ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

AUTOR: ANDRÉS JONATHAN VEINTIMILLA

LUIS FERNANDO CUENCA

TEMA: ESTUDIO DE LA TÉCNICA TEST DRIVEN DEVELOPMENT (TDD) Y DESARROLLO DEL SISTEMA PARA LA ADMINISTRACIÓN DE CONSULTORIOS MÉDICOS

DIRECTOR: ING. JENNY RUIZ

CODIRECTOR: ING. MARIO RON

SANGOLQUÍ, JULIO 2014

CERTIFICADO

Ing. Jenny Ruiz

Ing. Mario Ron

CERTIFICAN

Que el trabajo titulado “ESTUDIO DE LA TÉCNICA TEST DRIVEN DEVELOPMENT (TDD) Y DESARROLLO DEL SISTEMA PARA LA ADMINISTRACIÓN DE CONSULTORIOS MÉDICOS” realizado por la Sr. ANDRÉS JONATHAN VEINTIMILLA PESANTEZ y el Sr. LUIS FERNANDO CUENCA GIRÓN, ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas por la Universidad de las Fuerzas Armadas “ESPE”.

Sangolquí, julio del 2014

Ing. Jenny Ruiz	Ing. Mario Ron
DIRECTOR	CODIRECTOR

DECLARACIÓN DE RESPONSABILIDAD

ANDRÉS JONATHAN VEINTIMILLA PESANTEZ

LUIS FERNANDO CUENCA GIRÓN

DECLARO QUE:

El proyecto de grado denominado “ESTUDIO DE LA TÉCNICA TEST DRIVEN DEVELOPMENT (TDD) Y DESARROLLO DEL SISTEMA PARA LA ADMINISTRACIÓN DE CONSULTORIOS MÉDICOS”, ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme a las fuentes que se incorporan en la bibliografía.

Consecuentemente este trabajo es nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, julio del 2014

_____ _____

Andrés Veintimilla Luis Cuenca

AUTORIZACIÓN

Nosotros, ANDRÉS JONATHAN VEINTIMILLA PESANTEZ

LUIS FERNANDO CUENCA GIRÓN

Autorizamos a la UNIVERSIDAD DE LA FUERZAS ARMADAS “ESPE”, la publicación, en la biblioteca virtual de la Institución del trabajo “ESTUDIO DE LA TÉCNICA TEST DRIVEN DEVELOPMENT (TDD) Y DESARROLLO DEL SISTEMA PARA LA ADMINISTRACIÓN DE CONSULTORIOS MÉDICOS”, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Sangolquí, julio del 2014

_____ _____

Andrés Veintimilla Luis Cuenca

DEDICATORIA

El presente trabajo está dedicado a mi maestro celestial Jesús que siempre me acompaña y me dio la oportunidad de hacer realidad mis sueños.

A mis padres Julio y Teresa, cuyo sacrifico y amor constante siempre me tendrá en deuda.

A mis hermanos Dianita, Mayra, Diego, Mercy que me apoyaron y empujaron al trabajo constante, enseñándome que la humildad se maneja en silencio y con paciencia se logra lo imposible.

A mis familiares que la distancia me apoyaron y motivaron para no desfalleces en el largo camino del éxito.

A las amistades que Dios puso en mi camino para acompañar mi vida y darme a conocer la perseverancia y los valor para superar los desafíos de la vida.

Luis Fernando Cuenca

A Dios ser supremo que me ha dado la suficiente fuerza espiritual para culminar esta etapa de aprendizaje en mi vida.

De igual forma dedico esta tesis a mis PADRES, Patricio y Luci quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Su tenacidad y lucha insaciable han hecho de ellos el gran ejemplo a seguir y destacar.

A mis hermanos Javier y Bryan que estuvieron siempre a mi lado ayudándome.

Y a toda mi familia en general, porque me han brindado su apoyo incondicional y por compartir conmigo buenos y malos momentos.

Andrés Jonathan Veintimilla Pesántez

AGRADECIMIENTO

Agradezco a Dios que sin EL no podría hace realidad el sueño de convertirme en ingeniero, constantemente me enseña que con amor los retos de la vida se tornan una bendición y oportunidad de crecer como persona.

Gracias a mis padres Julio y Teresa, cuyo amor supero todas las barreras y juntos pudimos demostrar que si se puede conseguir lo que se anhela, los quiero demasiado y eternamente estaré agradecido, para plasmar el apoyo que día a día obtuve junto a ellos cito la siguiente frase:

"Un padre y una madre deben tener FE ciega en sus hijos"

Y su confianza ahora se ve reflejada en su profesional.

A mi hermano diego que siempre estuvo esperando este momento, juntos vivimos el proceso de desarrollo de este proyecto. Siempre te estaré agradecido Teniente.

A mis amigos que nunca dejaron de apoyarme y enseñarme que no debo dejar de soñar en grande y que siempre se espera los premios de la vida con esfuerzo y dedicación. Les estoy eternamente agradecido.

Luis Fernando Cuenca

En primer lugar a Dios por protegerme y por guiarme por todo mi camino, y darme fuerzas para superar obstáculos y dificultades.

A mis PADRES, por darme lo mejor en todos los aspectos, por el esfuerzo, por enseñarme a ser una mejor persona, por la confianza que tienen en mí a pesar de mis errores y por brindarme las fuerzas necesarias para alcanzar este logro, y lo que hoy soy es por ustedes.

Andrés Jonathan Veintimilla Pesántez

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

Con el continuo desarrollo de las aplicaciones distribuidas en el sector médico, poco a poco se presentan más a menudo los sistemas para automatizar los procesos internos de los consultorios médicos. El objetivo del presente trabajo es encontrar las ventajas de trabajar con la técnica de desarrollo guiado por pruebas frente al ciclo tradicional de testing dentro de la construcción de sistemas. Para llevar a cabo se realizó el análisis, diseño e implementación de una aplicación Web basada en la Metodología SCRUM conjuntamente con TDD. Como parte de diseño, se utilizó la Norma IEEE-830, que contempla los lineamientos para la recolección y análisis de la ingeniería de requerimientos. Para el análisis entre paradigmas del testing se utilizó las pruebas unitarias y de integración como marco de referencia para la elaboración de un cuadro comparativo frente a TDD. Los resultados obtenidos muestran que la técnica de desarrollo guiado por pruebas en colaboración con el framework JUnit no simplemente abarca el testing de la aplicación, sino que conduce el diseño de la misma, mejorando el modelo de codificación de software. Además se mantuvo un detalle de cuando se dio un fallo sobre un test, de esta forma se pudo detectar donde estuvo el error y corregirlo a tiempo. Además se muestra la aceptación por parte de los usuarios que catalogan la aplicación como confiable, disponible y eficaz a la vez para gestionar los procesos para la administración del consultorio médico.

PALABRAS CLAVES: TÉCNICA TDD, METODOLOGÍA SCRUM, ADMINISTRACIÓN DE CONSULTORIOS MÉDICOS, TEST DRIVEN DEVELOPMENT, DESARROLLO GUIADO POR PRUEBAS.

ABSTRACT

With the continuous development of distributed applications in the medical sector, gradually systems are more often present to automate the internal processes of medical offices. The aim of this work is to find the advantages of working with the technique of test-driven development to the traditional cycle testing in building systems. To perform the analysis, design and implementation of a Web-based application SCRUM methodology was conducted in conjunction with TDD. As part of design, the IEEE-830 standard, which provides guidelines for the collection and analysis of requirements engineering was used. For analysis between paradigms of unit testing and integration testing as a framework for the development of a comparative table in front of TDD was used. The results show that the technique of test-driven development in collaboration with the JUnit framework encompasses not just the testing of the application, but leads the design of it, improving the model coding software. Also held a detail of when a decision on a test like this where he could detect the error and correct it took time. Further acceptance by users cataloging application as reliable, available and effective at the same time to manage the processes for managing medical office shown.

KEYWORDS: TECHNICAL TDD, SCRUM METHODOLOGY, PRIVATE PRACTICES MANAGEMENT, TEST DRIVEN DEVELOPMENT, TESTING GUIDANCE DEVELOPMENT.

GLOSARIO DE NOMENCLATURAS

- TDD: desarrollo dirigido por pruebas.
- IEEE: Instituto de ingeniería eléctrica y electrónica.
- ERS: Especificación de Requerimientos de Software.
- IDE: Entorno de Desarrollo Integrado.
- QA: Aseguramiento de la calidad.
- XP: metodología de desarrollo ágil.
- MVC: Modelo vista controlador.
- JSF: Java Server Faces, framework de desarrollo.
- SUT: Sujeto sobre la prueba
- Refactorizar: Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.
- Tester: Investiga un producto de software con el objetivo de obtener información acerca de su calidad.
- JUnit: Conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas para
- IDE: Es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código.
- Framework: Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular
- Sprint Backlog Lista de tareas que el equipo elabora
- API: Interfaz de programación de aplicaciones.
- Sprint Planning: Planificación de la iteración
- Product Owner: Ciente
- Scrum Master: Facilitador que lidera al equipo
- ROI: Retorno de la inversión
- SOAP: Simple Object Access Protocol
- GPL: General Public License
- GNU: E s un sistema operativo basado en software libre

CAPÍTULO 1

INTRODUCCIÓN

El progreso en el desarrollo del software ha tomado nuevas prácticas en los proyectos actuales, donde el entorno del sistema es muy variado, exige disminuir costos, reducir los tiempos de desarrollo y mantener una calidad de alto nivel, es por eso que nacen las metodologías ágiles como una posible respuesta a los problemas en el desarrollo de software.

En las metodologías ágiles la técnica de desarrollo de software dirigido mediante pruebas (TDD), es una técnica fundamental del ciclo de desarrollo que permite al grupo de desarrollo modificar y refactorizar el código existente sin miedo a romper funcionalidades, pues cada una de ellas tiene un test asociado que debe cumplirse.

TDD requiere disciplina para llevarse a cabo, pues para cada funcionalidad que se desee agregar se debe escribir, antes que nada, uno o varios test que comprueben que la funcionalidad está implementada correctamente. Esta necesidad de escribir las pruebas antes de la funcionalidad propiamente, lo cual obliga a escribir pruebas que no dependen del código que se haya estructurado.

La obligación de escribir varios test antes de implementar funcionalidades, obliga a considerar con más detenimiento las responsabilidades y diseño de éstas, lo que generalmente se traduce en código más robusto y entendible para otros desarrolladores.

Luego del estudio de la citada técnica, se la aplicará en el desarrollo de un sistema para la administración de consultorios médicos, delineados por los requerimientos del consultorio médico del Dr. Wladimir Herrera.

Para los consultorios médicos la prioridad es brindar un servicio de calidad, íntegro y eficaz para salvaguardar la salud del paciente, sin embargo, se ha hecho de lado las potenciales ventajas que hoy en día se podría obtener de la tecnología, una de las mayores ventajas están en el tratamiento de las citas, consultas, historiales, pacientes y médicos a través de una página web la cual se encontrará disponible las 24 horas del día todos los días.

1.1 PLANTEAMIENTO DEL PROBLEMA

El desarrollo de software en colaboración con las prácticas tradicionales, influyen en el proceso de calidad, pues se ve opacado cuando estas prácticas se tornan predictivas, es decir, que es posible predecir el comportamiento de un sistema bajo ciertas condiciones (Romo, http://mundobyte-x.blogspot.com/2010/03/ultima-revision-17-de-marzo-2010.html, 2010), sin embargo, en el actual entorno de desarrollo de software, predecir un cambio de requerimientos o estimar la duración de un proyecto representa una tarea de alta complejidad, ya que la restructuración del sistema implica altos costos y demandan tiempo.

Además, en las prácticas tradicionales la etapa de pruebas es manejado por un grupo aparte del equipo de desarrollo, conjuntamente, esta fase se realiza al final de la etapa de codificación, dichos procesos ocasionan pérdida en la eficiencia en los recursos ya que los desarrolladores son los mejores testers para probar el correcta funcionamiento del sistema.

La insatisfacción de los clientes del software también se debe a las validaciones tardías de requerimientos, lo que produce una solución inconsistente a un caso específico, además, la falta de un previo proceso de pruebas conlleva a numerosos defectos de software en la fase de producción. Estas deficiencias se mencionan en el estudio de mercado The Chaos Report (Hut, http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure, 2009) realizado en el Centro Experimental de Ingeniería de software (CEIS) por Standish Group International en 2009, donde se concluyó que sólo un 32% de los proyectos de software son exitosos (terminan dentro de plazos y costos y cumplen los requerimientos acordados). Otro 44% sobrepasa costos y plazos y cumple parcialmente los requerimientos. El 24% restante ni siquiera llega al término.

1.2 JUSTIFICACIÓN

En el desarrollo de software basado en metodologías tradicionales, los cambios en los procesos son volátiles y considerados como amenazas, y no como oportunidades, por lo que la presente investigación pretende realizar un estudio sobre los beneficios que otorga el desarrollo dirigido por test aplicado en las metodologías ágiles, con la finalidad de interpretar esos cambios no sólo como inevitables, sino también necesarios, ya que la flexibilidad ante el cambio es la mayor ventaja de las metodologías ágiles.

El objetivo del desarrollo guiado por pruebas es, escribir las pruebas que validarán el sistema antes de comenzar a construirlo, idealmente que sea el propio cliente quien las escriba. De esta forma, ante cada modificación del sistema el conjunto de pruebas es ejecutado.

Esta técnica constituye, por tanto, la primera línea para garantizar la calidad del producto, pues ésta se considera en términos de la correcta

funcionalidad que se le ofrece al cliente. Entre las ventajas que aporta el desarrollo dirigido por pruebas se puede destacar que disminuye el tiempo dedicado a solucionar errores y genera un sentimiento de confianza de éxito del proyecto en el equipo. Tecnologías como JUnit son utilizadas para manejar y ejecutar conjuntos de pruebas automatizadas.

El objetivo del desarrollo guiado por pruebas es, ante todo, lograr un código limpio que funcione basado en los tres pilares que cumple esta técnica:

- La implementación de las funciones justas que el cliente necesita.
- La reducción del número de defectos que llegan al software en fase de producción.
- La producción de software modular, altamente reutilizable y preparado para el cambio.

Actualmente, El consultorio médico del Dr. Wladimir Herrera presenta problemas relacionados con la acumulación de documentos de diferente tipo, entre ellos historias médicas, resultados de laboratorios y curvas de crecimiento, por ende se vuelve inmanejable tal cantidad de información. Por otra parte el consultorio médico cuenta con los implementos necesarios para dar atender a sus pacientes, pero mantiene la información correspondiente a códigos de medicamentos, códigos de enfermedades y exámenes clínicos en forma manual; el proceso que se cumple con dicha información es aceptable pero no es eficaz, ya que un proceso en el cual se calcule y estime los resultados de exámenes médicos para un diagnóstico más acertado seria optimo, los datos sólo se convierten en información cuando han sido evaluados, filtrados, condensados, analizados y organizados para un propósito.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Realizar un estudio de la técnica Test Driven Development (TDD) utilizando el Framework JUnit y desarrollo del sistema para la administración de consultorios médicos en base a los lineamientos de la metodología SCRUM.

1.3.2 OBJETIVOS ESPECÍFICOS

- Investigar y analizar el funcionamiento y aplicación de la técnica del desarrollo dirigido por pruebas (TDD), al igual que el Framework JUnit.
- Implementar la técnica del desarrollo por pruebas (TDD) en el caso práctico del sistema web de administración de consultorios médicos.
- Gestionar el proyecto de desarrollo de software en base a la metodología SCRUM.
- Realizar un cuadro comparativo entre las pruebas (unitarias y de integración) y las pruebas que ofrece JUnit.
- Establecer conclusiones y recomendaciones de los resultados de las comparaciones.

1.4 ALCANCE

El proyecto se orientará hacia el estudio de la técnica (TDD) aplicada a la metodología SCRUM, con pruebas unitarias que serán justificadas para evaluar si pasa o no la prueba que se está aplicando.

Posteriormente se realizará un estudio sobre la metodología SCRUM, para aplicar al caso práctico en colaboración de la técnica (TDD) en base al documento de especificación de requerimientos (SRS) para el sistema web de administración de consultorios médicos.

El caso práctico será desarrollado en lenguaje java v.7, sobre el sistema operativo Windows v.7, utilizando el IDE Netbeans v.7.2, con base de datos MySQL v.5.1.68-cll y la herramienta para pruebas JUnit versión 4 con la finalidad de verificar, manejar y ejecutar conjuntos de pruebas automatizadas.

Los módulos a implementar son:

- Permitir la gestión de usuarios al sistema.
- Administrar las reservaciones de los pacientes para las consultas médicas con el Doctor Wladimir Herrera, de esta forma se optimizará la distribución horaria de la agenda y así brindar la disponibilidad de reservas necesarias.
- Gestionar las Historias Clínicas de cada paciente.
- Emitir Diagnósticos y recetas a los pacientes.
- Diseño y análisis de la curva de crecimiento de los pacientes en base a los parámetros dictados por el Doctor Wladimir Herrera.

- Desplegar información pertinente a los estudios realizados por el Doctor Wladimir Herrera y conocimientos sobre el área de Pediatría.

Como fase final el sistema será implantado sobre un servidor de aplicaciones dentro de un dominio en la Internet, se entregará la documentación técnica y de usuario para garantizar el uso adecuado y la posibilidad de realizar el mantenimiento necesario al aplicativo.

CAPÍTULO 2

MARCO TEÓRICO

2.1 DESARROLLO DIRIGIDO POR TESTS

2.1.1 DEFINICIÓN

En el desarrollo de una aplicación es importante verificar el código a implementarse, con el fin de medir la calidad del producto software. Normalmente el código es analizado por un equipo de Quality Assurance (QA), el cual trata de encontrar fallos de manera más o menos manual.

El desarrollo Dirigido por Tests (Test Driven Development), es una técnica de diseño e implementación de software, donde cada nueva línea de código que escribe un programador es en respuesta a una prueba que ha fallado. Sin ser definida como Metodología su implementación se basa en las pruebas unitarias y también en pruebas de aceptación, basándose en prácticas formalizadas de XP.

2.1.2 CARACTERÍSTICAS

Evitar escribir código innecesario, se intenta escribir el mínimo posible, y si el código pasa una prueba y falla, da una idea de cómo modificar nuestra lista de requerimientos agregando nuevos.

Generar pruebas para cada funcionalidad hace que el desarrollador confíe en el código escrito. Esto facilita futuras modificaciones, ya que si se logra asegurar el paso de todas las pruebas se obtendrá un código que funcione correctamente.

TDD requiere que el desarrollador haga fallar los casos de prueba. Se intenta asegurar el funcionamiento de los casos de prueba y de esta forma pueda recoger un error.

EL desarrollo guiado por pruebas (TDD) puede proporcionar un gran valor agregado en la creación de software, produciendo aplicaciones de calidad y en menos tiempo.

TDD tiene la capacidad de avanzar en pequeños pasos, permite al desarrollador centrarse en la tarea actual y hacer que el test pase.

Todo el proyecto se encuentra cubierto de pruebas, aumentado la confianza sobre lo desarrollado.

2.1.3 CICLO DEL DESARROLLO DEL TDD

A continuación se describe el ciclo de desarrollo del TDD y los niveles aplicables de esta técnica. (Araújo, Test Driven Development Fortalezas y Debilidades, pág. 2), (Jurado, Diseño Agil con TDD, 2010, pág. 53).

Test Driven Development (TDD) es una disciplina iterativa e incremental de diseño y programación. Es parte fundamental del desarrollo ágil derivado de XP, su principal objetivo es obtener “Código limpio que trabaje”

(Mugridge, 2006) Menciona, TDD puede ser aplicado en dos niveles:

- **Nivel de micro-iteraciones:**

En el nivel de micro-Iteraciones el desarrollo es guiado por pruebas unitarias diseñadas por el programador, (Mugridge, 2006) señala los siguientes pasos:

- Rápidamente agregar una prueba.
- Correr todas las pruebas y comprobar que solo la nueva falla.
- Hacer un pequeño cambio.
- Correr todas las pruebas y comprobar que pasan satisfactoriamente.
- Reconstruir para remover duplicaciones.

Los pasos anteriormente mencionados se describen en el siguiente diagrama de actividad de la ilustración 1.

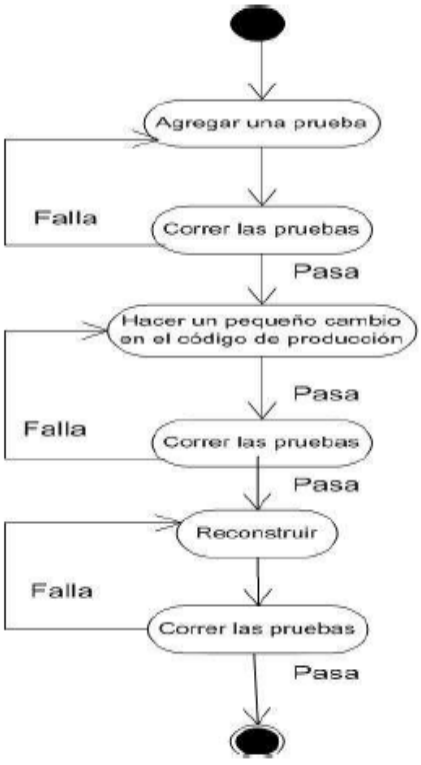


Ilustración 1: Ciclo TDD. (Araújo, Test Driven Development Fortalezas y Debilidades)

(Jurado, Diseño Agil con TDD, 2010), menciona el algoritmo descrito en el grafico anterior:

- Escribir la especificación del requisito (el ejemplo, el test).
- Implementar el código según dicho ejemplo.
- Refactorizar para eliminar duplicidad y hacer mejoras.

Escribir la especificación del requerimiento

Después de tener claro cuál es el requisito, el programador lo expresa en forma de código. La primera interrogante que se plantea es: ¿Cómo implementar un test para un código que no existe?, para esto se debe desarrollar la habilidad de imaginar cómo sería el código del SUT si ya estuviera implementado y cómo se comprobará su efectividad bajo una petición.

Lógicamente en esta fase el test no pasará, debido a la falta de una clase a al que haga referencia el test.

Implementar el código según dicho ejemplo

Una vez codificado el test, implemente el mínimo necesario para que el test pase. Es recomendable aplicar la programación por intención (*Programming by intention*) práctica extraída de XP, la cual implica no detenerse a pensar cómo conseguir un objetivo, sino en lo que se debe hacer para conseguir el objetivo. (Anderson, 2000, pág. Capítulo 14).

Refactorizar para eliminar duplicidad y hacer mejoras

(Martin, 2002) Define la refactorización como una práctica que se enfoca en mejorar el código existente, de manera disciplinada, sin alterar su funcionalidad externa, para hacer más fácil de entender y modificar. De esta forma la reconstrucción no significa reescribir el código, sino rastrear el código en busca de líneas duplicadas, además de verificar si el código cumple con ciertos principios de diseño, para este fin se puede implementar los criterios de S.O.L.I.D. (Jurado, Diseño Agil con TDD, 2010, pág. 111)

- **Nivel Iteración o Funcional:**

En este nivel el desarrollo es guiado por pruebas de aceptación (ATDD) (Jurado, Diseño Agil con TDD, 2010, pág. 63), se menciona los siguientes pasos:

- El usuario especifica pruebas antes que las funcionalidades sean implementadas.
- Una vez que el código es escrito la prueba sirve como un criterio de aceptación.

2.1.4 PARTES DEL TEST

En un test siempre hay que considerar tres partes fundamentales, AAA por sus siglas en inglés: ARRANGE (Preparar), ACT (Actuar), ASSERT (Afirmar).

- Parte de la codificación ARRANGE puede estar contenida en el método *setUp*, si es común para los test de la clase.
- ACT consiste en hacer la llamada al SUT.

- ASSERT se hacen sobre el resultado de la ejecución, utilizando validaciones de estado o validaciones de interacción.
  El Framework JUnit dispone de los siguientes métodos para hacer comprobaciones:

Tabla 1: Métodos de JUnit para hacer comprobaciones

Método ASSERT () JUnit	DESCRIPCIÓN
assertTrue(expresión)	Comprueba que expresión evalúe a true.
assertFalse(expresión)	Comprueba que expresión evalúe a false.
assertEquals(esperado, real)	Comprueba que esperado sea igual a real
Assert Null(Objeto)	Comprueba que objeto sea null
Assert NotNull(Objeto)	Comprueba que objeto no sea null
AssertSame(Objeto_esperado, objetivo_real)	Comprueba que objeto_esperado y objeto_real sean el mismo objeto
Assert NotSame(Objeto_esperado, objetivo real)	Comprueba que el objetivo_esperado no sea el mismo objetivo que el objetivo_real.
fail()	Hace que el test termine con fallo.

```
Package TDD;
import org.junit.Test;
import static org.junit.Assert.*;


public  class EjemploTDD{
@Test
Public void Metodo(){
//Arrange
……..
// Act
……..
//Assert
……..
}
}
```

Ilustración 2: Ejemplo de las partes de una prueba en TDD

En la Ilustración 2 las partes de una prueba se delimitan con comentarios (//), A continuación se describe cada una de ellas.

2.1.5 ARRANGE

Para esto se debe conocer el requisito, el cual será expresado en forma de Test. EL primer paso del algoritmo de TDD: **"Escribir la especificación del requerimiento"**, describe la necesidad de escribir un test antes de implementar el código que solventa el requerimiento.

Para poder codificarlo, tiene que pensar cómo quiere que sea la API del SUT. Para ejemplificar se muestra un test que realiza un operación matemática.

```
public void testSuma() {
Suma instance= new OperacionSuma();
int result;
result= instance.suma(1, 1);
assertEquals(2, result);
}
```

Ilustración 3: Primera parte de un Test (Preparar)

Para esta primera parte intente instanciar una clase llamada **OperacionSuma** que aún no existe, y a su vez un método dentro de esta clase llamado **Suma()**, por dichas razones el Test no pasará.

Se Debe crear una clase **OperacionSuma** la cual contenga un método llamado **Suma()** , dicho método recibirá dos parámetros y devolverá la suma de ambos.

2.1.6 ACT

Una vez planteado el requerimiento de forma clara, se codifica el mínimo posible para que el test pase, el mínimo código se desarrolla en el menor tiempo posible, no importa si el código no es el más óptimo, en la fase de refactorización se depurará el código.

Para el ejemplo se debe crear una clase **OperacionSuma** y añadir el método **Suma()** que recibirá dos parámetros, este método devolverá un entero cualquiera, para lograr compilar.

```
package Operaciones;
public class OperacionSuma {
    public  int suma(int a, int b) {
        return 0;
    }
```

Ilustración 4: Segunda parte de un Test (Actuar)

Si se prueba el test lógicamente fallará debido a que el **assertEquals();** espera un valor de 2 cuando el método suma() retorna 0.

En la Figura 2.5 se puede ver que efectivamente el test falla, describiendo los parámetros que causaron el error.

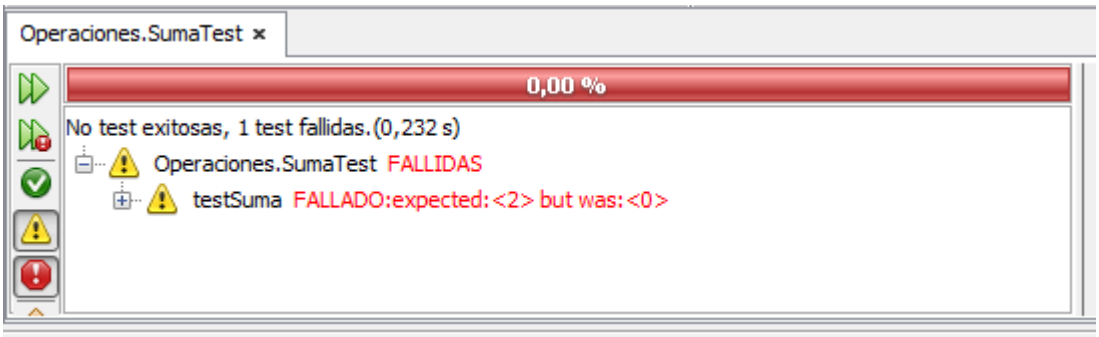


Ilustración 5: Test fallido de la función suma().

Para solucionar el fallo se modifica el mínimo código para que el test supere el error, y de forma inmediata se modifica el método **suma()** para que devuelva el número 2.

```
package Operaciones;
public class OperacionSuma {
    public  int suma(int a, int b) {
        return 2;
    }
`
```

Ilustración 6: Modificación del método Suma().

De esta manera el test automáticamente superará el error, y cumplirá con los parámetros deseados.

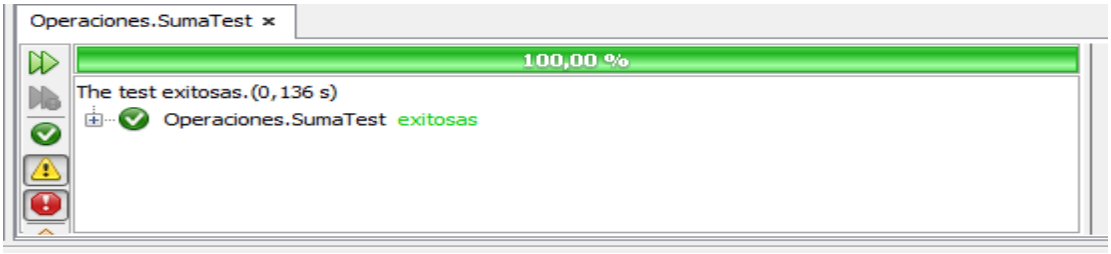


Ilustración 7: Test exitoso con JUnit.

2.1.7 ASSERT

Esta tercera parte consiste en refactorizar el código, reestructurando sin modificar su funcionalidad. La refactorización no se trata de reescribir el código sino, mejorar el diseño para hacerlo más óptimo.

Para esto se debe prestar mayor importancia a la duplicidad, que fácilmente se detecta en líneas de código duplicadas.

Para el ejemplo se reestructura el método **suma()** que se está contenida dentro de la clase **OperacionSuma** , para que el test pase sin errores, se modificó el resultado del método devolviendo el número 2, ahora optimice el código por algo más viable y coherente, la suma de los parámetros.

En la siguiente imagen se aprecia la modificación al código.

```
package Operaciones;
public class OperacionSuma {
    public  int suma(int a, int b) {
        return a+b;
    }
```

Ilustración 8: Tercera parte de un Test (Afirmar).

De esta forma se reestructuró el código sin alterar su funcionamiento. Puede ver que la cualidad de la refactorización es buscar la mejora al diseño de software contribuyendo al claro y fácil mantenimiento del mismo.

2.1.8 TIPOS DE TEST

Debido a la caótica nomenclatura sobre tests, sigue sin existir una clasificación universal. Dentro de los equipos se considera diferentes aspectos para denominar tests, por ejemplo, del aspecto visibilidad (si se sabe lo que contiene el SUT), de aspecto potestad (a quien es enfocado el test), etc. Dale H. Emery hace una referencia a los posibles aspectos o puntos de vista que se podría considerar. No son los únicos criterios que existen, pero proporciona una idea de la complejidad del problema.

Sin embargo, haciendo de lado la ideología que se tenga para definir los tests, es conveniente estar seguro de por qué se escribe el test y de qué está probándolo. Es muy importante tener claro que quiere afirmar con cada test y por qué se hace de esa manera.

En el posterior diagrama se muestra una clasificación de los test en un entorno ATDD/TDD, algunos tests contienen a otros tests.

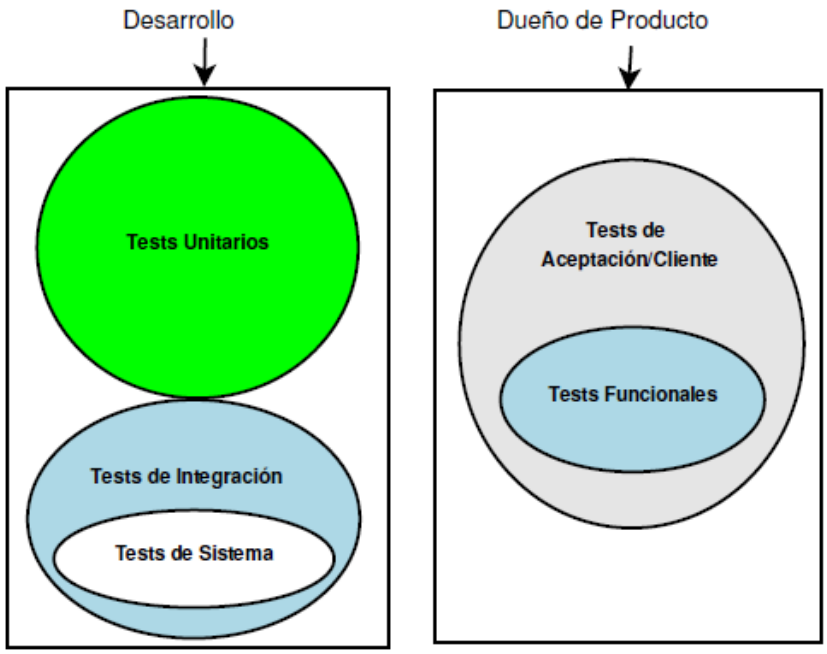


Ilustración 9: Clasificación de los tests en un entorno ATDD/TDD (Jurado, Diseño Agil con TDD, 2010)

2.1.9 TESTS DE ACEPTACIÓN

Es un test que permite comprobar que el software cumpla con los requerimientos del cliente. Un test de aceptación es un ejemplo descrito en lenguaje de usuario que puede ser interpretado por la máquina.

2.1.10 TESTS FUNCIONALES

En principio todos los tests son funcionales, ya que todos ejercen alguna función sobre el SUT, no obstante cuando se describe al aspecto funcional, se distribuyen en test funcionales y no funcionales, un test funcional es un subconjunto de los tests de aceptación. Es decir, comprueban alguna funcionalidad enfocado a un valor del negocio.

2.1.11 TESTS DE SISTEMA

Se denomina test de Sistema ya que integra varias partes de la aplicación y puede ir de extremo a extremo corroborando el funcionamiento del mismo.

Los tests de sistema son muy susceptibles a los cambios ya que cualquier cambio en algún componente del sistema puede romper el test, por eso no es recomendable escribir una gran cantidad de ellos por su fragilidad.

2.1.12 TESTS UNITARIOS

Son los tests más importantes para la práctica del TDD, todo test unitario debe ser:

- Atómico
- Independiente
- Inocuo
- Rápido

Atómico significa que el test prueba la mínima cantidad de funcionalidad posible.

Independiente significa que un test no puede depender de otros para producir un resultado positivo.

Inocuo significa que no altera el estado del sistema. Al ejecutarlo una vez, produce exactamente el mismo resultado que al ejecutarlo varias veces.

Rápido ya que ejecuta un gran número de test en poco tiempo, y de esta forma no disminuir la productividad.

Generalmente los test unitarios son usados por los desarrolladores para asegurar que el código funcione como se espera.

2.1.13 F.I.R.S.T

Las características de los test unitarios se agrupan bajo las siglas F.I.R.S.T que viene de: Fast, Independent, Repeatable, Small y Transparent.

2.1.14 TESTS DE INTEGRACIÓN

Los tests de integración viene hacer el complemento de los tests unitarios y los de sistema. Los tests de integración se pueden ver como pequeños tests de sistema. Normalmente tiene un aspecto parecido a los tests unitarios, sólo que estos pueden romper el procedimiento normal, esto significa que integra distintas partes del sistema, y logra escribir y leer datos de la base de información (BDD) para comprobar la lógica del negocio.

Un test de integración podría entenderse como un proceso donde se ejecuta la capa de negocio y después consulta la base de datos para afirmar que todo el proceso, desde el negocio hacia abajo, funciona correctamente.

2.2 METODOLOGÍA SCRUM

Scrum es una metodología ágil de desarrollo de proyectos, que toma su nombre y principios de Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80.

En el año de 1986 Hirotaka e Ikujijo publicaron el artículo “The New Product Development Game” (Hirotaka & Nonaka, 1986), el cual dio a conocer una nueva forma de gestionar proyectos en las que la agilidad, flexibilidad, y la incertidumbre son los principales elementos.

(Gallego, pág. 32) Menciona que, **Nonaka y Takeuchi** realizaron una investigación en la cual denotaron que en empresas tecnológicas que, estando en el mismo ambiente que se encontraban otras empresas, realizaban productos en menos tiempo, con mayor calidad y menos coste.

Al inicio este modelo surgió para el desarrollo de productos tecnológicos, pero también se emplea en entornos de trabajo con:

- **Incertidumbre:**

Sobre una variable se plantea los objetivos que se desea alcanzar sin detallar las estrategias a seguir. Generalmente esto genera un reto, y sobre las dudas que nacen la motivación decrecen en el equipo de trabajo.

- **Auto Organización:**

Los equipos de trabajo tienden a organizarse solos, sin necesitar roles para la gestión del proyecto

- **Control Moderado:**

Se define controles para enmarcar el trabajo bajo el “Auto Control entre iguales” para evitar que se desvié el trabajo a fines no comunes, pero sin impedir la creatividad de los integrantes del grupo de trabajo.

- **Transmisión del conocimiento:**

Entre todos se aprende, las personas superan poco a poco los proyectos y lo aprendido es transmitido al grupo de trabajo.

Todas las mencionadas son situaciones frecuentes en el desarrollo de sistemas de software.

Además esta práctica fue llamada **SCRUM** debido a una observación realizada por **Nonaka y Takeuchi** sobre renombradas empresas como Honda, HP, Canon… etc., en las cuales los productos no seguían un ciclo, con un grupo encargado de supervisar cada una de las etapas, si no que se basan en requerimientos muy generales y el

producto es desarrollado por un equipo multidisciplinario que trabaja desde el inicio hasta el fin del proyecto.

Se realizó una comparación de este estilo de trabajo, con la colaboración que hacen los jugadores de Rugby y el manejo de una formación denominada **SCRUM.**

SCRUM considera como una metodología ágil, se basa en la creación de ciclos breves para el desarrollo, que se conocen como **Iteraciones** y dentro de Scrum se llamarán "**SPRINTS**".

Antes de describir el ciclo de desarrollo de Scrum es primordial conocer las **5 fases** que definen el ciclo de desarrollo ágil:

- **Concepto:** se concreta de forma general las características del producto y se asigna el equipo que se confiará su desarrollo.
- **Especulación:** se presenta la información obtenida para asignar disposiciones, estableciendo límites que marcan el desarrollo del producto. Para profundizar esta fase, (Gallego, pág. 33), describe los rasgos que se considera para el análisis de este punto.

o **Desarrollar y revisar los requisitos generales.**
o **Mantener la lista de las funcionalidades que se esperan.** o **Plan de entrega. Se establece las fechas de las versiones, hitos e iteraciones (medirá el esfuerzo realizado en el proyecto).**

- **Exploración:** Se Realiza un incremento operativo al producto, poniendo en marcha las funcionalidades de la fase de especulación.
- **Revisión:** el equipo evalúa todo lo desarrollado y es verificado con el objetivo deseado. El equipo de revisión puede ser: las partes

interesadas del proyecto, gestores, desarrolladores y, en ocasiones los clientes.

- **Cierre:** se entrega en las fechas acordadas una versión del producto deseado. Al tratarse de una versión no significa que es la culminación del proyecto, al contrario se recepta opiniones y sugerencias para mejorar el producto.

Ilustración 10: (reserv, 2010) , ciclo del desarrollo Scrum.

2.2.1 COMPONENTES DE SCRUM

Los componentes del desarrollo del Scrum se agrupan en fases y roles, que más adelante se describirá de forma más concisa.

Scrum se divide generalmente en 3 fases, que se entiende como ***reuniones***. Las reuniones forman los pilares de esta metodología junto con los roles y el control de la evolución del proyecto.

Las Reuniones.

- **Planificación del Backlog.**

Se diseña un documento que especifica los requerimientos del proyecto bajo un orden de prioridad.

Además se define la planificación para el **Sprint 0,** para esto se define sus objetivos y el trabajo que hay que hacer en esta iteración.

El conjunto de características que forman parte de cada Sprint vienen definidas por el **Backlog,** que es un conjunto de requisitos de alto nivel que definen el trabajo que se debe realizar.

Dichos elementos para el Backlog son determinados durante la reunión de ***Sprint Planning,*** dentro de la reunión el ***Product Owner*** identifica y resalta los principales elementos del Backlog que desea ver finalizados. El equipo al frente del proyecto determina si los objetivos son alcanzables en el **Sprint 0,** si no lo fuera pasaría al **Sprint1.** Durante el tiempo que se establezca para el Sprint, nadie puede cambiar el Backlog, esto significa que los requerimientos se mantendrán estáticos.

- **Seguimiento del Sprint.**

Para esta fase se realizan reuniones diarias, donde se responde a las 3 preguntas para evaluar el avance de las tareas:

  - ¿**Qué trabajo se realizó desde la reunión anterior**?
  - **¿Qué trabajo se hará hasta una nueva reunión?**
  - **Inconvenientes que han surgido y qué hay que resolver para poder continuar. (el encargado de apoyar este punto es el *Scrum Master*).**

- **Revisión del Sprint.**

Terminado el Sprint se realiza una revisión del incremento que se ha desarrollado, para esto se presenta los resultados finales y una primera versión, esto ayuda a la retro-Alimentación con el cliente.

2.2.2 LOS ROLES

Los roles se dividen en 2 grupos: cerdos y gallinas, obtienen esta descripción en base al chiste sobre un cerdo y una gallina, los cuales intentan abrir un restaurant.

Ilustración 11: Chiste del cerdo y la gallina.

- **Los Cerdos.**

Son las personas COMPROMETIDAS con el proyecto y el proceso de Scrum.

- **Product Owner:** Es la persona que toma las decisiones, conoce sobre el negocio y la visión del producto. Se encarga de plasmar las ideas del cliente, para después ordenarlas prioritariamente y las coloca en el Product backlog.
- **Scrum Master:** Es la persona encargada de comprobar que el modelo y la metodología funciona. Eliminar todos los inconvenientes que hagan que el proceso no fluya e interactuará con el cliente y con los gestores.
- **Equipo de Desarrollo:** Es considerado un equipo de trabajo de 5 a 9 personas para organizar y tomar decisiones para

conseguir su objetivo. El equipo de desarrollo está considerado para estimar las tareas del Backlog.

- **Las Gallinas.**

No están consideras dentro del proceso de Scrum, es necesaria mantener una buena retroalimentación para revisar y planear cada Sprint.

- **Usuarios:** Destinatario final del producto.
- **StakeHolders:** terceras personas interesadas en el éxito del producto.
- **Managers:** Persona que toma las decisiones finales participando en la selección de los objetivos y de los requisitos.

2.2.3 ELEMENTOS DE SCRUM

Los elementos que forman a Scrum son:

- **Product Backlog:** lista de necesidades del cliente ordenas prioritariamente.
- **Sprint Backlog:** lista de tareas que se desarrollan en un Sprint.
- **Incremento:** Parte añadida o desarrollada en un Sprint, es una parte terminada y totalmente operativa.

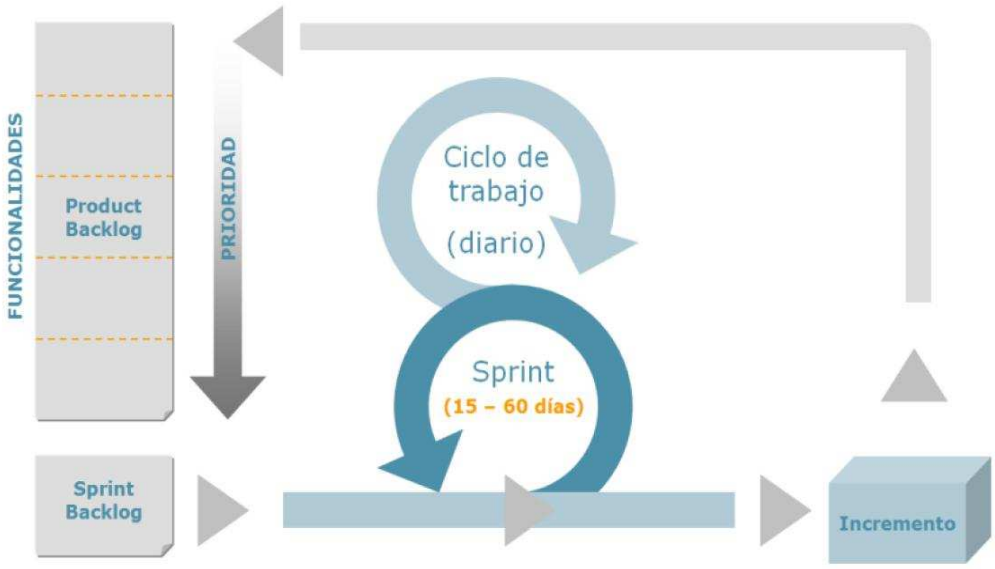


Ilustración 12: Elementos de Scrum (Gómez)

Product Backlog.

Es una lista en la que se detalla todas las funcionalidades o requisitos en forma priorizada, estos requisitos ira adquiriendo en forma secuencial el producto.

Esta lista es creada por el cliente y el Scrum Master, quien indicará el coste aproximado para completar un requisito, y además contendrá todo lo que aporte un valor final al producto.

(Gallego) Menciona las principales características que esta lista debe tener:

- **Contendrá los objetivos del producto, se suele usar para expresarlos las historias de usuario.**

- **En cada objetivo, se indicará el valor que le da el cliente y el coste estimado; de esta manera, se realiza la lista, priorizando por valor y coste, se basará en el ROI.**
- **En la lista se tendrán que indicar las posibles iteraciones y los realces que se han indicado al cliente.**
- **La lista ha de incluir los posibles riesgos e incluir las tareas necesarias para solventarlos.**

En primordial que antes de empezar el primer Sprint se definan cuáles van a ser los objetivos del producto y tener la lista de los requisitos ya definida.

Posterior a la definición de los requisitos se tendrá que acordar cuando un requisito está terminado o completo.

El Product Backlog irá evolucionando mientras el producto exista en el mercado. Esta es la forma para evolucionar y tener un valor de producto para el cliente suficiente para ser competitivo.

Historias de usuario.

Son las descripciones de las funcionalidades que va a tener el Software.

Las Historias de usuario, serán el resultado de la colaboración entre el cliente y el equipo, e irán evolucionando durante toda la vida del proyecto.

Las historias de usuario se componen de tres frases denominadas “las 3 C”:

- **Card:** Será una breve descripción escrita que servirá como recordatorio.
- **Conversation:** Es una conversación que servirá para asegurarse de que se ha entendido bien todo, y concretar el objetivo.
- **Confirmation:**Tests funcionales para fijar detalles que sean relevantes e indicar cuál va a ser límite.

En cuanto al formato, un modelo podría ser como el que se muestra en la siguiente imagen:

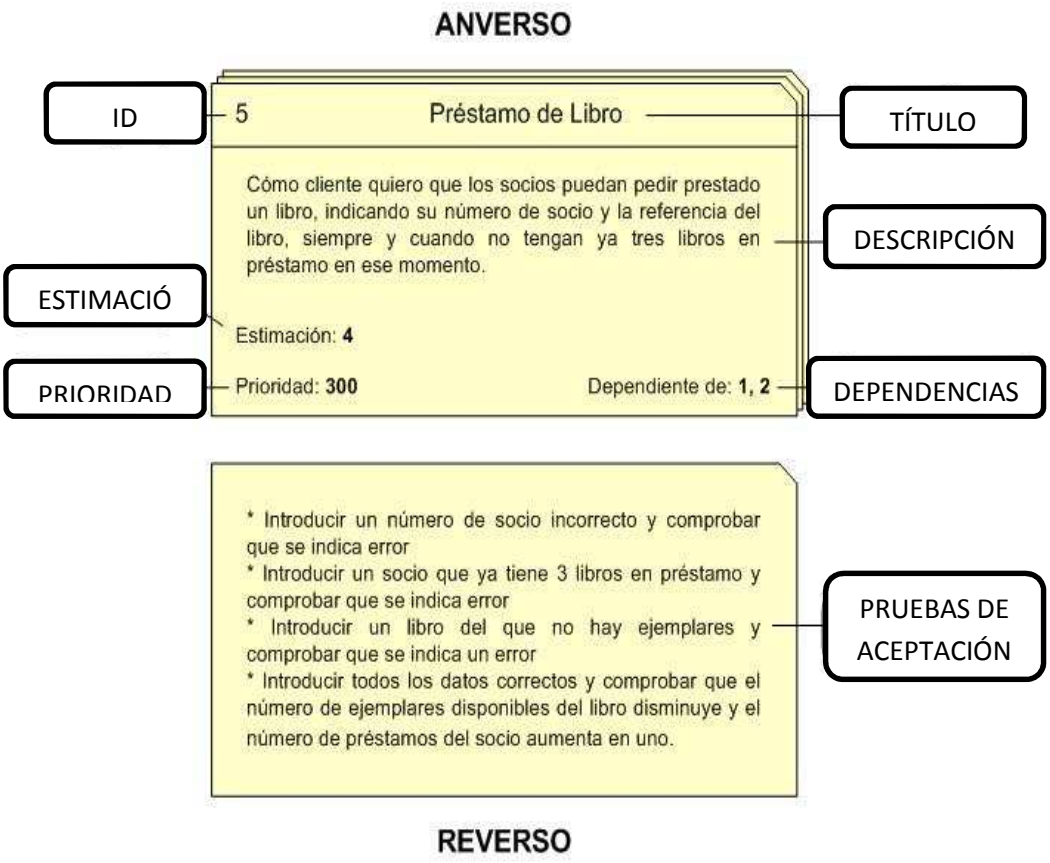


Ilustración 13: Ejemplo de Historias de Usuario.

Descripción de las partes de las historias de usuario:

- **ID:** Identificador de la Historia de Usuario.
- **TÍTULO:** Título descriptivo de la historia de usuario.
- **DESCRIPCIÓN:** Descripción sintetizada de la historia de Usuario.
- **ESTIMACIÓN:** Evaluación del coste de implementación en unidades de desarrollo. Estas unidades representan el tiempo teórico que se haya estimado al comienzo del proyecto.
- **PRIORIDAD:** Prioridad en la implementación de la historia de Usuario respecto al resto de las historias de usuario. A mayor

número, mayor prioridad. Otra aproximación a la priorización de tareas se hace a través del método MoSCoW:

  - **M – Must:** se debe completar este requerimiento para finalizar el proyecto.
  - **S – Should:** se debe completar este proyecto por todos los medios, pero el éxito del proyecto no depende de él.
  - **C** – **could:** se deberá completar este requerimiento si su implementación no afecta la consecución de los objetivos principales del proyecto.
  - **W – Would:** se puede completar este requerimiento si sobra tiempo de desarrollo.

- **DEPENDENCIAS:** Una Historia de Usuario no deberá ser dependiente de otra historia, pero a veces es inevitable. En este apartado se indicarán los ID's de las tareas de las que depende una tarea.

Sprint Backlog.

Es la lista de tareas que elabora el equipo durante la planificación de un Sprint. Se asignan las tareas a cada persona y el tiempo que queda para terminarlas.

De esta manera el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en qué tareas no se está avanzando e intentar eliminar el problema.

Requisito	Tarea	Quien	Estado (No iniciada / en progreso / completada)	Dia:	1	2	3	4	5	6	7	8	9	10
				Horas pendientes	1120	1088	1076	1048	1040	1032	1020	1008	992	972
Requisito A	Tarea 1	Joao	Completada		16	8								
Requisito A	Tarea 4	Laura	Completada		4									
Requisito A	Tarea 5	Laura	Completada		4									
Requisito A	Tarea 3	Gabri	Completada		8									
Requisito A	Tarea 2	Laura	Completada		16	8	4							
Requisito A	Tarea 6	Gabri	Completada		8	8	8							
Requisito A	Tarea 7	Joao	Completada		16	16	16	8						
Requisito A	Tarea 8	Laura	Completada		8	8	8							
Requisito A	Tarea 9	Laura	Completada		8	8	8	8	8					
Requisito A	Tarea 10	Laura	Completada		8	8	8	8	8	8	4			
Requisito A	Tarea 11	Joao	Completada		16	16	16	16	16	16	8			
Requisito B	Tarea 12	Gabri	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 13	Laura	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 14	Joao	En progreso		8	8	8	8	8	8	8	8	8	4
Requisito B	Tarea 15	Gabri	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito B	Tarea 16	Laura	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 17	Joao	No iniciada		4	4	4	4	4	4	4	4	4	4
Requisito C	Tarea 18	Gabri	No iniciada		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 19	Laura	No iniciada		16	16	16	16	16	16	16	16	16	16
Requisito C	Tarea 20	Joao	No iniciada		8	8	8	8	8	8	8	8	8	8

Ilustración 14: Ejemplo de Sprint Backlog.

Funcionamiento de la lista:

- Es una lista ordenada por prioridades para el cliente.
- Puede haber dependencias entre una tarea y otra, por lo tanto se tendrá que diferenciar de alguna manera.
- Todas las tareas tienen que tener un coste semejante que será entre 4-16 horas.

Incremento.

Representa los requisitos que se han completado en una iteración y que son perfectamente operativos. Según los resultados que se obtengan, el cliente puede ir haciendo los cambios necesarios y replanteando el proyecto.

2.3 HERRAMIENTAS DE DESARROLLO

2.3.1 JAVA PLATFORM, JAVA ENTERPRISE EDITOR (JAVA EE) V.7

Java Platform, Enterprise Edition (Java EE) es el estándar sectorial para la informática Java empresarial. (Oracle)

La tecnología Java es tanto un lenguaje de programación y una plataforma. El lenguaje de programación Java es un lenguaje de alto nivel orientado a objetos que tiene una sintaxis y estilo particular. Una plataforma Java es un entorno particular en el que las aplicaciones de lenguaje de programación Java se ejecutan.

La plataforma Java EE proporciona una API y el entorno de tiempo de ejecución para el desarrollo y ejecución de aplicaciones a gran escala, de varios niveles, escalables, fiables y seguras de red.

2.3.2 COMPONENTES DE JAVA ENTERPRISE EDITOR (JAVA EE)

- Los servlets, páginas JSP, aplicaciones JSF, filtros y detectores de eventos web suelen ejecutar en un contenedor web y puede responder a las peticiones HTTP de los clientes web. Los servlets , páginas JSP , aplicaciones JSF , y los filtros pueden ser utilizados para generar páginas HTML que son interfaz de usuario de una aplicación . También pueden ser utilizados para generar datos de formato XML o de otro tipo que se consume por otros componentes de la aplicación. Un tipo especial de servlet proporciona soporte para servicios web utilizando el protocolo SOAP / HTTP. Servlets , las páginas creadas con la tecnología JavaServer Pages ™ o la

tecnología JavaServer ™ Faces , filtros web , y detectores de eventos web se denominan colectivamente en esta memoria como **"componentes web ."**

- Los componentes Enterprise JavaBeans (EJB) se ejecutan en un entorno administrado que soporta transacciones. Los Enterprise JavaBeans suelen contener la lógica de negocio para una aplicación Java EE, pueden proporcionar directamente los servicios web utilizando el protocolo SOAP / HTTP. (Oracle)

2.3.3 NETBEANS IDE

Es un entorno de desarrollo - una herramienta para escribir, compilar, depurar y ejecutar programas. Está escrito en Java. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans IDE proporciona soporte completo para los últimos estándares Java EE7.

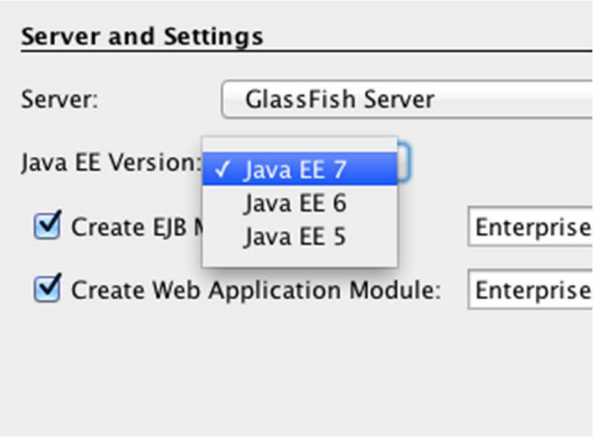


Ilustración 15: Soporte Java EE7 NetBeans.

El IDE Netbeans soporta Servicios web basados en SOAP proporcionando herramientas para trabajar con las anotaciones de servicios web

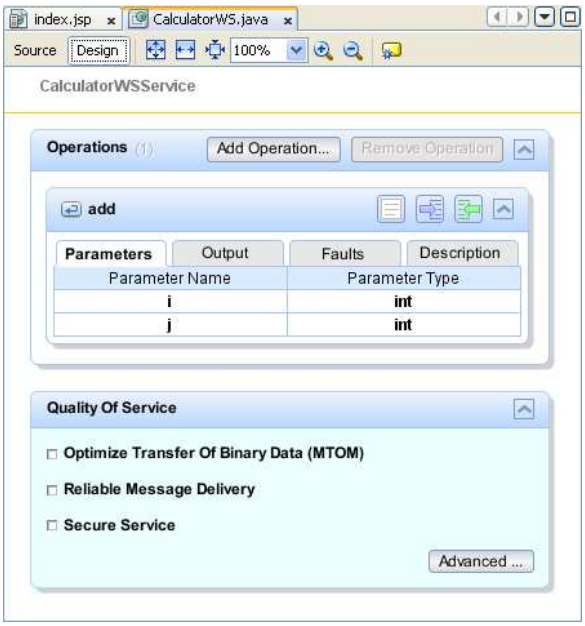


Ilustración 16: Soporte a Servicios SOAP NetBeans.

El IDE también soporta probar y crear aplicaciones cliente que acceden a los servicios web.

2.3.4 SOA (SERVICE ORIENTED ARCHITECTURE) ARQUITECTURA ORIENTADA A SERVICIOS

SOA es una unidad de software con una funcionalidad mínima, con las siguientes características:

- **Interfaz definida o Contrato de Servicio**

Descripción de cómo el servicio va a ser usado desde cualquier otro servicio o programa: Nombre, Parámetros, Resultado y Ubicación.

- **Reutilizable y/o Componible con otros**

Pueda ser utilizado por más de una aplicación y/u otros servicios o Intranet o Internet: SaS, Cloud Computing

- **Desacoplado**

Que para prestar su funcionalidad dependa en lo mínimo de otro servicio.

2.3.5 SERVICIO WEB

Un servicio web es un componente de Software que utiliza un conjunto de protocolos y estándares para intercambiar datos entre aplicaciones sobre una red. Los Servicios Web suelen ser considerados como APIs Web que pueden ser accedidos dentro de una red (principalmente Internet) y ejecutados en el sistema que los aloja. (Quispe, 2011).

2.3.6 FRAMEWORK JUNIT

JUnit es un Framework simple para escribir pruebas repetibles. Están compuestas por un conjunto de librerías creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

El Framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.

2.3.7 MYSQL V.5

MYSQL es un sistema de gestión de base de datos relacional, su funcionamiento se da bajo la licencia GPL de la GNU.

Consta de un sistema multihilo que le permite soportar una gran carga de instrucciones de forma muy eficiente, es fácil de instalar y configurar.

Se puede acoplar a una variedad de lenguajes de programación, además de herramientas y librerías que facilitan su uso.

2.3.8 CARACTERÍSTICAS DE MYSQL

- Escrito en C y en C++.
- Probado con un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un nivel adecuado de seguridad en los datos.

2.3.9 PRIMEFACES FRAMEWORK

PrimeFaces es un componente de código abierto que cuenta con un conjunto de componentes que facilitan la creación de aplicaciones web.

PrimeFaces funciona bajo la licencia de Apache License V2. Una de las potenciales ventajas de manejar PrimeFaces es la integración con otras tecnologías como RichFaces.

2.3.10 CARACTERÍSTICAS

- Conjunto de componentes (Editor de HTML, autocompletar, cartas, gráficas o paneles, entre otros).
- Soporte de ajax con despliegue parcial, lo que permite controlar cuáles componentes de la página actual se actualizarán y cuáles no.
- 25 temas prediseñados
- Componente para desarrollar aplicaciones web para móviles-celulares, especiales para Iphones, Palm, Android y teléfonos móviles Nokia.

CAPÍTULO 3

ANÁLISIS, DISEÑO Y DESARROLLO DEL CASO PRÁCTICO

3. ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE (ERS)

3.1. INTRODUCCIÓN

3.1.1 PROPÓSITO

La presente especificación de requerimientos de software (ERS) del sistema a construir, surge para ser un conjunto de información necesaria que ayuda a los desarrolladores del software a analizar y entender todos los requisitos y requerimientos que nuestro cliente desea, de la misma forma, como este constituye un informe útil para que el cliente del producto final describa lo que el realmente desea obtener, y de esta manera diseñar un documento necesario cuya información en el futuro servirá para el desarrollo del software, es decir en la codificación correcta del mismo.

Se describirá en forma detallada las interfaces de usuario, del software, del hardware y comunicaciones, así como de los requerimientos del cliente, atributos del sistema entre otros.

Este documento está sujeto a revisiones, especialmente por el usuario, hasta alcanzar su aprobación. Una vez aprobado servirá de base al equipo de desarrollo para la construcción del nuevo software.

3.1.2 ÁMBITO DEL SISTEMA

Se ha constatado la necesidad de diseñar un sistema para solventar los siguientes objetivos:

- Permitir la gestión de usuarios al sistema.
- Administrar las reservaciones de los pacientes para las consultas médicas con el Doctor Wladimir Herrera, de esta forma se optimizará la distribución horaria de la agenda y así brindar la disponibilidad de reservas necesarias.
- Gestionar las Historias Clínicas de cada paciente.
- Manipular eficientemente los códigos CIE10 Y VADEMECUM.
- Emitir Diagnósticos y recetas a los pacientes.
- Diseño y análisis de la curva de crecimiento de los pacientes en base a los parámetros dictados por el Doctor Wladimir Herrera.
- Desplegar información pertinente a los estudios realizados por el Doctor Wladimir Herrera y conocimientos sobre el área de Pediatría.

3.1.3 DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

DEFINICIONES:

Software.- Se conoce como software al equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware.

Hardware.- El término hardware se refiere a todas las partes tangibles de un sistema informático; sus componentes son: eléctricos, electrónicos, electromecánicos y mecánicos.

Sistema.- conformado por la suma de hardware, software y los usuarios.

Desarrollador.- Es un programador que se dedica a uno o más aspectos del proceso de desarrollo de software. Se trata de un ámbito más amplio de la programación.

Interface.- Medio que permite la comunicación entre el usuario y el sistema.

VADEMECUM.- libro donde reúne toda la información de relevancia relacionada con los productos farmacéuticos, y está destinado a, médicos odontólogos, farmacéuticos y otros profesionales de la salud.

pre-escribir.- Recetar algo.

TCP/IP.- es un modelo de descripción de protocolos de red desarrollado en los años 70 por Vinton Cerf y Robert E. Kahn. Fue implantado en la red ARPANET, la primera red de área amplia, desarrollada por encargo de DARPA, una agencia del Departamento de Defensa de los Estados Unidos, y predecesora de la actual red Internet. EL modelo TCP/IP se denomina a veces como Internet Model, Modelo DoD o Modelo DARPA.

Web.- Se canaliza a través del URL o identificador único de cada página de contenidos. Este sistema permite a los usuarios iniciar una solicitud de trámite y a los funcionarios del Agua Potable atender las solicitudes e ingresar datos de las inspecciones realizadas.

Internet.- Internet es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. Sus

orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como Arpanet, entre tres universidades en California y una en Utah, Estados Unidos.

ACRÓNIMOS:

ERS.- Especificación de Requerimientos de Software.

CIE10.- La CIE-10 es el acrónimo de la Clasificación internacional de enfermedades, décima versión correspondiente a la versión en español de la (en inglés) ICD, siglas de International Statistical Classification of Diseases and Related Health Problems) y determina la clasificación y codificación de las enfermedades y una amplia variedad de signos, síntomas, hallazgos anormales, denuncias, circunstancias sociales y causas externas de daños y/o enfermedad.

URL.- Un URL es una cadena de caracteres que identifica el tipo de documento, la computadora, el directorio y los subdirectorios en donde se encuentra el documento y su nombre.

3.2 DESCRIPCIÓN GENERAL

3.2.1 PERSPECTIVA DEL PRODUCTO

- Para la gestión de usuarios que dispone el sistema, se llevará a cabo un proceso que abarca desde la creación de nuevos usuarios hasta la distribución de permisos a las diferentes secciones del sistema.
  Incluye eliminar, modificar y buscar a cada uno de los usuarios.

Los actores que intervienen en este proceso son:

Médico: Administrador de los usuarios y del sistema.

Paciente: Puede acceder al sistema y solicitar una reserva médica, y anular la misma, además de visualizar información del doctor y conocimientos del área de pediatría.

- La administración de las citas médicas dentro de la agenda del Doctor Wladimir Herrera conlleva un proceso que inicia desde que se solicita una reserva hasta que se otorga la misma. Incluye la cancelación, modificación y anulación de la cita.

  En este proceso intervienen los siguientes actores:

  Médico: Podrá ver su número de reservas tanto por día como totales.

  Paciente: Puede acceder al sistema y solicitar una reserva, para la cual tendrá el privilegio de anular la misma.

  - Para la administración de las Historias clínicas, el doctor con cada paciente nuevo creará una historia clínica, posteriormente se incluirán las opciones de modificación y búsqueda de esta información.
  - Para utilizar los códigos CIE10 Y VADEMECUM, dicha información pasará por un proceso de digitalización, para posteriormente ser utilizados dentro de la emisión de recetas, diagnósticos médicos y otros documentos propios del doctor.
  - Para la emisión de diagnósticos y recetas médicas, el doctor establece un proceso que abarca :
    - La creación de nuevas historias clínicas.
    - Búsqueda de información de pacientes (Historia Clínica).
    - Manejo de códigos de CIE10 y VADEMECUM.

- Redacción de diagnósticos médicos
- De esta manera el proceso concluye con la entrega de la receta al paciente.

Los actores que intervienen en este proceso:

Médico: Busca o crea, dependiendo el caso historias clínicas.

Paciente: recibe la receta pre-escrita por el médico.

- Dentro del diseño y análisis de la curva de crecimiento; el doctor en base a parámetros propios del análisis clínicos, graficará una curva de crecimiento para monitorear el estado evolutivo de sus pacientes.

3.2.2 FUNCIONES DEL PRODUCTO

- El doctor podrá administrar a los clientes dentro del sistema con cada una de sus cuentas. Este proceso incluye la creación, modificación, búsqueda y eliminación de estas.
- La agenda Médica deberá ser capaz de proporcionar a los usuarios una forma amigable de administrar recursos.
- Abarca múltiples aspectos en la gestión de la agenda electrónica, tales como: Gestión de Historias Clínicas, Horarios y Turnos de pacientes para consultas médicas.
- Dentro de la agenda electrónica se presenta la opción de búsqueda por disponibilidad.
- Se puede generar una cita por 3 vías: Se recepta la solicitud de atención del paciente, en forma telefónica, vía web o personalmente.
- Gestión de agenda en tiempo real.

- El doctor tendrá los privilegios de: creación, modificación, búsqueda y eliminación de historias clínicas.
- Búsqueda directa en la base de información de los códigos CIE10 y VADEMECUM.
- Para detallar un diagnóstico se tiene conexión directa con los exámenes médicos del paciente y códigos CIE 10.
- Para pre-escribir una receta, el doctor tiene fácil acceso a los códigos VADEMECUM.
- El doctor podrá ingresar los parámetros necesarios para graficar la curva de crecimiento.

3.2.3 CARACTERÍSTICAS DE LOS USUARIOS

El sistema estará orientado a 2 tipos de usuarios, el Usuario con conocimientos en pediatría, quien será el encardo de Administrar el sistema, y los usuarios Estándar quienes se prevé que pueden ser de diversa formación académica desde aquellos que conocen el uso de las tecnologías de la información hasta aquellos que recién se encuentren conociendo sus características y beneficios.

3.2.4 RESTRICCIONES

- No hará control de inventario.
- No generar reportes.
- No realizará recordatorios de citas próximas.
- No administrará exámenes clínicos.

3.2.5 SUPOSICIONES Y DEPENDENCIAS

El sistema debe depender de una conexión a internet.

3.3 REQUISITOS ESPECÍFICOS

El sistema estará sujeto a la plataforma en el que sea desarrollado.

3.3.1 REQUISITOS DE LAS INTERFACES EXTERNAS

3.3.1.1 INTERFAZ CON EL USUARIO

El sistema debe ser amigable y predecible con el usuario ya que lo podrá utilizar cualquier usuario que sepa manejar un computador.

3.3.1.2 INTERFAZ CON EL HARDWARE

El sistema requiere de periféricos de entrada y salida: mouse, teclado, monitor, así como tarjeta de red para trabajar en red.

3.3.1.3 INTERFAZ DE COMUNICACIONES

El sistema podrá ser operable en red con la infraestructura que cuente el consultorio médico, en este caso el sistema estará en un dominio sobre la internet y el protocola que se necesita es el TCP/IP.

3.3.2 REQUISITOS FUNCIONALES

3.3.2.1 RE01 INGRESO AL SISTEMA

El sistema deberá autentificar al cliente para ingresar al aplicativo Web, ingresando su nombre de cédula de identidad y la contraseña.

Entrada

Cédula de identidad, contraseña.

Proceso

Ingresar el número de cédula y la contraseña correspondiente para proceder con el ingreso al sistema.

Salida

Confirmación acceso, ingresando a la ventana correspondiente a cada tipo de usuario.

3.3.2.2 RE02 REGISTRAR USUARIO

El administrador del sistema podrá registrar un nuevo usuario al aplicativo web.

Entrada

Datos del cliente: Cédula, nombres, apellido paterno, apellido materno, fecha de nacimiento, sexo, teléfono, dirección, correo electrónico, seguro médico, contraseña.

Proceso

El usuario ingresará a la pantalla de registro donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro del nuevo usuario con un mensaje.

3.3.2.3 RE03 BUSCAR USUARIO

El administrador del sistema podrá buscar la información de un usuario determinado por cédula.

Entrada

Cédula del usuario.

Proceso

El usuario ingresa a la pantalla de Administradores, donde ingresará la cédula del usuario a buscar.

Salida

Despliegue de la información del usuario que se solicita o a su vez un mensaje de error en caso de no encontrarlo.

3.3.2.4 RE04 MODIFICAR USUARIO

El administrador del sistema podrá modificar la información de un usuario determinado.

Entrada

Datos del cliente: cédula, nombres, apellidos, contraseña, teléfono, dirección, correo electrónico.

Proceso

El usuario accede a la pantalla de administradores donde ingresa y seleccionará los parámetros descritos anteriormente. El usuario deberá llenar todos los campos remarcados con un asterisco (*).

Salida

Confirmación de datos del usuario modificados exitosamente con un mensaje.

3.3.2.5 RE05 ELIMINAR USUARIO

El administrador del sistema podrá eliminar datos de un usuario de la base de datos.

Entrada

Selección del usuario a eliminar.

Proceso

El usuario en la pantalla de edición de datos del usuario, será capaz de borrar la correspondiente a través de un botón de confirmación.

Salida

Mensaje de confirmación de eliminación del usuario.

3.3.2.6 RE06 CREAR PACIENTE

El administrador del sistema podrá registrar un nuevo paciente al aplicativo web.

Entrada

Datos del Paciente: apellido paterno, apellido materno, nombres, cedula del paciente, fecha de nacimiento, sexo, teléfono, dirección, correo electrónico, seguro médico, contraseña.

Proceso

El usuario ingresará a la pantalla de registro de pacientes, donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro del nuevo paciente con un mensaje.

3.3.2.7 RE07 MODIFICAR PACIENTE

El administrador del sistema podrá modificar la información de un paciente determinado.

Entrada

Datos del Paciente: apellido paterno, apellido materno, nombres, cedula del paciente, fecha de nacimiento, sexo, teléfono, dirección, correo electrónico, seguro médico, contraseña.

Proceso

El usuario accede a la pantalla de gestión de pacientes, donde buscará un paciente y modificará los parámetros descritos anteriormente. El usuario deberá llenar todos los campos remarcados con un asterisco (*).

Salida

Confirmación de datos del paciente modificados exitosamente con un mensaje.

3.3.2.8 RE08 BUSCAR PACIENTE

El administrador del sistema podrá buscar la información de un paciente determinado bajo el parámetro: número de cédula.

Entrada

Número de cédula del paciente.

Proceso

El usuario ingresa a la pantalla de pacientes, donde ingresará la cédula del paciente a buscar.

Salida

Despliegue de la información del paciente que se solicita o a su vez un mensaje de error en caso de no encontrarlo.

3.3.2.9 RE09 ELIMINAR PACIENTE

El administrador del sistema podrá eliminar datos de un paciente de la base de datos.

Entrada

Búsqueda del paciente a eliminar.

Proceso

El cliente en la pantalla de gestión de paciente, será capaz de borrar el correspondiente a través de un botón de confirmación.

Salida

Mensaje de confirmación de eliminación del paciente.

3.3.2.10 RE10 REALIZAR RESERVA

El usuario del sistema podrá reservar una cita médica con el doctor, en una hora y fecha determinada.

Entrada

Selección del Paciente, selección del horario, selección de la fecha y el motivo de la consulta.

Proceso

El usuario accederá a la pantalla de la agenda electrónica donde Seleccionará al Paciente, seleccionará el horario, seleccionará la fecha y detallará el motivo de la consulta.

Salida

Confirmar la cita médica con un mensaje exitoso, caso contrario se despliega un mensaje de error informando que no está disponible la cita para la hora y fecha detallada anteriormente.

3.3.2.11 RE11 ELIMINAR RESERVA

El sistema deberá diferenciar con color azul las reservas realizadas y en blanco los espacios disponibles para agendar. Deberá existir un botón que permita eliminar dicha reserva.

Entrada

Reserva médica.

Proceso

El usuario accederá a la pantalla de la agenda electrónica donde será capaz de eliminar la cita por medio de un botón de confirmación.

Salida

Mensaje de confirmación de la eliminación de la cita médica.

3.3.2.12 RE12 REPORTE DE RESERVAS MÉDICAS

El sistema deberá desplegar las reservas médicas dentro de un rango de fechas, la descripción del reporte debe tener la siguiente información:

Fecha de la cita médica, nombre y apellido del paciente, hora de inicio y fin de la consulta médica y descripción del motivo de la cita médica.

Entrada

Los parámetros de búsqueda serán: fechas inicio y fin para visualizar el las citas médicas dentro de rango de fechas.

Proceso

El usuario accederá a la pantalla de reportes, donde ingresará los parámetros antes descritos.

Salida

Se desplegará información correspondiente a las reservas médicas dentro del rango de fechas.

3.3.2.13 RE13 ADMINISTRACIÓN DE RECURSOS

El sistema deberá diferenciar por colores cada recurso, deberá aparecer la casilla vacía cuando no se inserte nada en ese horario.

Entrada

Casilleros vacíos que indican un espacio disponible o casilleros con registros introducidos.

Proceso

El usuario ingresará a la agenda, donde podrá diferenciar la disponibilidad de la misma a través de colores que indican los horarios que se puede tomar.

Salida

La diferencia entre turnos disponibles y turnos asignados.

3.3.2.14 RE14 CREAR ANTECEDENTE MÉDICO

El administrador del sistema podrá registrar un nuevo antecedente médico del paciente con sus respectivos datos clínicos, los cuales son: fecha, antecedentes patológicos personales, enfermedades de hereditarios, quirúrgico traumático, farmacológico, transnacionales, hábitos, Gineco Obstétricos, alérgicos, personales sociales, vacunas, otros.

Entrada

Datos del antecedente médico del paciente: fecha, antecedentes patológicos personales, enfermedades de hereditarios, quirúrgico traumático, farmacológico, transnacionales, hábitos, Gineco Obstétricos, alérgicos, personales sociales, vacunas, otros.

Proceso

El usuario ingresará a la pantalla de administración de historia clínica donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro del nuevo antecedente médico con un mensaje.

3.3.2.15 RE15 MODIFICAR ANTECEDENTE MÉDICO

El administrador del sistema podrá modificar la información de un antecedente médico determinado.

Entrada

Datos del antecedente médico del paciente: fecha, antecedentes patológicos personales, enfermedades de hereditarios, quirúrgico

traumático, farmacológico, transnacionales, hábitos, Gineco Obstétricos, alérgicos, personales sociales, vacunas, otros.

Proceso

El usuario accede a la pantalla de administración de historia clínica donde ingresa y seleccionará los parámetros descritos anteriormente. El usuario deberá llenar todos los campos remarcados con un asterisco (*).

Salida

Confirmación de antecedente médico modificado exitosamente con un mensaje.

3.3.2.16 RE16 CONSULTAR ANTECEDENTE MÉDICO

El administrador del sistema podrá buscar la información de un antecedente médico determinado.

Entrada

Buscar un paciente por medio de la cédula para el cual se desee ver sus antecedentes médicos.

Proceso

El usuario ingresa a la pantalla de pacientes y buscará uno en especial para mostrar sus antecedentes médicos.

Salida

Despliegue de la información del antecedente médico que se solicita.

3.3.2.17 RE17 CREAR DIAGNÓSTICO

El administrador del sistema podrá registrar un nuevo diagnóstico para el paciente.

Entrada

Datos del diagnóstico: códigos CIE 10, fecha y detalle del diagnóstico.

Proceso

El usuario ingresará a la pantalla de diagnósticos donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro del nuevo diagnostico con un mensaje “Diagnóstico guardado”.

3.3.2.18 RE18 CREAR RECETA

El administrador del sistema podrá registrar una nueva receta médica para el paciente.

Entrada

Datos de la receta: seleccionar un diagnóstico para detallar una receta, indicaciones para la receta y selección del código vademécum correspondiente.

Proceso

El usuario ingresará a la pantalla de recetas donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro de la nueva receta con un mensaje.

3.3.2.19 RE19 BUSCAR CÓDIGO CIE10

El administrador del sistema podrá buscar la información de los códigos CIE10.

Entrada

Nombre de la enfermedad.

Proceso

El usuario ingresa a las distintas pantallas donde se utilizan los códigos CIE 10: recetas, diagnósticos. Aquí se ingresará el nombre de la enfermedad a buscar.

Salida

Despliegue de la información de los códigos CIE10 que se solicita.

3.3.2.20 RE20 BUSCAR CÓDIGO VADEMECUM

El administrador del sistema podrá buscar la información de los códigos VADEMECUM.

Entrada

Nombre del medicamento.

Proceso

El usuario ingresa a las distintas pantallas donde se utilizan los códigos VADEMECUM: recetas, diagnósticos. Aquí se ingresará el nombre o código del fármaco a buscar.

Salida

Despliegue de la información de los códigos VADEMECUM que se solicita.

3.3.2.21 RE21 RECUPERAR CLAVE DE USUARIO

El sistema permitirá recuperar la clave de ingreso al sistema de un usuario determinado.

Entrada

Correo electrónico del usuario.

Proceso

El usuario ingresa a la pantalla de recuperación de clave, donde ingresa el correo del usuario e ingresa los caracteres de seguridad.

Revisar la bandeja de entrada del correo para visualizar el mensaje con la clave del usuario.

Salida

Mensaje de confirmación de envío de un correo electrónico con la clave del usuario.

3.3.2.22 RE22 REPORTE DE DIAGNÓSTICOS

El usuario administrador podrá emitir un listado de los diagnósticos realizados a cada paciente. La información que se debe emitir es: fecha del diagnóstico, motivo de la cita médica, detalle del diagnóstico, receta dirigido a cada diagnóstico.

Entrada

Seleccionar un paciente de la lista de pacientes.

Proceso

Ingresar a la página de administración de pacientes, al seleccionar un paciente se desplegará una ventana con el reporte anteriormente descrito.

Salida

Se desplegará reportes con la información anteriormente definida.

3.3.2.23 RE23 REGISTRAR USUARIO ADMINISTRADOR

El usuario del sistema podrá registrar un nuevo administrador al aplicativo web.

Entrada

Datos del cliente: Cédula, nombres, apellidos, teléfono, dirección, correo electrónico, contraseña.

Proceso

El usuario ingresará a la pantalla de registro donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Confirmación del registro del nuevo usuario administrador con un mensaje.

3.3.2.24 RE24 CURVA DE CRECIMIENTO

El administrador del sistema podrá graficar la curva de crecimiento de un paciente.

Entrada

Peso en kg, talla en cm, perímetro cefálico y fecha de nacimiento.

Proceso

El usuario ingresará a la pantalla de curva de crecimiento donde ingresará los parámetros antes descritos, el usuario deberá ingresar los campos marcados con (*) lo que implica que son obligatorios.

Salida

Un punto sobre el plano cartesiano en base a los parámetros descritos.

3.3.2.25 RE25 HORARIOS DE ATENCIÓN

El administrador del sistema podrá crear horarios de atención al cliente para el consultorio médico.

Entrada

Hora de inicio y hora de fin del nuevo turno. Los turnos de atención son de 30 minutos cada uno.

Proceso

- Ingrese a la página de horarios de atención dentro del sistema web.
- Ingrese los parámetros antes descritos
- Confirme la acción dando clic sobre el botón de guardar.

Salida

Confirmación con un mensaje el registro del nuevo horario de atención.

3.3.3 REQUISITOS NO FUNCIONALES

3.3.3.1 FUNCIONALIDAD

A continuación se describirán requerimientos que aportan alguna funcionalidad adicional al sistema, sin ser considerados principales.

MODULAR

El sistema deberá estar divido por módulos independientes que se acoplarán, para la creación de un módulo se deberá tener en cuenta que este pueda funcionar independientemente de los demás.

ACCESO A LA WEB

La funcionalidad del sistema estará disponible a través de un browser. Se requiere instalar un explorador por ejemplo: MS Internet Explorer, Google Chrome o Mozilla Fire Fox.

3.3.3.2 USABILIDAD

A continuación se mostrarán los requerimientos que afectan a la capacidad del uso del sistema.

FACILIDAD DE USO

La interfaz de usuario será diseñada de tal manera que ofrezca la mayor facilidad de uso apropiada para usuarios que no cuenten con un entrenamiento adicional del sistema.

ELEMENTOS Y ACCESOS

Los elementos y accesos al sistema deben estar colocados en una parte visible y fácil de acceder.

COMPATIBILIDAD CON LOS BROWSERS

El sistema puede ejecutarse sobre MS Internet Explorer, Mozilla Fire Fox, o Google Chrome.

COMPATIBILIDAD CON WINDOWS

La interfaz será compatible con Windows.

LENGUAJE

El portal deberá tener la capacidad de exhibir la información en lenguaje Español.

3.3.3.3 CONFIABILIDAD

En este apartado se describirán los requerimientos que hacen que el sistema sea confiable.

DISPONIBILIDAD

El sistema deberá estar operable aun cuando se presenten fallas en la red de datos que afecten el acceso a la base de datos.

PRECISIÓN

Los datos registrados y entregados por el sistema deberán ser precisos, ya que se manipula tratamientos médicos que pueden afectar la salud de un paciente si estos no corresponden al paciente en correcto.

3.3.3.4 DESEMPEÑO

A continuación se mencionan los requerimientos que buscan asegurar un rendimiento óptimo del sistema.

USUARIOS CONCURRENTES

El sistema deberá soportar al menos 15 usuarios concurrentes.

SOPORTE

A continuación se detallan los requerimientos que afectan al mantenimiento del sistema.

ESTÁNDARES DE CODIFICACIÓN

Reglamentar la forma en que se implementará el código fuente del proyecto, pasando, por las variable, controles, ficheros, archivos y todo aquello que esté implicado en el código.

BIBLIOTECAS DE RECURSOS

Se creará bibliotecas en las que se almacenará cualquier recurso empleado en la creación de la aplicación, ya sean estos clases, plugins,wsdl etc.

3.3.3.5 RESTRICCIONES DE DISEÑO

En esta sección se enumera las restricciones de diseño en la aplicación que se desea desarrollar.

REQUISITOS DE LA PLATAFORMA

La parte cliente del sistema funcionará en cualquier computador con un procesador Intel Pentium 4 o superior.

LOS NAVEGADORES DE INTERNET

La interfaz basada en web para el sistema se extenderá en Firefox 4 o superior, Google Chrome 14, Internet Explorer 7 o superior, Safari 3 o superior.

3.2 CASOS DE USO

3.2.1 INTRODUCCIÓN

El análisis es una etapa fundamental dentro de la realización de una aplicación, esta etapa se trata de entender y definir un marco más amplio sobre el problema, esto conlleva a dividir los puntos principales a tratar para estudiar el comportamiento de la aplicación en el tiempo.

3.2.2 DESCRIPCIÓN GENERAL DE ACTORES

El sistema necesitará de dos actores:

Cliente: quien posee acceso a la información del consultorio como del doctor Wladimir Herrera, además de realizar citas médicas en la agenda electrónica.

Administrador: Quien es el encargado de gestionar la información relacionada con la administración de usuarios y gestionar la información relacionada con los pacientes y la agenda electrónica.

3.2.3 MODELO DE CASOS DE USO

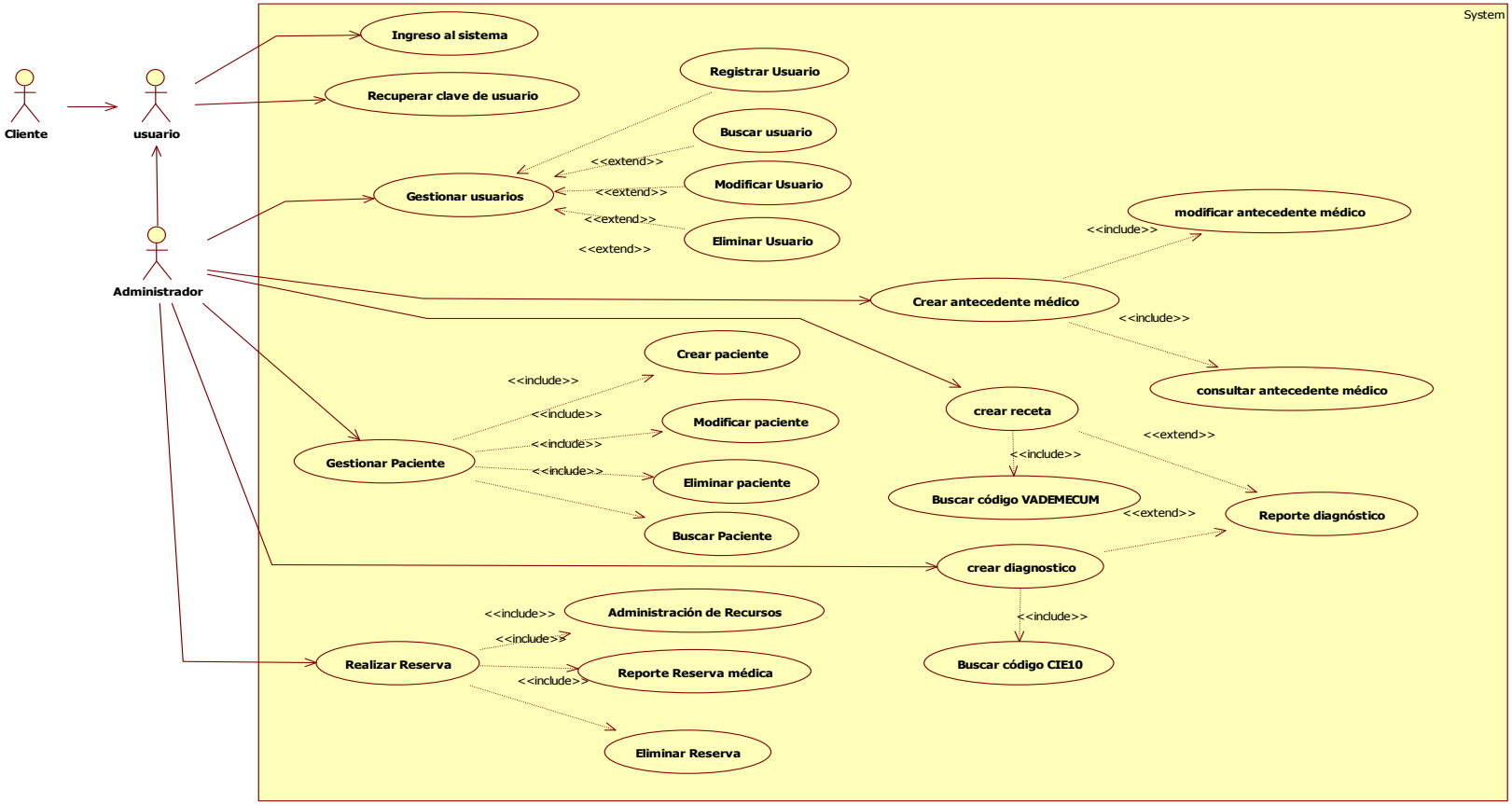


Ilustración 17: Diagrama decaso de uso general

Para tener más claro el funcionamiento se mostrará a continuación cada uno de los casos de uso a un nivel más específico.

3.2.4 ESPECIFICACIÓN DE CASO DE USO: INGRESO AL SISTEMA

Descripción

Autenticar al usuario que accede al sistema e ingresar su número de cédula y la contraseña en la página Web.

Meta

Ingresar al sistema mediante un logueo, ingresando el número de cedula y contraseña, para salvaguardar la integridad del mismo.

Actores

Cliente, administrador.

Flujo de eventos

Flujo básico

- Ingresar el número de cédula y la contraseña correspondiente para proceder con el logueo.

Flujos alternativos

- No acceso al sistema.

Precondiciones

Se debe contar con cuenta en el sistema.

Condición de éxito

El usuario ingresa al portal con sus credenciales.

Condición de fallo

Se despliega un mensaje de error “Datos Incorrectos”.

3.2.5 ESPECIFICACIÓN DE CASO DE USO: REGISTRAR USUARIO.

Descripción

Permite registrar un nuevo usuario.

Meta

Registrar un nuevo usuario con datos como: Cédula, nombres, apellido paterno, apellido materno, fecha de nacimiento, sexo, teléfono, dirección, correo electrónico, seguro médico, contraseña.

Actores

Usuario, administrador.

Flujo de eventos

Flujo básico

- El cliente accederá a la pantalla de registro de usuario.
- El cliente ingresará los datos listados anteriormente.
- El cliente pedirá a la página que guarde este nuevo registro en la base de datos.
- La página Web actualizará la base de datos con la información correspondiente.
- El cliente terminará el proceso de registro de usuario.

Flujos alternativos

- Si los campos no están completos o mal ingresados, indica un mensaje de error.

Precondiciones

El cliente debe haber accedido a la página de registro de nuevo usuario.

Condición de éxito

Se registra el usuario con éxito.

Condición de fallo

Se despliega un mensaje de error “No se pudo registrar”.

3.2.6 ESPECIFICACIÓN DE CASO DE USO: BUSCAR USUARIO.

Descripción

El administrador deberá realizar una búsqueda de un usuario determinado.

Meta

Encontrar un usuario, objeto de la búsqueda bajo los parámetros descritos.

Actores

Administrador.

Flujo de eventos

Flujo básico

- El administrador ingresa la cédula del usuario.
- Se envía la petición de búsqueda a la base de datos.
- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados obtenidos manualmente hasta dar con el deseado.
- La página Web mostrará la información del usuario seleccionado.

Flujos alternativos

- La página Web informará que el usuario no se encuentra registrado en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberá existir usuarios registrados en la base de datos.

Condición de éxito

Actualizar la página de búsqueda con la información del usuario solicitado.

Condición de fallo

Mensaje de error: “usuario no encontrado”.

3.2.7 ESPECIFICACIÓN DE CASO DE USO: MODIFICAR USUARIO.

Descripción

Permite editar datos de un usuario existente en la base de datos.

Meta

Modificar datos específicos, seleccionados de un usuario para lograr la actualización de la base de datos.

Actores

Administrador.

Flujo de eventos

Flujo básico

- El administrador podrá acceder a la vista de modificar usuarios, seleccionando anteriormente el usuario.
- El administrador puede editar los datos actuales del usuario.
- Se registra la modificación en la base de datos.

Flujos alternativos

- Si el administrador no completa los campos marcados con un asterisco (*) la información no será registrada y la actualización no se llevará a cabo.

Precondiciones

- El administrador debe estar logueado en la página Web.
- El administrador debe haber seleccionado el usuario para modificar sus datos.

Condición de éxito

Se modifica el registro de usuario y se actualiza la base de datos.

Condición de fallo

Mensaje de error: “No se pudo modificar”.

3.2.8 ESPECIFICACIÓN DE CASO DE USO: ELIMINAR USUARIO.

Descripción

Eliminar los datos de un usuario en la base de datos.

Meta

Eliminar todos los campos y el registro de un usuario seleccionado de la base de datos.

Actores

Administrador.

Flujo de eventos

Flujo básico

- El administrador podrá borrar el correspondiente usuario de la base de datos.

Flujos alternativos

- Si no existen registros en la base de datos para ser eliminados se desplegará un mensaje de error previniendo la inexistencia de registros.

Precondiciones

- El administrador debe estar logueado en la página Web.
- El administrador debe haber seleccionado un usuario.

Condición de éxito

Se despliega un mensaje de éxito indicando que el registro seleccionado se ha eliminado satisfactoriamente de la base de datos.

Condición de fallo

No eliminar el registro del usuario debido a la inexistencia del mismo, y desplegar un mensaje de error: “No se pudo eliminar”

3.2.9 ESPECIFICACIÓN DE CASO DE USO: CREAR PACIENTE.

Descripción

Permite registrar un nuevo paciente.

Meta

Registrar un nuevo paciente con datos como: apellido paterno, apellido materno, nombres, cedula del paciente, fecha de nacimiento, sexo, teléfono, dirección, correo electrónico, seguro médico, contraseña.

Actores

Administrador.

Flujo de eventos

Flujo básico

- El cliente accederá a la página de registro de pacientes.
- El cliente ingresará los datos listados anteriormente.
- El cliente pedirá a la página que guarde este nuevo registro en la base de datos.
- La página Web actualizará la base de datos con la información correspondiente.
- El cliente terminará el proceso de registro de pacientes.

Flujos alternativos

- Si los campos no están completos o mal ingresados, indica un mensaje de error.

Precondiciones

El cliente debe estar logueado en la página Web.

Condición de éxito

Se registra el paciente con éxito.

Condición de fallo

Se despliega un mensaje de error “Datos Incorrectos o Paciente ya Registrado”.

3.2.10 ESPECIFICACIÓN DE CASO DE USO: BUSCAR PACIENTE.

Descripción

El administrador deberá realizar una búsqueda de información dentro de los pacientes registrados.

Meta

Encontrar la información de un paciente bajo el parámetro: Número de cédula del paciente.

Actores

Administrador.

Flujo de eventos

Flujo básico

- El administrador ingresará a pantalla donde se gestionan los pacientes.
- El administrador ingresa la cédula del paciente.
- Se envía la petición de búsqueda a la base de datos.
- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados obtenidos manualmente hasta dar con el deseado.
- La página Web mostrará la información del paciente seleccionado.

Flujos Alternativos

- La página Web informará que el paciente deseado no se encuentra registrado en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberán existir pacientes registrados en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información del paciente solicitado.

Condición de Fallo

Mensaje de error: “Usuario no encontrado”.

3.2.11 ESPECIFICACIÓN DE CASO DE USO: MODIFICAR PACIENTE.

Descripción

Permite editar datos de un paciente existente en la base de datos.

Meta

Modificar datos específicos, seleccionados de un paciente para lograr la actualización de la base de datos.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador podrá acceder a la pantalla de gestión de pacientes, y modificar los datos descritos anteriormente.
- El administrador puede editar los datos actuales del paciente.
- Se registra la modificación en la base de datos.

Flujos Alternativos

- Si el administrador no completa los campos marcados con un asterisco (*) la información no será registrada y la actualización no se llevará a cabo.
- Desplegar mensaje de error indicando que no se pudo realizar la operación.

Precondiciones

- El administrador debe estar logueado en la página Web.
- El administrador debe haber seleccionado un paciente para realizar la modificación de datos.

Condición de Éxito

Se modifica el registro del paciente y se actualiza la base de datos.

Condición de Fallo

Mensaje de error: “No se pudo modificar”.

3.2.12 ESPECIFICACIÓN DE CASO DE USO: ELIMINAR PACIENTE.

Descripción

Eliminar los datos de un paciente en la base de datos.

Meta

Eliminar todos los campos y el registro de un paciente seleccionado de la base de datos.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador podrá borrar el correspondiente paciente de la base de datos.

Flujos Alternativos

- Si no existen registros en la base de datos para ser eliminados se desplegará un mensaje de error previniendo la inexistencia de registros.

Precondiciones

- El administrador debe estar logueado en la página Web.
- El administrador debe haber seleccionado un paciente.

Condición de Éxito

Se despliega un mensaje de éxito indicando que el registro seleccionado se ha eliminado satisfactoriamente de la base de datos.

Condición de Fallo

No eliminar el registro del paciente debido a la inexistencia del mismo, y desplegar un mensaje de error: “No se pudo Eliminado”.

3.2.13 ESPECIFICACIÓN DE CASO DE USO: REALIZAR RESERVA.

Descripción

Añade una nueva reserva médica con el doctor en una hora y fecha determinada.

Meta

Registrar una nueva reserva médica no existente en la base de datos del sistema correspondiente a un paciente registrado con anterioridad.

Actores

Administrador, cliente.

Flujo de Eventos

Flujo Básico

- El usuario ingresa a la pantalla de registro de reservas médicas y es asignado a un paciente determinado.
- El usuario accederá a la pantalla de la agenda electrónica donde Seleccionará al Paciente, seleccionará el horario, seleccionará la fecha y detallará el motivo de la consulta.
- Posteriormente, el usuario decidirá si guardar la nueva reserva o descarta los cambios.
- Al momento de guardar los cambios, se registrará los campos nombrados anteriormente en la base de datos.

Flujos Alternativos

- Mensaje de error indicando que no se pudo guardar: “cita médica no guardada”.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Debe existir pacientes registrados.

Condición de Éxito

Se registra de manera correcta una reserva médica en la base de datos, reserva correspondiente a un paciente seleccionado en la pantalla de registro de nueva reserva.

Condición de Fallo

No incrementar en la base de datos ningún registro.

3.2.14 ESPECIFICACIÓN DE CASO DE USO: ELIMINAR RESERVA.

Descripción

Eliminar los datos de una reserva de la base de datos.

Meta

Eliminación de una reserva médica existente en la base de datos correspondiente a un paciente registrado con anterioridad.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El usuario ingresa a la pantalla de reservas médicas y buscar la reserva asignada a un paciente determinado.
- El administrador será capaz de anular la cita médica por medio de un botón de confirmación.

Flujos Alternativos

- Si no existen registros en la base de datos para ser anulados se desplegará un mensaje de error previniendo la inexistencia de registros.

Precondiciones

- El administrador debe estar logueado en la página Web.
- El administrador debe haber seleccionado una reserva médica.

Condición de Éxito

Se despliega un mensaje de éxito indicando que el registro seleccionado se ha eliminado satisfactoriamente de la base de datos.

Condición de Fallo

No eliminar el registro de la reserva médica debido a la inexistencia del mismo.

3.2.15 ESPECIFICACIÓN DE CASO DE USO: REPORTE DE RESERVAS MÉDICAS.

Descripción

El administrador deberá realizar una búsqueda de una reserva médica determinada por un rango de fechas.

Meta

Encontrar una reserva médica, objeto de la búsqueda bajo los parámetros descritos.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador ingresará a la página de reportes.
- Los parámetros de búsqueda serán: fechas inicio y fin para visualizar el las citas médicas dentro de rango de fechas
- Se envía la petición de búsqueda a la base de datos.
- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.

Flujos Alternativos

- La página Web mostrará los datos encontrados en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberá existir reservas médicas registradas en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información de la reserva médica buscada.

Condición de Fallo

Visualizar el reporte de recetas médicas vació.

3.2.16 ESPECIFICACIÓN DE CASO DE USO: ADMINISTRACIÓN DE RECURSOS.

Descripción

Diferenciar la disponibilidad de la agenda electrónica a través de colores.

Meta

Diferenciar por colores cada recurso, deberá aparecer la casilla vacía cuando no se inserte nada en ese horario.

Actores

Administrador, cliente.

Flujo de Eventos

Flujo Básico

- El usuario ingresa a la pantalla de la agenda electrónica.
- El usuario podrá diferenciar la disponibilidad de la misma a través de colores que indican los horarios disponibles.

Flujos Alternativos

- No contar con disponibilidad en fecha y hora dentro de la agenda electrónica.

Precondiciones

- El usuario debe estar logueado en la página Web.
- Deberán existir disponibilidad de agenda para realizar reservas médicas.

Condición de Éxito

Disponer de casillas para agendar nuevas reservas médicas.

Condición de Fallo

Mensaje de error indicando que no hay disponibilidad en el horario escogido.

3.2.17 ESPECIFICACIÓN DE CASO DE USO: CREAR ANTECEDENTE MÉDICO

Descripción

Permite registrar un nuevo antecedente médico.

Meta

Registrar un nuevo antecedente médico del paciente con sus respectivos datos clínicos, los cuales son: fecha, antecedentes patológicos personales, enfermedades de hereditarios, quirúrgico traumático, farmacológico,

transnacionales, hábitos, Gineco Obstétricos, alérgicos, personales sociales, vacunas, otros.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El usuario ingresa a la pantalla de historia clínica, donde se detallan los antecedentes.
- El usuario ingresará los datos listados anteriormente.
- El usuario pedirá a la página que guarde este nuevo registro en la base de datos.
- La página web actualizará la base de datos con la información correspondiente.
- El cliente terminará el proceso de registro de antecedentes.

Flujos Alternativos

- Si los campos no están completos no se guardará los antecedentes médicos.

Precondiciones

- El usuario debe estar logueado en la página Web.

Condición de Éxito

Se registra el antecedente con éxito.

Condición de Fallo

Desplegar un mensaje de error: “No se pudo guardar el antecedente”.

3.2.18 Especificación de caso de Uso: Modificar antecedente médico

Descripción

Permite editar datos del antecedente medico de un paciente existente en la base de datos.

Meta

Modificar datos específicos fecha, antecedentes patológicos personales, enfermedades de hereditarios, quirúrgico traumático, farmacológico, transnacionales, hábitos, Gineco Obstétricos, alérgicos, personales sociales, vacunas, otros.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador podrá acceder a la vista de historia clínica donde se podrá modificar el respectivo antecedentes médicos.
- El administrador puede editar los datos actuales del antecedente médico.
- La página web actualizará la base de datos con la información correspondiente.
- El administrador terminará el proceso de Modificación del antecedente médico.

Flujos Alternativos

- Si el administrador no completa los campos marcados con un asterisco (*) la información no será registrada y la actualización no se completará.
- Desplegar mensaje de error previniendo la falta de ciertos registros necesarios.

Precondiciones

- El administrador debe estar logueado en la página web.
- El administrador debe haber seleccionado un antecedente médico para proceder a la modificación del mismo.

Condición de Éxito

Se modifica el registro del antecedente médico y se actualiza la base de datos.

Condición de Fallo

Mensaje de error: “No se pudo modificar”.

3.2.19 ESPECIFICACIÓN DE CASO DE USO: CONSULTAR ANTECEDENTE MÉDICO.

Descripción

El administrador deberá realizar una búsqueda de un antecedente medico determinada, ubicando un paciente mediante el número de cedula del mismo.

Meta

Encontrar un antecedente médico, bajo los parámetros de búsqueda descritos.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador ubica un paciente ingresa el número de cedula del mismo.
- Se envía la petición de búsqueda a la base de datos.
- La página web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados.
- La página web mostrará la información del paciente seleccionado.

Flujos Alternativos

- La página web informará los resultados de la base de datos.

Precondiciones

- El administrador debe estar logueado en la página web.
- Deberá existir antecedentes médicos en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información solicitada.

Condición de Fallo

Mostrar la información per tienen a los parámetros de búsqueda.

3.2.20 ESPECIFICACIÓN DE CASO DE USO: CREAR DIAGNÓSTICO.

Descripción

Permite registrar un nuevo diagnóstico.

Meta

Registrar un nuevo diagnóstico para un paciente con datos como: códigos CIE 10, fecha y detalle del diagnóstico.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador podrá acceder a la vista de registro de diagnósticos.
- El administrador ingresará los datos listados anteriormente.
- El administrador pedirá a la página que guarde este nuevo registro en la base de datos.
- La página web actualizará la base de datos con la información correspondiente.
- El administrador terminará el proceso de registro de diagnóstico.

Flujos Alternativos

- Si los campos no están completos o mal ingresados, no se guardará el diagnóstico.

Precondiciones

- El administrador debe estar logueado en la página web.

Condición de Éxito

Se registra un diagnóstico con éxito.

Condición de Fallo

Mensaje de error indicando que no se guardó el diagnóstico: “diagnóstico no se pudo guardar”

3.2.21 ESPECIFICACIÓN DE CASO DE USO: CREAR RECETA.

Descripción

Permite registrar una nueva receta médica para un paciente.

Meta

Registrar una nueva receta médica para el paciente con los datos de: seleccionar un diagnóstico para detallar una receta, indicaciones para la receta y selección del código vademécum correspondiente.

Actores

.Administrador

Flujo de Eventos

Flujo Básico

- .El administrador podrá acceder a la vista de registro de recetas.
- El administrador ingresará los datos listados anteriormente.
- El administrador pedirá a la página que guarde este nuevo registro en la base de datos.
- La página web actualizará la base de datos con la información correspondiente.
- El administrador terminará el proceso de registro de diagnóstico.

Flujos Alternativos

- . Si los campos no están completos o mal ingresados, no se guardará la receta.

Precondiciones

- El administrador debe estar logueado en la página web.

Condición de Éxito

- Se registra un diagnóstico con éxito.

Condición de Fallo

Mensaje de error indicando que no se guardó la receta: “receta no se pudo guardar”.

3.2.22 Especificación de caso de Uso: Buscar código CIE 10.

Descripción

El administrador deberá realizar una búsqueda de información dentro de los códigos CIE 10.

Meta

Encontrar un código y nombre de una enfermedad determinada.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador ingresará a pantalla de recetas, diagnósticos, donde habitualmente se utilizan los códigos CIE 10.
- El administrador ingresa el nombre de la enfermedad.
- Se envía la petición de búsqueda a la base de datos.
- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados obtenidos manualmente hasta dar con el deseado.

- La página Web mostrará la información del código seleccionado.

Flujos Alternativos

- La página Web informará que el código deseado no se encuentra registrado en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberán existir los códigos CIE 10 registrados en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información del código solicitado.

Condición de Fallo

Mensaje de que no lo encontró ningún resultado.

3.2.23 ESPECIFICACIÓN DE CASO DE USO: BUSCAR CÓDIGO VADEMECUM.

Descripción

El administrador deberá realizar una búsqueda de información dentro de los códigos VADEMECUM.

Meta

Encontrar un código y nombre de un medicamento determinada.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador ingresará a pantalla de recetas, diagnósticos, donde habitualmente se utilizan los códigos VADEMECUM.
- El administrador ingresa el nombre del medicamento.
- Se envía la petición de búsqueda a la base de datos.

- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados obtenidos manualmente hasta dar con el deseado.
- La página Web mostrará la información del código seleccionado.

Flujos Alternativos

- La página Web informará que el código deseado no se encuentra registrado en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberán existir los códigos VADEMECUM registrados en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información del código solicitado.

Condición de Fallo

Mensaje de que no lo encontró ningún resultado.

3.2.24 ESPECIFICACIÓN DE CASO DE USO: RECUPERAR CLAVE DE USUARIO.

Descripción

Permite recuperar la clave de ingreso al sistema de un usuario determinado.

Meta

Obtener la clave de acceso al sistema.

Actores

Administrador, usuarios.

Flujo de Eventos

Flujo Básico

- Ingresar a la página de recuperar cuenta.
- Ingresar el correo electrónico del usuario que se desee recuperar la cuenta.
- Ingresar los caracteres de seguridad del captcha.
- Confirmar el proceso con el botón de enviar.
- Revisar el mensaje en la bandeja de entrada del correo electrónico del usuario.

Precondiciones

- Tener cuenta en el sistema.
- Haber accedido a la pantalla de recuperar cuenta.

Condición de Éxito

Recibir un correo electrónico con la clave del usuario.

Condición de Fallo

Mensaje indicando: “El correo no está registrado”.

3.2.25 ESPECIFICACIÓN DE CASO DE USO: BUSCAR CÓDIGO VADEMECUM.

Descripción

El administrador deberá realizar una búsqueda de información dentro de los códigos VADEMECUM.

Meta

Encontrar un código y nombre de un medicamento determinada.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El administrador ingresará a pantalla de recetas, diagnósticos, donde habitualmente se utilizan los códigos VADEMECUM.
- El administrador ingresa el nombre del medicamento.
- Se envía la petición de búsqueda a la base de datos.
- La página Web mostrará todos los resultados que cumplan con los criterios de búsqueda.
- El administrador revisa los resultados obtenidos manualmente hasta dar con el deseado.
- La página Web mostrará la información del código seleccionado.

Flujos Alternativos

- La página Web informará que el código deseado no se encuentra registrado en la base de datos.

Precondiciones

- El administrador debe estar logueado en la página Web.
- Deberán existir los códigos VADEMECUM registrados en la base de datos.

Condición de Éxito

Actualizar la página de búsqueda con la información del código solicitado.

Condición de Fallo

Mensaje de que no lo encontró ningún resultado.

3.2.26 ESPECIFICACIÓN DE CASO DE USO: REPORTE DE DIAGNÓSTICOS.

Descripción

Permite emitir un listado de los diagnósticos realizados a cada paciente.

Meta

Visualizar un reporte con los diagnósticos de cada paciente.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- Ingresar a la página de pacientes.
- Seleccionar un paciente.
- Dar clic sobre el paciente deseado.
- Visualizar la información solicitada.

Precondiciones

- Tener cuenta en el sistema.
- Haber accedido a la pantalla de recuperar cuenta.

Condición de Éxito

Visualizar un reporte con los diagnósticos y recetas dirigidas al paciente seleccionado.

Condición de Fallo

Mensaje de que no lo encontró ningún resultado.

3.2.27 ESPECIFICACIÓN DE CASO DE USO: REGISTRAR USUARIO ADMINISTRADOR.

Descripción

Permite registrar un nuevo usuario administrador al sistema.

Meta

Registrar un nuevo usuario con datos como: Cédula, nombres, apellidos, teléfono, dirección, correo electrónico, contraseña.

Actores

Administrador.

Flujo de Eventos

Flujo Básico

- El cliente accederá a la pantalla de registro de usuario administrador.
- El cliente ingresará los datos listados anteriormente.
- El cliente pedirá a la página que guarde este nuevo registro en la base de datos.
- La página Web actualizará la base de datos con la información correspondiente.
- El cliente terminará el proceso de registro de usuario.

Flujos alternativos

- Si los campos no están completos o mal ingresados, indica un mensaje de error.

Precondiciones

El cliente debe haber accedido a la página de registro de nuevo usuario administrador.

Condición de éxito

Se registra el usuario con éxito.

Condición de fallo

Se despliega un mensaje de error “No se pudo registrar”.

3.3 MODELO DE DATOS

3.3.1 MODELO LÓGICO

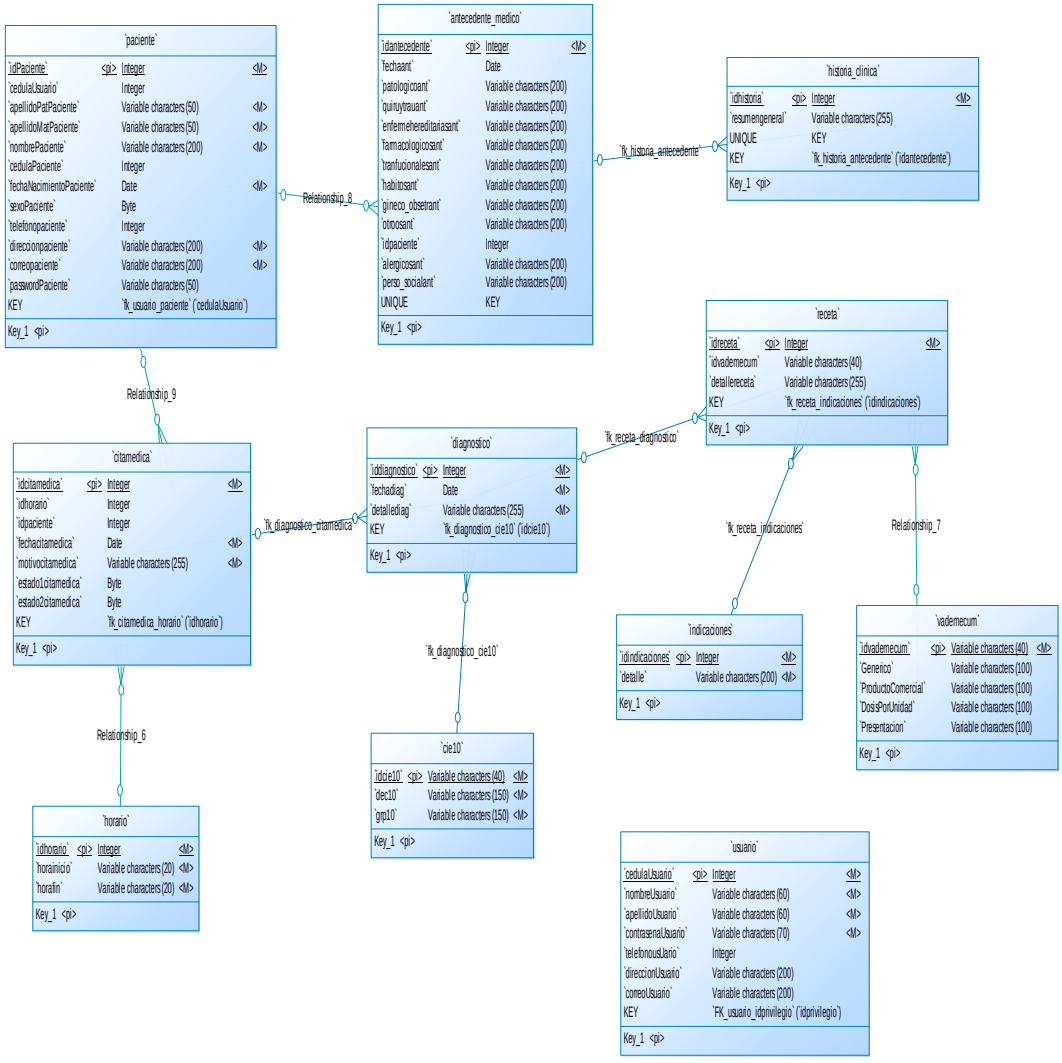


Ilustración 18: Modelo lógico

3.3.2 MODELO FÍSICO

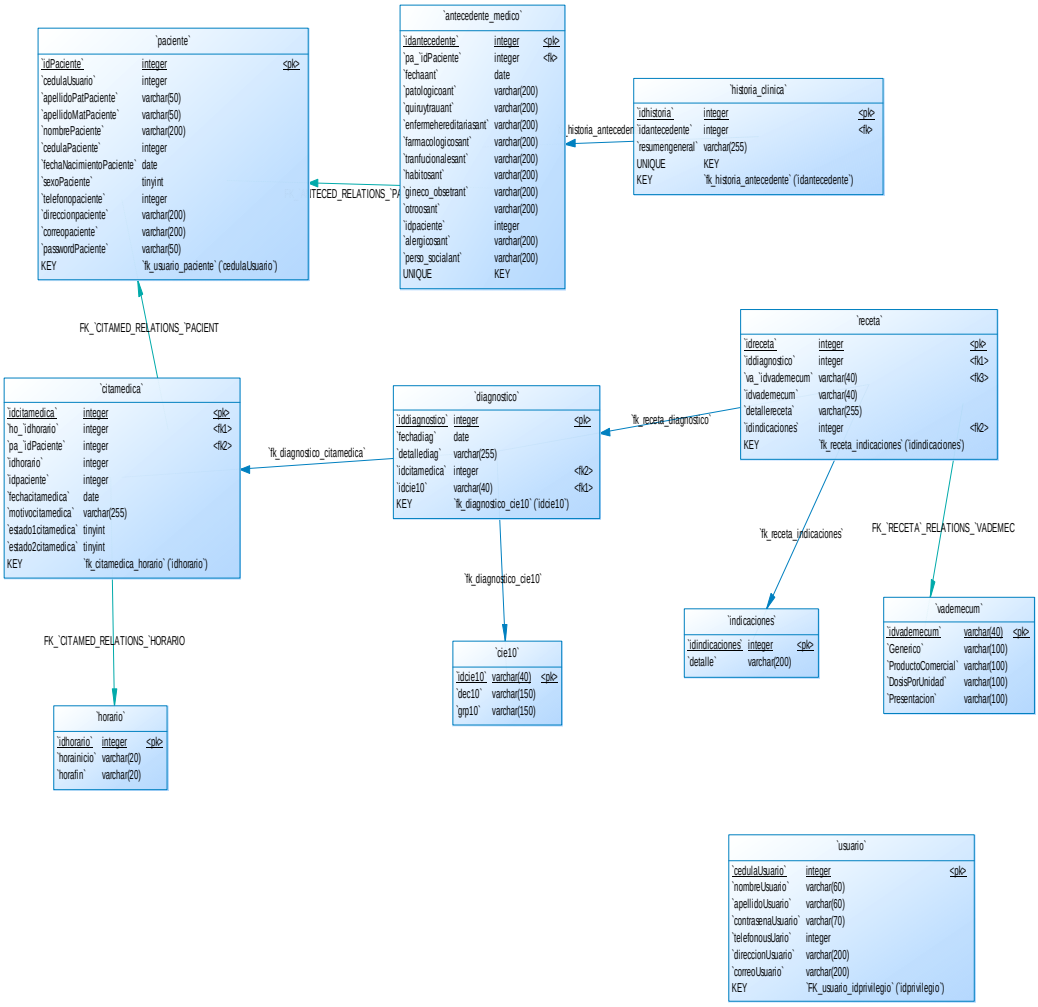


Ilustración 19: Modelo Físico

3.4 DISEÑO DE LA ARQUITECTURA

Uno de los patrones más conocidos en el desarrollo web es el patrón MVC (Modelo Vista Controlador). Este patrón es el que permite/obliga a separar la lógica de control, lógica de negocio y la lógica de presentación.

3.4.1 ARQUITECTURA LÓGICA

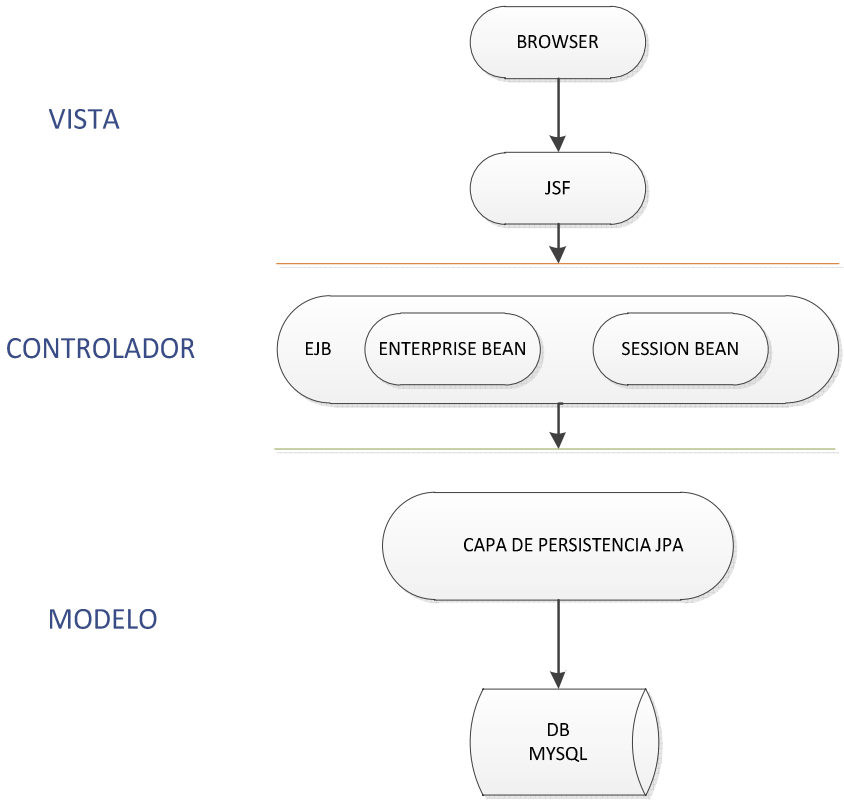


Ilustración 20: Arquitectura Lógica del Sistema.

3.4.2 ARQUITECTURA FÍSICA

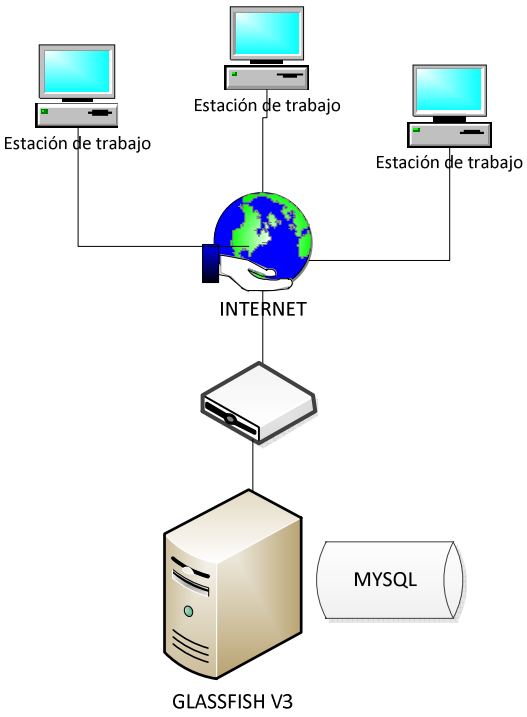


Ilustración 21: Arquitectura Física del Sistema.

3.5 PRUEBAS

3.5.1 PRUEBAS DE ACEPTACIÓN

Administración de Usuarios

- **¿Qué pasaría si se ingresa un nombre de usuario y contraseñas inválidas?**

El sistema no permitirá el ingreso al aplicativo web, indicando un mensaje de error “Datos incorrectos”.

- **¿Qué pasaría si se registra un usuario con datos incorrectos?**

El sistema validará cada entrada de la información de registro, indicando el campo errado con un mensaje “Error en el campo”.

- **Ingresar al sistema con usuario estándar y visualizar únicamente la opción de la agenda electrónica.**
- **Ingresar al sistema con usuario administrador y visualizar todas las opciones del sistema.**

Administración de Pacientes

- **¿Qué pasaría si se registra información incorrecta de un paciente?**

El sistema valida cada entrada de datos, indicando con un mensaje la falta de algún campo o error de digitación “Error en el campo”.

- **Buscar paciente por cédula.**
- **Modificar información de un paciente y la lista se debe actualizar.**
- **Registrar un nuevo paciente y la lista se debe actualizar.**

Antecedente Médico

- Modificar la información del antecedente satisfactoriamente.
- Eliminar la información del antecedente satisfactoriamente.

Lista de diagnósticos

Al seleccionar un paciente se debe desplegar la lista de diagnósticos correspondientes al paciente seleccionado.

Curvas de crecimiento

- ingresar los parámetros correspondientes y visualizar el punto actual del paciente masculino.

- Ingresar los parámetros correspondientes y visualizar el punto actual del paciente femenino.

Agenda Electrónica para Usuario Estándar

- **Realizar una reversa médica en un día determinado.**
- **¿Qué pasaría si se realiza dos reversa médica en un mismo horario?**

  El sistema informará con un mensaje de error que ya se encuentra ocupado dicho horario de atención “Horario no disponible”.
- **Eliminar únicamente las reservas médicas del usuario que se encuentra logueado.**

Agenda Electrónica para Usuario Administrador

- **Realizar una reversa médica en un día determinado.**
- **¿Qué pasaría si se realiza dos reversa médica en un mismo horario?**

El sistema informará con un mensaje de error que ya se encuentra ocupado dicho horario de atención “Horario no disponible”.

- **Eliminar cualquier reserva médica.**
- **Modificar cualquier reserva médica.**
- **Atender una cita médica.**

Diagnósticos y Recetas

- Buscar un código CIE 10.
- Buscar un código VADEMECUM.

Integración CIE10 Y VADEMECUM

- Introducir las primeras letras del nombre de una enfermedad, y se debe desplegar las posibles alternativas de la enfermedad deseada.
- Introducir las primeras letras del nombre de un medicamento, y se debe desplegar las posibles alternativas del medicamento deseado.

Página Informativa

- Crear un horario atención.
- Eliminar un horario de atención.

Listado de atención de pacientes.

Ingresar un rango de fechas y visualizar el listado pertinente.

3.5.2 PRUEBAS DEL SISTEMA

Tabla 2: Pruebas del sistema

ESTADO	DESCRIPCIÓN DEL TEST	RESULTADO	TIEMPO RESPUESTA
Culminado	Permitir al Usuario Administrador y al Usuario Estándar recuperar su clave de acceso al sistema.	Pasa, no se ha informado de ningún error.	8 segundos
Culminado	Permitir registrar un nuevo usuario estándar al sistema.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir ingresar al sistema como usuario administrador y usuario estándar.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir realizar agendas médicas escogiendo el horario deseado, como usuario estándar.	Pasa, no se ha informado de ningún error.	2 segundos

CONTINÚA

Culminado	Permitir registrar un usuario administrador al aplicativo web.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir modificar los datos personales de usuario Administrador que se encuentra registrado en el aplicativo.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir eliminar un usuario administrador del aplicativo web	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir registrar un paciente (usuario estándar) por parte de un Usuario Administrador al aplicativo web.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir modificar los datos personales de un paciente (usuario estándar) que se encuentra registrado en el aplicativo.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir eliminar un paciente del aplicativo web.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir registrar un nuevo antecedente médico por parte de un Usuario Administrador al aplicativo web.	Pasa, no se ha informado de ningún error.	2 segundos

CONTINÚA

Culminado	Permitir modificar un antecedente médico de un paciente (usuario estándar) que se encuentra registrado en el aplicativo.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir visualizar todos los diagnósticos y recetas de un paciente.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir graficar las curvas de crecimiento para niños y niñas	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir realizar agendas médicas escogiendo un horario y paciente deseado, como usuario administrador.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir atender a un paciente desde la agenda del médico.	Pasa, no se ha informado de ningún error.	6 segundos
Culminado	Permitir prescribir un diagnóstico y receta para la cita médica que se está atendiendo.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir crear horarios de atención.	Pasa, no se ha informado de ningún error.	2 segundos
Culminado	Permitir visualizar todos los datos de las citas médicas.	Pasa, no se ha informado de ningún error.	2 segundos

3.5.3 PRUEBAS UNITARIAS

El Desarrollo Dirigido por Test al fundamentarse en la ejecución de pruebas, utiliza herramientas para llevar a cabo pruebas unitarias, para aquello se utilizará framework JUnit. Este tipo de herramientas hacen que el esfuerzo y el trabajo en el Desarrollo Dirigido por Test se reduzcan, permitiendo que el desarrollador pueda centrarse en fragmentos de código que cumplen determinado funciones.

3.5.3.1 Administración de Usuarios

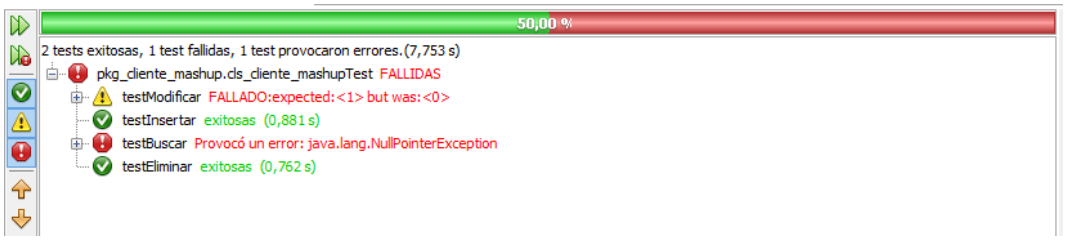
Test 1

```
@Test
   public void testInsertar() {
      System.out.println("insertar");
      String cedula="1718384215";
      String nombre="Dario";
      String apellido="Gualoto";
      String contrasenia="darios1234";
      String telefono="23660188";
      String direccion="Ejercito";
      String correo="dario12@hotmail.com";
      String privilegio="1";
      int result=port.insertar(cedula, nombre, apellido, contrasenia, telefono, direccion,
correo, privilegio);
       assertEquals(1, result);
```

```
@Test
    public void testEliminar() {
        System.out.println("eliminar");
        String cedula="1718384215";
        int result=port.eliminar(cedula);
        assertEquals(1, result);
        //fail("The test case is a prototype.");
    }
```

Resultado de la prueba

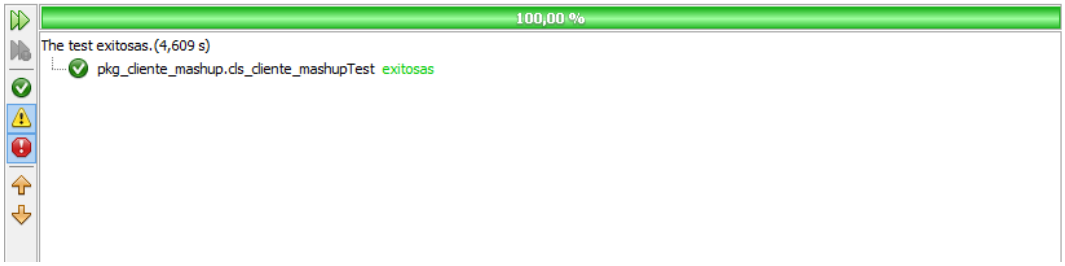
Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



Test 2

```
@Test
    public void testModificar() {
        System.out.println("modificar");
        String cedula="1718384215";
        String nombre="Dario Santiago";
        String apellido="Gualoto Alvarez";
        String contrasenia="darios1";
        String telefono="23660183";
        String direccion="Ejercito 2";
        String correo="dario11@hotmail.com";
        String privilegio="1";
        int result=port.modificar(cedula, nombre, apellido, contrasenia, telefono, direccion,
correo, privilegio);
```

Resultado de la prueba



Al ejecutar las pruebas conjuntamente el test pasa los requerimientos.

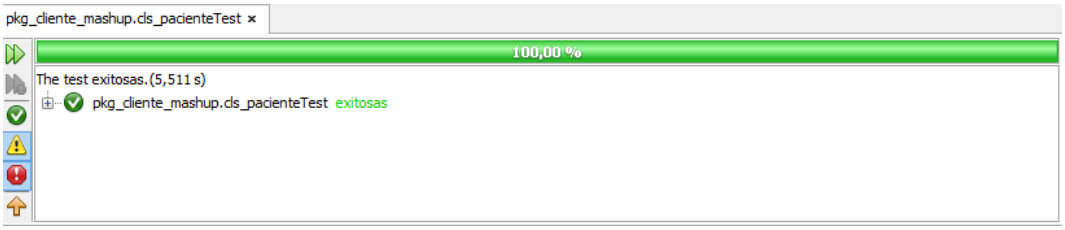
3.5.3.2 Administración de Pacientes

Test 1

```
@Test
    public void testInsertar1() {
        System.out.println("insertar1");
        String c1="1234567890";
        String apellidoPaterno="Quiroz";
        String apellidoMaterno="Donoso";
        String nombre ="Diego";
        String Cedula ="1718160680";
        String fecha = "1986/03/12";
        String sexo="1";
        String telefono="02343212";
        String direccion="Chillogallo";
        String correo="diegoDonoso@hotmail.com";
        String password="diego1234";
        String seguro="Ecuasanita";
        int result= port.insertarPaciente(c1, apellidoPaterno, apellidoMaterno, nombre, Cedula, fecha,
sexo, telefono, direccion, correo, password, seguro);
        assertEquals(1, result);
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.

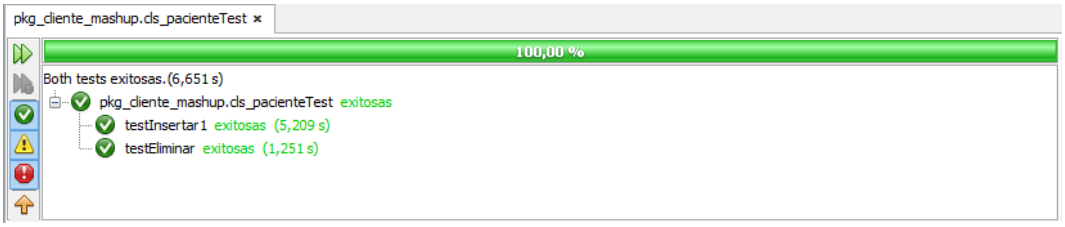


Test 2

```
@Test
    public void testInsertar1() {
        System.out.println("insertar1");
        String c1="1234567890";
        String apellidoPaterno="Quiroz";
        String apellidoMaterno="Donoso";
        String nombre ="Diego";
        String Cedula ="1718160680";
        String fecha = "1986/03/12";
        String sexo="1";
        String telefono="02343212";
        String direccion="Chillogallo";
        String correo="diegoDonoso@hotmail.com";
        String password="diego1234";
        String seguro="Ecuasanita";
        int result= port.insertarPaciente(c1, apellidoPaterno, apellidoMaterno, nombre, Cedula, fecha,
sexo, telefono, direccion, correo, password, seguro);
        assertEquals(1, result);
        //fail("The test case is a prototype.");
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.

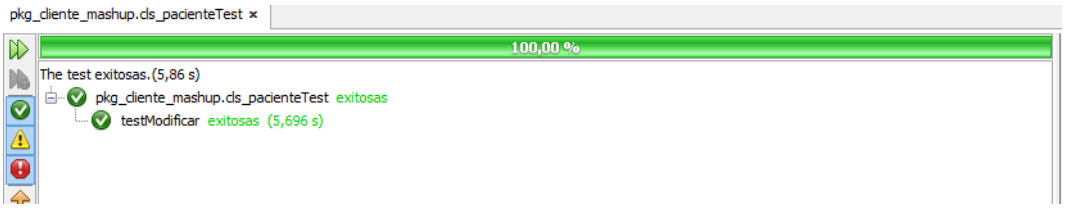


Test 3

```
@Test
   public void testModificar() {
      System.out.println("modificar");
     String c1="1234567890";
      String apellidoPaterno="Quiroz";
      String apellidoMaterno="Cuenca";
      String nombre ="Fabricio";
      String Cedula ="1718160680";
      String fecha = "1986/04/12";
      String sexo="1";
      String telefono="02343222";
      String direccion="Ecuatoriana";
      String correo="diegoDonoso1234@hotmail.com";
      String password="dieg";
      String seguro="Ecuasanita123";
      String id="46";
      int result=port.modificarPaciente(id,c1, apellidoPaterno, apellidoMaterno, nombre, Cedula,
fecha, sexo, telefono, direccion, correo, password, seguro);
      assertEquals(1, result);
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



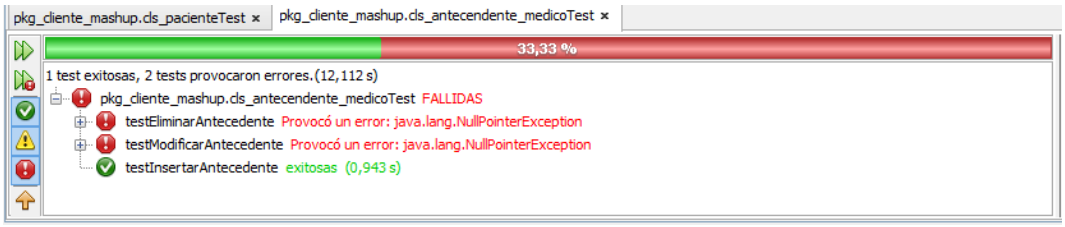
3.5.3.3 Antecedente Médico

Test 1

```
@Test
    public void testInsertarAntecedente() {
        System.out.println("insertarAntecedente");
        String fecha="2014/04/15";
        String patologico="patologico1";
        String quirur="qurur1";
        String enfer="enfer1";
        String farma="farma1";
        String trans="trans1";
        String habitos="habitos1";
        String gineco="gineco1";
        String otros="otros1";
        String idpaciente="46";
        String alergicos="alergicos1";
        String personales="persoanles1";
        String vacunas="vacunas1";
        int
result=port.insertarAntecedente(fecha,patologico,quirur,enfer,farma,trans,habitos,gineco,otros,idpa
ciente,alergicos,personales,vacunas);
```

Resultado de la prueba

El código necesita refactorizar para pasar las pruebas conjuntamente.

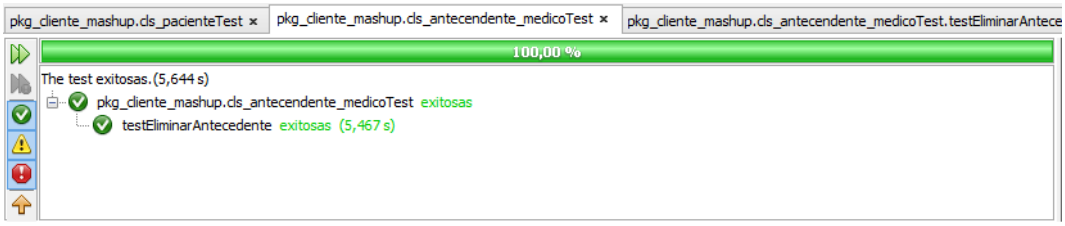


Test 2

```
@Test
    public void testEliminarAntecedente() throws Exception {
        System.out.println("eliminarAntecedente");
        String idpaciente="46";
        port.eliminarAntecedente(idpaciente);
    }
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



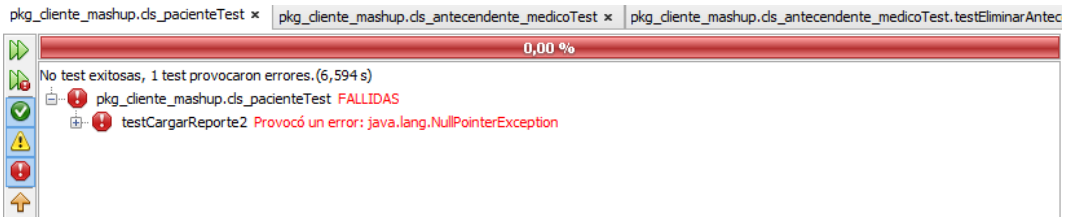
3.5.3.4 Lista de diagnósticos

Test 1

```
@Test
    public void testCargarReporte2() throws Exception {
        System.out.println("cargarReporte2");
        cls_paciente instance = new cls_paciente();
        instance.cargarReporte2();
}
```

Resultado de la prueba

Al inicio el test no pasa debido a la falta de código.

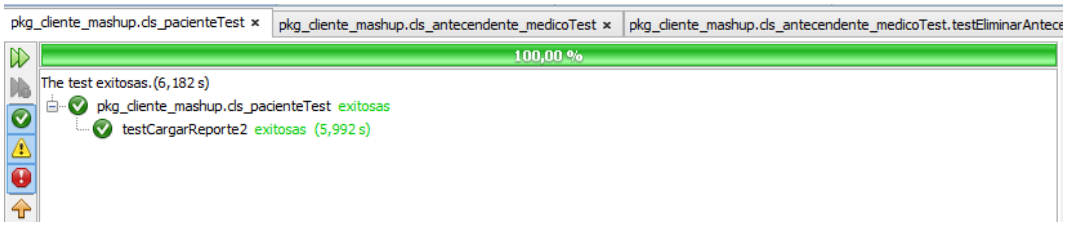


Test 2

```
@Test
    public void testCargarReporte2() throws Exception {
        System.out.println("cargarReporte2");
        cls_paciente instance = new cls_paciente();
        List<AnyTypeArray> Lista1 = null;
        List<AnyTypeArray> Lista2 = null;
        port.reporteDiagnostico("40");
        assertEquals(Lista1, Lista2);
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



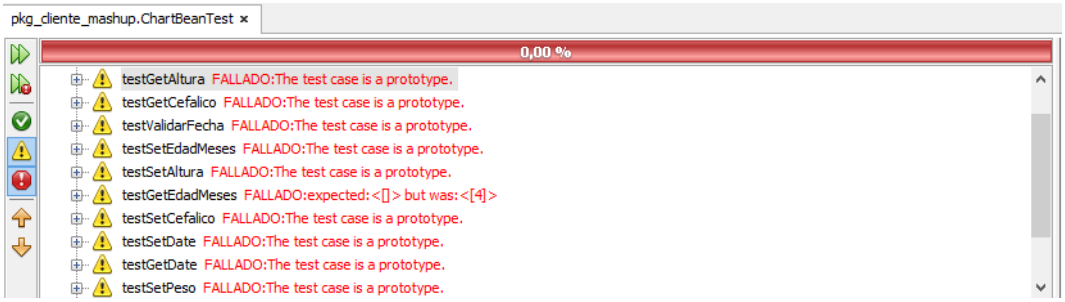
3.5.3.5 Curvas de crecimiento

Test 1

```
@Test
    public void testGetCartesianModel() {
        System.out.println("getCartesianModel");
        ChartBean instance = new ChartBean();
        CartesianChartModel expResult = null;
        CartesianChartModel result = instance.getCartesianModel();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call
to fail.
        fail("The test case is a prototype.");
    }
```

Resultado de la prueba

Al inicio el test no pasa debido a la falta de código.

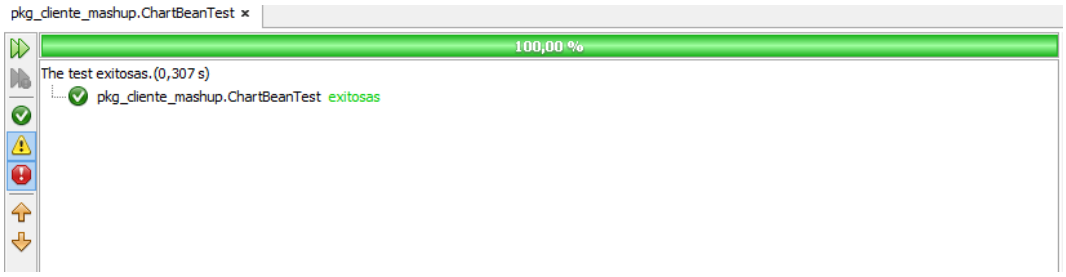


Test 2

```
@Test
    public void testCreateCartesianModel() {
        System.out.println("createCartesianModel");
        ChartBean instance = new ChartBean();
        CartesianChartModel cartesianModel = new CartesianChartModel();
        instance.createCartesianModel();
        ChartSeries linea = new ChartSeries();
        ChartSeries result = new ChartSeries();
        linea.set(14,74);
        cartesianModel.addSeries(linea);
    }
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



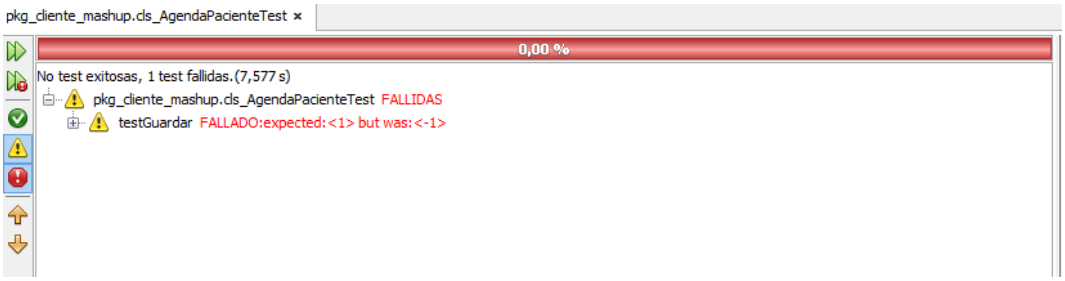
3.5.3.6 Agenda Electrónica para Usuario Estándar

Test 1

```
@Test
    public void testGuardar() throws Exception {
        System.out.println("guardar");
        cls_AgendaPaciente instance = new cls_AgendaPaciente();
        ActionEvent actionEvent = null;
        String idhoraio="108";
        String idpaciente="46";
        String fechaCitamedica="2014/04/15";
        String motivoCitamedica="Fiebre";
        String parame1="";
        String parame2="";
        int result= port.insertarCita(idhoraio, idpaciente, fechaCitamedica, motivoCitamedica,
parame1,parame2);
        assertEquals(1, result);
```

Resultado de la prueba

Al inicio el test no pasa debido a la falta de código.

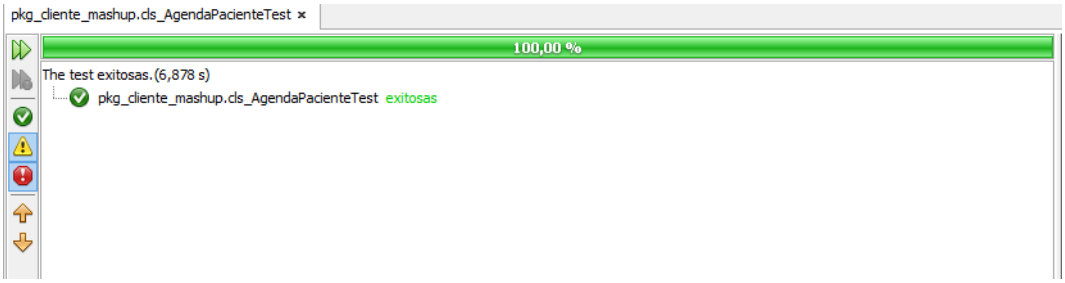


Test 2

```
@Test
   public void testGuardar() throws Exception {
      System.out.println("guardar");
      cls_AgendaPaciente instance = new cls_AgendaPaciente();
      ActionEvent actionEvent = null;
      String idhoraio="108";
      String idpaciente="46";
      String fechaCitamedica="2014/04/15";
      String motivoCitamedica="Fiebre";
      String parame1="0";
      String parame2="0";
      int result= port.insertarCita(idhoraio, idpaciente, fechaCitamedica,
motivoCitamedica, parame1,parame2);
      assertEquals(1, result);
   `
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



Test 3

```
@Test
    public void testValidarFecha_3args() throws
DatatypeConfigurationException, ParseException {
        System.out.println("validarFecha");
        int dia = 0;
        int mes = 0;
        int ano = 0;
        cls_AgendaPaciente instance = new cls_AgendaPaciente();
        String expResult = "2014 04 15";
        String result = instance.validarFecha(dia, mes, ano);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call
to fail.
        //fail("The test case is a prototype.");
    }
```

Resultado de la prueba

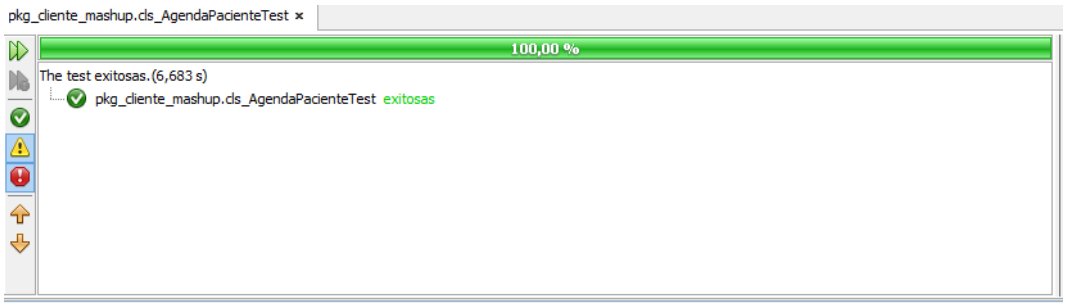
Al inicio el test no pasa debido a la falta de código.

Test 4

```
@Test
    public void testValidarFecha_3args() throws
DatatypeConfigurationException, ParseException {
        System.out.println("validarFecha");
        int dia = 15;
        int mes = 04;
        int ano = 2014;
        cls_AgendaPaciente instance = new cls_AgendaPaciente();
        String expResult = "2014 04 15";
        String result = instance.validarFecha(dia, mes, ano);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call
to fail.
        //fail("The test case is a prototype.");
    }
```

Resultado de la prueba

Al enviar datos correctamente al método descrito, se puede garantizar que el caso de prueba ha pasado exitosamente.



3.6 CUADRO COMPARATIVO

TDD Y PRUEBAS UNITARIAS

Tabla 3: Comparación entre TDD y las Pruebas Unitarias.

PARÁMETROS	TDD	PRUEBAS UNITARIAS
Propósito	TDD se utiliza para conducir el diseño de una aplicación. TDD se utiliza para expresar qué código debe diseñarse antes de escribirlo.	El propósito de una prueba unitaria es la confianza del desarrollador que una parte en particular de su aplicación se comporta de la manera que espera. Esto es muy valioso para las pruebas de regresión. Si se modifica el código que está cubierto por las pruebas unitarias, entonces puede utilizar las pruebas para determinar inmediatamente si se ha cambiado la funcionalidad existente.

CONTINÚA

Surge para	TDD surgió de una reacción al desarrollo en cascada. Una meta importante del TDD es el diseño incremental y el diseño evolutivo. En lugar de diseñar una aplicación de una sola vez, se diseña una aplicación incrementalmente-por-prueba.	Al escribir una prueba unitaria, sólo se debe considerar lo que la aplicación pueda hacer. Después de escribir la prueba, se toman decisiones de implementación. Finalmente, después de la implementación de código, se tiene en cuenta cómo se puede refactorizar la aplicación para tener un mejor diseño.
Diseño	Cuando se aplica TDD, se escribe la prueba sin tomar ninguna decisión de diseño.	Las pruebas unitarias en su mayoría están ligadas a la respuesta de una petición realizada desde la interfaz externa del proyecto.
Cubre	Eventualmente el diseño de la aplicación va evolucionando, el código que originalmente figuraba en una clase podría terminar en múltiples clases.	Las pruebas unitarias solo cubren una unidad de código a la vez.

Tabla 4: Comparación entre TDD y las Pruebas de Integración.

PARÁMETROS	TDD	PRUEBAS DE INTEGRACIÓN
Propósito	TDD se utiliza para conducir el diseño de una aplicación. TDD se utiliza para expresar qué código debe diseñarse antes de escribirlo.	Se utiliza para verificar que una aplicación se comporta en la forma en que un cliente espera. Una prueba de aceptación se crea normalmente en colaboración con un cliente como una forma de determinar si se han cumplido los requisitos del cliente.
Dirigido a	Una prueba de TDD, se crea para el beneficio de un desarrollador.	La audiencia para una prueba de aceptación es diferente a la audiencia para una prueba de TDD. Se crea una prueba de aceptación para el beneficio del cliente.
Marco de trabajo	TDD se crea un marco de pruebas unitarias.	Una prueba de aceptación no se crea un marco de pruebas unitarias. Las pruebas de aceptación del cliente se crean con un marco de pruebas de aceptación.

CONTINÚA

		Estos marcos de pruebas de aceptación le permiten probar de extremo a extremo la aplicación.
Desempeño	TDD necesitan ejecutarse muy rápido. Se debe ejecutar las pruebas TDD periódicamente y cada vez que se realiza un cambio en el código. Debido a que una aplicación podría incluir cientos de pruebas TDD.	Una prueba de aceptación significa que todo está conectado. Realizar una prueba de aceptación en el servidor web real y el servidor de base de datos.

Conclusión

- TDD es similar tanto a las pruebas unitarias como a las pruebas de integración, pero no es lo mismo. se utiliza TDD para conducir el diseño de una aplicación. TDD expresa lo que el código debe hacer antes de que realmente se escribe el código de la aplicación.

- Al igual que las pruebas unitarias, TDD pueden ser utilizado para las pruebas de regresión. Se puede utilizar TDD para determinar inmediatamente si un cambio altero la funcionalidad de la aplicación. Sin embargo, a diferencia de las pruebas unitarias, TDD no prueba necesariamente una parte de código de forma aislada, si no, cubre

una funcionalidad consecutiva, un desarrollo futuro en el código de la aplicación.

- TDD funciona como mini-pruebas de aceptación. Se utiliza TDD para expresar la funcionalidad que necesita ser implementada a continuación. Sin embargo, a diferencia de una prueba de aceptación, una prueba TDD no es una prueba que cubre el proyecto de extremo a extremo. Una prueba TDD no interactúa con una base de datos en vivo o un servidor web.

3.7 DESARROLLO

Ingreso al sistema

Para el ingreso al sistema se necesita la cédula y contraseña del usuario, estos datos pasan por un proceso de verificación que constata la existencia de esta información en la base de datos.

Ilustración 22: ingreso al Sistema

Recuperar contraseña

Cada usuario registrado en el sistema cuenta con una dirección de correo electrónico único, el cual es ingresado en la pantalla de recuperación de contraseña.

El sistema validará la cuenta de correo electrónico y el código captcha, posteriormente se envía la información al correo electrónico, la cual es constatada por el usuario.

Ilustración 23: Recuperar clave.

Registro de nuevo usuario al sistema

El usuario ingresará la información solicitada por el sistema, la cual pasa por un proceso de verificación para almacenarla en la base de datos.

Ilustración 24: Nuevo usuario.

Administradores

El usuario ingresa la información solicitada por el sistema, la cual es verificada y almacenada en la base de datos.

Ilustración 25: Administradores.

Administración de pacientes

La administración de pacientes, consta de funciones como:

- Nuevo paciente.
- Buscar paciente.
- Modificar información de paciente.
- Eliminar Paciente.
- Visualizar antecedentes.
- Visualizar diagnósticos y recetas.

Los campos pasan por un proceso de validación para almacenar información correcta en la base de datos.

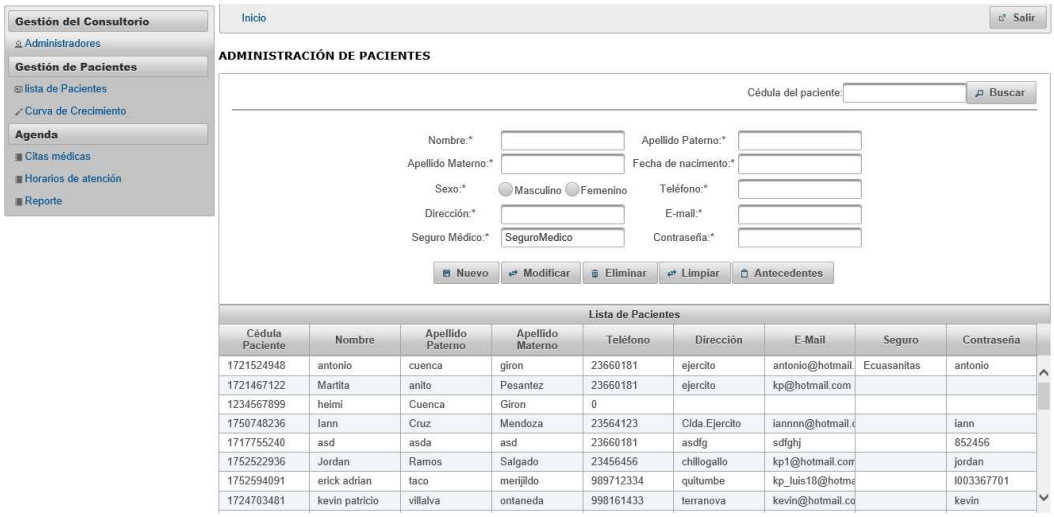


Ilustración 26: Administración de pacientes.

Antecedentes médicos

Después de localizar un paciente se ingresa a los antecedentes del mismo, el proceso se basa en buscar la información utilizando el id del paciente.

Los antecedentes pueden modificarse e eliminarse dependiendo la necesidad.

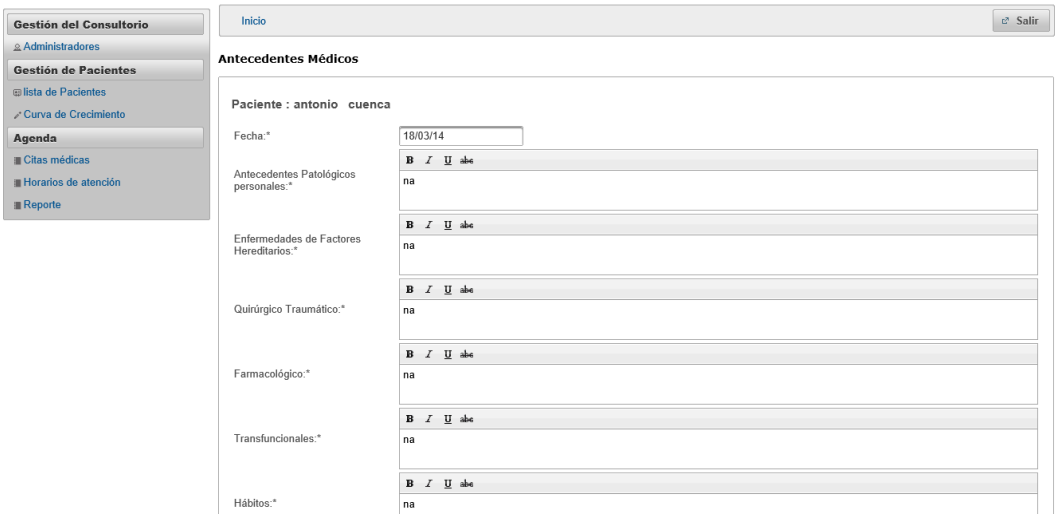


Ilustración 27: Antecedentes médicos.

Curva de crecimiento

La curva de crecimiento se fundamenta en graficar un punto dentro de un rango de edad, peso, longitud y perímetro encefálico.

Estos datos pasan por un proceso el cual delimita los puntos x,y para dibujarlos en un cuadro cartesiano.

Ilustración 28: Curva de crecimiento.

Ilustración 29: Curva de crecimiento.

Agenda electrónica

La agenda está diseñada para almacenar citas médicas bajos los siguientes parámetros:

- Horarios.
- Motivo de la consulta
- Fecha de la cita.
- Paciente.

No se puede agendar antes de la fecha actual, no se puede realizar la acción si no ingresa todo los campos descritos.

Además el usuario que se encuentre logueado puede visualizar las citas médicas pero no puede manipularlas, solo las creadas por el usuario actual.

La agenda del administrador cuenta con la funcionalidad adicional de atender directamente a un paciente, y acceder a sus antecedentes, diagnósticos y recetas médicas.

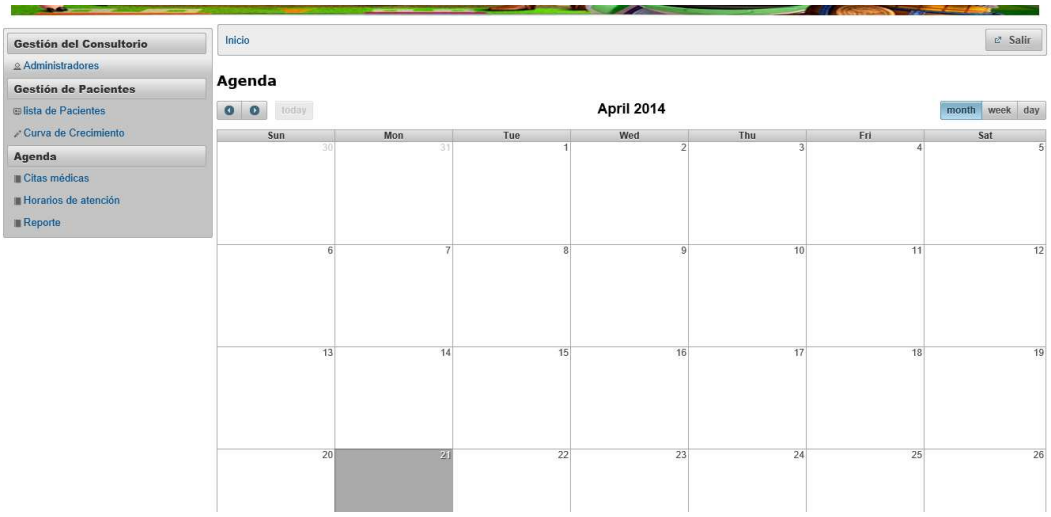


Ilustración 30: Agenda electrónica.

Horarios de atención

El usuario puede diseñar sus horarios de atención de consulta desde esta pantalla, ingresando una hora de inicio y fin de consulta médica.

Ilustración 31: Horarios de atención.

Reporte de atención de pacientes

Para el reporte se parametriza un rango de fechas para realizar un consulta a la base de datos, de esta forma se obtiene un listado con una sumatoria de las citas médicas atendidas por el doctor Wladimir Herrera.

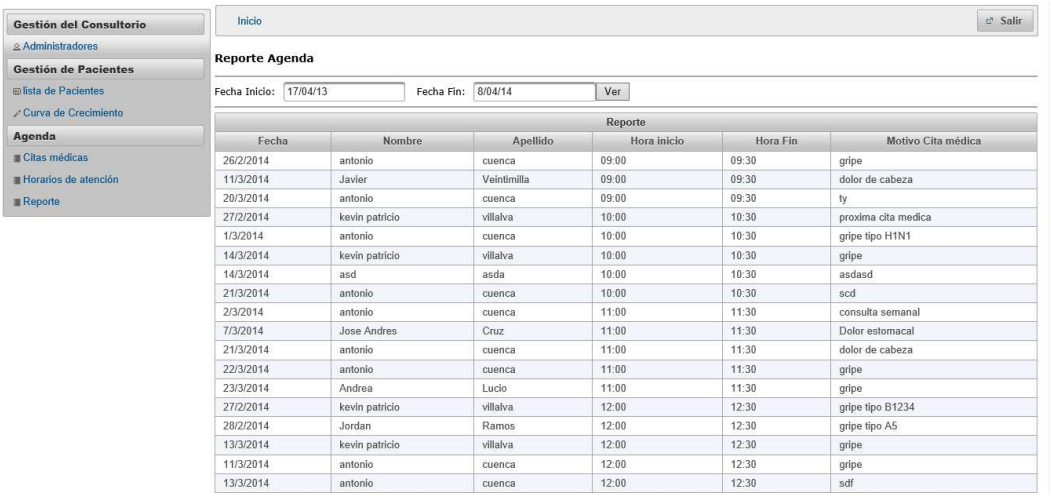


Ilustración 32: Reporte de atención de pacientes.

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Después del estudio de la técnica del desarrollo dirigido por pruebas conjuntamente con el framework JUnit, se ha podido determinar que la técnica no simplemente abarca el testing de la aplicación, sino que conduce el diseño de la misma, mejorando el paradigma de desarrollo del software, obteniendo un código de calidad.

Después de realizar el estudio de las pruebas unitarias, pruebas de integración y TDD, se realizó una comparación visualizada en las tablas N° 3.2 y N° 3.3, donde se determinó que TDD es similar tanto a las pruebas unitarias como a las pruebas de integración, pero no es lo mismo, ya que TDD expresa lo que el código debe hacer antes de que realmente este se escriba en la aplicación.

TDD puede ser utilizado para las pruebas de regresión. Se puede utilizar TDD para determinar inmediatamente si un cambio altero la funcionalidad de la aplicación. Por lo tanto a diferencia de las pruebas unitarias, TDD no prueba necesariamente una parte de código de forma aislada, sino que cubre una funcionalidad consecutiva, un desarrollo futuro en el código de la aplicación.

TDD funciona como mini-pruebas de aceptación, pero no cubre el testing de todo el proyecto. Sino que TDD expresa la funcionalidad que necesita ser implementada a continuación, por lo tanto TDD integra las pruebas que Junit proporciona para evaluar el sistema de forma modular.

La metodología Scrum aplica técnicas ágiles como el Desarrollo Dirigido por Pruebas (TDD), Modelado Ágil y Gestión de Cambios Ágil, por lo que se ajustó de manera consistente en el desarrollo del caso práctico, permitiendo tener un control y una distribución adecuada de las actividades de trabajo y a la vez centrarse en actividades prioritarias, logrando una aplicación distribuida y sencilla.

La primera vez que se usa Scrum, es complicado diseñar código de calidad dentro del sistema, porque no se conoce las limitaciones y fortalezas que ofrece la metodología. En este caso Scrum, cuenta con los Sprints. Un sprint es una iteración de actividades que se desarrolla con el grupo de trabajo, bajo los lineamientos de un conjunto de requerimientos. De esta forma se escribe una serie de pruebas para dichos requisitos y, una vez diseñados, se desarrolla el código más sencillo que haga que las pruebas pasen. De esta manera se asegura que se escribe el código necesario para cubrir dichos requisitos, a la vez que se cumplen los criterios de aceptación del cliente.

4.2 RECOMENDACIONES

Trabajar con TDD dirige el paradigma orientado a objetos a buscar un test que fuerce al SUT a tener una estructura o una API que dé buenos resultados en términos de legibilidad y reusabilidad. Así, se anticipa futuros casos de uso, futuros problemas y se cuidará el diseño de la API test tras test, aplicando buenas prácticas de diseño. Se recomienda el uso de TDD como una técnica de diseño sobre todo tipo de proyectos.

Cuando se usa TDD se recomienda crear las pruebas antes de la implementación del sistema, ya que si se realizan las mismas después del desarrollo se está orientando el diseño hacia las practicas tradicional, por lo que se pierde las principales características de implementar TDD.

Cuando se enfoca TDD hacia mini-pruebas de aceptación es importante dividir el proyecto por módulos de funcionalidad, de esta forma se cubre el testing de todo el sistema, y se evita que este proceso se torne complejo y sea necesaria la aplicación de herramientas adicionales de pruebas como Test Studio y WebInject.

BIBLIOGRAFÍA Y WEBGRAFÍA

Anderson, J. R. (2000). *Extrme Programming Installed.*

Araújo, A. (s.f.). *Test Driven Development Fortalezas y Debilidades.* Montevideo, Uruguay.

Araújo, A. (s.f.). *Test Driven Development Fortalezas y Debilidades.* Montevideo, Uruguay .

Gallego, M. T. (s.f.). *Metodología Scrum.*

Gómez, M. Á. (s.f.). *mifergo*. Obtenido de http://www.mifergo.es/tag/scrum/

Hirotaka, T., & Nonaka, I. (1986). *The New Product Development Game.*

Hut, P. (16 de Junio de 2009). *http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure*.

Hut, P. (16 de Junio de 2009). *http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure*.

JUnit. (s.f.). *JUnit.* Recuperado el 22 de Octubre de 2013, de http://junit.org

Jurado, C. B. (2010). *Diseño Agil con TDD.*

Jurado, C. B. (2010). *Diseño Agil con TDD.*

Martin, F. (2002). *Refactoring: Improving the Design of Existing Code.*

Mugridge, R. (2006). *Test Driven Development and the Scientific Method.* New Zealand.

Oracle. (s.f.). *Java*. Recuperado el 21 de Octubre de 2013, de http://www.oracle.com/es/technologies/java/overview/index.html

Oracle. (s.f.). *JSR-000342 JavaTM Platform, Enterprise Edition 7 Final Release for Evaluation.* Recuperado el 22 de Octubre de 2013, de JAVA EE Plataform: http://download.oracle.com/otndocs/jcp/java_ee-7-fr-eval-spec/index.html

Quispe, L. M. (Noviembre de 2011). *Arquitectura Orientada a Servicios (SOA).* Recuperado el 22 de Octubre de 2013, de http://www.slideshare.net/Mache007/arquitectura-orientada-a-servicios-soa-12818946

reserv. (2010). *reserv*. Recuperado el 12 de septiembre de 2013, de http://www.reserv.com.ar/metodologia.php

Romo, A. (17 de Marzo de 2010). *http://mundobyte-x.blogspot.com/2010/03/ultima-revision-17-de-marzo-2010.html*.

Romo, A. (17 de Marzo de 2010). *http://mundobyte-x.blogspot.com/2010/03/ultima-revision-17-de-marzo-2010.html*.