UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMATICA

PROYECTO DE GRADO

"SISTEMA DE CONTROL Y GESTION DE HISTORIALES CLINICOS APOYADO EN DISPOSITIVOS MOVILES" CASO: CENTRO MEDICO LA PAZ

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERIA DE SISTEMAS INFORMÁTICOS

POSTULANTE: MELISSA CENTELLAS COARITE
TUTOR METODOLOGICO: LIC. JAVIER REYES PACHECO
ASESOR: LIC. JHONNY FELIPEZ ANDRADE

LA PAZ – BOLIVIA

2015

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

RESUMEN

El presente trabajo nos muestra el proceso para el desarrollo del Sistema de Control y Gestión de Historiales Clínicos Apoyado en Dispositivos Móviles para el Centro Medico La Paz. El proceso de registro de historiales clínicos es manual, razón por la cual se necesita una herramienta que automatice estas tareas debido al crecimiento de pacientes que consultan al Centro Medico. Por otra parte la mayoría de los pacientes por diversos motivos no realizan un seguimiento a tiempo de su tratamiento médico, razón por la cual se les brinda una aplicación móvil que permita recordarle al paciente su tratamiento médico.

ABSTRACT

The present work shows us the process for the development of the System of Control and Supported Management of Clinical Records in Mobile Devices for the La Paz Medical Center. The process of registration of clinical records is manual, reason for which a tool is needed which automates these tasks due to the growth of patients that you consult to the Center Prescribes. On the other hand most of the patients for diverse reasons don't carry out a pursuit on time of their medical treatment, reason for which you are offered a mobile application that allows to remind the patient their medical treatment.

AGRADECIMIENTOS

Agradezco a mi casa de estudios Universidad Mayor de San Andrés, a mis docentes de carrera, por brindarme sus conocimientos y guiarme en este camino profesional.

A mi docente tutor Lic. Javier Reyes Pacheco por la colaboración y paciencia que me brindo en la revisión de mi Proyecto.

A mi docente asesor Lic. Jhonny Felipez Andrade por el apoyo incondicional en este desafío.

Agradezco al Dr. Freddy Yana, por facilitarme la información necesaria para el desarrollo del presente proyecto.

Agradezco a mis compañeros de carrera por el aporte de ideas a mi proyecto y colaboración en el desarrollo de la misma.

GRACIAS

DEDICATORIA

A DIOS por ser mi fortaleza.

A mis PADRES Willy y Edith, a mis HERMANOS Erick, Briseida y Bárbara. Los amo, gracias por el aliento en el camino de la vida.

Tabla de contenido

CAPÍTULO I

1. INTRODUCCIÓN

1.1 INTRODUCCION

Las nuevas tecnologías de información y comunicaciones (NTIC) para el proceso del conocimiento, han permitido a la humanidad estar aún más comunicada (globalización de las comunicaciones), desarrollando proyectos de software orientados a mejorar el servicio al cliente y disminuir costos.

El comercio actual ya no tiene fronteras de mercado gracias al e-commerce y la educación puede ser masificada mediante el e-learning. Por tanto las NTIC han posibilitado una nueva forma de interpretar el mercado.

Sabiendo que la distribución de servicios de salud, donde la distancia y el tiempo de respuesta son factores críticos, y donde los profesionales en salud deben usar información oportuna, relevante y confiable apoyada en la tecnología de comunicaciones para el intercambio de información válida para el diagnóstico, tratamiento y prevención de enfermedades o daños, investigación y evaluación es necesario desarrollar aplicaciones de software orientados a documentar y apoyar la gestión de tratamiento de la población en salud pública.

La plataforma de seguimiento de historiales clínicos apoyada por dispositivos móviles puede ser una herramienta tecnológica para el intercambio de información y comunicar a los pacientes su tratamiento a seguir en el momento oportuno.

El campo de estudio de este trabajo, es todavía en nuestra sociedad un área no desarrollada ya que la telefonía celular con acceso a internet recién se está ampliando y está abocada al diseño de una Plataforma de seguimiento de historial clínico bajo normas internacionales como HL7 y comunicando en el momento oportuno los tratamiento farmacológicos que debe seguir el paciente logrando así una recuperación satisfactoria del paciente.

Es entonces que este proyecto trata de la aplicación de las nuevas tecnologías orientadas al seguimiento de historias clínicas generadas en los servicios de salud utilizando plataformas móviles.

1.2 ANTECEDENTES

1.2.1 ANTECEDENTES DE PROYECTOS SIMILARES

En la carrera de Informática se pudo revisar los siguientes proyectos:

- "SISTEMA DE ADMINISTRACIÓN DE HISTORIAS CLÍNICOS: CLINICA SANJINES" (2009) elaborado por Américo Guillermo Machicao Rejas, el cual automatizo el registro de historias clínicas de forma electrónica, la consulta de historias clínicas, además de realizar el seguimiento y control de pacientes y asignar una consulta, con metodología UML.
- "GESTIÓN DE HISTORIAS CLÍNICAS Y CUADROS ESTADÍSTICOS APLICANDO AGENTES INTELIGENTES. CASO: CENTRO DE SALUD DE ASISTENCIA PÚBLICA" (2011) elaborado por Vanesa Paola Cárdenas Capari, automatiza el registro de un nuevo paciente, asignación de consulta médica y el registro del diagnóstico médico. Hace el uso de un agente para el análisis de datos del paciente y otro agente para generar un identificador único para cada paciente. Se aplica la metodología Scrum para el sistema y la metodología orientada a agentes MaSE.
- "SISTEMA DE CONTROL DE INVENTARIO Y MANEJO DE HISTORIALES CLÍNICOS PARA EL CENTRO DE SALUD 18 DE MAYO" (2011) elaborado por Marisol Alvarado Quisbert, el cual automatiza el registro y seguimiento de un paciente a través del historial clínico, además de hacer un control automatizado de inventario de farmacéuticos aplicando la metodología RUP.

1.2.2 ANTECEDENTES DE LA INSTITUCIÓN

Centro Medico La Paz

El centro médico La Paz ubicado en la Av. Tumusla de la Ciudad de La Paz, tiene como objetivo brindar atención médica en medicina general, medicina familiar, ginecología y pediatría.

Objetivo General.- Garantizar que el servicio sea básico y equitativo, en bienestar de la población respecto a su salud, cumpliendo las normas establecidas por el Ministerio de Salud.

Funciones.- Las funciones del Centro Medico La Paz, están divididas en: Funciones Administrativas y Funciones Técnicas.

- **Funciones Administrativas.-** La función Administrativa es la que engloba las actividades que incluye todo el proceso de organización, dirección, integración, control y evaluación de las actividades del servicio del centro médico.
- **Funciones Técnicas.-** Está constituido por las funciones de:
    - Atención consulta medicina general
    - Atención consulta medicina familiar
    - Atención consulta pediatría
    - Atención consulta ginecología

La organización de la información tanto de pacientes, como los procesos administrativos del centro médico son manuales.

1.3 PLANTEAMIENTO DEL PROBLEMA

1.3.1 FORMULACIÓN DEL PROBLEMA

¿De qué manera se puede realizar un control y seguimiento de los historiales clínicos de los pacientes del Centro Médico La Paz para brindar una asistencia clínica inmediata?

1.3.2 PROBLEMAS SECUNDARIOS

- La entidad de salud realiza un registro manual del historial clínico.

- No se conoce el nivel de ocupación del Centro Médico, lo que genera múltiples remisiones de los pacientes.
- Los recursos tecnológicos no se comparten entre las diferentes unidades, por lo que se remiten pacientes a unidades médicas externas.
- En cuanto al tratamiento se deja a voluntad del paciente el control del mismo y no se ofrece medio de colaboración que permita realizar el seguimiento y recordatorios.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Implementar un sistema de control y gestión de Historiales clínicos del Centro Médico La Paz que permita administrar esta información clínica presentándolo de manera oportuna al médico, además de permitir los pacientes a través de la aplicación móvil recibir recordatorios para la ingesta de medicamentos en el momento planificado por el médico según su tratamiento.

1.4.2 OBJETIVOS ESPECÍFICOS

Entre los objetivos específicos se tienen:

- Desarrollar un subsistema de registro de pacientes de forma automatizada.
- Desarrollar un subsistema de administración del personal médico.
- Desarrollar un subsistema medico de consulta médica.
- Desarrollar un subsistema de Citas Médicas en plataforma móvil.
- Desarrollar un subsistema de recordatorio en plataforma móvil que permita al paciente realizar la ingesta de medicamentos en el momento indicado por el médico.

1.5 JUSTIFICACIÓN

1.5.1 JUSTIFICACIÓN TÉCNICA

Desde el punto de vista técnico, este trabajo se justifica por que la estructura Cliente servidor basada en las tecnologías de hardware, software, bases de datos y redes

permitirá una mayor interactividad del flujo de información y usuarios (médico - paciente); también el presente trabajo colaborara de alguna manera a la investigación de la forma en cómo se puede realizar consultas electrónicas utilizando las ultimas herramientas de desarrollo de páginas Web dinámicas en este caso PHP y MySQL.

1.5.2 JUSTIFICACIÓN SOCIAL

El Centro médico La Paz ofrecerá al grupo social a su cargo un medio de seguimiento a sus pacientes a distancia, logrando así que el paciente se considere el centro de atención del médico.

1.5.3 JUSTIFICACIÓN ACADÉMICA

Para lograr tener un uso práctico de lo que es la plataforma se tomará como referencia las siguientes áreas:

- Ingeniería de software
- Programación móvil
- Base de Datos
- Redes y Comunicaciones
- Ingeniería web

1.6 LÍMITES Y ALCANCES

1.6.1 LÍMITES

- Seguimiento y control de Historiales clínicos.
- Administración de Médicos.
- Asignación de consultas.
- Consulta Médica.
- Esta plataforma se visualizara en la red intranet y en dispositivos móviles.

1.6.2 ALCANCES

Con el proyecto propuesto se pretende conseguir un manejo del Historial clínico de los pacientes de manera rápida y confiable, evitando los procesos manuales. Así

también realizar el seguimiento de los tratamientos de los pacientes mediante dispositivos móviles.

1.7 APORTES

- La plataforma junto con su prototipo será de gran utilidad para mejorar el desarrollo de las actividades de salud.
- Constituirá una herramienta para que cualquier persona con conocimientos de Internet básicos, pueda hacer su consulta
- Intercambio de información válida para el diagnostico
- Mayor información en temas relacionados con la salud
- La utilización de una aplicación móvil para recordatorios en la prescripción médica es de gran ayuda, ya que esta tecnología nos permite la utilización de recursos de teléfonos inteligentes además de un nivel de seguridad de acuerdo a la aplicación.

1.8 METODOLOGÍA

El presente proyecto estará basado en la metodología XP como línea orientadora para el análisis y el diseño del sistema.

- **Método descriptivo.**
  Utilizado para determinar e identificar de forma independiente, precisa y detallada el contexto conceptual del proyecto. Se puede indicar que la Institución tiene una sola oficina central y una sucursal.
- **Método de la observación.**
  Será utilizado para la obtención de los datos relacionados a las variables e identificar cuáles serán objeto de estudio para determinar los procesos que serán automatizados con el sistema. Se realizará principalmente en las instalaciones en La Paz donde acontecen los hechos en su ambiente natural, para luego analizar y descifrar los resultados obtenidos.
- **UML**
  Para el desarrollo de cualquier aplicación de software, es necesario seguir una metodología de la Ingeniería de Sistemas principalmente para tener

una orientación sobre los procesos que se deben seguir en todo momento y llegar a la conclusión con un trabajo perfectamente documentado. Se utilizará XP[1], como metodología de desarrollo basado en tecnología de la Ingeniería Web.

- **Métricas de Calidad**

  Las métricas de calidad necesarias para este proyecto están basadas en metodologías WEB. Para el establecimiento de la calidad del software se utilizará la métrica de calidad ISO 9126.

1.9 TÉCNICAS

Respecto de las técnicas de investigación, existen diferentes técnicas para obtener la información más importante relacionada con los procesos en actual estudio, cada una de ellas tienen ventajas y desventajas, por lo que es aconsejable usar más de una y así verificar los datos obtenidos en ellas. Las técnicas que se usarán para el relevamiento de información son las siguientes:

- **La entrevista personal.**

Consiste en un intercambio directo de información entre el Analista y un integrante de la organización. Se tiene una gran flexibilidad en la búsqueda de datos y brinda la oportunidad de entrar en contacto directo con el personal. Insume costos elevados de tiempo por la cantidad de horas de trabajo dedicadas.

- **La encuesta escrita.**

Se desarrolla mediante el diseño de cuestionarios específicos que se dirigen a los empleados de la institución vinculados con la investigación. La información que se obtiene es de manera inmediata aunque debe ser tabulada e interpretada. Se puede aplicar de manera simultánea a un número elevado de personas. No permite la obtención de información adicional, reticencia de los encuestados.

[1]XP Programación Extrema, es un método de ingeniería del software para el desarrollo de aplicaciones basado en UML.

- **El estudio de documentación**

Involucra la búsqueda y análisis de documentos existentes vinculados al estudio. Estos documentos deben ser seleccionados según los procesos involucrados respecto a los registros de los pacientes.

1.10 HERRAMIENTAS

Se utilizó herramientas CASE de apoyo para el diseño del sistema, como el StarUML, que permite realizar el modelado del sistema. Estas herramientas permiten acelerar el diseño del sistema, logrando reducir el tiempo de desarrollo y permiten además la coordinación de equipos de desarrollo, sincronizando las actividades y tareas que se ejecutan tanto en el diseño, desarrollo como en la implementación del mismo.

Para elaborar el código del programa, se lo realizo en el lenguaje PHP ya que ofrece diversas características de desarrollo y uso, pero lo más importante es que es de uso libre y por tanto no requiere licencias de uso. Esta plataforma permite utilizar otros lenguajes de programación y manejo de hipertexto como HTML5, HTML y archivos JSON.

Para el almacenamiento, diseño y construcción de la base de datos, se hace uso del gestor de base de datos MySQL en su versión 2011 y gracias a su robustez y madurez, ofrece escudos de seguridad a distintos niveles.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 ESTANDAR DE HISTORIA CLINICA HL7

Health Level Seven (HL7) es un conjunto de estándares cuyo principal objetivo es especificar mensajería para la comunicación de información clínica, demográfica y financiera, entre sistemas informáticos. Existen algunos estándares dentro de HL7 que tienen otros focos, pero la mensajería es uno de los aspectos más fuertes de HL7 tiene su origen en los EEUU, y sus estándares reflejan el modelo de atención en ese país. Por ejemplo, en su modelo de referencia, junto con la información clínica se encuentra la información para facturación. Igualmente, hoy en día HL7 tiene alcance internacional, debido a los capítulos locales del HL7 que se han creado en distintos países. [Subcomité Técnico HL7, 2007]

Se tienen dos estándares de Historias clínicas electrónicas (HCI Human Computer Interaction) en la actualidad: HL7 y EN13606. Para este proyecto se ha elegido HL7 por ser el estándar de facto en los sistemas informáticos sanitarios actualmente.

HL7 permiten realizar el intercambio electrónico de información clínica creado por HL7 International, una organización sin ánimo de lucro creada en 1987 como consecuencia de la problemática que exista en los sistemas de información sanitarios en cuanto a su heterogeneidad se refiere.

El nombre “Health Level Seven” hace referencia a la última capa (la séptima) del modelo de referencia OSI de la ISO, conocida como la capa de aplicación. El nombre indica que HL7 se ocupa del nivel de aplicación del modelo OSI únicamente, dejando el resto de niveles (sesión, transporte, etc.) a otros estándares y protocolos, a los que HL7 considera como meras herramientas.

El propósito de HL7 International, con sede social en EE.UU y organizaciones aliadas en diferentes pases, es desarrollar estándares para el intercambio electrónico de datos en el ámbito sanitario.

Actualmente en los hospitales y centros sanitarios coexisten una gran cantidad de aplicaciones diferentes entre sí pero al mismo tiempo dependiente.

Los principales estándares que conforman a HL7 son:

- Mensajería v2.x: mensajería basada en formatos EDI y XML, sin un modelo de referencia detrás.
- Reference Implementation Model: modelo de referencia de HL7, en el que se basan todos los mensajes v3.
- Mensajería v3: mensajería XML basada en el RIM.
- Dominios: la mensajería v3 se divide en dominios específicos de aplicación.
- Clinical Document Architecture: representación de documentos clínicos con XML basados en el RIM.
- EHR System Functional Model: especifica las funcionalidades que debería implementar un sistema de Historia Clínica Electrónica.

2.2 DISPOSITIVOS MOVILES

Se denomina smartphone o teléfono inteligente a un tipo de dispositivo móvil que ya no tiene como objetivo principal ser emisor y receptor de llamadas telefónicas. Estos móviles suelen estar permanentemente conectados a Internet y, además, tienen una capacidad de procesamiento mucho más cercana a la de un ordenador que a la capacidad con la que contaban sus predecesores.

Las características principales que diferencian a un smartphone de un teléfono móvil tradicional son:

- **Sistema operativo.** Estos dispositivos están basados en un sistema operativo sobre el que ejecutan sus aplicaciones, por ejemplo Android.
- **Aplicaciones.** Mientras que los móviles convencionales tenían una serie de funciones muy limitadas, éstos tienen la habilidad de hacer un mayor número de tareas, desde ver documentos de texto a obtener direcciones mediante GPS. Además, son extremadamente personalizables, ya que cada persona puede descargarse las aplicaciones que más le convengan.

- **Acceso a la web.** Cada vez más *smartphones* pueden acceder a la web a velocidades más altas, gracias a las tecnologías 3G y 4G así como al soporte de conexión Wi-Fi de estos terminales.
- **Teclado QWERTY.** Los teléfonos móviles siempre han tenido un teclado alfabético, pero en este momento en el que tenemos funciones para las que es necesario escribir textos más extensos, es mucho más cómodo contar con un teclado QWERTY.

2.2.1 ARQUITECTURA DE LA PLATAFORMA ANDROID

La arquitectura de la plataforma Android está compuesta por cinco capas: el núcleo de linux y herramientas de bajo nivel, las bibliotecas nativas, el entorno de ejecución de Android, el marco de aplicaciones y encima de todas, las aplicaciones.

- **Núcleo linux:** Proporciona servicios básicos como drivers de hardware, gestión de energía y de memoria y una capa de abstracción entre el hardware y el resto de las partes.
- **Bibliotecas nativas:** Android incluye un conjunto de bibliotecas C y C++, a las que se puede acceder mediante el marco de aplicaciones.
- **Entorno de ejecución de Android:** En esta capa se encuentran las bibliotecas de funciones básicas de Java y las específicas de Android. Además, de una máquina virtual, que está basada en registros y ejecuta las clases compiladas por el compilador de Java que se transforman al formato de la máquina virtual ejecutables por la misma.
- **Marco de aplicaciones:** Los desarrolladores tienen acceso total al código fuente usado en las aplicaciones básicas. De esta forma no se generan duplicados de componentes básicos que realizan las mismas acciones si no que se utilizan los que ya existen para evitar empezar a programar desde cero.
- **Aplicaciones:** En esta parte podemos ver las aplicaciones creadas con Android, tanto las nativas del sistema operativo como las que están creadas por terceros. [Paredes, Santa Cruz & Dominguez, 2012]

2.3 SERVICIO WEB

Se lo puede describir como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella. Desde la perspectiva de la empresa, un servicio realiza una tarea concreta: puede corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato como "Código del Cliente". Los servicios pueden acoplarse dentro de una aplicación completa que proporcione servicios de alto nivel, con un grado de complejidad muy superior, por ejemplo, "introducir datos de un pedido", un proceso que, desde su inicio hasta su conclusión, puede involucrar varias aplicaciones de negocio. [W3C, 2010]

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan en un mismo ordenador, el modelo de servicios facilita el acceso y consumo de los recursos de IT a través de la red. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de cada momento en el seno de una organización. Las aplicaciones compuestas (también llamadas "dinámicas") son lo que permite a las empresas mejorar y automatizar sus procesos manuales, disponer de una visión consistente de sus clientes y socios comerciales y orquestar sus procesos de negocio para que cumplan con las regulaciones legales y políticas internas. El resultado final es que las organizaciones que adoptan la orientación a servicios pueden crear y reutilizar servicios y aplicaciones y adaptarlos ante los cambios

evolutivos que se producen dentro y fuera de ellas, y con ello adquirir la agilidad necesaria para ganar ventaja competitiva. [W3C, 2010]

La adopción de una solución de diseño basada en SOA no exige implantar servicios Web. No obstante, los servicios Web son la forma más habitual de implementar SOA. Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Los servicios Web permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, tanto dentro de las organizaciones como con partners[2] de negocios. Los servicios Web se basan en un conjunto de estándares de comunicación, como son XML[3] para la representación de datos, SOAP (Simple Object Access Protocol) para el intercambio de datos y el lenguaje WSDL (Web Services Description Language) para describir las funcionalidades de un servicio Web. Existen más especificaciones, a las que se denomina genéricamente como la arquitectura WS-*, que definen distintas funcionalidades para el descubrimiento de servicios Web, gestión de eventos, archivos adjuntos, seguridad, gestión y fiabilidad en el intercambio de mensajes y transacciones. [W3C, 2010]

2.3.1 FORMATO DE INTERCAMBIO DE DATOS JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que

[2] Socios

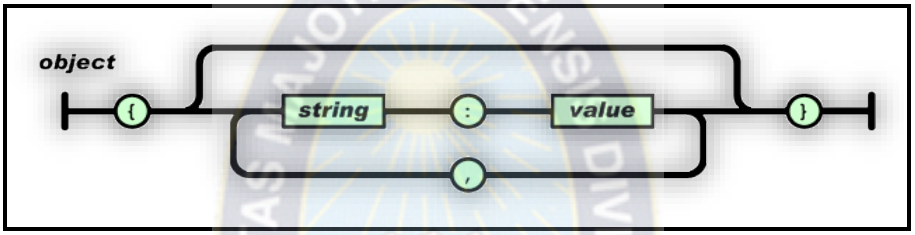
[3] XML, siglas en inglés de eXtensible Markup Language

JSON sea un lenguaje ideal para el intercambio de datos. [Potencier & Zaninotto, 2008]

En JSON, se presentan de estas formas:

- Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con “{” (llave de apertura) y termine con “}” (llave de cierre). Cada nombre es seguido por “:” (dos puntos) y los pares nombre/valor están separados por “,” (coma).

Figura 2.1 Objeto JSON



Fuente: Manejo de objetos JSON. (Potencier & Zaninotto, 2008)

- Un arreglo es una colección de valores. Un arreglo comienza con “[” (corchete izquierdo) y termina con “]” (corchete derecho). Los valores se separan por “,” (coma).

Figura 2.2 Arreglo JSON

Fuente: Manejo de arreglos JSON. (Potencier & Zaninotto, 2008)

2.3.2 INGENIERIA WEB

La ingeniería web es la aplicación de metodologías sistemáticas, disciplinadas y cuantificables al desarrollo eficiente, operación y evolución de aplicaciones de alta

calidad en la World Wide Web[4].La ingeniería web se debe al crecimiento desenfrenado que está teniendo la Web, ocasionando un impacto en la sociedad y el nuevo manejo que se le está dando a la información en las diferentes áreas en que se presenta ha hecho que las personas tiendan a realizar todas sus actividades por esta vía. [Pressman, 2002]

Uno de los aspectos más tenidos en cuenta, en el desarrollo de sitios web es sin duda alguna el diseño gráfico y la organización estructural del contenido. En la actualidad la web está sufriendo grandes cambios, que han obligado a expertos en el tema a utilizar herramientas y técnicas basadas en la ingeniería del software, para poder garantizar el buen funcionamiento y administración de los sitios web.

Cabe destacar que la ingeniería de la web hace una diferencia entre un sitio web y un aplicativo, ya que la ingeniería de la web no se dedica a la construcción de sitios web si no a la construcción de aplicativos web, la principal característica que los distingue (aplicativos de sitios web) es que los sitios web son sitios en la web en donde se publica contenido generalmente estático o un muy bajo nivel de interactividad con el usuario, mientras que los aplicativos son lugares con alto contenido de interactividad y funcionalidades que bien podrían ser de un software convencional, el aplicativo web más sencillo seria uno que contenga formularios y subiendo de nivel encontramos los que realizas conexión con bases de datos remotas, y administradores de contenidos entre otras.

Entonces la ingeniería de la Web es la aplicación de metodologías sistemáticas, disciplinadas y cuantificables al desarrollo eficiente, operación y evolución de aplicaciones de alta calidad en la World Wide Web[5]. En este sentido, la ingeniería de la Web hace referencia a las metodologías, técnicas y herramientas que se utilizan

[4] La World Wide Web es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet.

[5] Definición de Ingeniería de la Web extraída de www.webengineering.org

en el desarrollo de aplicaciones Web complejas y de gran dimensión en las que se apoya la evaluación, diseño, desarrollo, implementación y evolución de dichas aplicaciones.

2.3.3 CATEGORÍAS DE SITIOS WEB

Los sitios web pueden ser categorizados de la siguiente forma:

- Sólo estático que se enfoca en la organización de la estructura y el contenido, en la forma como se va a presentar la información y que sea fácil de manejar para cualquier usuario, pero debe tener en cuenta la eficiencia y la confiabilidad.
- Sitio estático con formularios de entrada este sitio tiene las mismas características que el anterior, adicionándole que él le permite a los usuarios la interacción por medio de cuestionarios, comentario y sugerencias.
- Sitio con acceso de datos dinámicos aquí, además de las características antes mencionadas, cuenta con bases de datos en las cuales el usuario puede realizar consultas y búsquedas.
- Sitio creado dinámicamente en este sitio los requerimientos son parecidos pero deben suplir con las necesidades de cada usuario; creando sitios dinámicos que sean compatibles con el entorno de navegación de cada usuario.
- Aplicación de software basada en la Web este sitio puede tener todas las características antes mencionadas, pero logrando un parecido con una implementación cliente/servidor comúnmente conocido que a un sitio web estático.

Con el pasar del tiempo y la constante evolución tecnológica que atraviesa nuestro mundo circundante hemos podido observar la necesidad y la utilidad de la red de redes; Internet para mejorar de cierta manera nuestras condiciones de vida y así fortalecer más nuestro proceso de formación educativa y contribuir con un mejoramiento del global de las necesidades de cada quien observemos que un proyecto que comenzó meramente con fines militares para no centralizar los datos,

ha tenido un crecimiento significativo, hoy en día el mundo se mueve con la web, ayudando a pequeñas, medianas y grandes empresas así como todo entidad educativa.

Se debe tener en cuenta que para la efectiva comunicación en la web, se tienen protocolos que es como el lenguaje, para que se haga efectiva el intercambio de comunicación, vale la pena preguntarse, así para poder acceder a toda la información que nos puede suministrar Internet sólo debes poseer un servicio de algún proveedor de Internet un navegador como Mozilla o Google Chrome.

Todas estas consideraciones nos llevan a la conclusión de que un sitio web bien logrado no es únicamente un espacio en la red para ver el mismo comercial que en televisión; es en realidad una extensión de las empresas o instituciones, así mismo teniendo en cuenta la importancia y aplicabilidad que tiene la ingeniería Web en nuestro desarrollo cognitivo, social y vivencial es fácil visionar que cada una de las funciones que ella emana estarán siempre ligadas a la vanguardia del desarrollo progresivo de la tecnología y del hombre

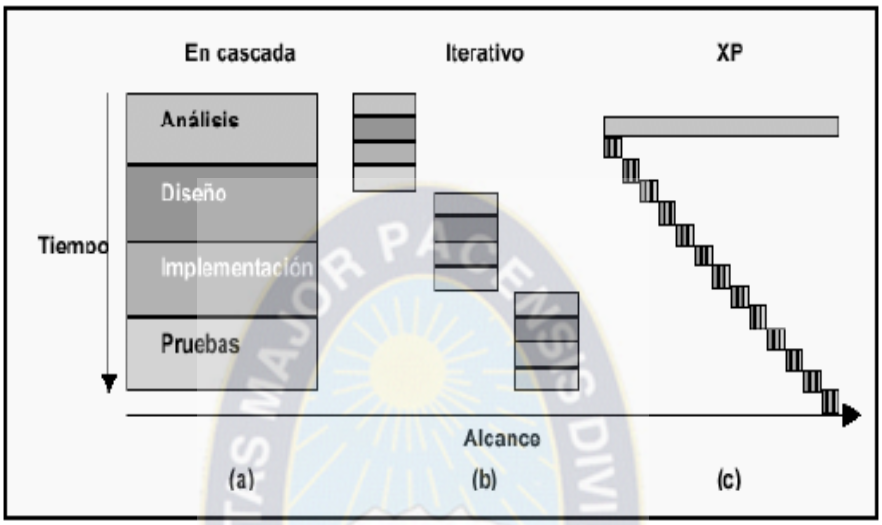
2.4 METODOLOGÍA XP

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software[6]. Forma parte del conjunto de métodos ágiles que centran sus prioridades en las personas, no en los procesos, en la actualidad XP se proyecta a ser un modelo de desarrollo común, sencillo y adaptable a las características cambiantes y exigentes de empresas y clientes. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

[6] Nace de la mano de Kent Beck en 1995, cuando trabajaba para Chrysler Corporation.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Figura 2.3 El coste del cambio a Extreme Programming.



Fuente: Introducción a la programación extrema. (Beck, 2004)

2.4.1 PRINCIPIOS DE XP

Los principios originales de la programación extrema son: simplicidad, comunicación, retroalimentación (feedback) y coraje. Un quinto principio, respeto, fue añadido en la segunda edición de Extreme Programming Explained[7]. Los cinco principios se detallan a continuación:

a. Simplicidad: La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí

[7] Libro escrito por Kent Beck en 1999

que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no decrementan la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

b. Comunicación: La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de cómo utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas.

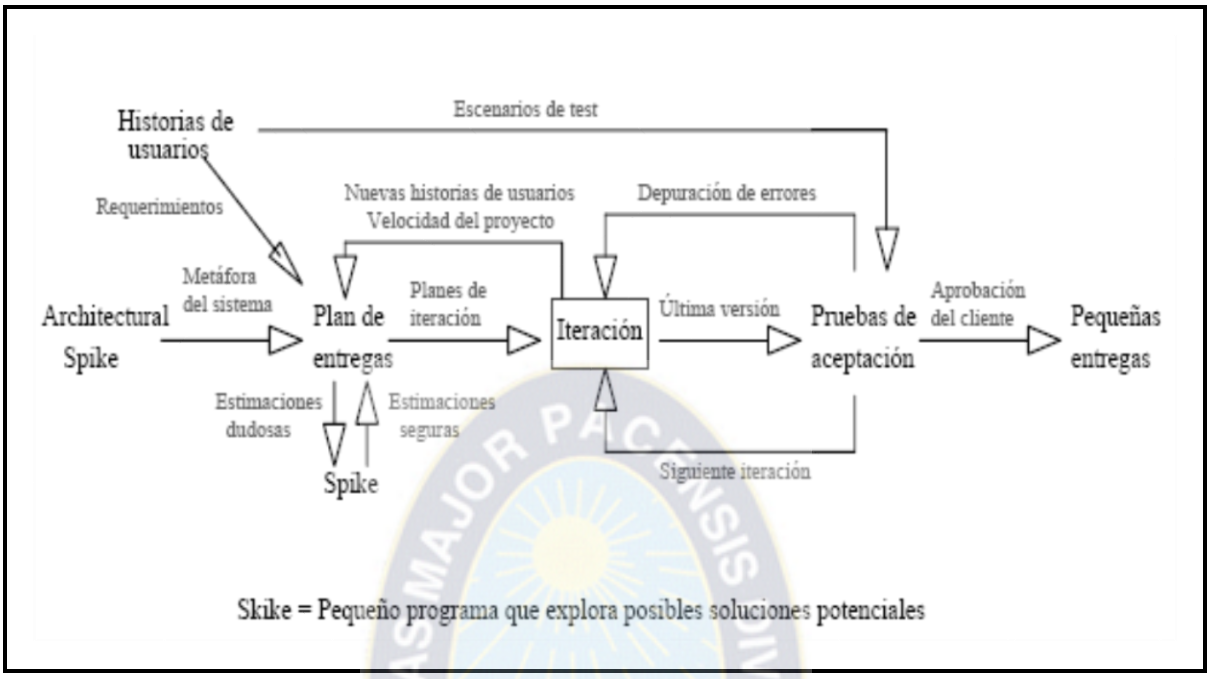
c. Retroalimentación (feedback): Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código.

Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

d. Coraje o valentía: Los puntos anteriores parecen tener sentido común, entonces, ¿por qué coraje? Para los gerentes la programación en parejas puede ser difícil de aceptar, porque les parece como si la productividad se fuese a reducir a la mitad ya que solo la mitad de los programadores está escribiendo código. Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad. La simplicidad es uno de los principios más difíciles de adoptar. Se requiere coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permita futuras modificaciones. No se debe emprender el desarrollo de grandes marcos de trabajo (*frameworks*) mientras el cliente espera. En ese tiempo el cliente no recibe noticias sobre los avances del proyecto y el equipo de desarrollo no recibe retroalimentación para saber si va en la dirección correcta. La forma de construir marcos de trabajo es mediante la refactorización del código en sucesivas aproximaciones.

e. Respeto: El respeto se manifiesta de varias formas. Los miembros del equipo se respetan los unos a otros, porque los programadores no pueden realizar cambios que hacen que las pruebas existentes fallen o que demore el trabajo de sus compañeros. Los miembros se respetan su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo o más eficiente para la solución a través de la refactorización del código.

Figura 2.4 Trabajando con Extreme Programming.



Fuente: Introducción a la programación extrema. (Beck, 2004)

2.4.2 CARACTERISTICAS DE XP

Las características fundamentales del método son:

a. Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
b. Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión[8]. Se aconseja escribir el código de la prueba antes de la codificación. Véase, por ejemplo, las herramientas de prueba JUnit orientada a Java, DUnit orientada a Delphi y NUnit para la plataforma.NET. Estas dos últimas inspiradas en JUnit.
c. Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la

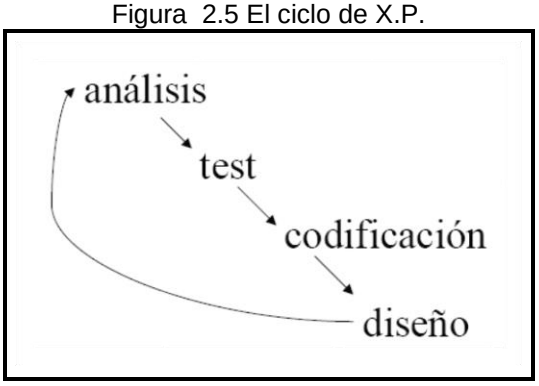
[8] Se denominan Pruebas de regresión a cualquier tipo de pruebas de software que intentan descubrir las causas de nuevos errores (bugs), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, inducidos por cambios recientemente realizados en partes de la aplicación que anteriormente al citado cambio no eran propensas a este tipo de error. Esto implica que el error tratado se reproduce como consecuencia inesperada del citado cambio en el programa.

mayor calidad del código escrito de esta manera, el código es revisado y discutido mientras se escribe, es más importante que la posible pérdida de productividad inmediata.

d. Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
e. Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
f. Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
g. Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
h. Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Figura 2.5 El ciclo de X.P.

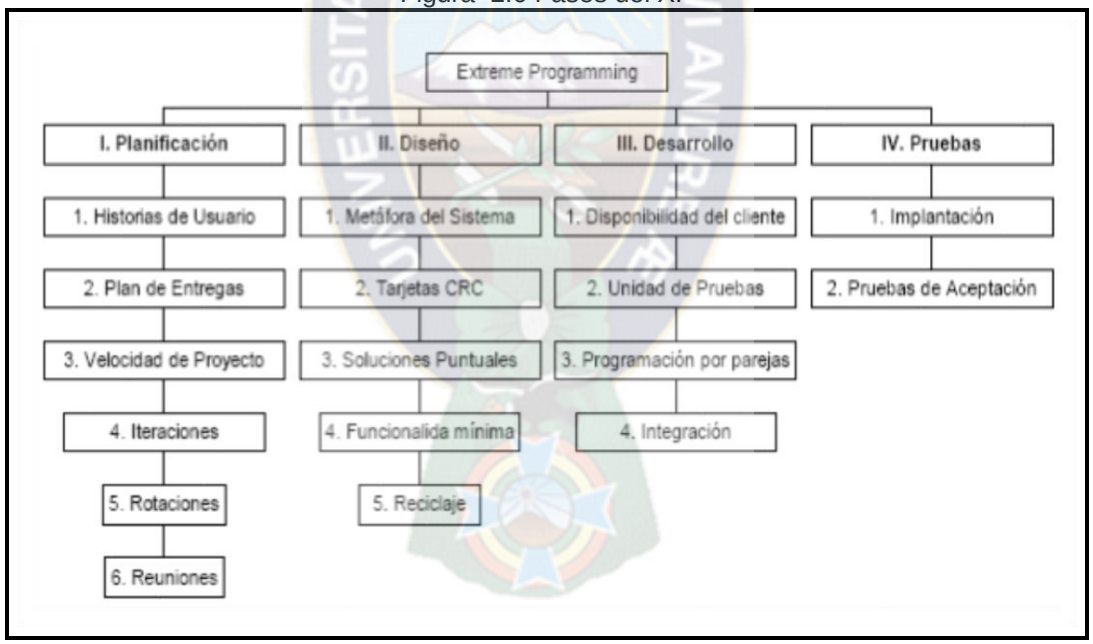


Fuente: Introducción a la programación extrema. (Beck, 2004)

2.4.3 FASES DE XP

El grafico siguiente muestra las fases de la programación extrema para el desarrollo e implementación de sistemas de información.

Figura 2.6 Fases del XP



Fuente: Introducción a la programación extrema. (Beck, 2004)

2.4.3.1 PLANIFICACIÓN

a. Historias de usuario: El primer paso de cualquier proyecto que siga la metodología X.P es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas

diferencias: Constan de 3 o 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

Release planning*: Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés "Release plan", donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un "Release plan" es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de un "Release plan" tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado. (Release plan: Planificación de publicaciones).

b. Iteraciones Todo proyecto que siga la metodología X.P. se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en el "Release planning" que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior. Estas historias de usuario son divididas en tareas de entre 1 y 3 días de duración que se asignarán a los programadores.

c. Velocidad del proyecto**:** La velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el proyecto; estimarla es muy sencillo, basta con contar el número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto controlaremos que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración. Es conveniente reevaluar esta medida cada 3 ó 4 iteraciones y si se aprecia que no es adecuada hay que negociar con el cliente un nuevo "Release Plan".

d. Programación en pareja: La metodología X.P. aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado. El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. De esta forma se consigue un código y diseño con gran calidad.

Reuniones diarias. Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

2.4.3.2 FASE 2: DISEÑOS SIMPLES

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementarlo que a la larga costará menos tiempo y esfuerzo desarrollar.

Glosarios de términos: Usar glosarios de términos y una correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

Riesgos: Si surgen problemas potenciales durante el diseño, X.P sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema.

Funcionalidad extra: Nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10% de la misma es utilizada, lo que implica que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

Refactorizar. Refactorizar es mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad. Refactorizar supone revisar de nuevo estos códigos para procurar optimizar su funcionamiento. Es muy común rehusar códigos ya creados que contienen funcionalidades que no serán usadas y diseños obsoletos. Esto es un error porque puede generar código completamente inestable y muy mal diseñado; por este motivo, es necesario refactorizar cuando se va a utilizar código ya creado.

Tarjetas C.R.C. El uso de las tarjetas C.R.C (Class, Responsabilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

2.4.3.3 FASE 3: CODIFICACIÓN

Como ya se dijo en la introducción, el cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de X.P. A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar

presente cuando se realicen los test que verifiquen que la historia implementada cumple la funcionalidad especificada.

La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad.

Crear test que prueben el funcionamiento de los distintos códigos implementados nos ayudará a desarrollar dicho código. Crear estos test antes nos ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y sabremos que una vez implementado pasará dichos test sin problemas ya que dicho código ha sido diseñado para ese fin. Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas unidades, de esta forma se crearán primero los test para cada unidad y a continuación se desarrollará dicha unidad, así poco a poco conseguiremos un desarrollo que cumpla todos los requisitos especificados.

Como ya se comentó anteriormente, X.P opta por la programación en pareja ya que permite un código más eficiente y con una gran calidad.

X.P sugiere un modelo de trabajo usando repositorios de código dónde las parejas de programadores publican cada pocas horas sus códigos implementados y corregidos junto a los test que deben pasar. De esta forma el resto de programadores que necesiten códigos ajenos trabajarán siempre con las últimas versiones. Para mantener un código consistente, publicar un código en un repositorio es una acción exclusiva para cada pareja de programadores.

X.P también propone un modelo de desarrollo colectivo en el que todos los programadores están implicados en todas las tareas; cualquiera puede modificar o ampliar una clase o método de otro programador si es necesario y subirla al repositorio de código. El permitir al resto de los programadores modificar códigos que no son suyos no supone ningún riesgo ya que para que un código pueda ser publicado en el repositorio tiene que pasar los test de funcionamiento definidos para el mismo.

La optimización del código siempre se debe dejar para el final. Hay que hacer que funcione y que sea correcto, más tarde se puede optimizar.

X.P afirma que la mayoría de los proyectos que necesiten más tiempo extra que el planificado para ser finalizados no podrán ser terminados a tiempo se haga lo que se haga, aunque se añadan más desarrolladores y se incrementen los recursos. La solución que plantea X.P es realizar un nuevo "Release plan" para concretar los nuevos tiempos de publicación y de velocidad del proyecto.

A la hora de codificar no seguimos la regla de X.P que aconseja crear test de funcionamiento con entornos de desarrollo antes de programar. Nuestros test los obtendremos de la especificación de requisitos ya que en ella se especifican las pruebas que deben pasar las distintas funcionalidades del programa, procurando codificar pensando en las pruebas que debe pasar cada funcionalidad.

2.4.3.4 FASE 4: PRUEBAS

Uno de los pilares de la metodología X.P es el uso de test para comprobar el funcionamiento de los códigos que vayamos implementando. El uso de los test en X.P es el siguiente:

- Se deben crear las aplicaciones que realizarán los test con un entorno de desarrollo específico para test.
- Hay que someter a test las distintas clases del sistema omitiendo los métodos más triviales.
- Se deben crear los test que pasarán los códigos antes de implementarlos; en el apartado anterior se explicó la importancia de crear antes los test que el código.
- Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará.
- Hay que crear los test abstrayéndose del futuro código, de esta forma aseguraremos la independencia del test respecto al código que evalúa.

Los resultados obtenidos de las distintas pruebas o test deben ser almacenadas en el repositorio de código, acompañados del código que verifican. Ningún código puede ser publicado en el repositorio sin que haya pasado su test de funcionamiento, de esta forma, aseguramos el uso colectivo del código. El uso de los test es adecuado para observar la refactorización. Los test permiten verificar que un cambio en la estructura de un código no tiene por qué cambiar su funcionamiento.

Test de aceptación**.** Los test mencionados anteriormente sirven para evaluar las distintas tareas en las que ha sido dividida una historia de usuario. Para asegurar el funcionamiento final de una determinada historia de usuario se deben crear "Test de aceptación"; estos test son creados y usados por los clientes para comprobar que las distintas historias de usuario cumplen su cometido. Al ser las distintas funcionalidades de nuestra aplicación no demasiado extensas, no se harán test que analicen partes de las mismas, sino que las pruebas se realizarán para las funcionalidades generales que debe cumplir el programa especificado en la descripción de requisitos.

2.5 MÉTODOS DE PRUEBA DEL SISTEMA

Entre los distintos métodos que existen para realizar pruebas estos se pueden dividir en prácticamente 2, los cuales son métodos de caja blanca y métodos de caja negra.

2.5.1 MÉTODO DE CAJA BLANCA

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo, ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa, ejecuten todos los ciclos en sus límites y con sus límites operacionales y ejerciten las estructuras internas de datos para asegurar su validez. Con este método se determina cuáles son los casos de prueba a partir del código fuente del software y se produce un resultado correcto, así como que la integridad de la información externa.

La prueba de caja blanca del software se basa en el minucioso examen de los detalles procedimentales. Se comprueba los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o ciclos. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

2.5.2 MÉTODO DE CAJA NEGRA

Con este método los casos de prueba y los resultados se determinan a partir de la especificación funcional del método de una clase. Es decir, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

2.6 MODELO DE CALIDAD ISO 9126

El estándar ISO-9126 establece que cualquier componente de la calidad del software puede ser descrito en términos de una o más de seis características básicas, las cuales son: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad; cada una de las cuales se detalla a través de un conjunto de subcaracterísticas que permiten profundizar en la evaluación de la calidad de productos de software. [Joc & Curran, s.f.]

A continuación se detalla cada una de las características que establece el estándar ISO-9126:

- **Funcionalidad.**

Esta característica permite agrupar un conjunto de atributos que permiten calificar si un producto de software maneja en forma adecuada el conjunto de funciones que satisfacen las necesidades. Para este propósito se establecen:

- Exactitud. Que permite evaluar si el software presenta resultados acordes a las necesidades.

- Interoperabilidad. Permite evaluar la habilidad del software de interactuar con otros sistemas previamente especificados.
- Conformidad. Evalúa si el software se adhiere a estándares, convenciones o regulaciones en leyes y prescripciones similares.
- Seguridad. Se refiere a la habilidad de prevenir el acceso no autorizado, ya sea accidental o premeditado, a los programas y datos.
- Adecuación. Se enfoca a evaluar si el software cuenta con un conjunto de funciones apropiadas para efectuar las tareas que fueron especificadas en su definición.

- **Confiabilidad.**

Agrupa atributos que se refieren a la capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido. Las subcaracterísticas que el estándar sugiere son:

- Nivel de Madurez. Permite medir la frecuencia de falla por errores en el software.
- Tolerancia a fallas. Es la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software o de cometer infracciones de su interfaz específica.
- Recuperación. Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla, así como al tiempo y el esfuerzo necesarios para lograrlo.

- **Usabilidad.**

Consiste de un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema.

- Comprensibilidad. Se refiere al esfuerzo requerido por los usuarios para reconocer la estructura lógica del sistema y los conceptos relativos a la aplicación del software.

- Facilidad de Aprender. Establece atributos del software relativos al esfuerzo que los usuarios deben hacer para aprender a usar la aplicación.
- Operabilidad. Agrupa los conceptos que evalúan la operación y el control del sistema.

- **Eficiencia.**

Esta característica permite evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados. Los aspectos a evaluar son:

- Comportamiento con respecto al Tiempo. Atributos del software relativos a los tiempos de respuesta y de procesamiento de los datos.
- Comportamiento con respecto a Recursos. Atributos del software relativos a la cantidad de recursos usados y la duración de su uso en la realización de sus funciones.

- **Mantenibilidad.**

Se refiere a los atributos que permiten medir el esfuerzo necesario para realizar modificaciones al software, ya sea por la corrección de errores o por el incremento de funcionalidad. En este caso, se tienen los siguientes factores:

- Capacidad de análisis. Relativo al esfuerzo necesario para diagnosticar las deficiencias o causas de fallas, o para identificar las partes que deberán ser modificadas.
- Capacidad de modificación. Mide el esfuerzo necesario para modificar aspectos del software, remover fallas o adaptar el software para que funcione en un ambiente diferente.
- Estabilidad. Permite evaluar los riesgos de efectos inesperados debidos a las modificaciones realizadas al software.
- Facilidad de Prueba. Se refiere al esfuerzo necesario para validar el software una vez que fue modificado.

- **Portabilidad.**

Es la habilidad del software de ser transferido de un equipo considerando:

- Adaptabilidad. Evalúa el grado de adaptación del software a diferentes ambientes sin necesidad de aplicarle modificaciones.
- Facilidad de Instalación. Es el esfuerzo necesario para instalar el software en un ambiente determinado.
- Conformidad. Permite evaluar si el software se adhiere a estándares o convenciones relativas a portabilidad.

Capacidad de reemplazo. Se refiere a la oportunidad y el esfuerzo usado en sustituir el software por otro producto con funciones similares.

2.7 MODELO DE COSTOS - COCOMO

El modelo de costos COCOMO es un modelo empírico de estimación de costes del software, que se obtuvo recopilando datos de varios proyectos grandes. Estos datos fueron analizados para descubrir las fórmulas que mejor se ajustaban a las observaciones. Estas fórmulas vinculan el tamaño del sistema y del producto, factores del proyecto y del equipo con el esfuerzo necesario para desarrollar el sistema [9]. Se ha elegido COCOMO por las siguientes razones:

- Está bien documentado, es de dominio público y lo apoyan el dominio público y las herramientas comerciales.
- Se ha utilizado y evaluado ampliamente.
- Tiene una gran tradición desde su primera versión en 1981, pasando por un refinamiento para el desarrollo de software en ADA[9], hasta su versión más reciente, COCOMO II, publicada en 2000.

[9] Lenguaje de Programación que fue desarrollado por el Departamento de Defensa de los Estados Unidos como un lenguaje estándar para el desarrollo de software militar.

Los modelos COCOMO son comprensibles, con un gran número de parámetros, los cuales pueden tomar un rango de valores. Éstos son complejos y no se puede dar una descripción completa aquí. Basta una simple exposición de las características esenciales para comprender los modelos algorítmicos de costes.

Los niveles de COCOMO II son los siguientes:

- *Nivel de construcción de prototipos.* Este presume que el sistema es creado mediante componentes reutilizables, scripts y programación de base de datos. Fue diseñado para hacer estimaciones de desarrollo de prototipos. Las estimaciones de tamaño del software están basadas en puntos de aplicación, y se utiliza una fórmula simple (tamaño/productividad) para estimar el esfuerzo requerido.
- *Nivel de diseño inicial.* Este nivel se utiliza en etapas tempranas del diseño del sistema, después de que los requerimientos hayan sido establecidos. Las estimaciones están basadas en puntos de función, los cuales se convierten a número de líneas de código. La fórmula permite seguir el estándar expuesto anteriormente con un conjunto de siete multiplicadores.
- *Nivel de re utilización.* Este nivel se utiliza para calcular el esfuerzo requerido para integrar componentes reutilizables y/o el código que es automáticamente generado por herramientas de diseño o programas de traducción. Normalmente es utilizado junto con el nivel de post-arquitectura.
- *Nivel de post arquitectura.* Una vez diseñado el sistema, se puede hacer una estimación más precisa del tamaño del software. Otra vez se utiliza la fórmula estándar para la estimación del coste expuesta anteriormente. Sin embargo, ésta incluye un conjunto de 17 multiplicadores que reflejan las habilidades del personal y las características del producto y del proyecto. [Sommerville, 2005]

Por supuesto, en sistemas grandes, diferentes partes pueden ser desarrolladas utilizando diversas tecnologías, y puede que no sea preciso estimar todas las partes con el mismo nivel de precisión. En estos casos, se puede utilizar el modelo apropiado para cada parte del sistema y combinar los resultados para crear una estimación del sistema completo.

2.8 ESTANDAR DE SEGURIDAD OWASP

OWASP (Open Web Application Security Project) es, según su propio sitio web, un proyecto de código abierto dedicado a determinar y a combatir las causas que hacen que las aplicaciones web sean inseguras.

Los documentos y proyectos más destacados de OWASP son, probablemente, la *Guía OWASP* y el documento de autoevaluación *OWASP Top 10*.

2.8.1 Guía OWASP

Muchos profesionales de la seguridad ven todavía la comprobación de seguridad dentro del terreno de las pruebas de intrusión. Como se ha discutido anteriormente, aunque las pruebas de intrusión tienen un papel que jugar, generalmente es ineficaz en encontrar bugs, y depende excesivamente de la capacidad del probador. Debería ser considerado únicamente como una técnica de implementación, o suscitar una mayor atención sobre incidencias de producción. Para mejorar la seguridad de las aplicaciones, se debe antes mejorar la calidad de seguridad del software. Eso significa comprobar la seguridad en las fases de definición, diseño, desarrollo, implementación y mantenimiento, y no confiar en la costosa estrategia de esperar hasta que el código este construido por completo.

Tal y como se expuso en la introducción, hay muchas metodologías de desarrollo, como la de proceso racional unificado, desarrollo ágil y extremo, y las metodologías tradicionales de cascada. La intención de la guía no es ni apuntar a una metodología de desarrollo en particular ni proporcionar unas directrices específicas que se ajusten a una metodología especifica. En vez de eso, se presenta un modelo de desarrollo genérico, que se debería seguir de acuerdo al proceso que emplee su compañía.

Este marco de pruebas consta de las siguientes actividades:

- Antes de empezar el desarrollo.
- Durante el diseño y definición.
- Durante el desarrollo.
- Durante la implementación.

- Mantenimiento y operaciones

2.8.2 OWASP Top 10

El objetivo principal del Top 10 es educar a los desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones; sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web. El Top 10 provee técnicas básicas sobre cómo protegerse en estas áreas de alto riesgo – y también provee orientación sobre los pasos a seguir.

Figura 2.7 Características de seguridad del Estándar OWASP Top 10

Fuente: Introducción a OWASP Top 10. (OWASP Foundation, 2008)

CAPÍTULO III

3. MARCO APLICATIVO

3.1 INTRODUCCIÓN

Los sistemas médicos brindan beneficios como la disminución de los tiempos de atención, establecer diagnósticos y tratamientos más oportunos, reduce costos de transporte del paciente y puede ofrecer una atención continua estableciendo recordatorios de los tratamientos a los pacientes logrando una mayor incidencia de los medicamente en el paciente.

La implementación del sistema de control y gestión de Historiales Clínicos se realizó en base a la metodología XP, UML y archivo JSON.

Con este objetivo, se utilizó el framework PHP Designer v.8 que brinda la ventaja de integrarse perfectamente unos con otros sistemas mediante JSON además permite la comunicación entre ellos e integra los componentes de las diferentes capas.

Adicionalmente se utilizó el App Inventor 2 ofrecido on line por la MIT y Google Inc. previo registro en su sistema. Este aplicativo permite el desarrollo de aplicaciones para dispositivos móviles mediante código de bloques y generando un APK para ser instalado en el celular.

3.2 FASE I. PLANIFICACIÓN

3.2.1 ACTORES DEL SISTEMA

La gestión de usuarios es un elemento fundamental en la arquitectura de la plataforma porque trata aspectos tan relevantes como la autenticación personal, los permisos de acceso a la información y a los dispositivos o los tipos de usuarios.

3.2.1.1 USUARIO

En esta clase se almacenaran los datos básicos de todos los usuarios, independientemente de su tipo o rol. Los atributos almacenan el nombre y apellidos

del usuario y tienen correspondencia con campos del segmento de identificación del paciente (PID). Además, se requiere que cada usuario mantenga un autenticador y una contraseña de acceso para mantener un sistema de autenticación de usuarios. Por último, se almacena en la fecha y hora del último acceso del usuario al sistema.

3.2.1.2 ROL

Los roles definen una serie de permisos que tiene un usuario para acceder a los datos y a los dispositivos. De esta manera se separa la descripción de los tipos de usuarios de la definición de sus privilegios.

Un usuario puede asumir varios roles simultáneamente de tal forma que cada rol cubra una serie de necesidades, lo que permite una asignación flexible de permisos. Los equipos pueden estar o no disponibles según el rol que tenga el usuario.

3.2.1.3 ADMINISTRADOR

Según los requisitos del sistema, se requiere de un administrador que se encargará de dar de alta y de baja a los usuarios, modificar sus datos y gestionar los registros del sistema. Esta clase hereda de “User" y define una serie de roles por defecto con acceso total al sistema.

3.2.1.4 MÉDICO

El médico tendrá acceso a la información médica de los pacientes a su cargo y trabajará con enfermeros y asistentes que son otro tipo de usuarios.

Hereda de “User" y define una serie de roles por defecto que permitan al médico acceder a los datos necesarios para realizar su labor.

3.2.1.5 PACIENTE

Será necesario guardar una serie de atributos específicos para los usuarios de la clase Paciente, como su fecha de nacimiento, su domicilio , número de teléfono, aviso de recordatorios y toda su información médica además del sexo de la persona, todo esto basado en las recomendaciones estándar HL7.

3.2.2 ESCENARIOS

En este proyecto se define tres escenarios, según la intervención del personal médico o administrativo: consultas, registro de los datos del paciente y administración del sistema.

Consultas: se trata de chequeos médicos donde el paciente debe estar en contacto con el personal médico.

Registro de los datos, donde el personal médico registra mediante el sistema web los datos personales y/o clínicos del paciente, además de incluir los medicamentos que se le recomiendan para aliviar la enfermedad, principalmente las horas de ingesta de cada píldora para establecer los horarios de recordatorio.

Administración: este módulo está encargado de realizar la configuración de los usuarios y del sistema: añadir usuarios, establecer permisos, registrara pacientes, médicos, es decir puede realizar todas las acciones del sistema.

3.2.3 HISTORIAS DE USUARIO

Como primera actividad para el desarrollo del sistema, la metodología XP establece que deben realizarse las historias de usuario para determinar las necesidades que esperan satisfacer los clientes, en este caso la Clínica, especialmente los médicos.

HISTORIA DE USUARIO Sistema de Control y Gestión de Historial Clínico

A partir de las necesidades establecidas tras una entrevista y trabajo de campo, se logró establecer la necesidad principal expresada en la siguiente historia de usuario.

Tabla 3.1 H.U. Descripción general del sistema

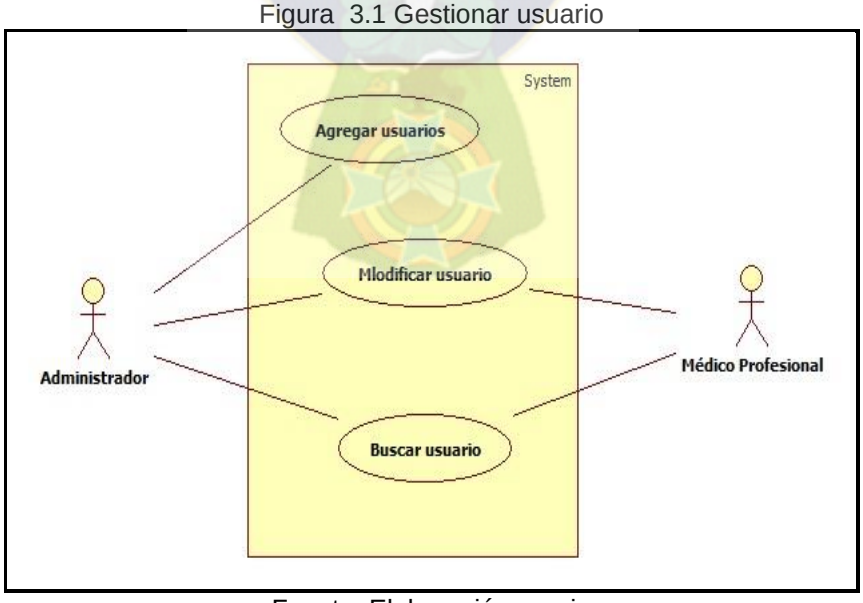
Historia de Usuario Nº 1			
Nombre Historia de Usuario		Sistema de Control y gestión de Historial clínico	
Usuario	Administrador y personal médico	**Iteración Asignada**	Aún no establecida
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Desarrollo del sistema contemplando la metodología de desarrollo XP logrando así la centralización de los datos en una base de datos.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Aún no corresponde.
Descripción	Brindar un servicio médico a los pacientes, registrando sus datos y tratamiento según el caso individual de cada paciente e introducción de datos mediante vía web.		
Observaciones	Se logró el desarrollo del sistema considerando el factor tiempo, con un retraso de 2 semanas. El tiempo estimado era de nueve semanas.		

Fuente: Elaboración propia

Con la determinación de la historia de usuario, se especifican de forma concreta otras necesidades que serán desarrolladas en el futuro.

CASO DE USO Gestionar usuarios

Figura 3.1 Gestionar usuario



Fuente: Elaboración propia

Para una mejor interpretación del requerimiento, es necesario formalizarlo en un diagrama de historia de usuario que se presenta a continuación para la administración de usuarios del sistema los que son de dos tipos: usuarios administradores del sistema y los usuarios operadores del sistema como son los médicos.

Tabla 3.2 H.U. Gestionar usuarios

Historia de Usuario Nº 2			
Nombre Historia de Usuario		Gestión de usuarios	
Usuario	Administrador	**Iteración Asignada**	1
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Agregar nuevo usuario. Editar datos de usuario Borrar registro de usuario Listar todos los registros de los usuarios.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	Este proceso está orientado a trabajar con los datos de los usuarios que se harán cargo de la aplicación. Son los usuarios los que tienen acceso a todos los módulos referidos a médicos, pacientes, consultas, diagnóstico, medicación, consultas y reportes en general.		
Observaciones	Se desarrollaron los módulos sin retraso alguno. Se utilizó un módulo de encriptación de datos para el almacenamiento de la contraseña usando funciones propias del lenguaje de programación PHP: MD5.		

Fuente: Elaboración propia

HISTORIA DE USO Gestionar médicos

Un médico es un profesional que practica la medicina y que intenta mantener y recuperar la salud humana mediante el estudio, el diagnóstico y el tratamiento de la enfermedad o lesión del paciente.

La institución cuenta al menos con cuatro médicos que responden a diversas especialidades y un plantel de al menos 2 enfermeras y personal de apoyo que integran el equipo médico del Centro Medico.

Tabla 3.3 H.U. Gestión de médicos

Historia de Usuario Nº 4			
Nombre Historia de Usuario		Gestión de médicos	
Usuario	Administrador	**Iteración Asignada**	1
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Agregar nuevo médico. Editar datos del médico Dar de baja lógica al médico. Listar todos los registros de los médicos activos e inactivos.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	Este proceso de registro de los médicos requiere que se tenga ya registrado la tabla de especialidades para que al momento de registrar al médico se pueda elegir una especialidad del médico		
Observaciones	Se desarrollaron los módulos sin retraso alguno. Solo se considera una especialidad por médico aunque pueda tener más especialidades por razones administrativas ya que un médico es contratado para dar servicios de solo una especialidad.		

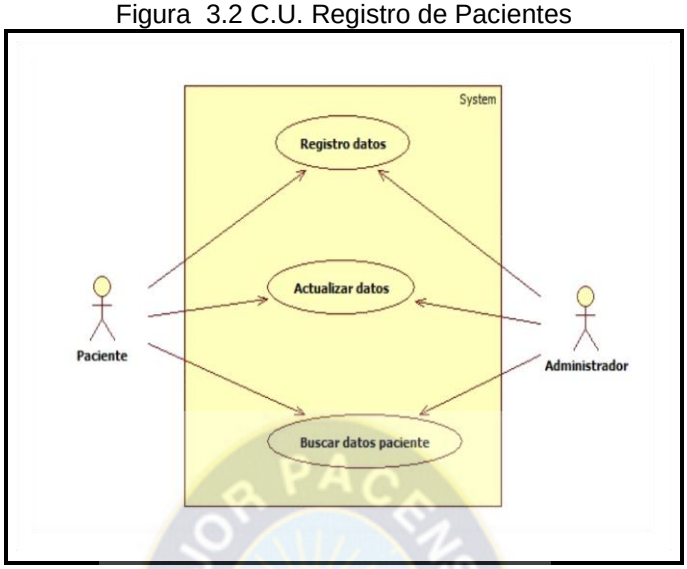
Fuente: Elaboración propia

CASO DE USO Gestionar pacientes

En la medicina y en general en las ciencias de la salud, el paciente es alguien que sufre dolor o malestar (muchas enfermedades causan molestias diversas, y un gran número de pacientes también sufren dolor).

Antes de atender a cualquier consulta de un paciente es necesario registrarlo en el sistema para lo que se procede a establecer el proceso de registro de pacientes.

Figura 3.2 C.U. Registro de Pacientes



Fuente: Elaboración propia

El diagrama anterior refleja que solamente el usuario administrador está autorizado para el llenado de los datos del paciente, identificándose tres procesos: registro de datos de los pacientes, actualización de los datos y finalmente la búsqueda de los datos.

Tabla 3.4 H.U. Registro de datos de los pacientes

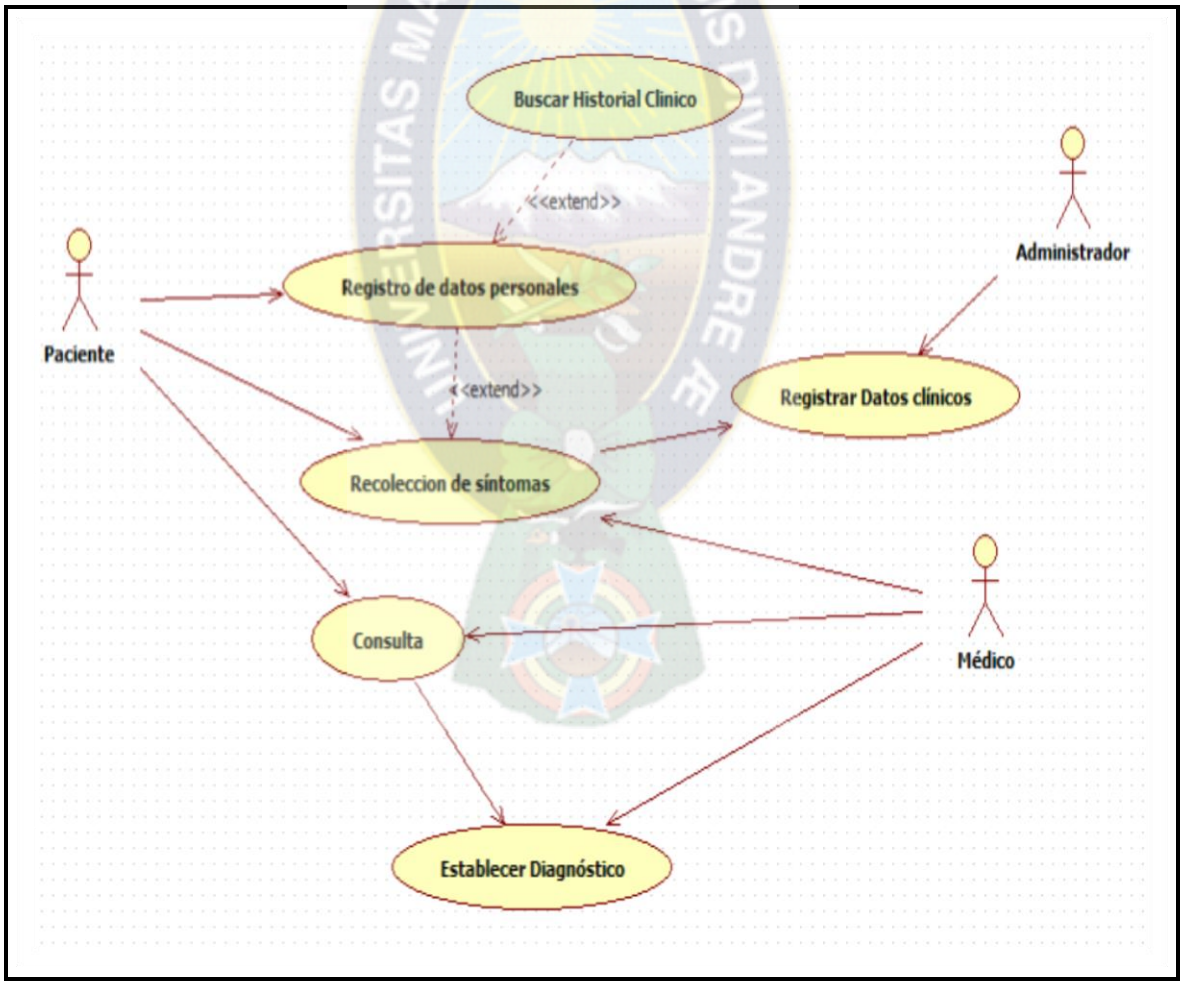
Historia de Usuario Nº 5			
Nombre Historia de Usuario		Gestión de pacientes	
Usuario	Administrador enfermera, médico	**Iteración Asignada**	2
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Agregar nuevo paciente. Editar datos del paciente Dar de baja lógica al paciente. Listar todos los registros de los pacientes activos e inactivos.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	Administrar los datos del paciente que solicita el servicio de tele asistencia médica.		
Observaciones	Se desarrollaron los módulos sin retraso alguno. El proceso de captura de datos se corta si se presentan ausencia de datos en las casillas de los formularios de entrada de datos.		

Fuente: Elaboración propia

CASO DE USO Consulta médica

El episodio de atención se inicia con la razón por la que el paciente consulta al médico. Es un periodo de comunicación directa y franca, en el que el paciente transmite su situación o dudas, donde da sus explicaciones y se relaja, y el médico realiza una escucha activa, creando un clima de serenidad y seguridad, para captar sus necesidades. En la historia clínica se recoge con la expresión literal ofrecida por el enfermo, y sirve de desencadenante de las decisiones y acciones que tomará el profesional médico. A continuación se desarrolla el caso de uso de la consulta reflejada en el Historial clínico.

Figura 3.3 C.U. Consulta médica



Fuente: Elaboración propia

Tabla 3.5 H.U. Consulta médica

Historia de Usuario Nº 6			
Nombre Historia de Usuario		Consulta médica	
Usuario	Administrador	**Iteración Asignada**	2
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Realizar el registro de la consulta médica. Si ya se tiene registrado al paciente, recuperar los datos. Preparar el registro de diagnóstico. Preparar el registro de medicación. Listar todos los registros de los historiales médicos del paciente.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	En esta etapa se prepara el sistema para realizar el registro del Historial clínico del paciente según las normas establecidas por HL7. Este formato será utilizado para la compatibilidad en el traspaso de historiales clínicos entre distintas instituciones de salud.		
Observaciones	Del modelo HL7, solo se consideró las recomendaciones establecidas por EHR System Functional Model: especifica las funcionalidades que debería implementar un sistema de Historia Clínica Electrónica.		

Fuente: Elaboración propia

A continuación las historias de usuario correspondientes al registro del diagnóstico y medicación relacionada con la consulta médica del paciente.

HISTORIA DE USUARIO Diagnóstico

La información almacenada en un EHR puede incluir los antecedentes médicos de un paciente (entre ellos el estado de las vacunas, resultados de pruebas y registros de crecimiento y desarrollo), medicación, información sobre el seguro médico y de facturación y otros datos relacionados con la salud.

En particular, en este sistema se trata de almacenar la información relacionada con el paciente, antecedentes, registro de las enfermedades según las cuatro especialidades y su medicación, con recordatorios mediante el uso de los celulares.

Estas historias de usuario están relacionadas al historial clínico que el sistema genera por cada uno de los pacientes.

Tabla 3.6 H.U. Diagnóstico

Historia de Usuario Nº 7			
Nombre Historia de Usuario		Diagnóstico	
Usuario	Médico	**Iteración Asignada**	3
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Se realiza de manera conjunta a la consulta. Puede ser un diagnóstico nuevo o un diagnóstico de una consulta de seguimiento.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	En esta etapa se prepara el sistema para realizar el registro del Historial clínico del paciente según las normas establecidas por HL7. Este formato será utilizado para la compatibilidad en el traspaso de historiales clínicos entre distintas instituciones de salud.		
Observaciones	Del modelo HL7[10] solo se consideró las recomendaciones establecidas por EHR System Functional Model que especifica las funcionalidades que debería implementar un sistema de Historia Clínica Electrónica.		

Fuente: Elaboración propia

HISTORIA DE USUARIO Medicación

Medicación es la administración de una o más medicinas para curar o prevenir una enfermedad o aliviar un dolor físico.

También se entiende que la medicación es un conjunto de medicinas y medios para curar o prevenir una enfermedad, o para aliviar un dolor físico.

[10] Health Level Seven (HL7) no es un estándar en sí, sino que es un conjunto de estándares cuyo principal objetivo es especificar mensajería para la comunicación de información clínica,

Tabla 3.7 H.U. Medicación

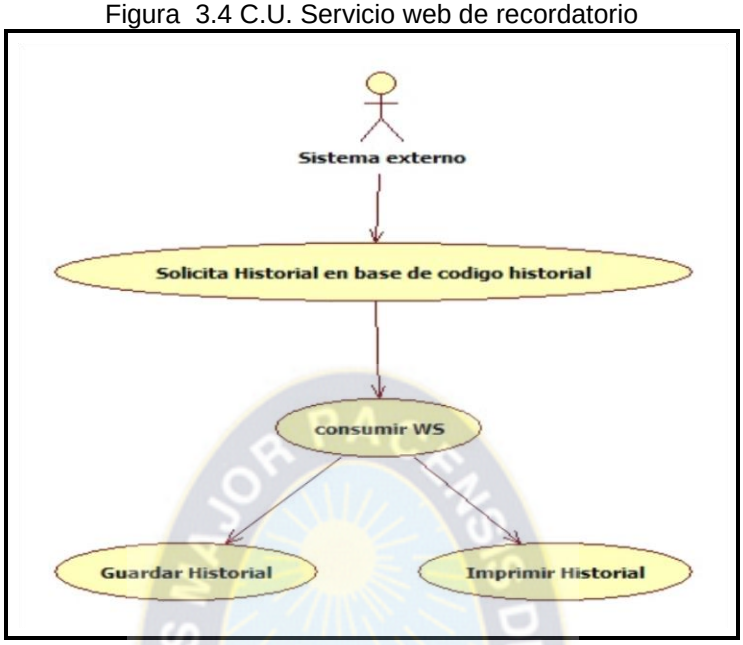
Historia de Usuario Nº 8			
Nombre Historia de Usuario		Medicación	
Usuario	Médico	**Iteración Asignada**	3
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Registro de los horarios de medicación por cada paciente. Envío de mensajes recordatorios al paciente según horario mediante el uso del dispositivo móvil. Recordatorios de medicamentos con similares composiciones. Consulta de medicación desde el dispositivo móvil.
Riesgo en Desarrollo Alta, Media, Baja	Media	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	Por cada paciente puede haber medicaciones compuestas por diversos medicamentos. Se debe establecer el horario en el que deben ser ingeridas por el paciente. Los estados de la medicación serán modificadas de acuerdo a los periodos de dosificación establecidas por los médicos. Cuando el periodo aún se encuentre activo, la aplicación enviará los recordatorios necesarios a los pacientes.		
Observaciones	Se contempla solo el recordatorio de los medicamentos. No se realiza ninguna recomendación en base a edad y peso del paciente.		

Fuente: Elaboración propia

CASO DE USO Recordatorio

Un detalle de la construcción del web services que será construido y como será consumido se presenta a continuación.

Figura 3.4 C.U. Servicio web de recordatorio



Fuente: Elaboración propia

Tabla 3.8 H.U. Servicio web de recordatorio

Historia de Usuario Nº 8			
Nombre Historia de Usuario		Servicio web para recordatorios	
Usuario	Médico	**Iteración Asignada**	3
Prioridad en Negocio: Alta, Media, Baja	Alta	**Puntos estimados**	Consulta a la base de datos sobre consultas cuyo estado sea ACTIVO y entonces según la hora inicial registrada en medicación se procede a extraer en formatos XML los registros de medicación y enviarlos desde el servidor a los dispositivos móviles de los pacientes.
Riesgo en Desarrollo Alta, Media, Baja	Alta	**Puntos reales**	Se lograron desarrollar los módulos especificados según la planificación establecida.
Descripción	Extraer horario de modificación en formato XML		
Observaciones	Se debe usar web services para la extracción de información de la base de datos.		

Fuente: Elaboración propia

3.2.4 PLAN DE ENTREGAS

Es una planificación donde los desarrolladores y clientes establecen:

- Tiempos de implementación ideales de las historias de usuario.
- Asignar una prioridad a cada historia de usuario.

Tabla 3.9 Cronograma de Plan de Entregas

Plan de entregas			
Nombre del proyecto: Sistema de Control y Gestión de Historiales Clínicos			
Fecha de reunión de planificación:	14 de Septiembre 2015		
Nombre de documentador:	Melissa Centellas		
Entrega nº:	1		
# de historia	Título	Prioridad	Fecha inicio
1	Establecimiento de requerimiento funcionales	Alta	21/Septiembre/2015
2	Subsistema de usuarios	Media	22/Septiembre/2015
3	Administración de pacientes	Alta	24/Septiembre/2015
4	Administración de Médicos	Alta	1/Octubre/2015
5	Análisis de consultas	Alta	8/ Octubre /2015
6	Subsistema de consultas	Alta	8/ Octubre /2015
7	Pruebas de interfaces para la explotación	Media	15/ Octubre /2015
8	Módulo de reportes	Media	19/ Octubre /2015
Información de aprobación del plan			
Melissa Centellas Coarite	Gerente Centro Medico		
Firma del entrenador (coach)	Firma del cliente		

Fuente: Elaboración propia

3.2.5 ITERACIONES

3.2.5.1 PRIMERA ITERACIÓN

En esta iteración se desarrollan las tareas de ingeniería y pruebas de aceptación para cada una de las historias de usuarios definidas, Cabe mencionar que de las 21 tareas de las iteraciones, se muestra 12 tareas.

Tabla 3.10 Tarea 1

Número de tarea: **1**	Número de historia: **1**
Nombre de tarea: Diseño de interfaces	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrollara las interfaces y menús que son de fácil manejo para los usuarios administradores del sistema y otra interfaz para los clientes.	

Fuente: Elaboración propia

Tabla 3.11 Tarea 2

Número de tarea: **2**	Número de historia: 4
Nombre de tarea: Registro de pacientes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrolla el Registro de los pacientes con los siguientes campos: Nombre, Paterno, Celular, Teléfono, Correo Electrónico, Fecha de Nacimiento, Sexo, Usuario, Contraseña. Obligatorios: Nombre del Paciente, Fecha de nacimiento, Sexo.	

Fuente: Elaboración propia

Tabla 3.12 Tarea 3

Número de tarea: **3**	Número de historia: **4**
Nombre de tarea: **Editar paciente**	
Tipo de tarea: **Desarrollo**	**Puntos estimados:** 0.3
Fecha inicio: **01/09/2015**	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrolla el módulo Editar paciente en el que se editan los datos de los pacientes registrados y se guardan los cambios en la base de datos del sistema.	

Fuente: Elaboración propia

Tabla 3.13 Tarea 4

Número de tarea: **4**	Número de historia: **5**
Nombre de tarea: Registro de recetas médicas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrolla el módulo Registro de Recetas médicas con los siguientes datos: En pedidos locales: - Registro de la secretaría de salud - Cedula profesional del médico - Nombre del médico y en su caso la especialidad - Nombre del paciente - Fecha de la expedición de la receta - Edad del enfermo - Peso del enfermo (opcional) - Talla del enfermo (opcional) - Temperatura - Tipo de medicamento y la sustancia activa del mismo.	

Fuente: Elaboración propia

Tabla 3.14 Tarea 5

Número de tarea: **5**	Número de historia: **6**
Nombre de tarea: Cambiar estado consulta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrolla las funciones para cambiar el estado de la consulta ya que puede haber no concluido y el paciente tenga que volver para una nueva consulta.	

Fuente: Elaboración propia

Tabla 3.15 Tarea 6

Número de tarea: **6**	Número de historia: **11**
Nombre de tarea: Reporte de pacientes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. Se desarrolla los reportes relacionados con los pacientes ya sean por tipo de síntomas, enfermedades, temporadas, quienes concluyeron su tratamiento, quienes no.	

Fuente: Elaboración propia

Tabla 3.16 Tarea 7

Número de tarea: **7**	Número de historia: **11**
Nombre de tarea: Recordatorio de recetas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 01/09/2015	**Fecha fin:** 07/09/2015
Programador Responsable: Melissa Centellas	
Descripción. En esta tarea se desarrolla los recordatorios que el sistema debe mandar a los pacientes cuando el tiempo de tomar una pastilla se cumple. Este recordatorio está en función de la receta médica permitiendo que el paciente siga su tratamiento sin demoras y así lograr una mejor recuperación de su enfermedad.	

Fuente: Elaboración propia

3.2.5.2 SEGUNDA ITERACIÓN

En esta iteración se desarrollan las historias de usuario, tareas de ingeniería y pruebas de aceptación correspondientes a la segunda iteración. Se realizaron correcciones de algunas historias de usuario como también se adicionaron nuevas funcionalidades al sistema.

Tabla 3.17 Historia de Usuario: Corrección y Mejora del módulo paciente

Número de Tarea: **8**	Usuario: **Administrador del sistema**
Nombre Historia: Corrección y Mejora del módulo Paciente	
Prioridad en negocio: Alta	**Prioridad en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador Responsable: Melissa Centellas	
Descripción: El sistema debe poder ubicar al paciente y enviar notificaciones de que se realiza el seguimiento al tratamiento médico desde su teléfono celular.	

Fuente: Elaboración propia

Tabla 3.18 Historia de Usuario: Corrección y Mejora del módulo consulta

Número de Tarea: **9**	Usuario: **Administrador del sistema**
Nombre Historia: Corrección y Mejora del módulo consulta	
Prioridad en negocio: Alta	**Prioridad en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador Responsable: Melissa Centellas	
Descripción: En esta tarea se mejora los siguientes puntos del módulo pacientes: • Funcionalidades de confirmación del tratamiento. • Se envía notificaciones a los correos del paciente para que tengan conocimiento de que sus datos fueron registrados en el sistema.	

Fuente: Elaboración propia

3.2.5.3 TERCERA ITERACIÓN

En esta iteración se desarrollan las historias de usuario, tareas de ingeniería y pruebas de aceptación correspondientes a la tercera iteración. Se crearon historias de usuario para brindar una mejor funcionalidad a los administradores.

Tabla 3.19 Historia de Usuario: Pacientes frecuentes y no frecuentes

Número de Tarea: **10**	Usuario: **Administrador del sistema**
Nombre Historia: Pacientes frecuentes y no frecuentes	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: Melissa Centellas	
Descripción: Se debe poder saber si un paciente es frecuente o no, se considera frecuente si su cumple su tratamiento de manera correcta.	

Fuente: Elaboración propia

Tabla 3.20 Historia de Usuario: Ayuda online de información

Número de Tarea: **11**	Usuario: **Administrador del Sistema, paciente, usuario.**
Nombre Historia: Ayuda online	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: Melissa Centellas	
Descripción: Se debe contar con un formulario para el paciente en que pueda contactarse con la empresa si necesita ayuda para usar el sistema o una sugerencia, y el Centro Médico pueda responder a su mensaje todo mediante vía web.	

Fuente: Elaboración propia

Tabla 3.21 Historia de Usuario: Respaldos de la Base de datos

Número de Tarea: 12	Usuario: **Administrador del sistema.**
Nombre Historia: Respaldos de la Base de datos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: Melissa Centellas	
Descripción: El sistema debe poder sacar respaldos de la base de datos del sistema para poder contar con copias de seguridad.	

Fuente: Elaboración propia

3.2.6 VELOCIDAD DEL PROYECTO

El plan de entregas desarrollado establece que el periodo de trabajo es de aproximadamente cinco semanas, donde cada semana está considerado con un periodo de siete días, realmente son cinco los días de trabajo hábiles.

Se consideran tres iteraciones que contemplan el desarrollo del sistema con tiempos distintos para cada iteración. Los resultados de los datos establecidos en las iteraciones, se muestran en la tabla conformada por las reuniones que se realizaron al inicio del proceso XP.

Tabla 3.22 Velocidad de Desarrollo del Proyecto

Actividad	Iteración 1	Iteración 2	Iteración 3
Establecimiento de requerimientos	2	2	2
Subsistema de usuarios	3	2	
Administración de pacientes	2	2	2
Administración de médicos		3	2
Subsistema de Análisis de datos		2	5
Subsistema de Consultas		3	3
Módulo de reportes	1	4	5

Fuente: Elaboración propia

3.3 FASE II. DISEÑO

3.3.1 TARJETAS CRC

Las tarjetas de Clases, responsabilidad y colaboración se usan para ayudar a enfocar el desarrollo del código del programa orientado a objetos.

Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

A continuación las tablas siguientes reflejan los casos identificados para el establecimiento de acciones que deben realizarse para implementar el sistema.

Tabla 3.23: Tarjeta CRC 1

Nombre de la Clase: Ingreso al sistema	
Responsabilidades	Colaboradores
Desplegar pantalla de ingreso.	Base de datos
Desplegar opciones de acceso	Sistema inicial
Pasar control al sistema.	

Fuente: Elaboración propia

Tabla 3.24: Tarjeta CRC 2

Nombre de la Clase: Registro de usuarios	
Responsabilidades	Colaboradores
Recoger datos del usuario	Usuario
Almacenar datos en la Base de datos.	Base de datos

Fuente: Elaboración propia

Tabla 3.25: Tarjeta CRC 3

Nombre de la Clase: Registro de paciente	
Responsabilidades	Colaboradores
Recoger datos del paciente mediante la interfaz del dispositivo móvil	Paciente
Almacenar datos en la Base de datos.	Base de datos

Fuente: Elaboración propia

Tabla 3.26: Tarjeta CRC 4

Nombre de la Clase: Consulta	
Responsabilidades	Colaboradores
Desplegar pantalla de consulta.	Base de datos
Desplegar opciones de registro de historial	Sistema inicial
Pasar control al sistema.	
Registrar Medicación.	

Fuente: Elaboración propia

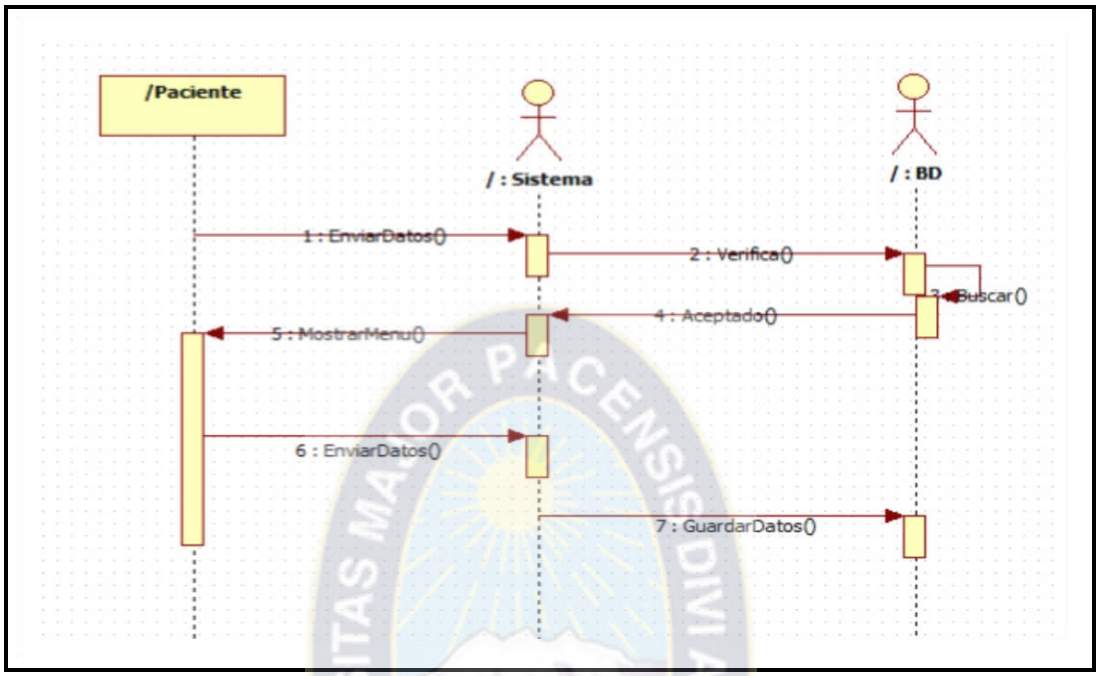
3.3.1 DIAGRAMA DE SECUENCIAS

Los diagramas muestran una interacción ordenada según la secuencia temporal de eventos del sistema de Control y Gestión de Historiales Clínicos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

A continuación se detallan los diagramas.

3.3.1.1 Registro del Paciente

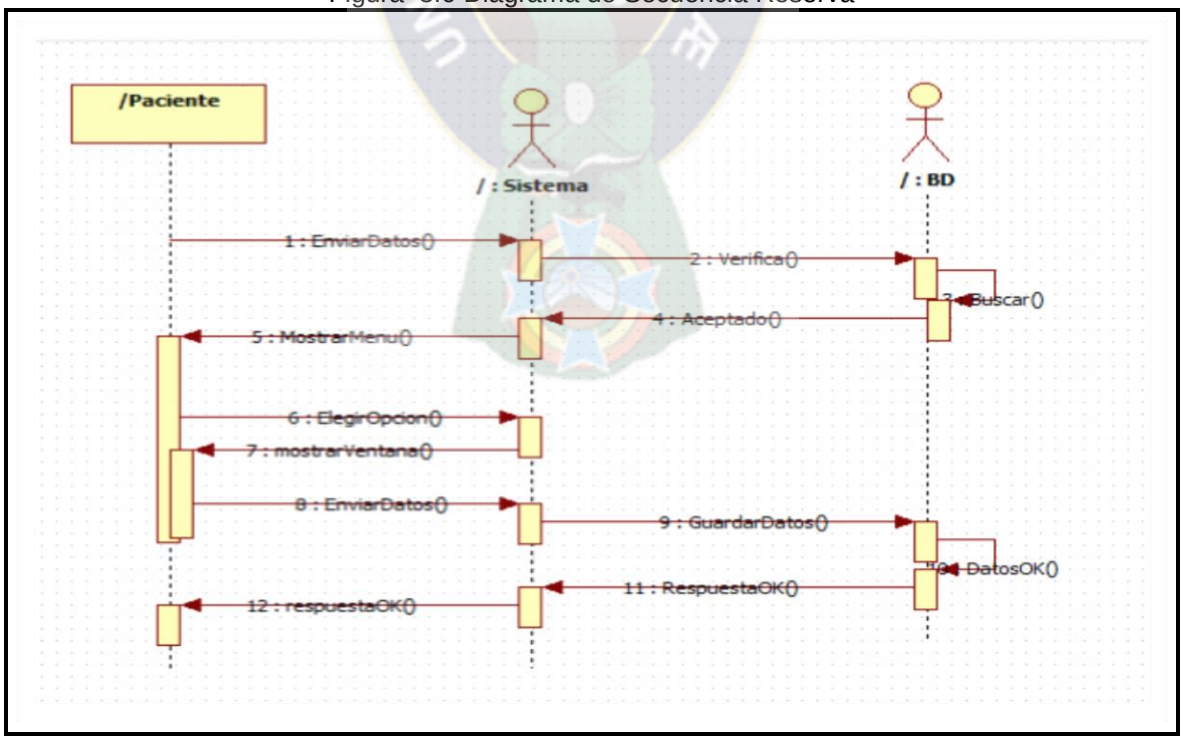
Figura 3.5 Diagrama de Secuencia Registro del Paciente



Fuente: Elaboración propia

3.3.1.2 Reserva

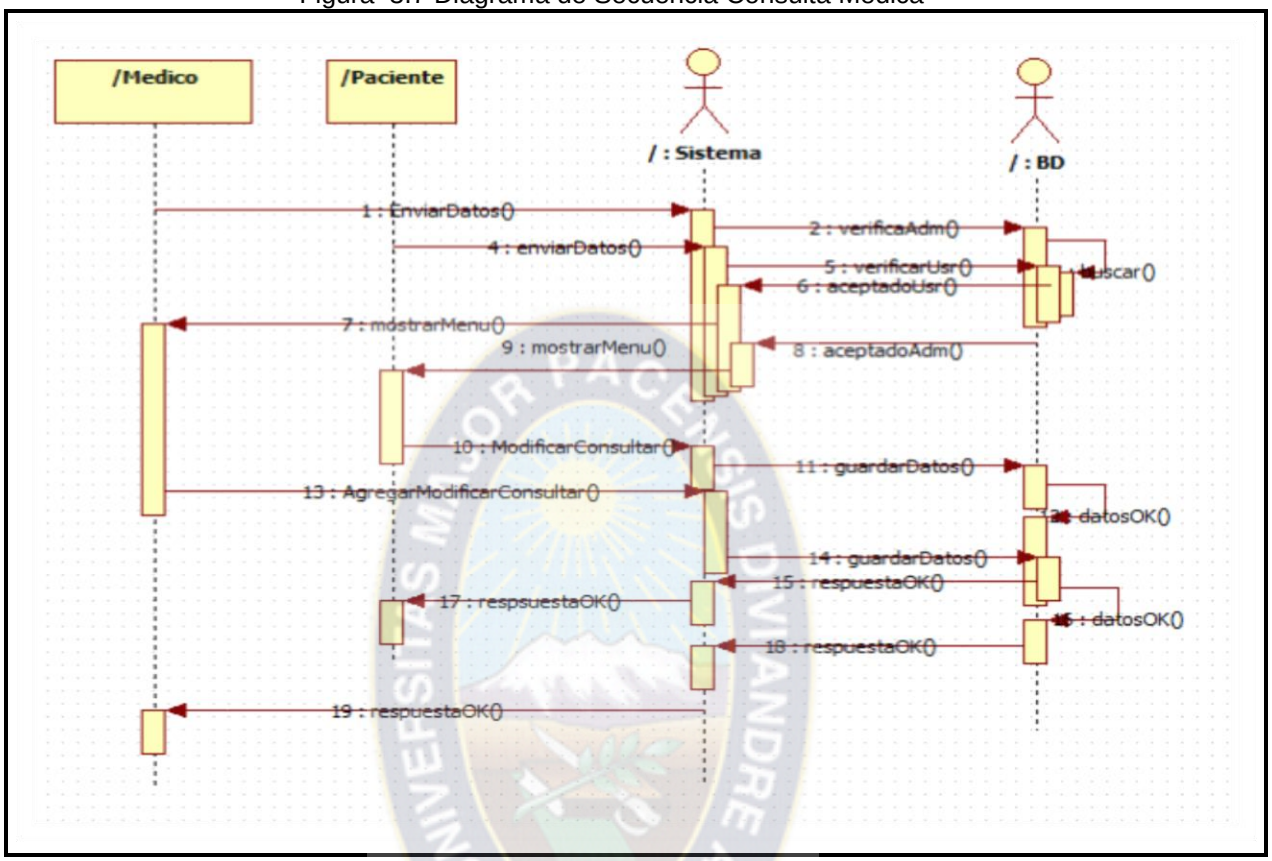
Figura 3.6 Diagrama de Secuencia Reserva



Fuente: Elaboración propia

3.3.1.3 Consulta médica

Figura 3.7 Diagrama de Secuencia Consulta Médica

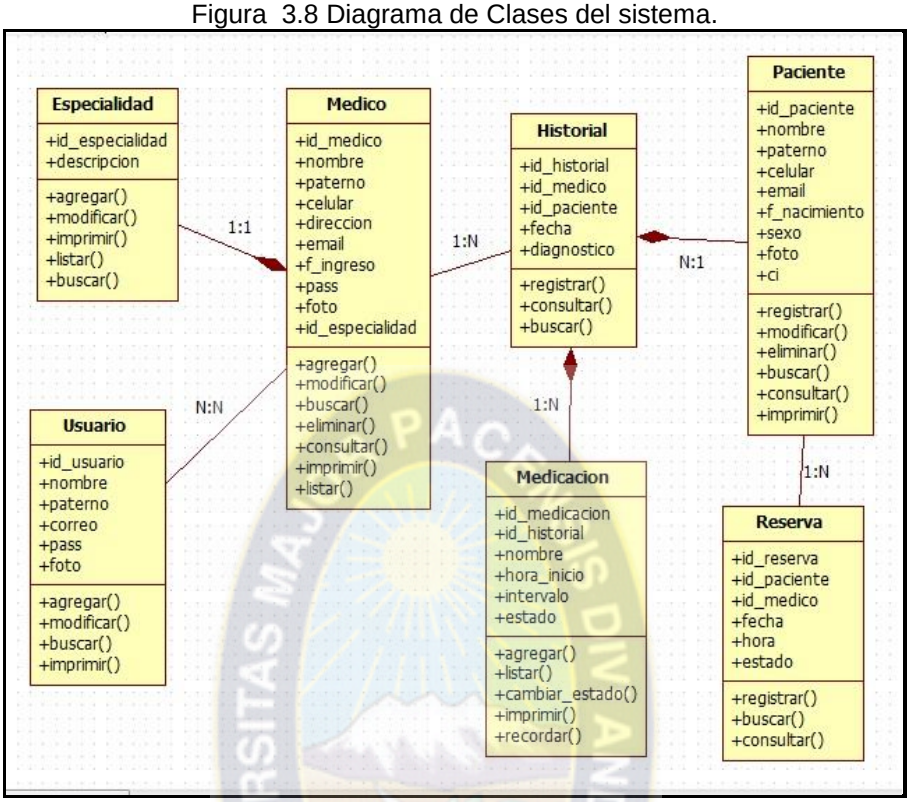


Fuente: Elaboración propia

3.3.2 DIAGRAMA DE CLASES

A continuación se realiza el diseño del diagrama de clases propuesto para el sistema, que incorpora los diferentes ámbitos donde se requiere gestionar los datos del paciente y el equipamiento que le proporciona servicio.

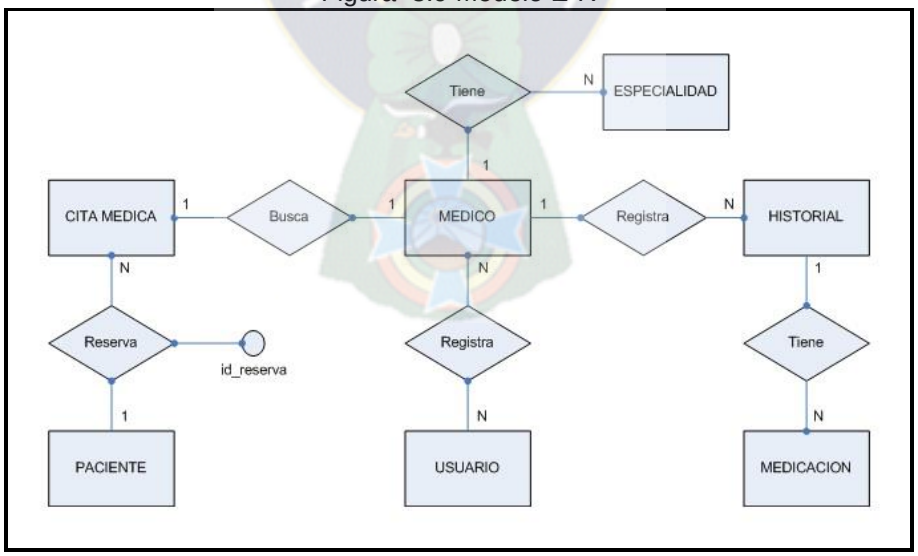
Figura 3.8 Diagrama de Clases del sistema.



Fuente: Elaboración propia

3.3.3 MODELO ENTIDAD-RELACIÓN

Figura 3.9 Modelo E-R

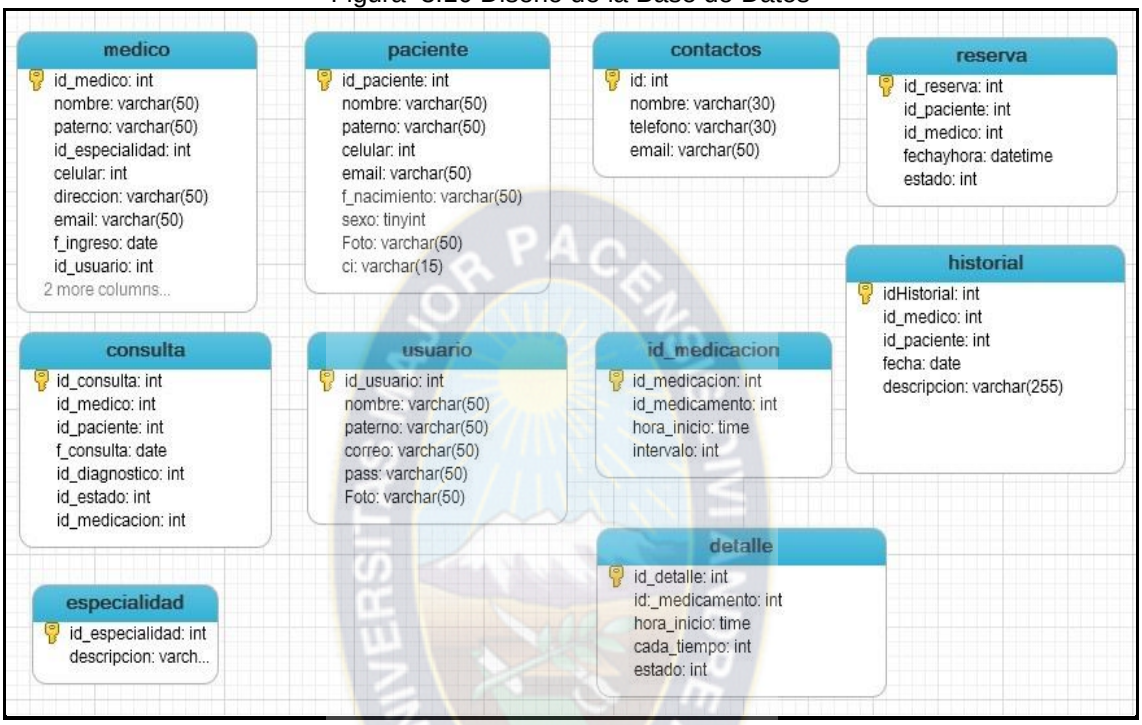


Fuente: Elaboración propia

3.3.4 DISEÑO DE LA BASE DE DATOS

La Figura siguiente muestra la estructura física de los datos del Sistema que son almacenados en la base de datos.

Figura 3.10 Diseño de la Base de Datos



Fuente: Elaboración propia

3.3.5 DISEÑOS DE INTERFACES

Después de haber realizado las iteraciones y diagramas UML, el cual provee una imagen de la estructura del Sistema, el diseño no estaría completo sin el diseño de la interfaz del sistema. A continuación se presentan algunas pantallas del funcionamiento del Sistema.

Ingreso al Sistema, el diseño del sistema contempló la seguridad del mismo, restringiendo niveles de acceso para diferentes tipos de usuarios, teniendo la administración completa del sistema, únicamente el usuario Administrador del sistema. Por lo tanto si el usuario no es autenticado, es rechazado en el intento del acceso al mismo.

Figura 3.11 Ventana de Ingreso al Sistema

Fuente: Elaboración propia

La pantalla principal del sistema presenta una ventana de ingreso que solicita el login y contraseña del usuario, datos obligatorios para ingresar al sistema.

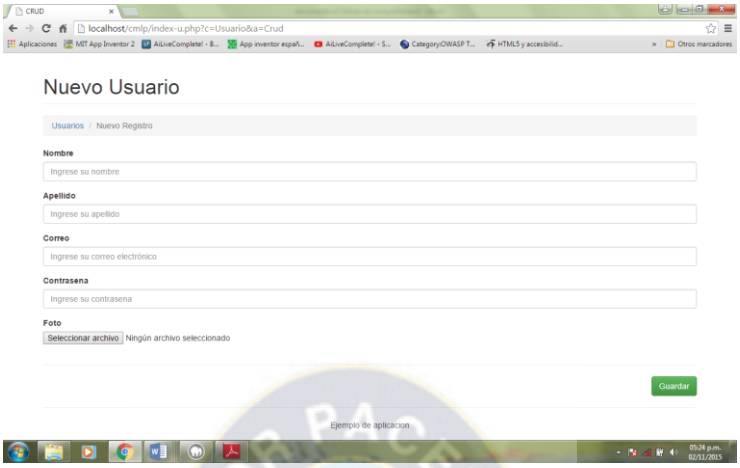
El menú principal muestra las opciones correspondientes al manejo del sistema. Se dispone de un menú con cuatro opciones:

Figura 3.12 Despliegue del menú principal del Sistema

Fuente: Elaboración propia

A partir de esta ventana, se puede iniciar el ingreso de los datos en principio de los usuarios, médicos profesionales y finalmente de los pacientes del sistema que harán uso del sistema.

Figura 3.13 Pantalla de inicio de Ingreso de datos de un Usuario del sistema

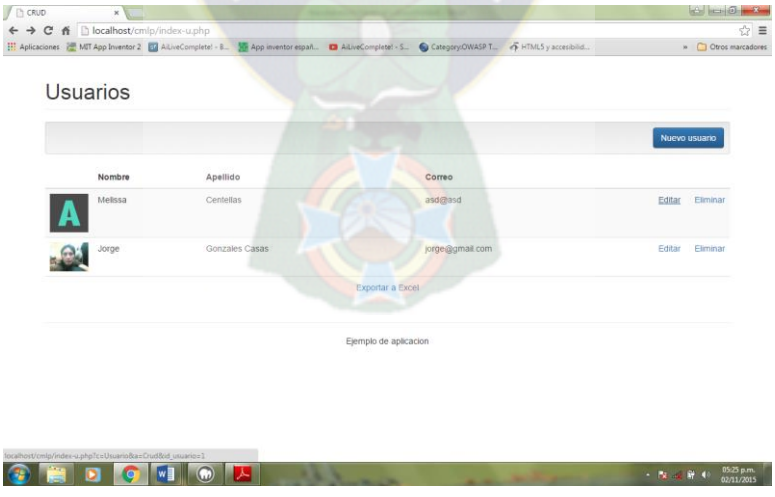


Fuente: Elaboración propia

La figura 3.13 muestra el formulario de captura de datos de usuarios del sistema y de los profesionales médicos que serán encargados de dar respuesta a las consultas de los pacientes que interactúen con el mismo.

Una vez registrados los datos, el sistema brinda la posibilidad de realizar los cambios de los datos registrados para brindar una mayor seguridad.

Figura 3.14 Cambiar datos del Profesional médico

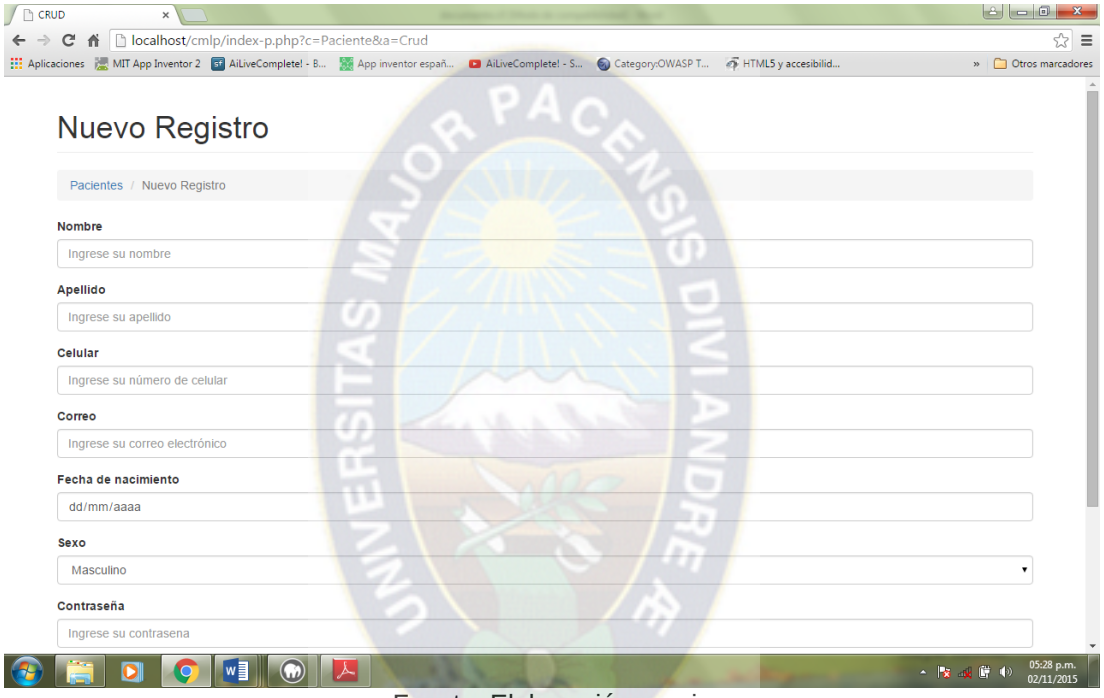


Fuente: Elaboración propia

Para trabajar con el sistema, después de registrar a los usuarios administradores y profesionales que apoyan el funcionamiento del sistema, se debe registrar los datos de los pacientes que desean ser atendidos.

La captura de los datos de los pacientes es un subsistema que incluye tanto los datos personales como la generación de los historiales clínicos del mismo.

Figura 3.15 Ventana de Captura de datos del paciente



Fuente: Elaboración propia

Una vez introducidos los datos requeridos el en formulario anterior, el usuario administrador debe realizar su confirmación haciendo clic en el botón Guardar.

Una vez ingresado los datos de los usuarios administradores del sistema, del personal médico profesional y de los pacientes, es necesario registrar el historial clínico de los mismos.

Figura 3.16 Historial clínico del paciente

Fuente: Elaboración propia

El ingreso de pacientes correspondiente a la aplicación móvil, se muestra en la siguiente figura:

Figura 3.17 Ingreso a la Aplicación Móvil

Fuente: Elaboración propia

Una vez ingresado al sistema, se tiene el siguiente menú:

Figura 3.18 Menú a la Aplicación Móvil

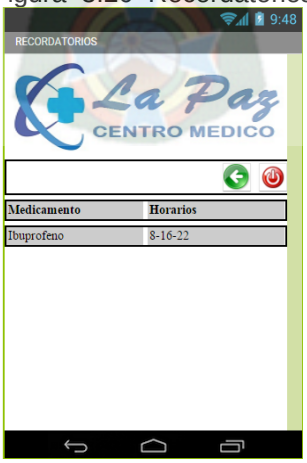
Fuente: Elaboración propia

A partir de este menú podemos realizar una reserva o consultar la medicación.

Figura 3.19 Reserva Médica

Fuente: Elaboración propia

Figura 3.20 Recordatorios



Fuente: Elaboración propia

3.4 FASE III. DESARROLLO

3.4.1 DISPONIBILIDAD DEL CLIENTE

Respecto a la disponibilidad del cliente, dado que se utilizó la metodología XP, de principio se estableció que la Secretaria encargada de las reservas y atención en el centro médico, trabajó de manera permanente con el equipo de desarrollo del software, logrando de manera inmediata resolver contingencias referidas a la dudas sobre los manejos y proceso de los formularios médicos.

El cliente entonces apoyó de manera permanente al equipo de desarrollo.

3.4.2 PROGRAMACIÓN POR PAREJAS

La programación en parejas en principio no fue del total agrado de la institución clínica dado que se requería un par de programadores para realizar la misma tarea, pero el lograr productos bien estructurados, consistentes y casi libre de errores, la institución clínica dio paso a esta forma de programar.

Entonces se dice que el código producto de 2 personas trabajando al mismo tiempo en la misma máquina contiene menos errores que el de 1 sola persona haciendo lo mismo, y en mérito a la verdad se logró un mejor producto software.

Sin embargo para acelerar el desarrollo de la aplicación, se trabajó de manera individual los módulos CRUD relacionados a Paciente, Médico, Especialidad. En cambio para los módulos Reserva, Consulta e Historial clínico se trabajó en parejas dado que son en éstos módulos los más importantes y complejos en su desarrollo.

3.5 FASE IV. IMPLANTACION Y PRUEBAS

3.5.1 PLAN DE IMPLANTACIÓN DEL SISTEMA

En cuanto a la programación de actividades se plantean las actividades de implantación del sistema, con la única diferencia que se le incluyen los tiempos de realización de las actividades, para llevar a cabo cada actividad, se utiliza para ello el Cuadro de Actividades expresado mediante un Diagrama de Gantt como se muestra en el cuadro siguiente.

Para cada actividad se le asigna un tiempo promedio normal (días) que requiere para su realización, además se asigna la secuencia de actividades en su orden de ejecución, otro punto importante que se incluye son las actividades precedentes a cada actividad.

Tabla 3.27 Cronograma de Trabajo

Actividad	Cantidad
Contratación de Personal	3 días
Divulgación de Actividades	1 día
Adquisición de materiales y equipos	1 día
Desarrollo del Sistema	30 días
Pruebas	20 días
Retroalimentación	5 días
Entrega del Proyecto	1 día

Fuente: Elaboración propia

3.5.2 PRUEBAS

La ejecución del sistema fue realizado por un periodo de 30 días a partir de la conclusión del trabajo.

Los primeros cinco días se llevó a cabo el llenado de los registros de los datos al sistema. Es en esta etapa se lograron detectar y solucionar un conjunto de fallas lógicas como físicas del sistema.

Se dio soluciones a cada uno de los problemas detectados logrando mejorar el sistema y hacerlo más robusto frente a fallas que se puedan presentar adelante.

Por otra parte la segunda fase de pruebas del sistema estuvo dirigido a la obtención de los reportes y consultas de forma que el sistema entregue los historiales clínicos tal y como estableció en la identificación de los requerimientos funcionales del sistema.

3.5.2.1 PRUEBAS DE CAJA BLANCA

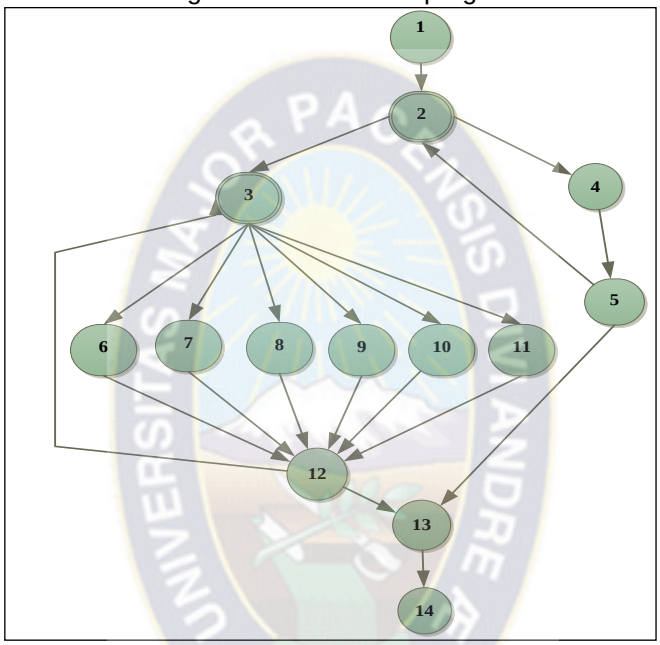
Esta prueba se orienta al cálculo de las regiones que deben ser consideradas como partes independientes del sistema, y estableciendo cuáles con las entradas para las

que se ejecutan cada una de las regiones, asegurando así que cada región se ejecuta al menos una vez.

De forma general, se debe seguir:

- Emplear el diseño del sistema para elaborar el grafo del programa.

Figura 3.21 Grafo del programa



Fuente: Elaboración propia

Dónde:

1. Inicio del sistema
2. Menú principal
3. Módulo Administración
4. Información del Centro Médico la Paz.
5. Mostrar información del Centro
6. Registro del Paciente
7. Registro de Reservas
8. Consultas
9. Consulta médica
10. Recordatorios
11. Ayuda on line
12. Fin ciclo Administrar
13. Fin ciclo Sistema
14. Fin del sistema

Analizado el grafo generado a partir de las características del sistema, ahora se procede a determinar la complejidad ciclomática del grafo mediante:

V (G) = A – N + 2

Dónde:
A = 21 (Aristas)
N = 14 (Nodos)
Por tanto V (G) = 21 – 14 +2 = 9

- Determinar el conjunto básico de caminos linealmente independientes.

Los caminos que deben ser probados dadas ciertas variables son 9. Estos caminos son los siguientes:

Camino 1: 1-2-3-6-12-13-14
Camino 2: 1-2-3-7-12-13-14
Camino 3: 1-2-3-8-12-13-14
Camino 4: 1-2-3-9-12-13-14
Camino 5: 1-2-3-10-12-13-14
Camino 6: 1-2-3-11-12-13-14
Camino 7: 1-2-4-5-13-14
Camino 8: 1-2-4-13-14
Camino 9: 1-2-3-13-14

- Preparar los casos de prueba para forzar la ejecución de cada camino.

Esta última condición establece que para la ejecución de ciertos caminos, se deben establecer las condiciones en las que al menos se ejecuta los nodos establecidos en el camino.

Camino 1: Se ejecuta cuando se realizan las reservas de los pacientes.
Camino 2: Este módulo se ejecuta en el instante en que el paciente es evaluado médicamente.
Camino 3: Cuando se realizan consultas sobre las consultas

Camino 4: Son los procesos relacionados con las consultas y la respectiva medicación de cada paciente.
Camino 5: Básicamente involucra los procesos de altas bajas y modificaciones de los médicos, usuarios, especialidades y otros.
Camino 6: Se ejecuta cuando el paciente requiere ayuda en línea, de forma que pueda tener una mayor información sobre su medicación.
Camino 7: Se ejecuta cuando el paciente que ingresa el sistema no tiene los permisos necesarios para ingresar al módulo de reservas o consultas. Solo es informativo.
Camino 8: El paciente se registra en el sistema y concluye.
Camino 9: El usuario del centro ingresa a Administrar y concluye.

3.5.2.2 PRUEBAS DE CAJA NEGRA

Estas pruebas se refieren al hecho de que el sistema debe ejecutar procesos solamente cuando los datos ingresados son relativamente coherentes, de lo contrario deben dar al usuario un informe completo del porque no se ejecuta la aplicación requerida.

Se diseñaron un caso de prueba de caja negra basado en la partición equivalente, un caso de prueba de caja negra basado en los valores límites y un caso de prueba de interfaz de usuario gráfica para el caso de uso INICIAR SESIÓN del sistema:

El flujo de eventos del caso de uso **INICIAR SESIÓN** del sistema: ENTRAR

EN EL SISTEMA

El usuario escribe su nombre y el password (Autenticación).

El sistema comprueba que existe una cuenta con ese nombre y password y, si es así, se le da permiso para entrar en el sistema.

Si existe el nombre de usuario pero el password es incorrecto permite reintroducir el password hasta un máximo de tres veces.

Requisitos no funcionales del caso de uso ENTRAR EN EL SISTEMA:

1. El password no debe ser visible.
2. Los nombres de usuarios sólo pueden contener letras y como mínimo diez letras

SOLUCION:

CASO DE PRUEBA DE CAJA NEGRA BASADO EN LA PARTICIÓN EQUIVALENTE

Se registró un usuario válido: admin con el password admin. Se considera bastante limitado un nombre de usuario de solo 7 caracteres, no obstante la clave si acepta hasta 18 caracteres.

1) Entrada: usuario “admin” password: “admin”.
Salida: dar paso (usuario y password válidos; se encuentran registrados en la base de datos).
2) Entrada: usuario “admin44” password “admin”.
Salida: no dar paso (usuario no válido por contener caracteres “no letras” y más de siete).
3) Entrada: usuario “admin” password “admin”.
Salida: no dar paso (usuario no válido por contener menos de siete letras).
4) Entrada: usuario “gvel” password “admin”.
Salida: no dar paso (usuario no válido por contener menos de siete letras, y algunas no ser letras).
5) Entrada: usuario “ ” password “admin”.
Salida: no dar paso (usuario “en blanco”).
6) Entrada: usuario “1234567” password “admin”.
Salida: no dar paso (usuario no válido por no contener ni una sola letra, aunque sí los siete caracteres).
7) Entrada: usuario “admin”, password: “ ”.

Salida: no dar paso (usuario válido pero password no, usuario se encuentra registrado en la base de datos).

CASO DE PRUEBA DE CAJA NEGRA BASADO EN LOS VALORES LÍMITE

1) Entrada: usuario “admin” password: “malPassw”.

Salida: no obtener paso intentar otra vez.

Usuario “admin” password: “malPassw”.

Salida: no obtener paso intentar otra vez.

Usuario “admin” password: “malPassw”.

Salida: no obtener paso y comprobar que se bloquea // Intenta 4 veces (usuario válido pero password incorrecto).

2) Entrada: usuario “admin” password: “malPassw”.

Salida: no obtener paso intentar otra vez.

Usuario “admin” password: “malPassw”.

Salida: no obtener paso intentar otra vez.

Usuario “admin” password: “admin”.

Salida: obtener paso // Intenta 3 veces (Usuario válido pero password incorrecto).

3) Entrada: usuario “mcesped” password: “malPassw” => no obtener paso intentar otra vez: usuario “mcesped” password: “malPassw” => no obtener paso intentar otra vez: usuario “mcesped” password: “malPassw”.

Salida: no obtener paso pero comprobar que no se bloquea // Intenta 3 veces (Usuario y password incorrectos).

4) Entrada: usuario “admin” password: “malPassw” => no obtener paso intentar otra vez: usuario “admin” password: “admin”.

Salida: obtener paso // Intenta 2 veces (usuario “admin” válido y password “admin” válido).

CASO DE PRUEBA DE INTERFAZ GRÁFICA DE USUARIO:

1) Entrada: usuario “melissa” password: “malPassw”.

Salida: comprobar passw. NO visible.

2) Entrada: pulsar botón “Acceder”.

Salida: comprobar que “responde”.

3) Entrada: pulsar icono “minimizar ventana”.

Salida: comprobar que se minimiza.

CASO DE PRUEBA: GESTIONAR USUARIOS

Se diseñaron un caso de prueba de caja negra basado en la partición equivalente, en los valores límites y la interfaz de usuario gráfica, para el caso de uso GESTIONAR USUARIOS del sistema.

El flujo de eventos del caso de uso GESTIONAR USUARIOS del sistema: AGREGAR USUARIO (usuarios responsables del sistema en la clínica).

EVENTOS DEL USUARIO Y DEL SISTEMA:

El usuario escribe su nombre, paterno, materno, correo, password, reescribe el password.

El sistema proporciona un código de usuario generado de forma secuencial y automática.

El sistema comprueba que no exista una cuenta con ese código, nombre y password, si se cumple esa condición, se da permiso para guardar la información en el sistema.

El sistema verifica que el llenado de ambos campos del password sea igualmente escrito para otorgar permiso de guardado.

Requisitos no funcionales del caso de uso AGREGAR USUARIO:

1. El password en ambos casos no debe ser visible y tener como máximo 10 caracteres.

2. Los nombres de usuarios y apellidos sólo pueden contener letras, como máximo 25 letras y mínimo 3.

SOLUCION:

CASO DE PRUEBA DE CAJA NEGRA BASADO EN LA PARTICIÓN EQUIVALENTE

Se registró un usuario de prueba válido: user, apellidos aleatorios, con el password: userclave, con el correo: user@hotmail.com, con el nivel: secundario, Se considera bastante limitado un nombre de usuario de solo 10 caracteres como mínimo permitido, así que para evitar perdida de información se mantuvieron los 25 caracteres.

1) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userclave”.
Salida: dar paso (código, usuario, apellidos, correo, password, repetir password y nivel válidos; se encuentran registrados en la base de datos).
2) Entrada: usuario: “user*12345678”; paterno: “Pérez”; materno: “Ríos”; correo: “user11@hotmail.com”; password: “userclave”; repita password: “userclave”.
Salida: no dar paso (entrada de campo usuario no válido por contener caracteres “no letras”).
3) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user987@hotmail.com”; password: “25”; repita password: “userclave”.
Salida: no dar paso (password no válido por contener menos de 6 letras y no coincidir).
4) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userlave”.
Salida: no dar paso (password no válido por no coincidir ambas claves).
5) Entrada: usuario: “ ”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userlave”.
Salida: no dar paso (usuario “en blanco”).

6) Entrada: usuario: “123”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userclave”.
Salida: no dar paso (usuario no válido por no contener ni una sola letra, aunque sí los tres caracteres).
7) Entrada: usuario: “user5”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userclave”.
Salida: no dar paso (usuario válido pero password no, usuario se encuentra registrado en la base de datos).

CASO DE PRUEBA DE CAJA NEGRA BASADO EN LOS VALORES LÍMITE

1) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userclve”; nivel: “secundario“.
Salida: no obtener paso intentar otra vez.
Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclav”; repita password: “userclave “.
Salida: no obtener paso intentar otra vez.
Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlave”; repita password: “userclave”.
Salida: no obtener paso intentar otra vez. // Intentos 3 veces (usuario válido pero contraseña no coincide en todos los casos).
2) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlve”; repita password: “usercave”.
Salida: no obtener paso intentar otra vez.
Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlave”; repita password: “userclve”.
Salida: no obtener paso intentar otra vez.
Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlave”; repita password: “userclae”.
Salida: no obtener paso intentar otra vez. Salida: no obtener paso y comprobar que se bloquea // Intentos 4 veces (datos usuario válido pero password incorrecto).

3) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlve”; repita password: “usercave”.

Salida: no obtener paso intentar otra vez.

Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlave”; repita password: “userclve”.

Salida: no obtener paso intentar otra vez.

Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlave”; repita password: “userclae”.

Salida: no obtener paso pero comprobar que no se bloquea // Intenta 3 veces (Usuario y password incorrectos).

4) Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userlve”; repita password: “usercave”.

Salida: no obtener paso intentar otra vez.

Entrada: usuario: “user”; paterno: “Pérez”; materno: “Ríos”; correo: “user@hotmail.com”; password: “userclave”; repita password: “userclave”.

Salida: obtener paso // Intenta 2 veces (usuario “user” válido y password “userclave” válido en ambos casos).

CAPITULO IV

4. SEGURIDAD, CALIDAD Y COSTOS

4.1 SEGURIDAD

Se debe asegurar la Confidencialidad, Integridad, y Disponibilidad de los datos que el sistema administra. Se recomienda determinar la seguridad física y la seguridad lógica.

4.1.1 SEGURIDAD FÍSICA

Se recomienda que el usuario administrador del sistema tenga que realizar copias de seguridad comúnmente llamado BACKUPS para asegurar que cuando se presenten fallas físicas de almacenamiento el usuario administrador pueda restaurar los registros originales a partir de la copia realizada anteriormente.

Es recomendable además que las copias de respaldo deban ser almacenados en ubicaciones geográficas diferentes, por lo común en un servidor distinto ya que así se asegura que no existan pérdidas.

Se establece como política la prohibición del ingreso de memorias flash o cualquier otro dispositivo de almacenamiento externo a los equipos que trabajan como servidores locales de base de datos.

Para mantener el equipo de las configuraciones del SETUP, se incluye una contraseña de administrador y solo el encargado podrá acceder a la BIOS. Finalmente se recomienda eliminar puertos seriales y/o paralelos que no se utilicen.

Debe insistirse una vez más que si un potencial intruso tiene acceso al hardware no hay ningún tipo de seguridad lógica inviolable.

4.1.2 SEGURIDAD LÓGICA

El sistema maneja principalmente las contraseñas de acceso al sistema.

Se tiene una contraseña a nivel de la base de datos para que se pueda acceder a la configuración de la base de datos.

Otra seguridad implantada en el sistema es el almacenamiento de las contraseñas de los usuarios utilizando algoritmos de encriptación como el MD5.

Una vista del resultado del algoritmo de encriptación aplicado a la tabla usuarios se muestra en la figura siguiente:

Figura 4.1 Encriptamiento de la contraseña

id_usu	login	tipo	nombre	password
35	roberto	ADMINISTRADOR	ROBERTO	c1bfc188dba59d2681648aa0e6ca8c8e
40	andrea	ADMINISTRADOR	ANDREA	1c42f9c1ca2f65441465b43cd9339d6c
41	x	ADMINISTRADOR	X	9dd4e461268c8034f5c8564e155c67a6

Fuente: Elaboración propia

Cada usuario al ingresar al sistema, la base de datos realiza el seguimiento de las actividades del mismo almacenando las acciones realizadas de forma que puedan ser asumir su responsabilidad de cualquier tarea mal ejecutada.

Se recomienda que el antivirus y firewall deban estar permanentemente actualizados para evitar cualquier daño lógico de programas intrusos.

4.2 CALIDAD DEL SOFTWARE ISO 9126

Para determinar si un sistema cumple con normas de calidad, es necesario recurrir a ciertas técnicas planteadas por la Ingeniería de software.

Para tal efecto se utilizarán métricas de calidad en conjunción de la obtención de índices y/o factores que varían para diferentes aplicaciones y los usuarios que lo soliciten.

La métrica de punto función se usa como medio para predecir el tamaño del sistema que se va obtener por medio de análisis siendo una medida indirecta de software.

4.2.1 FUNCIONALIDAD

Cuantifica el tamaño y la complejidad del sistema en términos de las funciones de usuario, para este fin se determina cinco características del dominio de información, tomando en cuenta su cantidad.

Número de entradas de usuario

Proporciona datos del sistema para poder realizar distintas operaciones (Altas, bajas y modificaciones) con el objetivo de satisfacer las necesidades primordiales de una aplicación.

Número de salidas de usuario

Las salidas de usuario se refieren a informes por pantalla, mensajes de error brindando información orientada al sistema.

Número de peticiones

Es una entrada interactiva que produce la generación de alguna respuesta de software en forma de salida interactiva.

Numero de Archivos

Es un conjunto lógico de datos que puede ser parte de una gran base de datos o de archivos independientes.

Número de interfaces Externas

Es el número de interfaces legibles por la máquina y que se utilizan para transmitir información a otra máquina.

Tabla 4.1 Cuenta total de entradas y salidas

Grupo	Característica
Entradas de Usuario	Registro de Pacientes Registro de Reservas Registro de Especialidades Registro de Médicos Registro de usuarios del sistema Registro de Consultas
Salidas de Usuario	Reportes de pacientes Reportes de usuarios del sistema Reportes de médicos Reportes de atenciones médicas Recordatorios
Peticiones de usuario	Listado de Consultas Listado de reservas Listado de Medicaciones.
Numero de archivos	Archivos de la base de datos=>7
Interfaces Externas	Backups (Copia de seguridad)

Fuente: Elaboración Propia

Tabla 4.2 Parámetros de medición

Parámetros de medición	Cuenta	Factor de Ponderación			Total
		Simple	Medio	Complejo	
Entradas de usuario	6	3	4	6	24
Salidas de usuario	5	4	5	7	25
Peticiones de usuario	3	3	4	6	12
Archivos	12	7	10	15	120
Interfaces externas	1	5	7	10	7
				Cuenta Total	**188**

Fuente: Elaboración propia

Donde Cuenta Total igual a 188 es la suma de todas las entradas obtenidas.

Valores de ajuste de la complejidad

Tabla 4.3 Valores de Ajuste de complejidad

0	1	2	3	4	5
No influencia	Incidencia	Moderado	Medio	Significativo	Esencial

Fuente: Elaboración propia

Utilizando esta escala de valores de debe cuantificar los siguientes factores:

Tabla 4.4 Ajuste de complejidad del Punto Función

	Factor	Valor
1	Requiere el Sistema de copias de seguridad y de recuperación de datos fiables	5
2	Se requiere de comunicación de datos	5
3	Existen funciones de procesamiento distribuido	3
4	Es crítico el rendimiento	1
5	Se ejecuta el Sistema en un entorno operativo existente	5
6	Requiere el Sistema la entrada de datos de forma interactiva, mostrando múltiple pantallas u operaciones.	4
7	Se actualizan los archivos maestros de forma automática	5
8	Son complejas las entradas, salidas o peticiones de usuario	4
9	Es complejo el procesamiento interno	3
10	Es compleja la utilización del Sistema	3
11	Se ha diseñado el código para ser reutilizable	5
12	Está incluida en el diseño la instalación	4
13	Se ha diseñado para facilitar los cambios y sea de fácil uso para el usuario	4
14	Se diseñó el Sistema para soportar múltiples instalaciones en diferentes departamentos	4
	Σ *Fi*	55

Fuente: Elaboración propia

Donde **Σ *Fi*** es la suma total de los factores de complejidad.

Para calcular puntos

$$PF = \text{Cuenta Total} \times [R(t) + 0.01 \times \Sigma Fi]$$

Dónde: Cuenta Total=Suma de todas las entradas obtenidas

0.01=Error de confiabilidad de Sistema

Σ *Fi*= Suma de factores de complejidad

Cálculos:

PF= 188(0.65+0.01*55)= 225.6

PF Máximo= 188(0.65+0.01*70)=282

Funcionalidad = (*PF*/*PF Máximo*)*100 = (225.6/282)*100= 80

Entonces la funcionalidad del Sistema es de 80%

4.2.2 FIABILIDAD

La fiabilidad se refiere al punto en que se puede esperar que el programa lleve a cabo su función predefinida con la exactitud requerida y está dado por:

1-(número de errores /número de líneas de código)

Tabla 4.5 Resultados de evaluación de ejecución del sistema

Tiempo de servicio	Peticiones realizadas	Fallas encontradas	Probabilidad fallo bajo demanda	Tiempo medio entre fallos
10 hrs	70	4	0.0286	2.5 hrs.
16 hrs	54	2	0.037	8 hrs.
168 hrs.	1500	13	0.0086	12 hrs.

Fuente: Elaboración propia

Fiabilidad = 1 – (6.33 errores / 600 líneas de código)

Fiabilidad = 0.9367*100

Fiabilidad = 93.67 %(Significa que en 94% el sistema es fiable)

4.2.3 FACILIDAD DE MANTENIMIENTO

Es el esfuerzo necesario para localizar y arreglar un error en el programa y está dada por la siguiente ecuación matemática:

= 1 – 0.1* (número medio de días - hombre por corrección)

Facilidad de mantenimiento = 1-0.1 (2 – 1 persona por corrección)

Facilidad de mantenimiento = 0.9 * 100

Facilidad de mantenimiento = 90 %(Significa que en un 90% es fácil de mantener)

4.2.4 EFICIENCIA

Para evaluar este factor de calidad, se consideraron los atributos correspondientes a la característica de eficiencia.

Tabla 4.6 Resultados de evaluación de eficiencia del sistema

CARACTERÍSTICA	INFORMACIÓN
Comprensibilidad del sistema	95
Mecanismos de ayuda y retroalimentación	80
Aspectos de la interfaz	90
Aspectos de exploración	89
Errores	10

Fuente: Elaboración propia

Los resultados obtenidos en base a un cuestionario resuelto tanto por los usuarios administradores así como los profesionales médicos y en menor número por los pacientes, haciendo un total de veinticinco encuestas.

Por tanto el promedio de eficiencia será:

Eficiencia = promedio (95, 80, 90, 89, (100-10)) = 88.8%

Este resultado indica que el Sistema elaborado logra un servicio con una eficiencia del 88.8%, lo que se interpreta como un rendimiento satisfactorio al momento de generar notas de salida, reportes y de ser comprensible en el manejo, ya que estas fueron diseñadas con elementos simples que reducen el tiempo de proceso.

4.2.5 FLEXIBILIDAD

La flexibilidad es el coste de modificación del producto cuando cambian sus especificaciones, y puede ser calculada a partir de la siguiente fórmula:

1-0.05 (número medio de días – hombre por cambio)

Flexibilidad = 1 – 0.05 (2- 1)
Flexibilidad = 0.95 * 100

Flexibilidad = 95%(Significa que en un 95% es flexible el Sistema)

4.3 COSTOS

Entre los distintos métodos de estimación de costes de desarrollo de software, el modelo COCOMO 2 (Constructive Cost Model) desarrollado por Barry M. Boehm, se engloba en el grupo de los modelos algorítmicos que tratan de establecer una

relación matemática que permite estimar el esfuerzo y tiempo requerido para desarrollar un producto.

Por un lado COCOMO define tres modos de desarrollo o tipos de proyectos, para nuestro caso el modelo intermedio será el que usaremos, dado que realiza las estimaciones con bastante precisión.

Así entonces las fórmulas serán las siguientes:

- E = Esfuerzo = a KLDC ^{e} * FAE (persona x mes)
- T = Tiempo de duración del desarrollo = c Esfuerzo ^{d} (meses)
- P= Personal = E/T (personas)

La suma de las líneas de código en cada una de los módulos se desglosa en la siguiente tabla:

Tabla 4.7: Estimación de líneas de código a desarrollar

Módulos	LDC estimadas			
	Optimista	Medio	Pesimista	Esperado
Base de datos (manipulación de objetos)	400	550	600	533
Subsistema de registro de usuarios	550	600	750	617
Subsistema de administración de escenarios.	320	500	650	495
Subsistema de administración de contenidos	350	450	570	453
Elaboración de Mundos Virtuales	450	500	650	517
Administración de ejercicios	560	750	800	727
Subsistema de evaluación	450	750	850	717
Interfaces de usuario- gráficas	420	560	700	560
TOTAL LINEAS DE CODIGO				4618

Fuente: Elaboración propia

Aplicando la fórmula respectiva para hallar la media ponderada (o valor esperado - VE) para estimar las LDC se tiene:

VE = (Sopt+4Sm+Spes) / 6

Dónde: Sopt es el valor optimista.

Sm es el valor medio.

Spes es el valor pesimista.

Si en base a estos datos tenemos un LDC total de 4618 líneas de código entonces nuestro KLDC será de: 4618/1000 = 4,618 KLDC

Por lo tanto se tiene la suma total del prototipo:

KLDC = 4,618 líneas de código

Para calcular el Esfuerzo, se requiere la variable KDLC (Kilo-líneas de código), estimadas anteriormente

Tabla 4.8 Conversión de Puntos función a líneas de código

LENGUAJE	LDC/PF
C	150
C++	64
PHP	12
SQL	12

Fuente: Estimación de Proyectos Software.

Así entonces tras saber que son 12 LDC por el hecho de que el sistema será desarrollado en PHP, se tiene que los puntos función son:

PF = 4.618 / 12

PF = 384,83

Como KLDC estimado es menor a 50, se debe utilizar el tipo orgánico y por consiguiente las constantes que se utilizarán son:

Tabla 4.9: Coeficiente de evaluación de software del tipo orgánico

Constante	a	b	c	d
Valor	3,2	1,05	2,5	0,38

Fuente: Estimación de Proyectos Software.

Adicionalmente se debe hallar un factor de ajuste que está determinado por las necesidades y ponderación de quince factores que se muestra en la siguiente tabla:

Tabla 4.10: Conductores de coste

Conductores de coste	VALORACIÓN			
	Bajo	***Nominal***	***Alto***	***Muy alto***
Fiabilidad requerida del software	0,88	1.00	1,15	1,40
Tamaño de la base de datos	0,94	1.00	1,08	1,16
Complejidad del producto	0,85	1.00	1,15	1,30
Restricciones del tiempo de ejecución	-	1.00	1,11	1,30
Restricciones del almacenamiento principal	-	1.00	1,06	1,21
Volatilidad de la máquina virtual	0,87	1.00	1,15	1,30
Tiempo de respuesta del ordenador	0,87	1.00	1,07	1,15
Capacidad del analista	1,19	1.00	0,86	0,71
Experiencia en la aplicación	1,13	1.00	0,91	0,82
Capacidad de los programadores	1,17	1.00	0,86	0,70
Experiencia en S.O. utilizado	1,10	1.00	0,90	-
Experiencia en el lenguaje de programación	1,07	1.00	0,95	-
Prácticas de programación modernas	1,10	1.00	0,91	0,82
Utilización de herramientas software	1,10	1.00	0,91	0,83
Limitaciones de planificación del proyecto	1,08	1.00	1,04	1,10

Fuente: Elaboración propia

FAE=1,15*1,00*0,85*1,11*1,00*1,00*1,07*0,86*0,82*0,70*1,00*0,95*1,00*0,91*1,08

FAE = 0,53

Esfuerzo

E = a KLDC ^{b} * FAE = 3,2 * (4.618)^1,05 * 0,53 = **8.4 personas /mes**

Tiempo

T = c Esfuerzo ^{d} = 2,5 * (8,4)^0,38 = **5.6 meses**

Productividad

PR = LDC/Esfuerzo = 4618/8,4 = **549 LDC/personas mes**

Personal promedio

P = E/T = 8,4/5,6 = **1.5 personas**

Estos resultados indican que trabajando 1.5 personas, el proyecto confluyen en 5.6 meses, sin embargo si el equipo de desarrollo fuera de tres personas, entonces el tiempo de trabajo debe reducirse a la mitad, aproximadamente 3 meses de desarrollo.

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se ha logrado desarrollar e implementar el Sistema de control y gestión de historiales clínicos, que permite mejorar las tareas de gestión de historiales clínicos de manera rápida y confiable, reduciendo de esta manera el tiempo en el registro de datos y manejo de esta información en el Centro Médico.
- Con la implementación del módulo de recordatorios en dispositivos móviles para las recetas médicas, el paciente recibe un recordatorio en el momento indicado, mejorando así el tratamiento de cada paciente.
- Se ha cumplido con los objetivos planteados realizando los siguientes módulos: *módulo de historial de paciente, módulo de administración de personal médico, módulo de cita médica, módulo de recordatorios.*

5.2 RECOMENDACIONES

- Se sugiere la implementación de un módulo de datos para cada historial clínico, que incluya el almacenamiento de estudios médicos propios de una especialidad.
- Se sugiere al personal del Centro Medico, encargada de administrar el sistema, incorporar normas y políticas de uso.

6. BILIOGRAFÍA

Beck, K. (2004). *Extrema Programming explained EMBRACE CHANGE.* Estados Unidos: Addison - Wesley.

Joc, S., & Curran, E. (s.f.). *Software Quality. A Framework for Success in Software Development and Support.* Addison Wesley.

Kroenke, D. (2003). *Procesamiento de bases de datos: fundamentos, diseño e implementación.* Pearson Educación.

Larman, C. (2007). *UML y Patrones.* Prentice Hall.

OWASP Foundation (2008): *Guia de Pruebas OWASP V3.0*. Creative Commons Attribution-ShareAlike.

Paredes, V., Santa Cruz, V., & Dominguez, M. (2012). *Programacion Multimedia y Dispositivos Moviles*. Ra-Ma.

Pilar Esparza Salanova, (2006). *MODELOS DE CALIDAD WEB. CLASIFICACIÓN DE MÉTRICAS.* Universidad Politécnica de Madrid, España.

Potencier, F., & Zaninotto, F. (2008). *Symfony 1.2, la guía definitiva.* Licencia de documentación libre de GNU.

Pressman, R. S. (2002). *INGENIERÍA DEL SOFTWARE. Un enfoque práctico. (5ta edición).* Concepción Fernández Madrid.

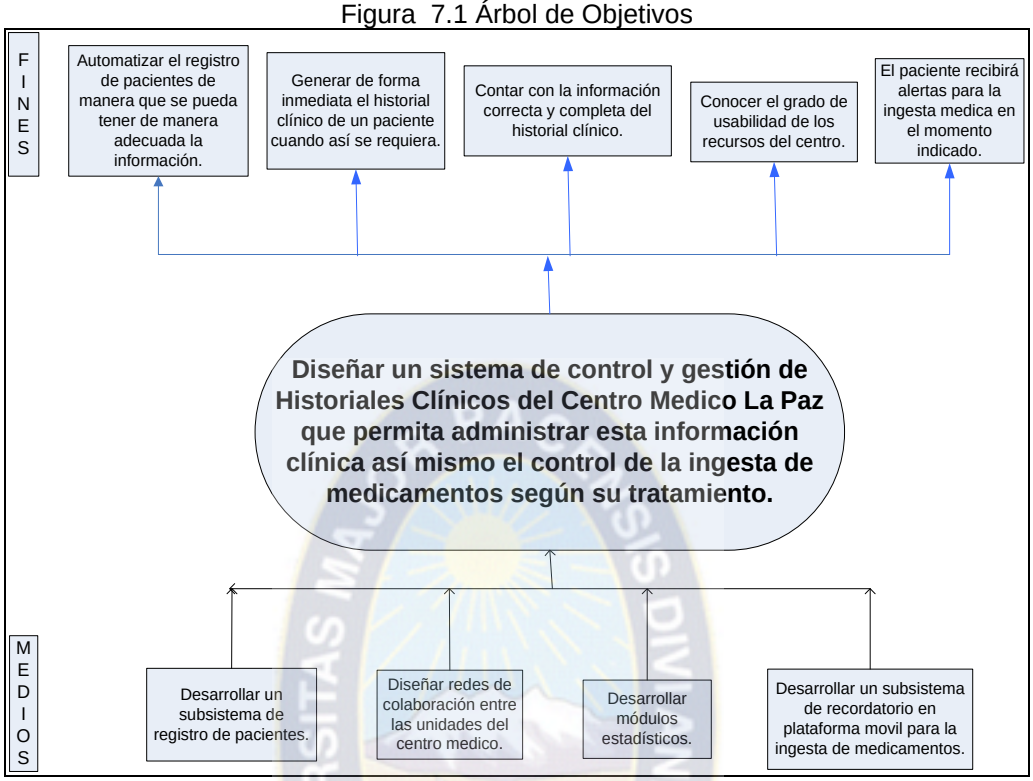
Subcomité Técnico HL7 Spain (2007). *Guía para el desarrollo de CDA (Clinical Document Architecture) - HL7(Health Level 7)*, España.

W3C. (2010). *World Wide Web Consortium*. Recuperado el 12 de Septiembre de 2015, URL: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb.

Sommerville, I. *(2005). Ingenieria de Software.* PEARSON, Addison Wesley.

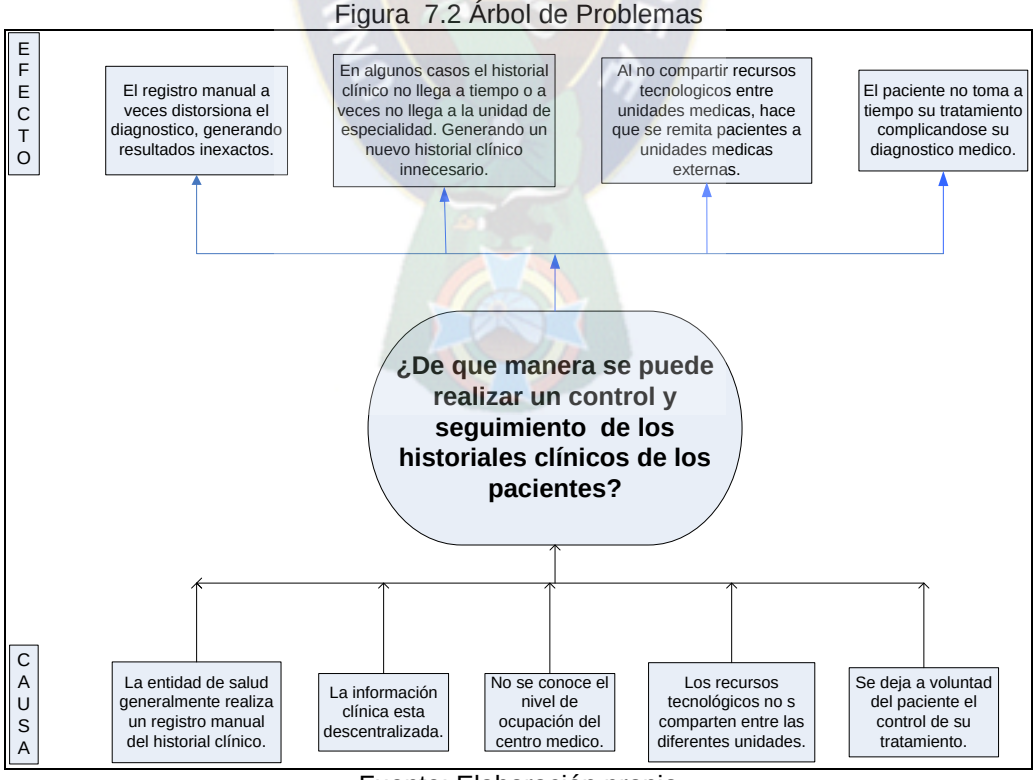
7. ANEXOS

Figura 7.1 Árbol de Objetivos



Fuente: Elaboración propia

Figura 7.2 Árbol de Problemas



Fuente: Elaboración propia