"  BorderBrush="#FF688CAF">
            <my:WpfStatusBar.Background>
                <LinearGradientBrush EndPoint="0.0,5.5" StartPoint="0,0">
                    <GradientStop Color="#4682B4" Offset="0" />
                    <GradientStop Color="White" Offset="1" />
                </LinearGradientBrush>
            </my:WpfStatusBar.Background>
            <StatusBarItem Name="statusBarItem" HorizontalAlignment="Left"
Foreground="White" />
            <StatusBarItem Name="statusBarNoModificable" HorizontalAlignment="Center"
Foreground="White" FontWeight="Bold" />
            <StatusBarItem Name="statusBarItemEstado" HorizontalAlignment="Right"
Foreground="White" />
        </my:WpfStatusBar>
        <my:WpfDataGrid AutoGenerateColumns="False"
Estilo="StlWpfDataGridAlternatingRows" Estilo_Header="WpfDataGridColumnHeaderAzul"
Margin="12,12,12,147" Name="dataGridTramoEdad"
SelectionChanged="dataGridTramoEdad_SelectionChanged" IsReadOnly="True">
            <my:WpfDataGrid.Columns>
                <mic:DataGridTextColumn Binding="{Binding
CFG_CAT_FAR_TramoEdad_CodTramoEdad}" Header="Código" SortDirection="Ascending"
SortMemberPath="CFG_CAT_FAR_TramoEdad" MaxWidth="80">
                    <mic:DataGridTextColumn.ElementStyle>
                        <Style TargetType="TextBlock">
                            <Setter Property="TextAlignment" Value="Right" />
                        </Style>
                    </mic:DataGridTextColumn.ElementStyle>
                </mic:DataGridTextColumn>
                <mic:DataGridTextColumn Binding="{Binding
CFG_CAT_FAR_TramoEdad_DescripcionTramo}" Header="Descripción"
SortDirection="Ascending" SortMemberPath="CFG_CAT_FAR_TramoEdad" Width="960" >
                    <mic:DataGridTextColumn.ElementStyle>
                        <Style TargetType="TextBlock">
                            <Setter Property="TextWrapping" Value="Wrap" />
                        </Style>
```

```
                    </mic:DataGridTextColumn.ElementStyle>
                </mic:DataGridTextColumn>
                <mic:DataGridCheckBoxColumn Binding="{Binding
CFG_CAT_FAR_TramoEdad_Modificable}" Header="Modificable"
SortMemberPath="CFG_CAT_FAR_TramoEdad" MaxWidth="100" />
                <mic:DataGridCheckBoxColumn Binding="{Binding
CFG_CAT_FAR_TramoEdad_Activo}" Header="Activo" SortMemberPath="CFG_CAT_FAR_TramoEdad"
MaxWidth="100" />
            </my:WpfDataGrid.Columns>
        </my:WpfDataGrid>
    </Grid>
</Page>
```

Código C# 3: ARCHIVO .XAML DE LA VISTA DE TRAMO DE EDAD

En la parte de gestión de la información de la vista, se compone por un **archivo .xaml.cs** el cual contiene todos los métodos necesarios para la gestión y carga de información en la vista. Con esta estructura se logra una gran independencia entre las diferentes capas de la aplicación. A continuación un ejemplo de archivo **.xaml.cs** de la pantalla tramo de edad:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using WpfCCLibrary.Shared.Presenter;
using CHGUV.Guia.Farmacoterapeutica.Business.HGUVHis;
using CHGUV.WPFControls.Controls;


namespace CHGUV.Guia.Farmacoterapeutica.Pages.Maestros.TramoEdad
{
    /// <summary>
    /// Lógica de interacción para TramoEdadPage.xaml
    /// </summary>
    public partial class TramoEdadPage : Page, TramoEdadInterface, IMVPPage
    {
        #region Atributes

        #region Presente
        private TramoEdadPresenter _presenter = new TramoEdadPresenter();
        /// <summary>
        /// Propiedades necesarias del Presenter
        /// </summary>
        public TramoEdadPresenter Presenter
        {
            get { return _presenter; }
            set { _presenter = value; }
        }
        #endregion

        #region Usuario
```

```
        private string _Usuario = "";
        /// <summary>
        /// Propiedades del usuario
        /// </summary>
        public string Usuario
        {
            get { return _Usuario; }
        }
        #endregion
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor por defecto de la clase TramoEdadPage
        /// </summary>
        public TramoEdadPage()
        {
            InitializeComponent();
            _presenter.MenuName = "TabAccionesGuiaFarmacoterapeutica";
            _presenter.View = this;
            Inicio();    //Iniciamos la página con el estado de inicio
        }
        /// <summary>
         /// Constructor para cuando se inicia sesión con usuario en la página
         TramoEdadPage
        /// </summary>
        /// <param name="oUsuario">El nombre del usuario que ha iniciado
       sesión</param>
        public TramoEdadPage(string oUsuario)
        {
            _Usuario = oUsuario;
            InitializeComponent();
            _presenter.View = this;
            Inicio();
        }
        #endregion

        #region Events
        #region Cerrar
        /// <summary>
        /// Evento del botón Cerrar
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnCerrar_Click(object sender, RoutedEventArgs e)
        {
            this.Cursor = Cursors.Wait;
            try
            {
                Presenter.CloseView();
            }
            catch
            {
            }
            finally
            {
                this.Cursor = Cursors.Arrow;
            }
        }
        #endregion

        #region Aceptar
        /// <summary>
        /// Evento del botón Aceptar
        /// </summary>
```

```
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAceptar_Click(object sender, RoutedEventArgs e)
{
    this.Cursor = Cursors.Wait;
    try
    {
        _presenter.Aceptar();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error en el botón Aceptar : " + ex.Message);
    }
    finally
    {
        this.Cursor = Cursors.Arrow;
    }
}

#endregion

#region Cancelar
/// <summary>
/// Evento del botón cancelar
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnCancelar_Click(object sender, RoutedEventArgs e)
{
    this.Cursor = Cursors.Wait;
    try
    {
        _presenter.Cancelar();
    }
    catch
    {
    }
    finally
    {
        this.Cursor = Cursors.Arrow;
    }
}
#endregion

#region Alta
/// <summary>
/// Evento del botón Alta
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAlta_Click(object sender, RoutedEventArgs e)
{
    this.Cursor = Cursors.Wait;
    try
    {
        _presenter.Alta();
    }
    catch
    {
    }
    finally
    {
        this.Cursor = Cursors.Arrow;
    }
}
```

```
#endregion

#region Tecla ESC
private void Page_PreviewKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        _presenter.Cancelar();
    }
}
#endregion

#region Tecla ENTER
private void TextBoxCodigo_PreviewKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Enter)
    {
        _presenter.Aceptar();
    }
}
private void TextBoxDescripcion_PreviewKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Enter)
    {
        _presenter.BusquedaDescripcion();
    }
}
#endregion
#endregion

#region Methods
#region Proteger
/// <summary>
/// Protege todos los elementos de la pantalla de visualizacion de Tramo Edad
/// </summary>
private void Proteger()
{
    this.TextBoxCodigo.IsReadOnly = true;
    this.TextBoxDescripcion.IsReadOnly = true;
    this.comboBoxPosologia.IsEnabled = false;
    this.CheckBoxActivo.IsEnabled = false;
    this.CheckBoxModificable.IsEnabled = false;

    dataGridTramoEdad.IsReadOnly = true;

    ////Botones acciones
    _presenter.BotonIsEnabled("Aceptar", false);
    _presenter.BotonIsEnabled("Nuevo", false);
    _presenter.BotonIsEnabled("Cancelar", false);
    _presenter.BotonIsEnabled("Cerrar", false);
}
#endregion

#region Limpiar
/// <summary>
/// Limpia la pantalla de visualización de Tramo Edad
/// </summary>
private void Limpiar()
{
    this.TextBoxCodigo.Text = "";
    this.TextBoxDescripcion.Text = "";

    this.CheckBoxActivo.IsChecked = false;
    this.CheckBoxModificable.IsChecked = false;
    this.comboBoxPosologia.SelectedItem = null;
```

```
            try
            {
                dataGridTramoEdad.ItemsSource = null;
                dataGridTramoEdad.Items.Clear();
            }
            catch { }
            this.statusBarItem.Content = "";
            this.statusBarItemEstado.Content = "";
            this.statusBarNoModificable.Content = "";
        }
        #endregion
        #endregion

        #region Estados
        #region Busqueda
        /// <summary>
        /// Activa los campos necesarios para cuando se realiza una busqueda
        /// </summary>
        public void Busqueda()
        {
            statusBarItemEstado.Content = "BÚSQUEDA";
            this.Proteger(); //Protegemos primero todos los campos
            this.TextBoxDescripcion.Focus();
            this.TextBoxCodigo.IsReadOnly = false;
            this.TextBoxDescripcion.IsReadOnly = false;

            ////Botones acciones
            _presenter.BotonIsEnabled("Aceptar", true);
            _presenter.BotonIsEnabled("Nuevo", true);
            _presenter.BotonIsEnabled("Cancelar", true);
            _presenter.BotonIsEnabled("Cerrar", true);
            //_presenter.BotonIsEnabled("Guardar", true);
        }
        #endregion

        #region Modificacion
        /// <summary>
        /// Activa todos los campos necesarios para la modificación de un registro
        /// </summary>
        public void Modificacion()
        {
            this.Proteger(); // Protegemos todos los campos

            statusBarItemEstado.Content = "MODIFICACIÓN";
            TextBoxDescripcion.Focus();
            this.TextBoxDescripcion.IsReadOnly = false;
            this.comboBoxPosologia.IsEnabled = true;
            this.CheckBoxActivo.IsEnabled = true;
            this.CheckBoxModificable.IsEnabled = true;

            ////Botones acciones
            _presenter.BotonIsEnabled("Aceptar", true);
            _presenter.BotonIsEnabled("Nuevo", true);
            _presenter.BotonIsEnabled("Cancelar", true);
            _presenter.BotonIsEnabled("Cerrar", true);
        }
        #endregion

        #region Inicio
        /// <summary>
        /// Activa los campos necesarios para cuando la aplicación se pone en el
estado Inicio
        /// </summary>
        public void Inicio()
        {
```

```
            this.Proteger(); //Protegemos todos los campos
            this.Limpiar(); //Limpiamos todo el contenido que pueda haber
            this.Busqueda(); // Iniciamos pantalla en estado de busqueda y
seleccion de tramo
            _presenter.LoadTramoEdad();
            _presenter.LoadCombo();
            _presenter.IniciarEstado();
            statusBarItemEstado.Content = "BÚSQUEDA";

        }
        #endregion

        #region Alta
        /// <summary>
        /// Activa los campos necesarios para la alta de un nuevo registro
        /// </summary>
        public void Alta()
        {
            this.Proteger(); //Protegemos todos los campos
            this.Limpiar(); //Limpiamos todo el contenido que pueda haber
            statusBarItemEstado.Content = "Alta";
            this.TextBoxCodigo.IsReadOnly = false;
            this.TextBoxDescripcion.IsReadOnly = false;
            this.comboBoxPosologia.IsEnabled = true;
            this.CheckBoxActivo.IsEnabled = true;
            this.CheckBoxModificable.IsEnabled = true;
            this.CheckBoxActivo.IsChecked = true;
            this.CheckBoxModificable.IsChecked = true;
            this.TextBoxCodigo.Focus();

            ////Botones acciones
            _presenter.BotonIsEnabled("Aceptar", true);
            _presenter.BotonIsEnabled("Cancelar", true);
            _presenter.BotonIsEnabled("Cerrar", true);
        }
        #endregion
        #region NoModificable
        public void NoModificable()
        {
            this.Proteger();
            // this.Limpiar();
            statusBarItemEstado.Content = "MODIFICACIÓN";
            statusBarNoModificable.Content = "NO MODIFICABLE";
            dataGridTramoEdad.IsEnabled = true;
            CheckBoxModificable.IsEnabled = true;
            comboBoxPosologia.IsReadOnly = true;

            ////Botones acciones
            _presenter.BotonIsEnabled("Aceptar", true);
            _presenter.BotonIsEnabled("Nuevo", true);
            _presenter.BotonIsEnabled("Cancelar", true);
            _presenter.BotonIsEnabled("Cerrar", true);
            this.CheckBoxModificable.Focus();

        }
        #endregion
        #region Cerrar
        /// <summary>
        /// Cierra el formulario
        /// </summary>
        public void Cerrar()
        {
            Presenter.CloseView();
        }
```

```
        #endregion
        #endregion

        #region Properties
        /// <summary>
        /// Definimos la propiedad get de Código para obtener su valor
        /// </summary>
        public string Codigo { get { return this.TextBoxCodigo.Text; } }
        /// <summary>
        /// Definimos la propiedad get de la Descripción para obtener su valor
        /// </summary>
       public string Descripcion { get { return this.TextBoxDescripcion.Text; } }

        /// <summary>
        /// Definimos la propiedad get de Activo para obtener su valor
        /// </summary>
       public bool Activo { get { return this.CheckBoxActivo.IsChecked.Value; } }
        /// <summary>
        /// Definimos la propiedad get de Modificable para obtener su valor
        /// </summary>
         public bool Modificable { get { return
         this.CheckBoxModificable.IsChecked.Value; } }
        /// <summary>
        /// Definimos la propiedad get de TipoSol
        /// </summary>
        public string TipoSol { get {
            if(comboBoxPosologia.SelectedValue == null){return "";}
            return comboBoxPosologia.SelectedValue.ToString(); } }
        #endregion

        #region  Load
        /// <summary>
        /// Se cargan todos los datos disponibles de TramoEdad en el datagrid
        /// </summary>
        /// <param name="lista">Lista con todos los TramosEdad existentes</param>
        public void Load(List<CFG_CAT_FAR_TramoEdad> lista)
        {
            try{


                List<CFG_CAT_FAR_TramoEdad> nueva = new List<CFG_CAT_FAR_TramoEdad>();
                foreach (CFG_CAT_FAR_TramoEdad t in lista)
                {
                      t.CFG_CAT_FAR_TramoEdad_CodTramoEdad=
                      t.CFG_CAT_FAR_TramoEdad_CodTramoEdad.Trim();
                     nueva.Add(t);
                }

                this.Limpiar();
                this.Cursor = System.Windows.Input.Cursors.Wait;
                if (nueva.Count == 1)
                {
                    dataGridTramoEdad.ItemsSource = nueva;
                    dataGridTramoEdad.SelectedIndex = 0;
                }
                else {
                    dataGridTramoEdad.ItemsSource = nueva;
                }

            }
            catch (Exception el) { MessageBox.Show("Error en el Load : " +
el.Message); }
            finally { this.Cursor = System.Windows.Input.Cursors.Arrow; }
            this.Cursor = System.Windows.Input.Cursors.Arrow;
```

```
        }
        public void LoadCombo(List<CFG_CAT_FAR_PosologiaTipo> lista)
        {
            try
            {
                  comboBoxPosologia.DisplayMemberPath =
               "CFG_CAT_FAR_PosologiaTipo_Descripcion";
                  comboBoxPosologia.SelectedValuePath =
               "CFG_CAT_FAR_PosologiaTipo_Codigo";
                  comboBoxPosologia.ItemsSource = lista;
            }
            catch (Exception e) { MessageBox.Show("Error cargando listado combo : " +
e.Message); }

        }
        /// <summary>
        /// Carga un tramo de edad en los campos correspondientes
        /// </summary>
        /// <param name="tramo"></param>
        public void LoadTramoCampos(CFG_CAT_FAR_TramoEdad tramo)
        {

            TextBoxCodigo.Text = tramo.CFG_CAT_FAR_TramoEdad_CodTramoEdad;
            TextBoxDescripcion.Text = tramo.CFG_CAT_FAR_TramoEdad_DescripcionTramo;
            this.comboBoxPosologia.SelectedValue =
            tramo.CFG_CAT_FAR_TramoEdad_TipoPosol;

            CheckBoxActivo.IsChecked = tramo.CFG_CAT_FAR_TramoEdad_Activo;
            CheckBoxModificable.IsChecked =
            tramo.CFG_CAT_FAR_TramoEdad_Modificable;

            string status = "Usuario alta: " +
            tramo.CFG_CAT_FAR_TramoEdad_UsuarioAlta + "\tFecha alta: " +
            tramo.CFG_CAT_FAR_TramoEdad_FechaAlta.ToString() + "\tUsuario
            modificación: " + tramo.CFG_CAT_FAR_TramoEdad_UsuarioModificacion +
            "\tFecha modificación: " +
            tramo.CFG_CAT_FAR_TramoEdad_FechaModificacion.ToString() + "\t";
            statusBarItem.Content = status;
        }
        #endregion
        #region Selected Item
        /// <summary>
        /// Se coge el elemento seleccionado y se obtienen sus datos
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dataGridTramoEdad_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            //this.Cursor = Cursors.Wait;
            this.Cursor = System.Windows.Input.Cursors.Wait;
            try
            {
                if (dataGridTramoEdad.SelectedItem != null)
                {
                      TextBoxCodigo.Text =
                      ((CFG_CAT_FAR_TramoEdad)dataGridTramoEdad.SelectedItem).CFG_CAT
                      _FAR_TramoEdad_CodTramoEdad.ToString();

                    _presenter.GridSeleccionado();
                }
                else { return; }

            }
```

```
        catch (Exception ex)
        { MessageBox.Show("Error en el Selection Changed: " + ex.Message); }


        this.Cursor = System.Windows.Input.Cursors.Arrow;
    }
    #endregion

    #region Miembros de IMVPPage

    public string MenuName
    {
        get
        {
            return _presenter.MenuName;
        }
        set
        {
            _presenter.MenuName = value;
        }
    }

    public Microsoft.Windows.Controls.Ribbon.RibbonTab MenuToolBar
    {
        get
        {
            return _presenter.MenuToolBar;
        }
        set
        {
            _presenter.MenuToolBar = value;
            if (value != null)
                Inicio();
        }
    }
    public void ButtonOptionPreset(string Name)
    {
        switch (Name)
        {
            case ACCIONES_VBase.CERRAR:
                btnCerrar_Click(null, null);
                break;
            case ACCIONES_VBase.ACEPTAR:
                btnAceptar_Click(null, null);
                break;
            case ACCIONES_VBase.CANCELAR:
                btnCancelar_Click(null, null);
                break;
            case ACCIONES_VBase.NUEVO:
                btnAlta_Click(null, null);
                break;
        }
    }

    #endregion
    /// <summary>
    /// Definición de variables constantes para el menú
    /// </summary>
    public static class ACCIONES_VBase
    {
        public const string NUEVO = "Nuevo";
        public const string ACEPTAR = "Aceptar";
        public const string CANCELAR = "Cancelar";
        public const string CERRAR = "Cerrar";
```

```
        }
    }
}
```

Código C# 4: ARCHIVO .XAML.CS DE LA VISTA DE TRAMO DE EDAD

12.8 Resumen capa business

El siguiente código representa la clase **dtGuia.cs**, que es la que contiene todos los métodos necesarios para la manipulación de la base de datos de la guía (recordar que toda la gestión de la base de datos se ha realizado con la tecnología LINQ). Como es muy extensa, se muestra un resumen de esta ya que todas las tablas tienen métodos similares (el ejemplo ilustrado hace referencia a los métodos, para realizar operaciones con la tabla, de Unidad de Medida).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Linq.SqlClient;


namespace CHGUV.Guia.Farmacoterapeutica.Business.HGUVHis
{
    public class dtGuia
    {


        #region Unidad De Medida

            #region GetUnidadMedidaByCodUnidad
        /// <summary>
        /// Obtiene las Unidades de Medida a partir de un código
        /// </summary>
        /// <param name="cod"> Código de la unidad de medida</param>
        /// <returns></returns>
        public CFG_CAT_FAR_UnidadMedida GetUnidadMedidaByCodUnidad(string cod)
        {
            lnqHGUVHisDataContext db = new lnqHGUVHisDataContext();

           CFG_CAT_FAR_UnidadMedida obj =
           db.CFG_CAT_FAR_UnidadMedida.FirstOrDefault(c =>
           c.CFG_CAT_FAR_UnidadMedida_Codigo == cod);

            return obj;

        }
        #endregion

            #region InsertUnidadDeMedida
        /// <summary>
        /// Inserción de una Unidad De Medida
        /// </summary>
        /// <param name="um"> Objeto Unidad de Medida</param>
        public void insertUnidadMedida(CFG_CAT_FAR_UnidadMedida um)
        {
            //conexion con bd
```

```
            lnqHGUVHisDataContext db = new lnqHGUVHisDataContext();

            //insertamos la fila
            db.CFG_CAT_FAR_UnidadMedida.InsertOnSubmit(um);
            //aplicamos cambio
            db.SubmitChanges();


        }
        #endregion

            #region UpdateUnidadMedida
        /// <summary>
        /// Actualiza una cuenta
        /// </summary>
        /// <param name="um"> Objeto de unidad de medida</param>
        public void UpdateUnidadMedida(CFG_CAT_FAR_UnidadMedida um)
        {
            lnqHGUVHisDataContext db = new lnqHGUVHisDataContext();


             CFG_CAT_FAR_UnidadMedida oUM =
             db.CFG_CAT_FAR_UnidadMedida.FirstOrDefault(c =>
             c.CFG_CAT_FAR_UnidadMedida_Codigo == um.CFG_CAT_FAR_UnidadMedida_Codigo);
                         oUM.CFG_CAT_FAR_UnidadMedida_Descripcion =
             um.CFG_CAT_FAR_UnidadMedida_Descripcion;
                         oUM.CFG_CAT_FAR_UnidadMedida_FechaAlta =
             um.CFG_CAT_FAR_UnidadMedida_FechaAlta;
                         oUM.CFG_CAT_FAR_UnidadMedida_FechaModificacion =
             um.CFG_CAT_FAR_UnidadMedida_FechaModificacion;
                         oUM.CFG_CAT_FAR_UnidadMedida_Modificable =
             um.CFG_CAT_FAR_UnidadMedida_Modificable;
                         oUM.CFG_CAT_FAR_UnidadMedida_UsuarioAlta =
             um.CFG_CAT_FAR_UnidadMedida_UsuarioAlta;
                         oUM.CFG_CAT_FAR_UnidadMedida_UsuarioModificacion =
             um.CFG_CAT_FAR_UnidadMedida_UsuarioModificacion;
                         oUM.CFG_CAT_FAR_UnidadMedida_Activo =
             um.CFG_CAT_FAR_UnidadMedida_Activo;

            db.SubmitChanges();

        }
        #endregion

            #region GetAllUnidadMedida
        /// <summary>
        /// Obtener un listado de UnidadMedida
        /// </summary>
        /// <returns>Listado de UnidadMedida</returns>
        public List<CFG_CAT_FAR_UnidadMedida> GetAllUnidadMedida()
        {

            List<CFG_CAT_FAR_UnidadMedida> lista = new
List<CFG_CAT_FAR_UnidadMedida>();

            lnqHGUVHisDataContext db = new lnqHGUVHisDataContext();

            lista = db.CFG_CAT_FAR_UnidadMedida.ToList();

            return lista;
        }

        #endregion

            #region GetUnidadMedidaByDescripcion
```

```
        public List<CFG_CAT_FAR_UnidadMedida> GetUnidadMedidaByDescripcion(string
descr)
        {
            lnqHGUVHisDataContext bd = new lnqHGUVHisDataContext();
            descr = "%" + descr + "%";
            List<CFG_CAT_FAR_UnidadMedida> lista =
bd.CFG_CAT_FAR_UnidadMedida.Where(c =>
System.Data.Linq.SqlClient.SqlMethods.Like(c.CFG_CAT_FAR_UnidadMedida_Descripcion,
descr)).ToList();
            return lista;
        }
        #endregion

            #region ExisteUnidadMedida
        public bool ExisteUnidadMedida(string cod)
        {
            lnqHGUVHisDataContext bd = new lnqHGUVHisDataContext();
            CFG_CAT_FAR_UnidadMedida lista =
bd.CFG_CAT_FAR_UnidadMedida.FirstOrDefault(c => c.CFG_CAT_FAR_UnidadMedida_Codigo ==
cod);
            if (lista == null)
                return false;
            else return true;

        }

        #endregion

            #region GetAllUnidadMedidaActivos
        public List<CFG_CAT_FAR_UnidadMedida> GetAllUnidadMedida_Activos()
        {

            List<CFG_CAT_FAR_UnidadMedida> lista = new
List<CFG_CAT_FAR_UnidadMedida>();

            lnqHGUVHisDataContext db = new lnqHGUVHisDataContext();

            lista = db.CFG_CAT_FAR_UnidadMedida.Where(c =>
c.CFG_CAT_FAR_UnidadMedida_Activo == true).ToList();

            return lista;
        }

        #endregion
}
}
```

Código C# 5: RESUMEN CAPA BUSINESS DE LA GUÍA

Por otra parte, la capa de persistencia también contiene el archivo **lnqHGUVHis.dbml** . Este archivo no lo detallamos ya que es la representación de las tablas de la base de datos de forma gráfica (cómo si fuera el diagrama de base de datos anteriormente ilustrado). Su funcionalidad es crear de forma automática los objetos y sus atributos referentes a cada tabla de la base de datos con la tecnología LINQ para facilitar el proceso de gestión de información de esta.

13 BIBLIOGRAFIA

Garcia, J. (27 de 9 de 2007). ***Ingenieros Software***. Recuperado el 15 de 11 de 2011, de http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php

IEE STD 830 . (s.f.). Recuperado el 08 de Agosto de 2011, de IEEE Standard Board : IEE Guide to Software Requirements Specifications: http://www.standards.ieee.org/

***Ingenieria del software*.** (s.f.). Recuperado el 08 de Agosto de 2011, de Guía del IEE para la especificación de requerimientos software.

Introducción a las consultas LINQ. (s.f.). Recuperado el 02 de Agosto de 2011, de Microsoft: Introducción a las consultas LINQ (C#): http://msdn.microsoft.com/es-es/library/bb397906.aspx

Introducción a WPF. (s.f.). Recuperado el 03 de Agosto de 2011, de Microsoft: Introducción a WPF: http://msdn.microsoft.com/es-es/library/aa970268.aspx#Y2280

LINQ : Language-Integrated Query. (s.f.). Recuperado el 02 de Agosto de 2011, de Microsoft: LINQ (Language-Integrated Query): http://msdn.microsoft.com/es-es/library/bb397926.aspx

Microsoft SQL SERVER 2005. (s.f.). Recuperado el 21 de 10 de 2011, de http://www.microsoft.com/latam/sql/2005/productinfo/

Model View Presenter (MVP) Bundle. (17 de Junio de 2008). Recuperado el 7 de Mayo de 2011, de Microsoft Patterns & Practices Web Client Developer Guidance : Model View Presenter (MVP) Bundle: http://webclientguidance.codeplex.com/wikipage?title=MVP_landing_page

Model-View-Presenter Pattern. (17 de Diciembre de 2007). Recuperado el 7 de Mayo de 2011, de Microsoft Patterns & Practices Web Client Developer Guidance : Model-View-Presenter Pattern: http://webclientguidance.codeplex.com/wikipage?title=ModelViewPresenterPatternDescription&referringTitle=MVPDocumentation

Pressman, R. S. *Ingeniería del Software*. *Un enfoque práctico.*

Sparx System. (2007). Recuperado el 15 de 11 de 2011, de http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html

Wikipedia. (s.f.). Recuperado el 21 de 10 de 2011, de http://es.wikipedia.org/wiki/Microsoft_SQL_Server

Wikipedia. (29 de Julio de 2011). Recuperado el 02 de Agosto de 2011, de Wikipedia - Desarrollo guiado por pruebas: http://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas

Wikipedia-UML. (10 de 11 de 2011). Recuperado el 15 de 11 de 2011, de http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

14 ILUSTRACIONES

15 TABLAS

16 CÓDIGO C#