









**INFORME FINAL DE RESIDENCIA PROFESIONAL**

**“ANÁLISIS Y DESARROLLO DE APLICACIONES DE  
ESCRITORIO PARA AUTODIAGNÓSTICO, DETECCIÓN DE  
POSIBLES PATOLOGÍAS Y SEGUIMIENTO DEL PACIENTE”.**



**PRESENTA:  
NOE NEFTALI MERCADO MELGOZA**

**CARRERA:  
INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**Tlaltenango de Sánchez Román, Zac., junio de 2016**

## RESUMEN EJECUTIVO

<b>Nombre de la institución</b>	Instituto Tecnológico Superior Zacatecas Sur
<b>Nombre de la empresa</b>	Medisist S.A de C.V.
<b>Nombre del proyecto</b>	Análisis y desarrollo de aplicaciones de escritorio para autodiagnóstico, detección de posibles patologías y seguimiento del paciente.
<b>Problemas a resolver</b>	<p>1.- La empresa Medisist requiere de personal con experiencia en el lenguaje Java para realizar aplicaciones del sector salud.</p> <p>2.- Los sistemas que han desarrollado en dicha empresa, no cuentan con documentación para darles mantenimiento ni un seguimiento de versiones.</p>
<b>Justificación</b>	La empresa Medisist, dedicada al desarrollo de tecnologías software para la salud y al ser una empresa innovadora dentro del ecosistema de la alta tecnología en Jalisco, requiere de personal con ideas creativas y que tengan experiencia en los lenguajes y plataformas que se manejan dentro de la empresa.
<b>Objetivo general</b>	Desarrollar aplicaciones de escritorio para el autodiagnóstico y seguimiento de parámetros clínicos del usuario, así como mejorar y establecer un mejor criterio de calidad a todos y cada uno de los productos desarrollados dentro de la organización.
<b>Objetivos específicos</b>	<p>1.- Recibir capacitación para ser un personal preparado y con capacidad de reacción ante errores y mejoras en procesos del sistema.</p> <p>2.- Seguir los procesos de desarrollo de software.</p>

	<p>3.- Mejorar y perfeccionar la biblioteca y/o documentación existente.</p> <p>4. Capacitación constante para lograr cubrir con las necesidades que las áreas de producción requieren.</p> <p>5. Crear un ambiente de trabajo adecuado para el equipo de desarrollo.</p>
<b>Alcance del proyecto</b>	<p>Realizar aplicaciones de escritorio enfocadas a la salud de los pacientes implementando programación en Java, manejo de bases de datos, creación de interfaces gráficas de usuario y manipulación de complementos de desarrollo como el uso de creadores de gráficas y controladores de versiones, así como documentar los procesos realizados dentro del periodo de residencia.</p>

# INDICE

INTRODUCCIÓN.....	1
JUSTIFICACIÓN.....	3
OBJETIVOS.....	4
OBJETIVO GENERAL .....	4
OBJETIVOS ESPECÍFICOS .....	4
CARACTERIZACIÓN DEL ÁREA EN QUE SE PARTICIPÓ .....	5
DATOS DE LA EMPRESA.....	5
INFORMACIÓN GENERAL DE LA EMPRESA.....	6
PROBLEMAS A RESOLVER CON SU RESPECTIVA PRIORIZACIÓN .....	7
ALCANCES Y LIMITACIONES.....	8
CAPÍTULO I FUNDAMENTO TEÓRICO.....	10
1.1 HERRAMIENTAS DE DESARROLLO .....	11
1.2 MVC (MODELO VISTA CONTROLADOR) .....	17
1.2.1 EJEMPLO DE MVC.....	19
1.3 UI - INTERFAZ DE USUARIO.....	20
1.4 UX – EXPERIENCIA DE USUARIO.....	20
1.5 DISEÑO ADAPTIVO .....	21
CAPÍTULO II PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS.....	23
2.1 ACTIVIDADES DEL PROYECTO FONDO DE OJO .....	24
2.1.1 Propuesta de diseño.....	24
2.1.2 Diseño de interfaces de usuario .....	24
2.1.3 Implementación de diseño adaptivo en Java.....	25
2.1.4 Tabla “sorter” para filtrar búsquedas de pacientes .....	26
2.1.5 Alertas personalizadas (JOptionPane) .....	26
2.1.6 Modificación de componente JComboBox.....	27
2.1.7 Importar y aplicar tipografías personalizadas .....	28
2.1.8 Diseño de tablas de base de datos para guardar información del cuestionario.....	28
2.2 ACTIVIDADES DEL PROYECTO ATENCIÓN VITAL V1.....	29



2.2.1	Codificación de cuestionario (almacenar y recuperar respuestas del usuario) .....	29
2.2.2	Tablas de base de datos y codificación para versión en inglés .....	29
2.2.3	Implementación de teclado virtual para la versión sueca .....	30
2.3	ACTIVIDADES DEL PROYECTO ATENCION VITAL V2.....	30
2.3.1	Propuesta de diseño.....	30
2.3.2	Diseño de selección de inicio de sesión .....	31
2.3.3	Animaciones del tipo de inicio de sesión .....	32
2.3.4	Diseño de inicio de sesión manual .....	32
2.3.5	Diseño del menú principal .....	33
2.3.6	Animaciones del menú principal.....	33
2.3.7	Diseño adaptivo .....	34
2.3.8	Desarrollo del sistema en MVC .....	34
CAPÍTULO III RESULTADOS (PLANOS, GRAFICAS, PROTOTIPOS Y PROGRAMAS).....		36
3.1	RESULTADOS DEL PROYECTO FONDO DE OJO.....	37
3.1.1	Resultados de la propuesta de diseño.....	37
3.1.2	Resultados del diseño de interfaces de usuario .....	41
3.1.3	Resultados del diseño adaptivo en Java .....	45
3.1.4	Resultados de la tabla con sorter para filtrar datos.....	46
3.1.5	Resultado de las alertas personalizadas .....	49
3.1.6	Resultado de la modificación del JComboBox.....	51
3.1.7	Resultado de importar y aplicar tipografías personalizadas: .....	53
3.2	RESULTADOS DEL PROYECTO ATENCIÓN VITAL V1 .....	55
3.2.1	Resultado del diseño de tablas de la base de datos (cuestionario).....	55
3.2.2	Resultado de la codificación del cuestionario .....	57
3.2.3	Resultado de la creación de tablas y código para versión en inglés.....	59
3.2.4	Resultado del teclado virtual versión sueca.....	60
3.3	RESULTADOS DEL PROYECTO ATENCION VITAL V2 .....	62
3.3.1	Propuesta de diseño.....	62
3.3.2	Resultado del diseño del tipo de inicio de sesión .....	69

3.3.3	Resultado de las animaciones para el tipo de inicio de sesión.....	73
3.3.4	Resultado del diseño del inicio de sesión manual .....	77
3.3.5	Resultado del diseño del menú principal .....	81
3.3.6	Resultado de las animaciones para el menú principal.....	83
3.3.7	Resultado del diseño adaptivo.....	84
3.3.8	Resultado del uso de la arquitectura MVC .....	90
CONCLUSIONES .....		91
RECOMENDACIONES.....		94
FUENTES DE INFORMACIÓN.....		96
GLOSARIO .....		98

## ÍNDICE DE CUADROS, GRÁFICAS Y FIGURAS

Figura 1. Propuesta de inicio de sesión proyecto "Fondo de Ojo" .....	37
Figura 2. Propuesta de pantalla principal para proyecto "Fondo de Ojo" .....	38
Figura 3. Diseño final para logueo "Fondo de Ojo" .....	39
Figura 4. Diseño final para pantalla principal "Fondo de Ojo" .....	39
Figura 5. Diseño final para pantalla de nuevo usuario "Fondo de Ojo" .....	40
Figura 6. Diseño final para pantalla de búsqueda de pacientes "Fondo de Ojo" .....	40
Figura 7. Diseño previo en NetBeans del inicio de sesión .....	41
Figura 8. Uso del grid layout en pantalla de logueo .....	42
Figura 9. Vista previa del inicio de sesión .....	43
Figura 10. Diseño de la interfaz principal en NetBeans.....	43
Figura 11. Uso del grid layout en la vista principal .....	44
Figura 12. Vista previa de la interfaz principal.....	44
Figura 13. Vista para registrar pacientes.....	45
Figura 14. Interfaz para buscar un paciente (aún sin aplicar filtro de búsqueda) .....	47
Figura 15. Aplicando un filtro para mostrar resultados de búsqueda .....	47
Figura 16. Alerta de instrucción de uso, informar al paciente como usar la interfaz..	49
Figura 17. Alerta para errores del sistema .....	50
Figura 18. Alerta para información o ayuda de uso.....	50
Figura 19. Alerta para conexión de base de datos .....	50
Figura 20. Plantilla genérica para alertas .....	51
Figura 21. Vista general de JComboBox personalizados .....	52
Figura 22. JComboBox personalizado .....	53
Figura 23. Paquete con tipografías .....	54
Figura 24. Tabla de respuestas para cuestionario .....	55
Figura 25. Estructura de tabla de respuestas.....	56
Figura 26. Estructura de tabla para almacenar respuestas del usuario .....	56
Figura 27. Tabla para guardar respuestas (sin datos).....	57
Figura 28. Tabla con respuestas en inglés.....	59
Figura 29. Diseño de teclado virtual en NetBeans .....	60
Figura 30. Resultado real del teclado virtual para versión sueca .....	61
Figura 31. Creación de recursos con Illustrator.....	62
Figura 32. Diseño de cuestionario en Illustrator .....	63
Figura 33. Diseño de pantallas generales en Illustrator .....	64
Figura 34. Archivos de diseño y recursos .....	64
Figura 35. Diseño de banner para menú principal.....	65
Figura 36. Propuesta para el tipo de inicio de sesión.....	66
Figura 37. Propuesta de diseño para la ayuda y menú principal.....	66
Figura 38. Propuesta de diseño para interfaz de peso y estatura .....	67

Figura 39. Propuesta de diseño para cuestionario de salud .....	68
Figura 40. Propuesta de diseño para interfaz de monitoreo.....	68
Figura 41. Propuesta de diseño para interfaz de premios.....	69
Figura 42. Vista de diseño del tipo de inicio de sesión.....	70
Figura 43. Diseño del tipo de inicio de sesión en NetBeans .....	70
Figura 44. Personalización de botones .....	71
Figura 45. Animación en estado inicial.....	73
Figura 46. Animación al presionar el icono de inicio de sesión manual .....	74
Figura 47. Animación de deslizamiento.....	75
Figura 48. Animación para mostrar (subir) un nuevo panel.....	75
Figura 49. Vista del diseño de logueo manual.....	78
Figura 50. Teclado virtual oculto .....	78
Figura 51. Diseño del logueo manual en NetBeans .....	79
Figura 52. Diseño de menú principal en NetBeans .....	81
Figura 53. Aplicando el grid layout al menú principal .....	82
Figura 54. Vista previa del menú principal .....	82
Figura 55. Elección de logueo a 1920 * 1080.....	86
Figura 56. Elección de logueo a 1366 * 768.....	86
Figura 57. Logueo manual a 1920 * 1080 .....	87
Figura 58. Logueo manual a 1366 * 768 .....	87
Figura 59. Interfaz de medición de peso a 1920 * 1080.....	88
Figura 60. Interfaz de medición de peso a 1366 * 768.....	88
Figura 61. Menú principal a 1920 * 1080.....	89
Figura 62. Menú principal a 1366 * 768.....	89
Figura 63. Estructura MVC del sistema "Atención Vital" .....	90

# INTRODUCCIÓN

Hoy en día el uso de software para cualquier tipo de negocio se ha incrementado, haciendo de este una herramienta indispensable para automatizar múltiples procesos llevados a cabo en distintas áreas.

En empresas del ramo de la medicina, el software se ha implementado para agilizar los procesos administrativos y también para brindar atención de calidad a los pacientes. Un ejemplo muy sencillo es la implementación de algún sistema (una aplicación móvil, un sitio web o programa de computadora) que le permita al usuario conocer los resultados de alguna prueba o medición, llevar el control de sus datos médicos o contar con una plataforma donde reciba noticias relacionadas a su salud.

Es así, que en este documento se describirán ciertos procesos realizados para llevar a cabo el desarrollo de aplicaciones relacionadas a la salud del paciente, que les permitirá tener un control sobre sus datos médicos.

El documento se encuentra dividido en capítulos en los cuales se describen los aspectos teóricos, las actividades llevadas a cabo, los resultados de dichas actividades y las conclusiones obtenidas dentro del periodo de la residencia profesional.

Para realizar las aplicaciones se necesitaron ciertas herramientas como editores de código, servidores de bases de datos y editores de gráficos e imágenes.

En el primer capítulo se describen estas herramientas y se muestran algunas tecnologías utilizadas para llevar a cabo el desarrollo de los sistemas.

El proceso de desarrollo de las aplicaciones comenzó con prototipos, propuestas de diseño y posteriormente comenzar con la codificación. Es en el capítulo dos en donde se muestran las actividades y procedimientos que se realizaron para conseguir los objetivos de las aplicaciones.

Para mostrar los resultados de las actividades que se hicieron durante el desarrollo de las aplicaciones, se muestran algunas capturas de pantalla que evidencian la funcionalidad y las interfaces de usuario creadas para los sistemas. En el cuarto capítulo se exponen los resultados de las actividades descritas en el capítulo tercero, además de capturas se plasma un poco del código utilizado para llevar a cabo los trabajos de programación.

## JUSTIFICACIÓN

La empresa Medisist, dedicada al desarrollo de tecnologías de software para la salud y al ser una empresa innovadora dentro del ecosistema de la alta tecnología en Jalisco, requerirá de personal con ideas creativas y que tengan experiencia en los lenguajes y plataformas que se manejan dentro de la empresa.

Dado que la mayoría de las aplicaciones con las que cuenta Medisist tienen el mismo aspecto, es decir, casi todas comparten cierta similitud en el diseño de la interfaz de usuario, será necesario realizar diseños nuevos, llamativos para los usuarios y que cumplan con la modalidad del diseño adaptivo para las distintas resoluciones de dispositivos en el mercado.

Al ser una empresa con varios años en el mercado de desarrollo de software, se ven en la necesidad de contar con personal nuevo, y que estén en constante capacitación, pues no solo se desarrollará nuevo software, sino también, mejorarán los sistemas existentes contra defectos, problemas y/o usabilidad, así como documentar los cambios.

La empresa, al tener ya experiencia en el mercado, ha adaptado su flujo de trabajo para cumplir con sus objetivos, es por eso que los nuevos integrantes necesitarán capacitación para entender los métodos de desarrollo, para así puedan adaptarse a los procesos y trabajar con su equipo de manera clara y fluida.

## **OBJETIVOS**

### **OBJETIVO GENERAL**

Desarrollar aplicaciones de escritorio para el autodiagnóstico y seguimiento de parámetros clínicos del usuario, así como mejorar y establecer un mejor criterio de calidad a todos y cada uno de los productos desarrollados dentro de la organización, todo esto en base al desarrollo, detección y predicción de posibles fallos en la usabilidad del sistema.

### **OBJETIVOS ESPECÍFICOS**

1. Recibir capacitación para ser un personal preparado y con capacidad de reacción ante errores y mejoras en procesos del sistema.
2. Seguir los procesos de desarrollo de software.
3. Mejorar y perfeccionar la biblioteca y/o documentación existente.
4. Capacitación constante para lograr cubrir con las necesidades que las áreas de producción requieren.
5. Crear un ambiente de trabajo adecuado para el equipo de desarrollo.



## CARACTERIZACIÓN DEL ÁREA EN QUE SE PARTICIPÓ

La prestación de servicios profesionales se realizó en la empresa Medisist, con la finalidad de apoyar en el desarrollo de software médico, llevando a cabo actividades como diseño de interfaces de usuario, manejo de bases de datos, desarrollo en el lenguaje Java, controlar versiones del sistema y realizar la documentación necesaria para las aplicaciones. Todo esto fue realizado en un periodo de 4 meses iniciando el día 11 de enero de 2016 y finalizando el día 11 de mayo del mismo año.

### DATOS DE LA EMPRESA

<b>Nombre:</b>	Medisist S.A de C.V.
<b>Razón social:</b>	Medisist S.A de C.V.
<b>Dirección:</b>	Centro del Software, Av. López Mateos Sur # 2077-Z-34, Col. Jardinesde Plaza del Sol, C.P. 44510. Guadalajara Jalisco, México.
<b>Teléfono:</b>	01 (33) 3030-7030.
<b>Asesor externo:</b>	Mónica Heredia (Líder de Software).

## **INFORMACIÓN GENERAL DE LA EMPRESA**

Dirigida por especialistas con más de 20 años de experiencia en el área de Informática Médica, la empresa se ha dedicado desde su inicio en 1995, a través de la innovación y la cooperación, a la aplicación de las tecnologías de información, comunicaciones y microelectrónica, al mejor costo beneficio y a mejorar la calidad e incrementar la competitividad de las instituciones latinoamericanas prestadoras de servicios de salud.

Empresa innovadora del ecosistema de la alta tecnología en Jalisco, en colaboración con el CINVESTAV y empresas de microelectrónica, Medisist desarrolla productos patentables para el telemonitoreo y la teleasistencia de pacientes con problemas crónicos y agudos de salud, quienes podrán beneficiarse de los adelantos de Internet y las telecomunicaciones para recibir una pronta y adecuada atención.

## **PROBLEMAS A RESOLVER CON SU RESPECTIVA PRIORIZACIÓN**

- La empresa Medisist requiere de personal con experiencia en el lenguaje Java para realizar aplicaciones del sector salud.
- Los sistemas que han desarrollado en dicha empresa, no cuentan con documentación para darles mantenimiento ni un seguimiento de versiones.
- Existen sistemas médicos que tienen defectos y necesitan ser reparados para liberar una versión mejorada.
- Las aplicaciones cuentan con interfaces de usuario un tanto obsoletas, es decir, no cuentan con una interfaz moderna ni llamativa para los usuarios.

## **ALCANCES Y LIMITACIONES**

### **Alcances**

Dentro de las actividades planeadas para llevar a cabo la residencia profesional se tiene como alcance:

- Realizar aplicaciones de escritorio enfocadas a la salud de los pacientes implementando programación en Java, manejo de bases de datos, creación de interfaces gráficas de usuario y manipulación de complementos de desarrollo como el uso de creadores de gráficas y controladores de versiones, así como documentar los procesos realizados dentro del periodo de residencia.

### **Limitaciones**

- Dado que no se especificó un proyecto en particular, se pudiera llegar a cambiar el proyecto, pues la empresa trabaja sobre prioridades de entrega y es decisión de los gerentes cambiar de actividades o posponerlas.
- Como la empresa trabaja en distintos proyectos, muy frecuentemente el personal se encuentra en capacitación, siendo un obstáculo para desarrollar las actividades planeadas.
- Las tareas asignadas para realizar una modificación o nuevo requerimiento, dependen también de otros departamentos, siendo así que para liberar una actividad es necesario esperar mucho tiempo, como en el caso del cambio de

interfaces, donde es necesario hacer una propuesta, después pasarla al área de diseño, esperar que la liberen y por ultimo implementarla en codificación, es un proceso de espera largo, pues el área de diseño trabaja también la imagen corporativa y a veces no se tiene el tiempo necesario para dar continuidad a las actividades.

# **CAPÍTULO I**

## **“FUNDAMENTO TEÓRICO”**

## 1.1 HERRAMIENTAS DE DESARROLLO

- **NetBeans:**

Es un entorno de desarrollo integrado (IDE) libre, diseñado en un principio para el lenguaje Java pero que con el pasar de los años ha sido crecido para trabajar con más lenguajes de programación, haciéndolo uno de los IDE de desarrollo más competitivos en el mercado.

“Una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software”.<sup>1</sup>

---

<sup>1</sup> NetBeans [En línea]. Disponible: marzo 10, 2016. [https://netbeans.org/index\\_es.html](https://netbeans.org/index_es.html)

El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactorización.

- **Java:**

Java es un lenguaje de programación utilizado para la creación de aplicaciones móviles, web y de escritorio. Cuenta con un sinnúmero de aplicaciones y dispositivos que utilizan a este lenguaje como base.

“Un lenguaje que evoluciona a partir de C y C++, pero que elimina diversos aspectos de estos lenguajes y se constituye en un lenguaje definitivamente orientado a objetos. El romper con distintos aspectos de C++ cuyo manejo inadecuado por parte de muchos programadores daba lugar a problemas en las aplicaciones ha sido un factor decisivo para convertir a Java en un lenguaje popular y de amplio uso”.<sup>2</sup>

---

<sup>2</sup> Aprende a Programar - ¿Qué es Java? [En línea]. Disponible: marzo 14, 2016.  
<http://www.aprenderaprogramar.com/index.php>



- **Illustrator:**

Adobe Illustrator es un programa que permite crear y editar archivos gráficos vectoriales. Es como si se utilizara el editor de imágenes Paint de Windows, pero con un sorprendente motor de gráficos que permite, mediante “mesas o espacios” de trabajo dibujar, vectorizar, renderizar, colocar y crear vectores para su uso en distintas plataformas y diversos formatos de exportado.

“Adobe Illustrator contiene opciones creativas, un acceso más sencillo a las herramientas y una gran versatilidad para producir rápidamente gráficos flexibles cuyos usos se dan en (maquetación-publicación) impresión, vídeo, publicación en la Web y dispositivos móviles.

Actualmente forma parte de la familia Adobe Creative Cloud y tiene como función única y primordial la creación de material gráfico-ilustrativo altamente profesional basándose para ello en la producción de objetos matemáticos denominados vectores”.<sup>3</sup>

- **Photoshop:**

A diferencia de Illustrator, Photoshop es un sistema que trabaja con imágenes de mapas de bits, es decir imágenes rasterizadas. Este sistema no maneja gráficos

---

<sup>3</sup> Wikipedia - Adobe Illustrator [En línea]. Disponible: marzo 14, 2016.  
[https://es.wikipedia.org/wiki/Adobe\\_Illustrator](https://es.wikipedia.org/wiki/Adobe_Illustrator)

vectoriales, pero es un excelente programa para manipulación de fotografías, tanto así que, traducido al español sería “Taller de fotos”.

“Photoshop se ha convertido, casi desde sus comienzos, en el estándar de facto en retoque fotográfico, pero también se usa extensivamente en multitud de disciplinas del campo del diseño y fotografía, como diseño web, composición de imágenes en mapa de bits, digital, fotocomposición, edición y grafismos de vídeo y básicamente en cualquier actividad que requiera el tratamiento de imágenes digitales”.<sup>4</sup>

- **JFreeChart:**

“Es una librería para Java, que permite la creación de gráficas para estadísticas. Esta librería cuenta con múltiples funciones para crear y personalizar gráficas, y también cuenta con distintos tipos de graficas como son, pasteles, de barras, 3D, puntos y de más”.<sup>5</sup>

JFreeChart es compatible con una serie de gráficas diferentes, incluyendo cuadros combinados. Después de tipos de gráficos son compatibles:

- Gráficos XY (línea, spline y dispersión). Es posible usar un eje del tiempo.
- Gráfico circular.
- Diagrama de Gantt.

---

<sup>4</sup> Imagen Digital UCP - ¿Qué es Photoshop? [En línea]. Disponible: marzo 14, 2016. <https://imagendigitalucp.wordpress.com/tutoriales-photoshop/>

<sup>5</sup> JFreeChart [En línea]. Disponible: marzo 14, 2016. <http://www.jfree.org/jfreechart/index.html>

- Gráficos de barras (horizontales y verticales, apiladas e independientes).  
También tiene incorporado un dibujador de histogramas.
- Single valued (termómetro, brújula, indicador de velocidad) que luego se pueden colocar sobre el mapa.
- Varias gráficas específicas (tabla de viento, gráfica polar, burbujas de diferentes tamaños, etc.).

Además, es posible colocar varios marcadores en el área de gráfica.

JFreeChart dibuja automáticamente las escalas de los ejes y leyendas. Con el ratón (mouse) se puede hacer zoom en la interfaz de la gráfica automáticamente y cambiar algunos ajustes a través del menú local. Las tablas existentes pueden actualizarse fácilmente a través de los oyentes (listeners) que la biblioteca tiene en sus colecciones de datos.

- **SQL Server 2008:**

Microsoft SQL Server es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos.

“SQL Server se ejecuta en T-SQL (Transact -SQL), un conjunto de extensiones de programación de Sybase y Microsoft que añaden varias

características a SQL estándar, incluyendo control de transacciones, excepción y manejo de errores, procesamiento fila, así como variables declaradas”.<sup>6</sup>

---

<sup>6</sup> TechTarget - ¿Qué es sql server? [En línea]. Disponible: marzo 15, 2016.  
<http://searchdatacenter.techtarget.com/es/definicion/SQL-Server>

## 1.2 MVC (MODELO VISTA CONTROLADOR)

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

Este patrón separa el código en tres capas:

- “Capa de modelo: que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia (aquí se encuentra la base de datos).

Esta capa es responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
  - Define las reglas de negocio (la funcionalidad del sistema).
  - Lleva un registro de las vistas y controladores del sistema.
  - Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).”<sup>7</sup>
- “Capa de vista: o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

---

<sup>7</sup> Onnes sourceforge - Aplicaciones en capas [En línea]. Disponible: marzo 15, 2016.  
<http://onnes.sourceforge.net/proyecto/html/ch03s02.html>

Las vistas son responsables de:

- Recibir datos del modelo y los muestre al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).
- Capa de controlador: actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Esta capa es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar ()".<sup>8</sup>

---

<sup>8</sup> Desarrollo web - ¿Qué es MVC? [En línea]. Disponible: marzo 15, 2016.  
<http://www.desarrolloweb.com/articulos/que-es-mvc.html>

### 1.2.1 EJEMPLO DE MVC

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario).
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

### **1.3 UI - INTERFAZ DE USUARIO**

Hace foco en el artefacto, o, dicho de otra manera, en lo que está dentro de la pantalla. Cuando se diseñan interfaces el problema que está resolviendo está en el diseño: selección y distribución de los elementos de la interfaz (ej. textos y campos del formulario), consistencia del diseño (con la plataforma, con otras pantallas).

“Es importante aclarar que Diseño de Interfaces no equivale a Diseño Gráfico: el diseño de la interfaz puede incluir o no diseño gráfico. Por ejemplo, cuando se hace un wireframe se está diseñando una interfaz pero no se está aplicando diseño gráfico, y cuando se requiere aplica reglas de estilo a una interfaz se está aplicando diseño gráfico pero no se está diseñando una interfaz”.<sup>9</sup>

### **1.4 UX – EXPERIENCIA DE USUARIO**

La experiencia de usuario hace referencia a lo “fácil” o “intuitivo” de un sistema, aquí entran distintos factores, como los colores elegidos para llamar la atención o desviar miradas, el acomodo de componentes, los efectos aplicados, la sencillez de encontrar o destacar algún apartado, etc.

---

<sup>9</sup> Kambrica - UI, UX, IxD: ¿Cuál es la diferencia? [En línea]. Disponible: marzo 15, 2016.  
<http://www.kambrica.com/blog/ui-ux-ixd-cual-es-la-diferencia/>



“Sin incorporar al usuario, no se puede hacer UX. Por eso, resulta fundamental en el diseño de la experiencia, comprender en primer lugar a los usuarios y sus verdaderas motivaciones y necesidades, considerar desde ese lugar qué interfaz, qué contenidos y qué interacciones lograrán el resultado buscado, y finalmente, validar con usuarios los resultados que produce la interfaz propuesta”.<sup>10</sup>

La validación se puede hacer de forma directa (pruebas con usuarios, entrevistas cualitativas, relevamiento de modelos mentales), o de forma indirecta (heat maps, A/B testing, click maps).

## **1.5 DISEÑO ADAPTIVO**

El diseño web adaptativo o adaptable (en inglés, Responsive Web Design) es una técnica de diseño y desarrollo web donde esta se adapte a la resolución de la pantalla o al dispositivo que está visitando su página web de manera inteligente.

El Responsive Web Design ayuda a que su página se adapte a los tamaños de casi cualquier pantalla, como por ejemplo, las de los Celulares, SmartPhones, Tablet's, Laptops, Notebook o PC de escritorios. Prácticamente a cualquier dispositivo que pueda navegar en internet.

---

<sup>10</sup> Kambrica - UI, UX, IxD: ¿Cuál es la diferencia? [En línea]. Disponible: marzo 15, 2016. <http://www.kambrica.com/blog/ui-ux-ixd-cual-es-la-diferencia/>

“Hoy día las páginas web se visualizan en multitud de tipos de dispositivos como tabletas, teléfonos inteligentes, libros electrónicos, portátiles, PC, etc. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, capacidad de memoria, entre otras. Esta tecnología pretende que con un solo diseño web, se tenga una visualización adecuada en cualquier dispositivo”.<sup>11</sup>

---

<sup>11</sup> ID Web - Qué es el "responsive web design" [Editable]. Disponible: marzo 16, 2016.  
[http://imasdeweb.com/index.php?pag=m\\_blog&gad=detalle\\_entrada&entry=12](http://imasdeweb.com/index.php?pag=m_blog&gad=detalle_entrada&entry=12)

# **CAPÍTULO II**

## **“PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS”**

## **2.1 ACTIVIDADES DEL PROYECTO FONDO DE OJO**

### **2.1.1 Propuesta de diseño**

Como paso fundamental para iniciar o actualizar un diseño de interfaces de usuario es crear un “boceto” el cual sirve para que ciertos miembros de un equipo puedan llegar a la conclusión de que el diseño propuesto es el adecuado para una nueva versión de un sistema. La propuesta de diseño se trabajó en equipo, tomando las opiniones, ideas y diseños del diseñador de UI y la diseñadora industrial que laboran en la empresa.

Se hicieron dos versiones de la propuesta, una por cada miembro del equipo; después de revisar las propuestas, se llegó a la conclusión de que las dos propuestas tenían pros y contras, entonces se tomaron características de ambas propuestas para hacer una tercera y dejarla como propuesta para la nueva versión del sistema.

### **2.1.2 Diseño de interfaces de usuario**

El diseño de las interfaces de todos los módulos se llevó a cabo siguiendo la estructura de diseño establecida en la propuesta de nuevo diseño. No fue posible dejar la interfaz idéntica a la propuesta ya que algunos componentes del lenguaje Java no se pueden modificar de la manera en que se mostró en la propuesta.

Al fin de cuentas se llegó a la conclusión de que se respetara lo máximo posible establecido en la propuesta tomando en cuenta las limitaciones del lenguaje.

### **2.1.3 Implementación de diseño adaptivo en Java**

El sistema de Fondo de Ojo se hizo pensando abarcar la mayor cantidad de resoluciones posibles. Por tal motivo se vio la necesidad de cambiar la mayoría de componentes y codificarlos para que se redimensionaran de acuerdo al tamaño de la resolución de la pantalla anfitriona. Dicho proceso de redimensión, se trabajó cien por ciento desde código y con la ayuda de layouts.

Se requirió de muchas pruebas para lograr que se adaptara a la mayor cantidad de resoluciones posibles, y que el diseño y los componentes quedaran en perfecto estado, tanto de visibilidad (letra visible) y acomodo (que no se mostraran fuera del rango de la pantalla).

Hacer este proceso en Java es un tanto laborioso, pues a diferencia de la programación web en donde se trabaja con porcentajes directos, en java se tienen que hacer todo ese cálculo de porcentajes desde código, y asignarlos a cada componente que conforma la interfaz.

#### **2.1.4 Tabla “sorter” para filtrar búsquedas de pacientes**

Una de las prácticas comunes al hacer consultas de búsqueda con filtros para obtener resultados precisos es mediante consultas directas a la base de datos, pero ¿Qué pasa si se tienen miles de registros?

Una buena práctica es, si ya se tienen los registros de la base de datos “vacíos” en alguna tabla, es aplicar un “sorter o clasificador” para evitar consumir la base de datos. Esto quiere decir que en vez de hacer consultas de filtrado directamente a la base de datos se hagan a la tabla, básicamente se enlazan los campos de texto, listas desplegables, o cualquier componente de entrada de datos a un clasificador de datos que tiene una tabla.

Con este pequeño cambio una aplicación puede incrementar su rendimiento notablemente y de esa manera se reservará memoria para otros procesos.

En el proyecto de Fondo de Ojo se aplicó dicho “sorter” para la tabla de búsqueda de pacientes así se evitaron las llamadas a la base de datos y la saturación de conexiones.

#### **2.1.5 Alertas personalizadas (JOptionPane)**

Para el proyecto Fondo de Ojo se pensó trabajar con un sistema de “alertas” personalizadas (cuadros de dialogo con diseño propio para notificar eventos).

Dado que se pretendía cambiar completamente la interfaz del sistema, se pensó también en crear este tipo de alertas para que no se desfasara en cuanto a las interfaces gráficas. Dichas alertas muestran un diseño similar a la interfaz principal, y cuentan con un funcionamiento casi idéntico a las ventanas de dialogo creadas por la librería swing de Java (JOptionPane).

Las alertas cuentan con animaciones de desvanecimiento. Este efecto es algo diferente a lo convencional de java, pues después de cierto tiempo (alrededor de 5-7 segundos) comienzan a desvanecerse hasta quedar completamente invisibles, como si el usuario la hubiera cerrado.

Al momento de que la alerta se está desvaneciendo, el usuario puede pasar el cursor por encima de esta, y automáticamente recuperará su color original, como si se actualizara y se mostrara nuevamente al pasar el cursor, de tal manera que el usuario puede volver a leer lo escrito en la alerta si algo no le quedó claro.

#### **2.1.6 Modificación de componente JComboBox**

En la propuesta de diseño se pretendió que las listas desplegables (JComboBox) también estuvieran personalizados, esto es algo complicado pues solo es posible modificar su clase abstracta de diseño.

A final de cuentas fue posible solo modificar pequeñas partes del diseño de estas listas, pudiendo modificar en si el botón que hace que se active la lista desplegable, solo es posible cambiar el color de fondo, así como el color de fondo de los elementos que conforman la lista.

#### **2.1.7 Importar y aplicar tipografías personalizadas**

Dado que la empresa maneja tipografías personalizadas, fue una obligación implementar estos tipos de letra en el proyecto, pero al ser tipografías que no están instaladas en las maquinas destino (computadoras de clientes) lo ideal fue “incrustar” estos tipos de letra en el sistema y que mediante código se cambiara la tipografía de todos los elementos del sistema.

Básicamente lo que se hace es copiar las fuentes requeridas dentro de un paquete y después mediante código “registrarla” es decir, crear un archivo temporal con la fuente y que sea leído desde ahí, para que no se instale en la máquina del cliente. De esta manera el sistema mostrara las tipografías propias de la empresa sin necesidad de que los clientes tengan que instalarlas.

#### **2.1.8 Diseño de tablas de base de datos para guardar información del cuestionario**

Uno de los nuevos requerimientos para la modificación del sistema “Atención Vital” era guardar los datos del cuestionario de cada usuario, ya que anteriormente solo se



guardaba el resultado final (recomendaciones). Como fue un nuevo requerimiento fue necesario modificar la base de datos para agregar nuevas tablas y relaciones. Para realizar dicha actividad fue necesario añadir tres tablas nuevas a la base de datos, las cuales son: catálogo de respuestas, respuestas del usuario español y respuestas del usuario en inglés.

## **2.2 ACTIVIDADES DEL PROYECTO ATENCIÓN VITAL V1**

### **2.2.1 Codificación de cuestionario (almacenar y recuperar respuestas del usuario)**

Una vez que fueron creadas las tablas para almacenar la información del cuestionario, fue turno de llevar a cabo la codificación necesaria para hacer que los datos se guardaran en caso de que el usuario nunca haya tenido interacción con el sistema o que estos datos se modificaran en caso contrario. Fue necesario implementar ciclos para detectar los elementos (check box) seleccionados y posteriormente hacer consultas para guardar los datos en las tablas.

### **2.2.2 Tablas de base de datos y codificación para versión en inglés**

Dado que el sistema cuenta con dos idiomas, también fue necesario crear tanto las tablas de la base de datos como la codificación necesaria para controlar los datos cuando el idioma del sistema cambie a inglés. Solo fue necesario agregar variables

auxiliares para detectar el cambio de idioma y en base a eso, cambiar la ruta de la tabla en la variable de la consulta.

### **2.2.3 Implementación de teclado virtual para la versión sueca**

La versión sueca del sistema fue implementada sobre el sistema operativo Windows 10 y tuvo algunos problemas en cuanto a utilizar el teclado virtual del propio Windows, es por eso que se tomó la decisión de utilizar un teclado virtual personalizado.

Para realizar ese teclado virtual se tomó una “plantilla” que fue hecha para la versión 2 del mismo sistema. Se cambiaron algunas teclas como la “ñ” y se colocaron las letras “ä”, “å” y “ö”.

## **2.3 ACTIVIDADES DEL PROYECTO ATENCION VITAL V2**

### **2.3.1 PROPUESTA DE DISEÑO**

Una de las actividades principales por realizar, el cambio, renovación, actualización de las interfaces de usuario. Al igual que en la propuesta de diseño del sistema “Fondo de Ojo”, fue necesario hacer un “prototipo” en un archivo de power point, para que posteriormente fuera revisado y aceptado o reenviado con retrospectiva.

Se tomó como base la interfaz anterior para tratar de no modificar el flujo de trabajo, y ver cómo es que funciona el sistema. También se tomó como base la versión anterior del sistema para revisar la paleta de colores y para darse una idea de cómo es que debían ir los iconos.

Para realizar esta propuesta se tomó como base un diseño limpio, minimalista, ya que va destinado a hospitales o a sitios que normalmente manejan colores discretos, no tan llamativos.

### **2.3.2 Diseño de selección de inicio de sesión**

En esta parte se debieron contemplar tres tipos de inicio de sesión, los cuales son: inicio de sesión manual (con usuario y contraseña), inicio de sesión mediante código QR y mediante huella dactilar, además de mostrar un botón para dar de alta nuevos pacientes. Otros elementos que contiene la interfaz de selección de inicio de sesión son los botones para cambiar el idioma de la aplicación y los botones de ayuda y el de cerrar la aplicación. Como es una aplicación a pantalla completa (solo se ve el sistema y no algún otro programa), se remueven los bordes que contienen las ventanas del sistema operativo que se esté utilizando, en este caso Windows, y como se remueven estos bordes entonces se debe contemplar añadir botones para cerrar, minimizar o maximizar el sistema.

### **2.3.3 Animaciones del tipo de inicio de sesión**

Dado que a la nueva versión se pretendía hacer más vistosa e interactiva con el usuario, se pensó en agregar efectos y animaciones a los componentes que conforman la interfaz, de tal manera que cuando el usuario presionara un botón, apareciera una alerta, o cambiara de pantalla (ventana) pudiera visualizar algún efecto, animación, algo llamativo que no hiciera tan plano al sistema.

Las animaciones que se implementaron en esta sección fueron efectos de barrido para alternar los iconos cuando son presionados y efectos de movimiento para cuando el usuario presionara un botón asociado a una pantalla de logueo esta se desplazara de abajo hacia arriba, algo como superponerse, pero con animación de desplazamiento.

### **2.3.4 Diseño de inicio de sesión manual**

Dado que el sistema de Atención Vital fue diseñado para dispositivos con pantalla touch principalmente (computadoras todo en uno y tabletas Windows) era un tanto incomodo o innecesario usar un teclado “físico” y se optó por la idea de implementar un teclado “virtual” propio del sistema y que no fuera el que ofrecen los sistemas operativos por defecto. Es por eso que se hizo un teclado personalizado que cubriera con las necesidades del sistema y que combinara en interfaz con el resto de los componentes.

A esa interfaz la componen más elementos, no solo el teclado, esos elementos son los campos de texto para el usuario y contraseña y botones de iniciar, regresar a la pantalla anterior y cerrar la aplicación.

Esa interfaz contaba con animaciones para mostrar y ocultar el teclado, que fueron descartadas a última hora por su falta de uso ya que el usuario solo se ve en la necesidad de escribir sus datos e iniciar, en ningún momento se demostró la utilidad de ocultar el teclado.

### **2.3.5 Diseño del menú principal**

En el menú principal se planeó poner la mayor cantidad de accesos directos a diferentes partes del sistema con la intención de hacer la pantalla más comprensible para el usuario. Para lograr su diseño fue necesario implementar layouts de rejilla y nulos, pues también se debió procurar cubrir diferentes resoluciones de pantalla.

### **2.3.6 Animaciones del menú principal**

Las animaciones que contiene esta pantalla son similares a las animaciones que presentan los dispositivos móviles, animaciones de “recorrido” o desplazamiento para simular una galería. Se implementaron eventos para hacer que la galería de accesos se deslice haciendo uso de los dedos, ya que el sistema es presentado a clientes en computadoras touch o en tabletas de Windows.

Para lograr estas animaciones fue necesario crear clases para mover componentes y asignar distintos tipos de eventos a paneles y botones, así como estar haciendo cálculos para determinar ciertas restricciones cuando el usuario arrastre los paneles, esto para evitar desfasar los componentes fuera de la visibilidad del usuario.

### **2.3.7 Diseño adaptivo**

Uno de los principales requerimientos del sistema Atención Vital V2 era hacerlo de manera “responsiva” o “adaptable”, es decir que las interfaces de usuario se ajusten al tamaño de la pantalla o monitor del dispositivo que esté mostrando el sistema.

Para poder realizar un diseño adaptable, fue necesario desde hacer imágenes a mayor tamaño o resolución posible (tomando en cuenta las resoluciones más grandes, pues una imagen puede ser reducida en tamaño sin perder calidad, pero no se pueden hacer más grandes, ya que perderían calidad y se deformarían) hasta realizar cálculos de porcentajes de tamaños para cada componente, imagen y tipografía que conforma la interfaz del sistema.

### **2.3.8 Desarrollo del sistema en MVC**

Con el motivo de realizar un sistema “mantenible”, se implementó el diseño en tres capas modelo – vista – controlador, esto para que en requerimientos futuros sea posible modificar el sistema sin tener tantas complicaciones que si se hiciera el proyecto de una forma tradicional o sin patrón de arquitectura de software.

El sistema se compone principalmente de tres paquetes, que son: paquete de modelo, en donde se almacenan las clases necesarias para realizar transacciones a bases de datos y conexión, paquete de vista, que es en donde se van las “ventanas” de los módulos, en estas clases se debe omitir escribir tanto código, código funcional, ya que en futuras modificaciones solo se deberá procurar cambiar el diseño y no la funcionalidad en sí.

Y por último el paquete de controlador, este contiene las clases necesarias para interconectar a las vistas con el modelo, es decir, el controlador recuperará los datos arrojados por el modelo y los mostrará al usuario por medio de una vista y viceversa.

# **CAPÍTULO III**

**“RESULTADOS (PLANOS,  
GRAFICAS, PROTOTIPOS  
Y PROGRAMAS)”**



## 3.1 RESULTADOS DEL PROYECTO FONDO DE OJO

### 3.1.1 Resultados de la propuesta de diseño

A continuación se muestra la propuesta del nuevo diseño en la cual se trabajó individualmente:

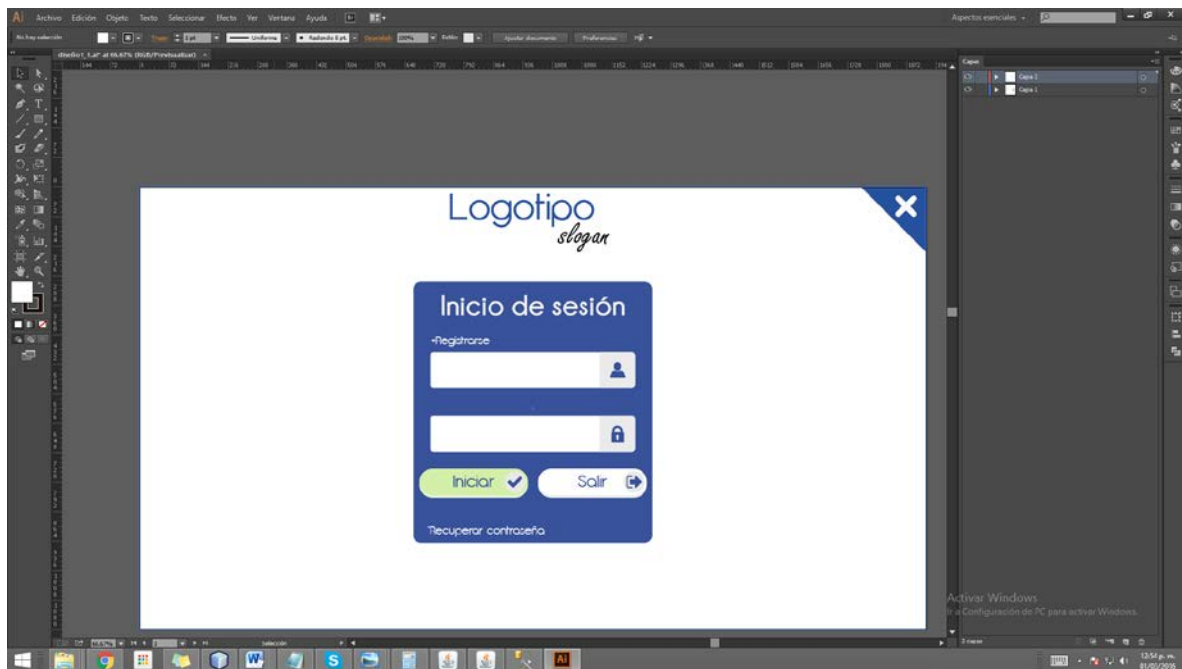


Figura 1. Propuesta de inicio de sesión proyecto “Fondo de Ojo”.

Nuevo diseño para la interfaz principal:

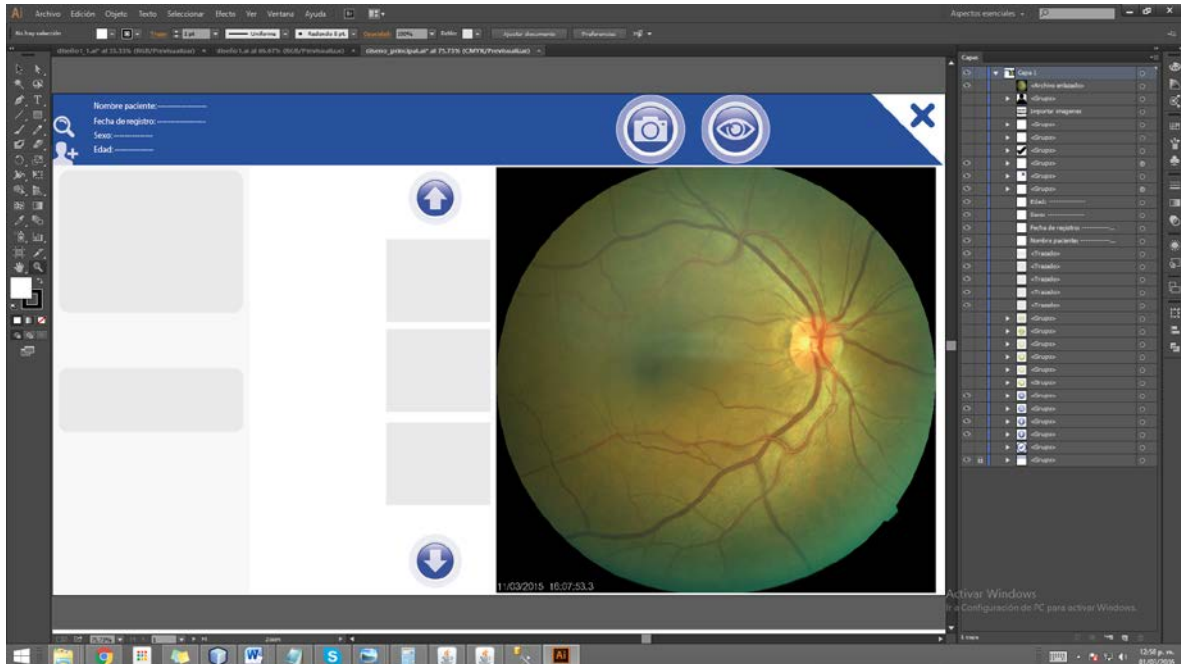


Figura 2. Propuesta de pantalla principal para proyecto "Fondo de Ojo".

Esta fue una propuesta sencilla, donde no se contemplaron todos los módulos del sistema, solo fue para mostrar cómo se verían los botones, un carrito con imágenes, información del usuario y paneles laterales de información del análisis.

Y a continuación se muestra la propuesta unificada, derivada de las dos propuestas iniciales:

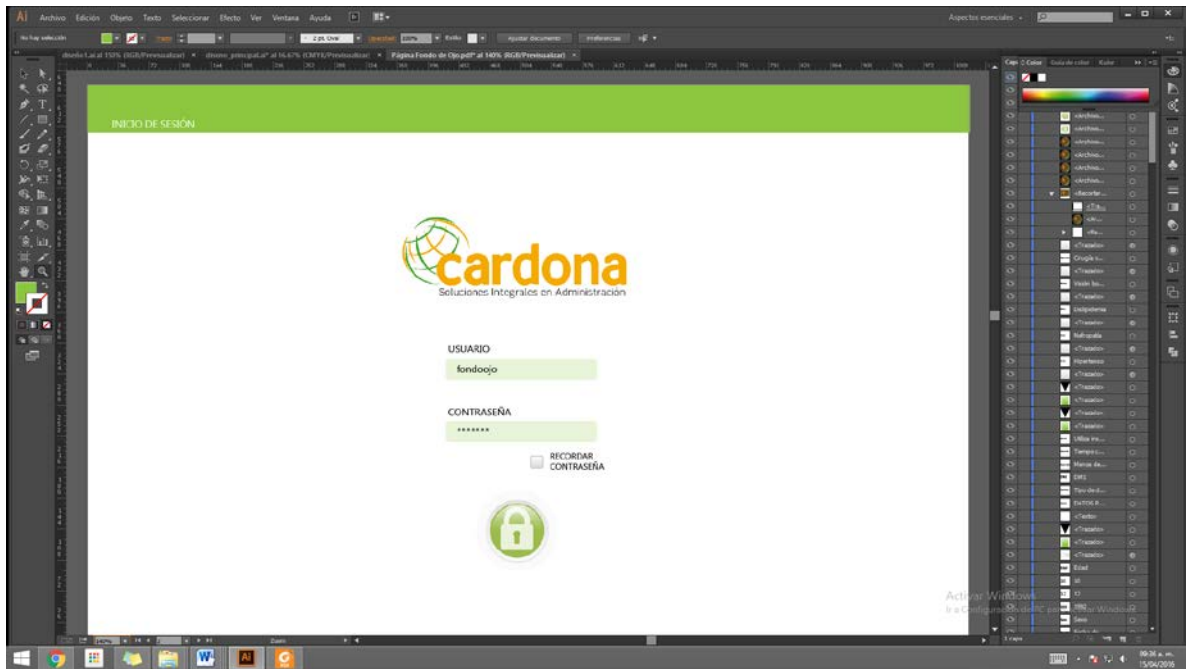


Figura 3. Diseño final para logueo "Fondo de Ojo".

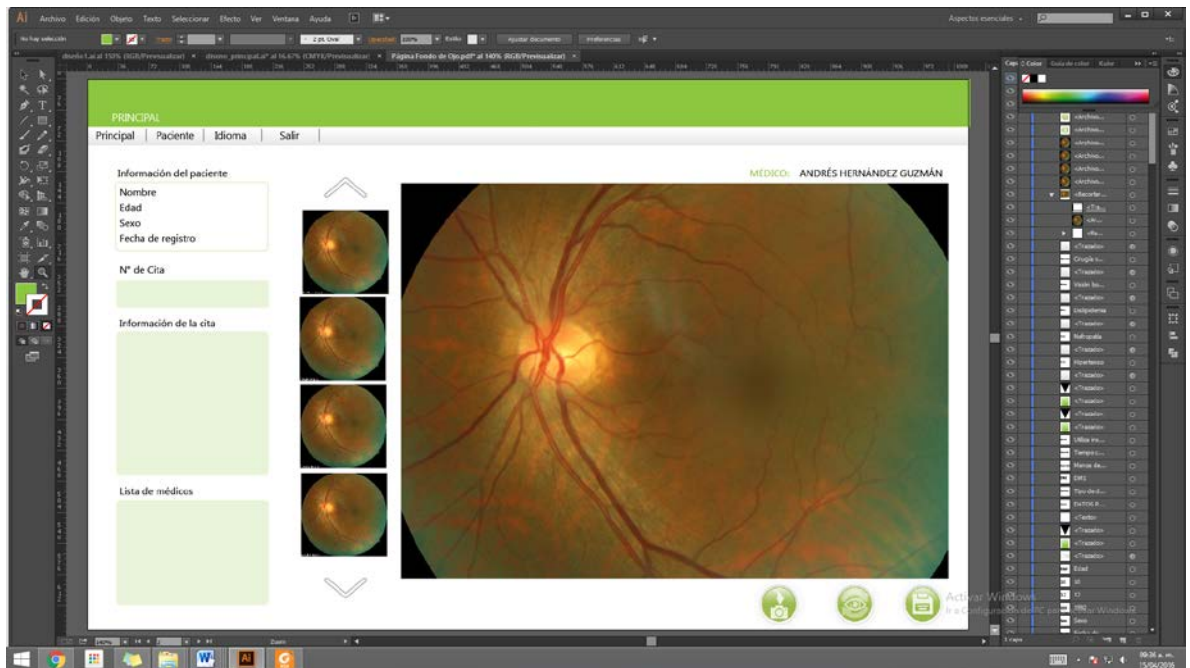


Figura 4. Diseño final para pantalla principal "Fondo de Ojo".

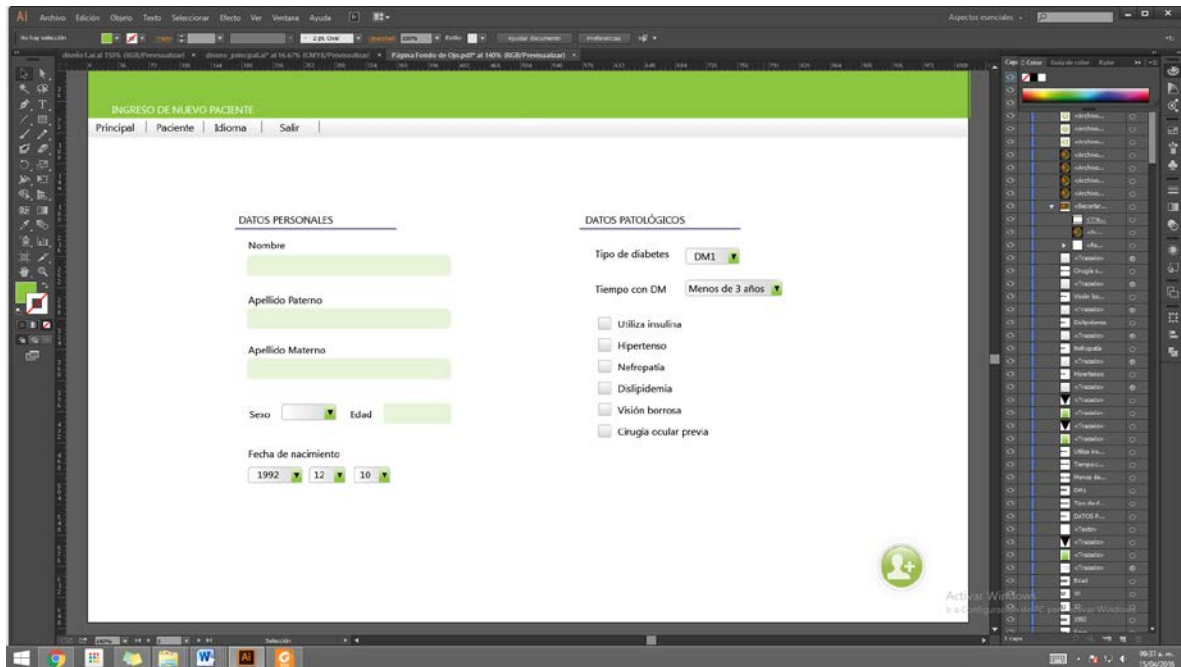


Figura 5. Diseño final para pantalla de nuevo usuario "Fondo de Ojo".

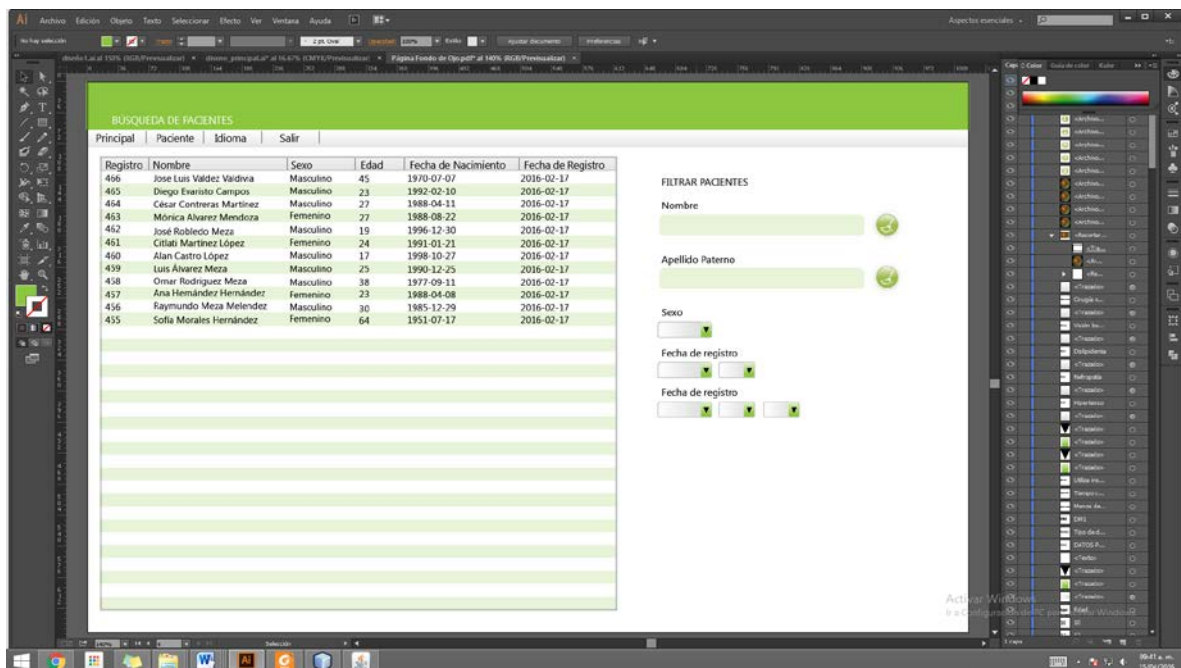


Figura 6. Diseño final para pantalla de búsqueda de pacientes "Fondo de Ojo".

Al principio no se había establecido una paleta de colores, es por eso que se muestra un diseño azul y otro verde. Al final se optó por dejar la interfaz en color verde y con ideas de ambas propuestas.

### 3.1.2 Resultados del diseño de interfaces de usuario

Aquí se muestra un poco del proceso que se llevó a cabo para realizar el diseño de las interfaces, para asimilarlo a la propuesta de diseño.

Esta es de la pantalla de inicio de sesión, utilizando el layout “Grid layout” o diseño de rejilla, se puede lograr a hacer algo como esto:

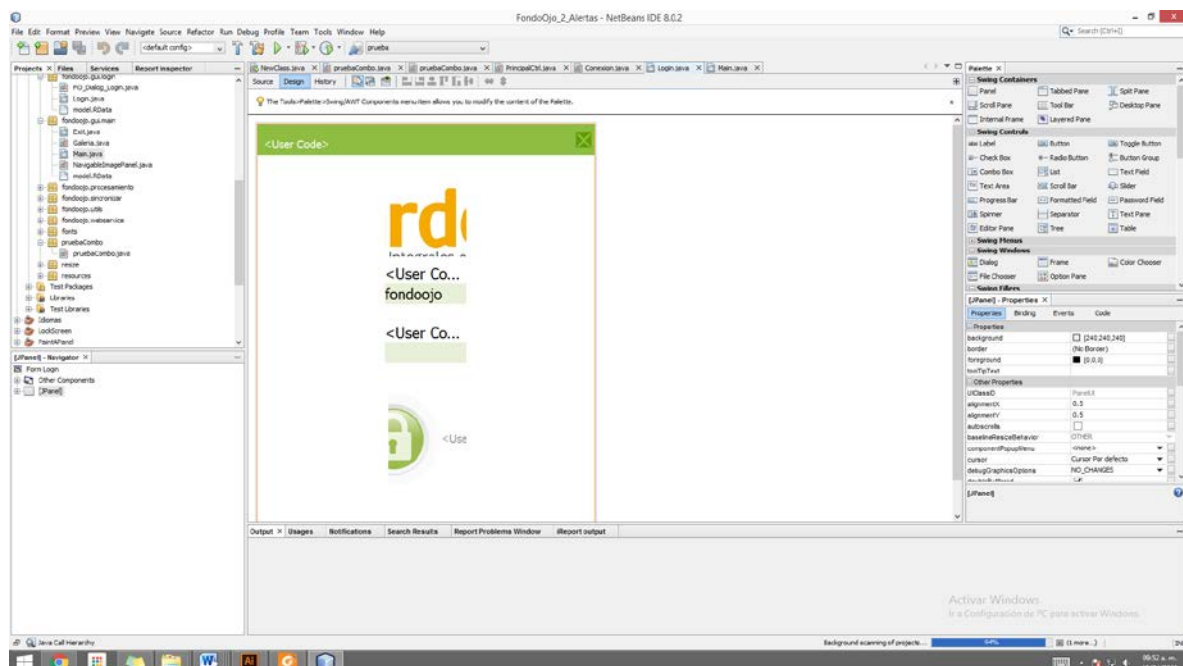


Figura 7. Diseño previo en NetBeans del inicio de sesión.

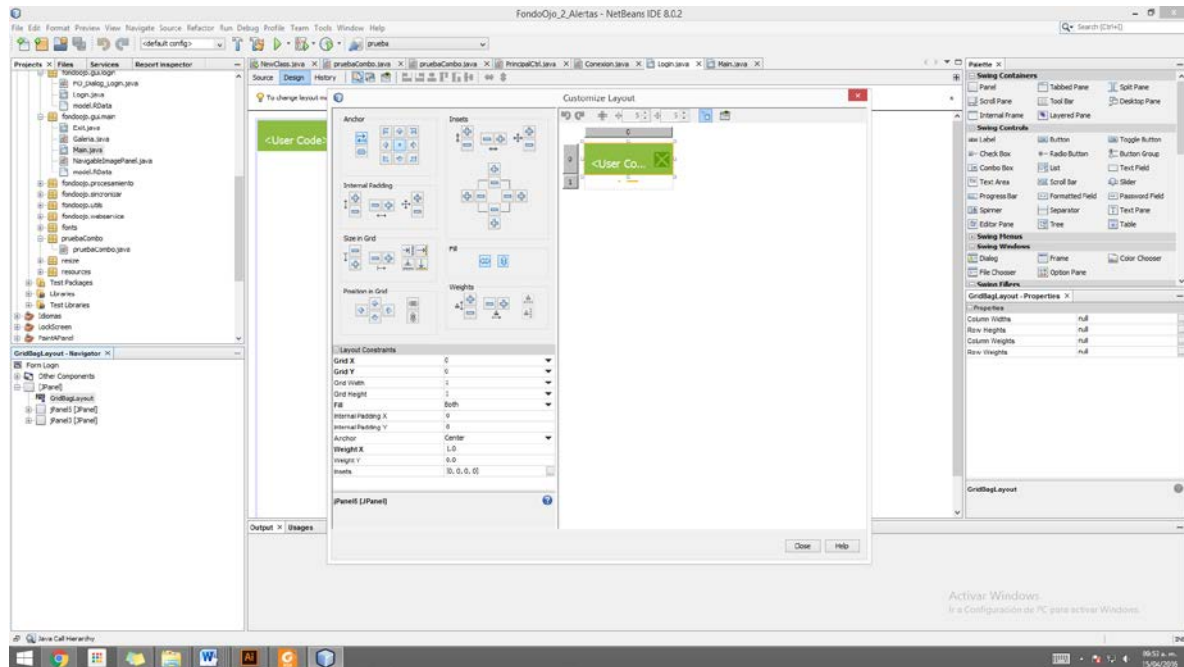


Figura 8. Uso del grid layout en pantalla de logueo.

Lo básico de este layout es no usar tamaños predefinidos, por eso no se muestra bien en la imagen como están ordenados los componentes, ya que al ejecutar el programa la “rejilla” hará el trabajo de ordenar los componentes, lo único que se debe modificar es el porcentaje de espacio que utilizará cada componente dentro de la rejilla.



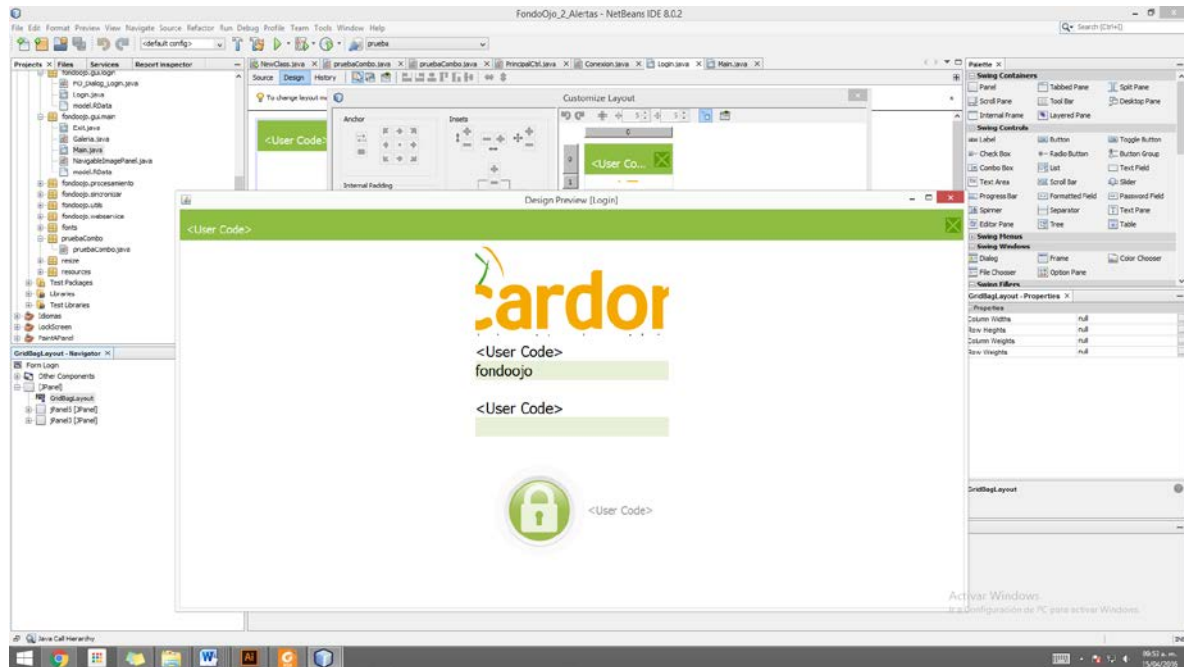


Figura 9. Vista previa del inicio de sesión.

Así se ve el diseño de la interfaz principal (dentro del entorno de desarrollo):

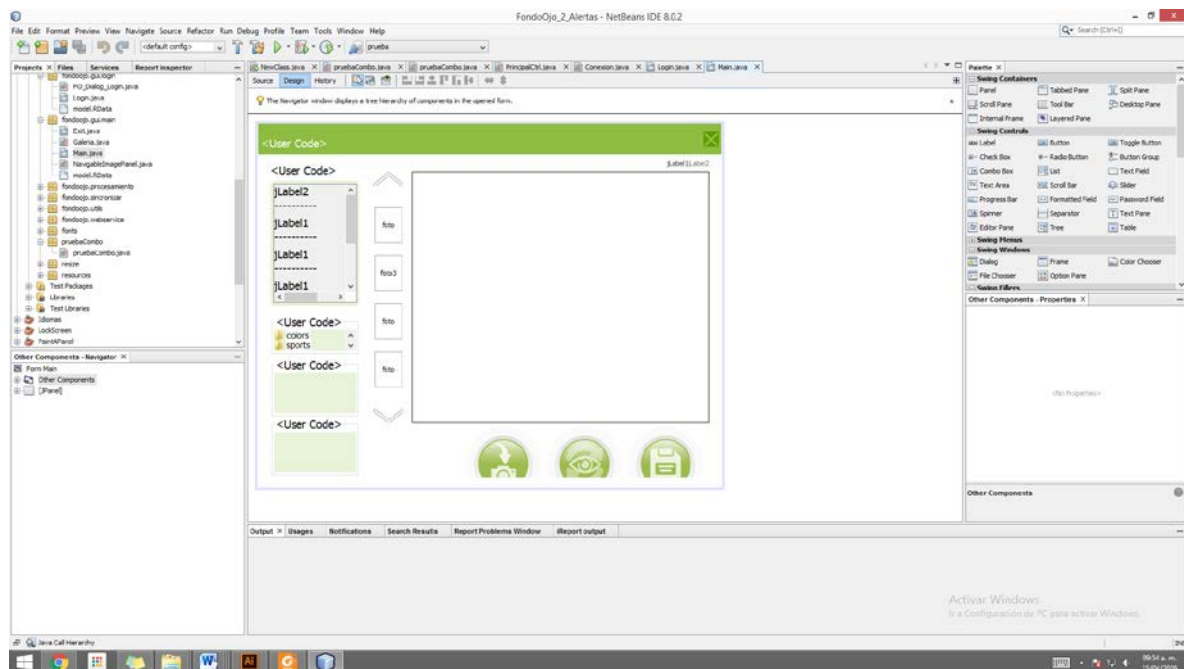


Figura 10. Diseño de la interfaz principal en NetBeans.





A continuación se muestra la vista utilizada para registrar un paciente:

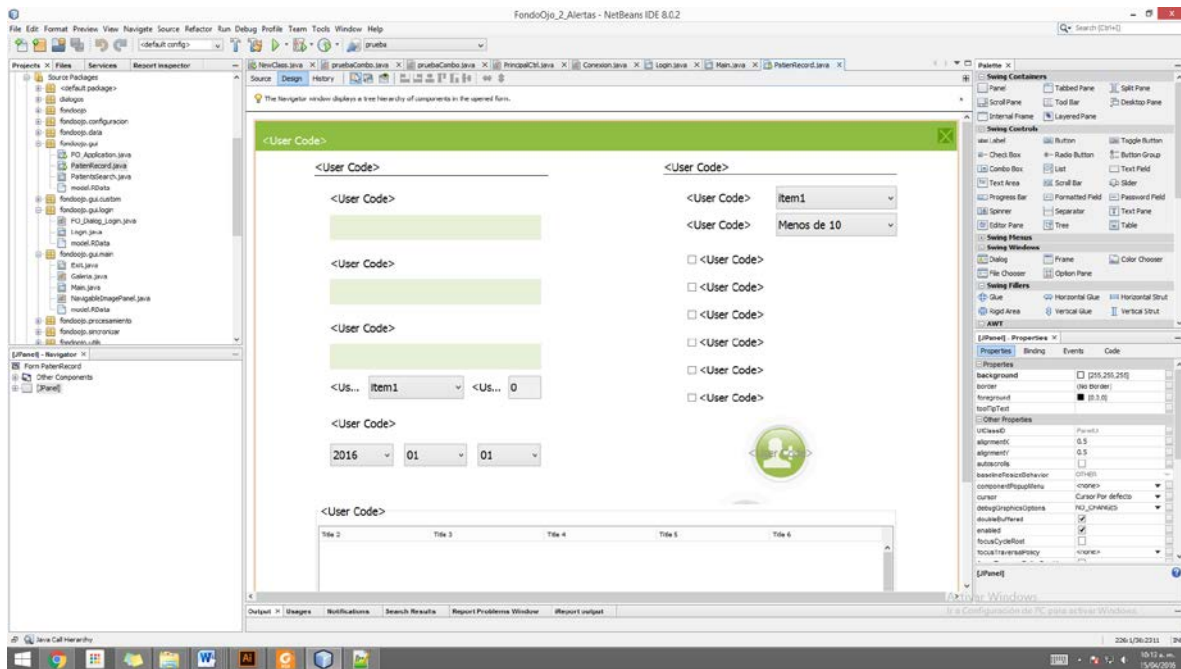


Figura 13. Vista para registrar pacientes.

### 3.1.3 Resultados del diseño adaptivo en Java

Se utilizó código para cambiar el tamaño de las ventanas, cambiar los iconos de los botones y redimensionarlos de acuerdo al porcentaje de escala de la resolución de la pantalla:

```
private void postInit() {  
  
loadList(FO_DBManager.get().getMedicos(FO_TempManager.get().getIdCliente()));  
  
jScrollPane3.getViewPort().setOpaque(false);  
  
jScrollPane3.setBorder(BorderFactory.createEmptyBorder());
```

```
if (height < 1040) {  
    sizeButton = (int) (height * .09);  
    Dimension tamano = new Dimension(sizeButton + 10, sizeButton + 40);  
    btnEnviarDatos.setPreferredSize(tamano);  
    btnEnviarDatos.setIcon(getIcon(btnguardar, sizeButton));  
    btnEnviarDatos.setRolloverIcon(getIcon(btnguardar2, sizeButton));  
    btnEnviarDatos.setPressedIcon(getIcon(btnguardar2, sizeButton));  
}
```

En todas las vistas se implementa parte del código mostrado anteriormente, ya que todas siguen la misma plantilla en cuanto a colores, fondos e iconos. El acomodo se da desde el diseñador de interfaces de NetBeans.

#### **3.1.4 Resultados de la tabla con sorter para filtrar datos**

Básicamente lo realizado fue enlazar los componentes del lado izquierdo (campos de texto y listas desplegables) con la tabla que ya contiene los registros de la base de datos. En la siguiente imagen se muestra como la tabla muestra toda la información, es decir, no muestra ningún filtrado de datos.

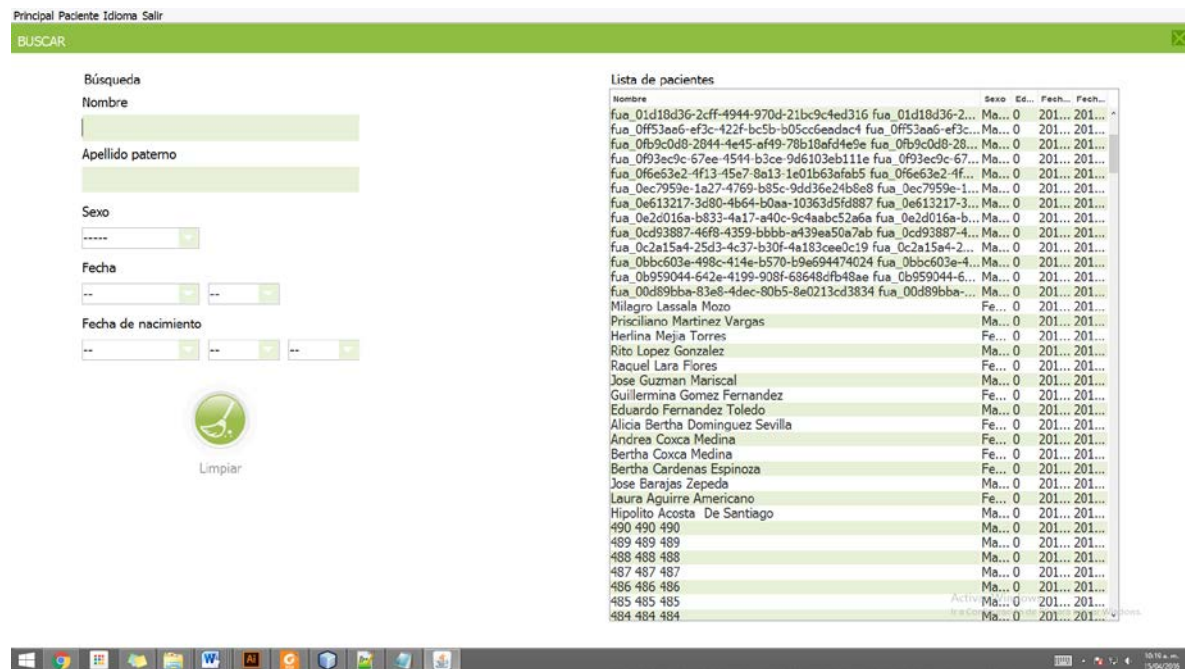


Figura 14. Interfaz para buscar un paciente (aún sin aplicar filtro de búsqueda).

Y en esta otra se puede ver como al escribir en algún campo de texto, automáticamente se comienza a filtrar los datos que contenía la tabla:

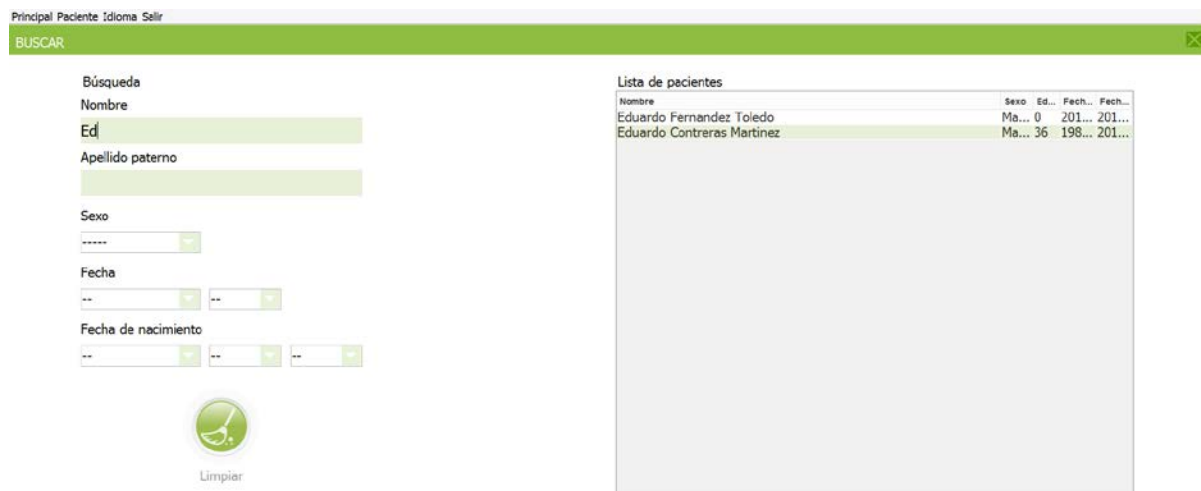


Figura 15. Aplicando un filtro para mostrar resultados de búsqueda.

A continuación se muestra algo de código implementado para lograr esta buena práctica:

```
TableRowSorter<TableModel> sorter;

RowFilter<TableModel, Object> filtro;

List<RowFilter<Object, Object>> rfs = new ArrayList<RowFilter<Object, Object>>();

TableModel model = (DefaultTableModel) jTable1.getModel();

sorter = new TableRowSorter<>(jTable1.getModel());

jTable1.setRowSorter(sorter);

private void updateFilter() {

    if (cbGender.getSelectedIndex() > 0) {

        rfs.add(RowFilter.regexFilter(cbGender.getSelectedItem().toString(), 2));

    }

    if (cbYearRegister.getSelectedIndex() > 0) {

        rfs.add(RowFilter.regexFilter(cbYearRegister.getSelectedItem().toString(), 5));

    }

    rfs.add(RowFilter.regexFilter(tfNombre.getText(), 1));

    rfs.add(RowFilter.regexFilter(tfApParterno.getText(), 1));

    filtro = RowFilter.andFilter(rfs);

    sorter.setRowFilter(filtro);

}
```

### 3.1.5 Resultado de las alertas personalizadas

Para realizar estas alertas se tomó como idea al componente JOptionPane ofrecido por Java, solo que con un diseño personalizado tanto en colores, como en iconos y animaciones. Esta es una captura de una alerta funcional del sistema:

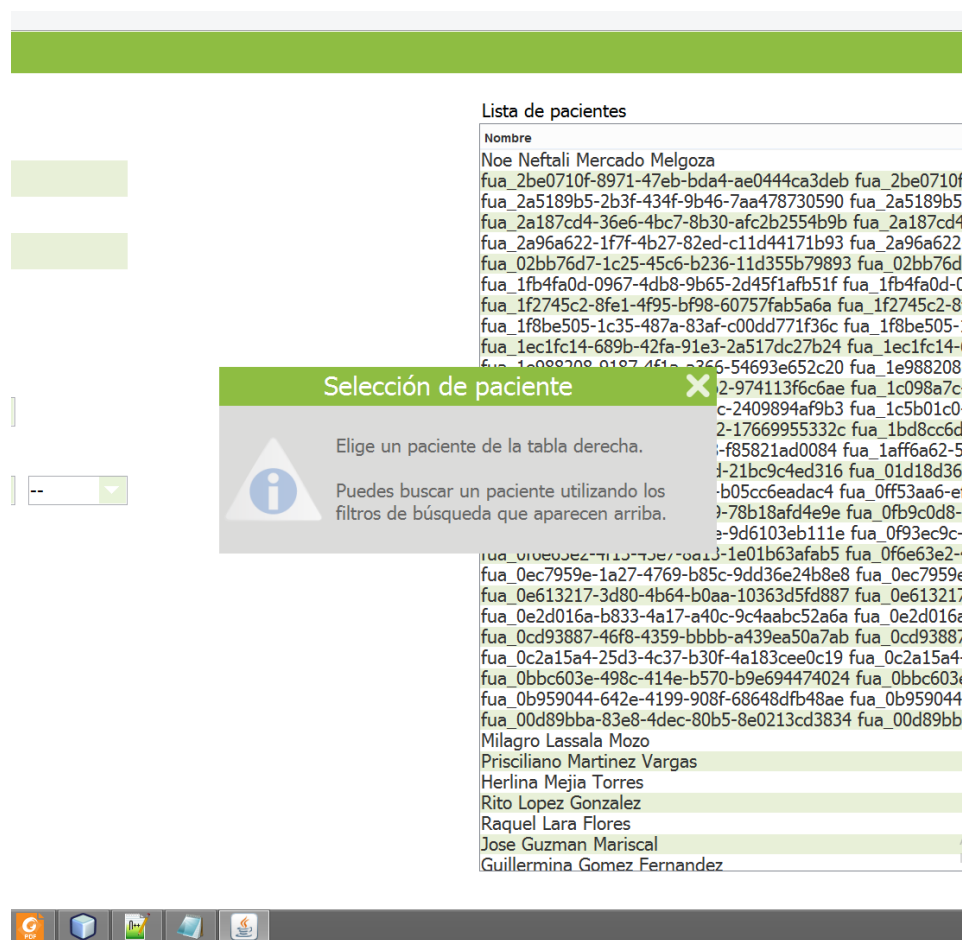


Figura 16. Alerta de instrucción de uso, informar al paciente como usar la interfaz.

Estos son los distintos diseños de alertas del sistema:

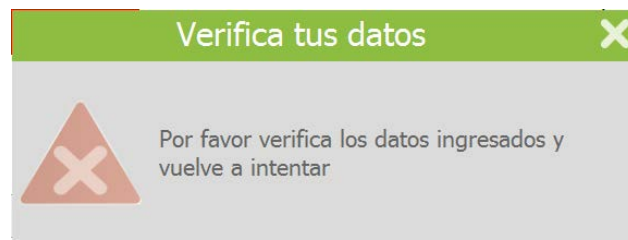


Figura 17. Alerta para errores del sistema.

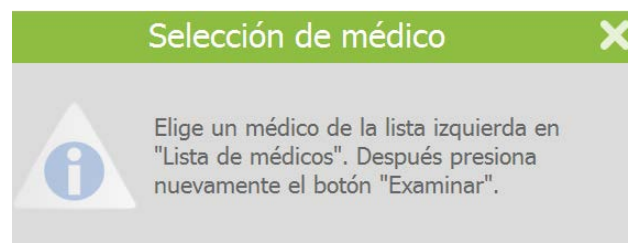


Figura 18. Alerta para información o ayuda de uso.

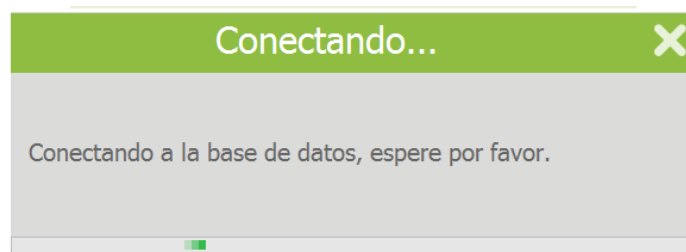


Figura 19. Alerta para conexión de base de datos.

Consta de un icono informativo, un título, un mensaje y un botón de cierre, así como una animación de desvanecimiento para que se cierre por sí misma en un intervalo de tiempo establecido. Así luce la plantilla de la alerta:

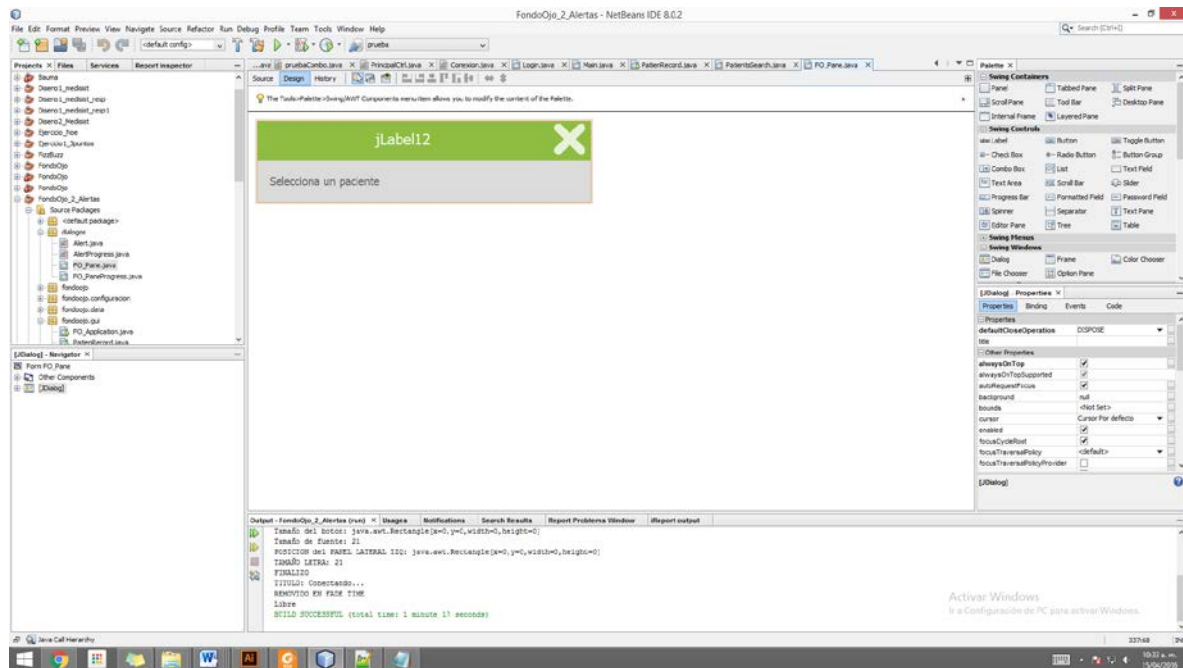


Figura 20. Plantilla genérica para alertas.

### 3.1.6 Resultado de la modificación del JComboBox

Para cambiar su aspecto basta con extender de la clase “BasicComboBoxUI” y sobrescribir el método “createArrowButton” de tal manera que se obtendrá un componente con distinto color, sombra y dirección de flecha:

```
public class pruebaCombo extends BasicComboBoxUI{

    public static ComboBoxUI createUI(JComponent c) {

        return new pruebaCombo();
    }
}
```

```

    }

    @Override

    protected JButton createArrowButton() {

        return new BasicArrowButton(

            BasicArrowButton.SOUTH,new Color(232,243,217),new

            Color(232,243,217),Color.white,new Color(232,243,217));

        }

    }

```

Donde “BasicArroButton” significa que es un botón “normal”, dentro se definen unos parámetros:

BasicArrowButton.SOUTH = botón con flecha apuntando al sur.

Los demás parámetros son colores, que definen: color de fondo, color de borde y color de flecha respectivamente.

Y así es como se puede llegar a ver implementado en una aplicación:

The screenshot displays a medical application form with two main sections: 'Datos personales' (Personal Data) and 'Datos patológico' (Pathological Data).

**Datos personales:**

- Nombre: A text input field with a light green background.
- Apellido paterno: A text input field with a light green background.
- Apellido materno: A text input field with a light green background.
- Sexo: A dropdown menu with 'Masculino' selected.
- Edad: A text input field with '0' entered.
- Fecha de nacimiento: Three dropdown menus for year (2016), month (01), and day (01).

**Datos patológico:**

- Tipo de diabetes: A dropdown menu with 'DM1' selected.
- Tiempo con DM: A dropdown menu with 'Menos de 10 años' selected.
- Utiliza insulina: ☐
- Hipertenso: ☐
- Nefropatía: ☐
- Dislipidemia: ☐
- Visión borrosa: ☐
- Cirugía ocular previa: ☐

At the bottom right, there are two circular buttons: 'Registrar' (with a person icon) and 'Cancelar' (with an 'X' icon).

Figura 21. Vista general de JComboBox personalizados.



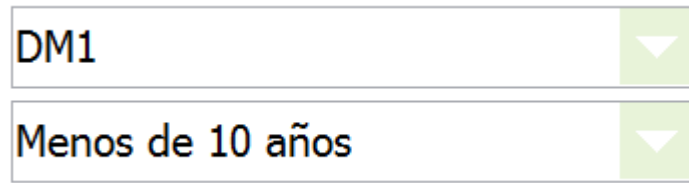


Figura 22. JComboBox personalizado.

### 3.1.7 Resultado de importar y aplicar tipografías personalizadas:

Para que los componentes tengan tipografía personalizada se debe “registrar” una fuente o tipografía para que después pueda ser utilizada:

```
public class CustomFonts {

    static Font font = new Font("Tahoma", Font.PLAIN, 24);

    public Font cuerpoFont(int size) {

        try {

            font = Font.createFont(Font.TRUETYPE_FONT,
                getClass().getResourceAsStream("LaoUI.ttf")).deriveFont(Font.PLAIN, size);

            GraphicsEnvironment ge =
                GraphicsEnvironment.getLocalGraphicsEnvironment();

            ge.registerFont(Font.createFont(Font.TRUETYPE_FONT,
                getClass().getResourceAsStream("LaoUI.ttf")));

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

```
return font;  
}
```

Los archivos de tipografías deben estar almacenados en algún paquete, dentro del proyecto:

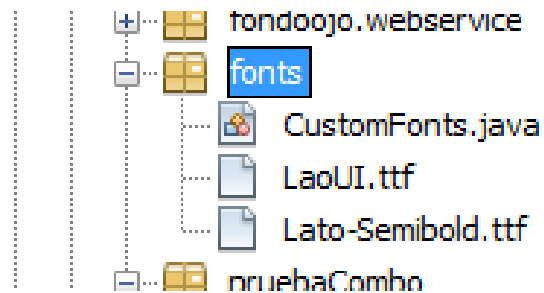


Figura 23. Paquete con tipografías.

Después solo se debe hacer la instanciación de la nueva clase de tipografías y establecerla a los componentes deseados:

```
customFont = new CustomFonts();  
btnNewImages.setFont(customFont);
```

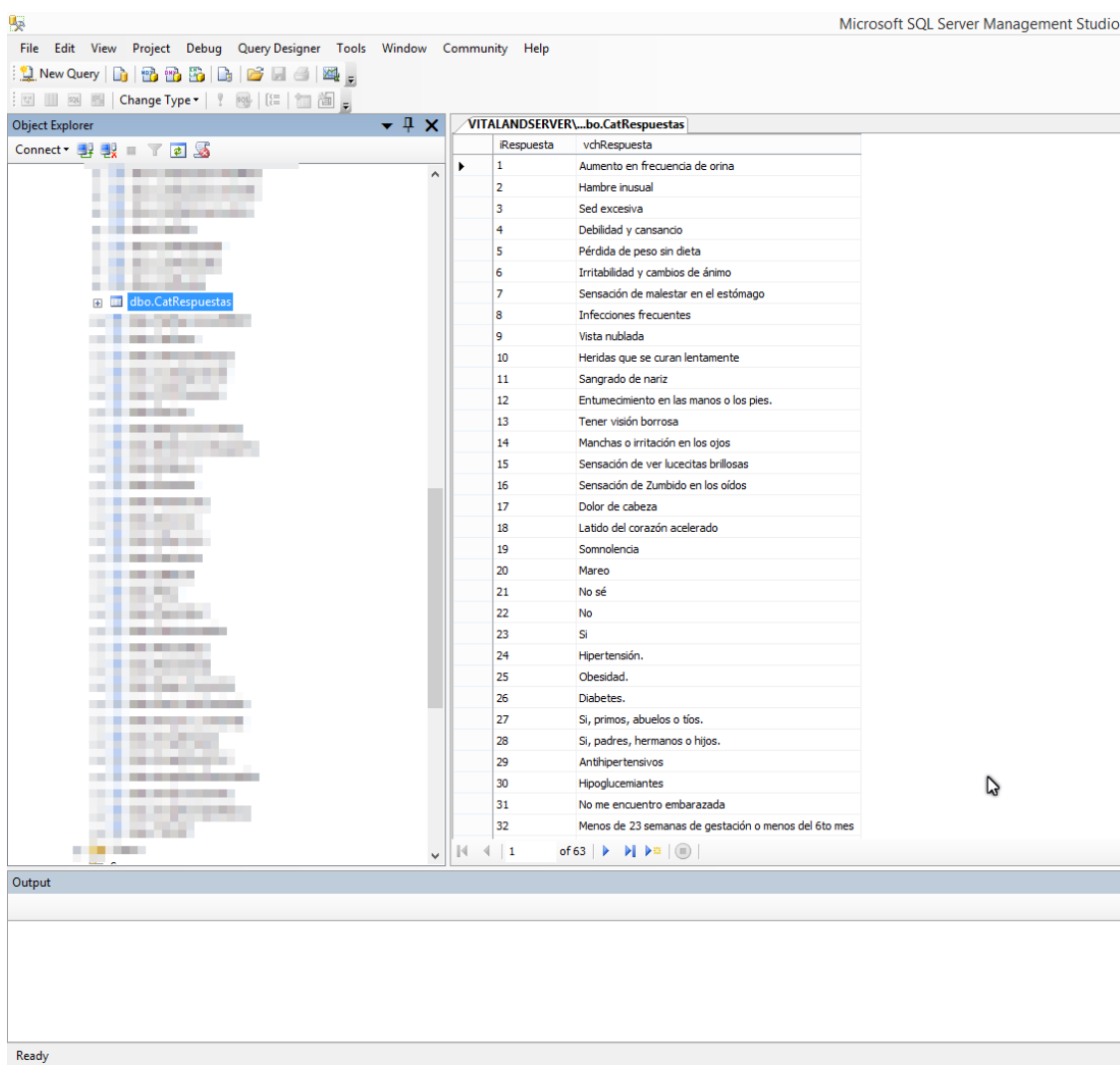
Con esto es posible implementar cualquier tipografía en sistemas que requieran alguna fuente personalizada.

## 3.2 RESULTADOS DEL PROYECTO ATENCIÓN VITAL V1

### 3.2.1 Resultado del diseño de tablas de la base de datos (cuestionario)

En las siguientes ilustraciones se muestra la creación de las tablas de la base de datos para almacenar la información del cuestionario:

Esta es la tabla con las respuestas del cuestionario.



Microsoft SQL Server Management Studio

Object Explorer: Connect, Server Explorer, Object Explorer, Query Designer, Tools, Window, Community, Help

dbo.CatRespuestas

IdRespuesta	vdRespuesta
1	Aumento en frecuencia de orina
2	Hambre inusual
3	Sed excesiva
4	Debilidad y cansancio
5	Pérdida de peso sin dieta
6	Irritabilidad y cambios de ánimo
7	Sensación de malestar en el estómago
8	Infecciones frecuentes
9	Vista nublada
10	Heridas que se curan lentamente
11	Sangrado de nariz
12	Entumecimiento en las manos o los pies.
13	Tener visión borrosa
14	Manchas o irritación en los ojos
15	Sensación de ver lucecitas brillosas
16	Sensación de Zumbido en los oídos
17	Dolor de cabeza
18	Latido del corazón acelerado
19	Somnolencia
20	Mareo
21	No sé
22	No
23	Si
24	Hipertensión.
25	Obesidad.
26	Diabetes.
27	Si, primos, abuelos o tíos.
28	Si, padres, hermanos o hijos.
29	Antihipertensivos
30	Hipoglucemiantes
31	No me encuentro embarazada
32	Menos de 23 semanas de gestación o menos del 6to mes

Output

Ready

Figura 24. Tabla de respuestas para cuestionario.

	Column Name	Data Type	Allow Nulls
▶	iRespuesta	smallint	<input type="checkbox"/>
	vchRespuesta	varchar(250)	<input type="checkbox"/>
			<input type="checkbox"/>

Figura 25. Estructura de tabla de respuestas.

La siguiente tabla contiene las respuestas de las preguntas, es decir, en esta tabla se guardarán los id de las respuestas, el usuario o paciente y un id de cuestionario ya que el formulario se conforma de tres partes: cuestionario de enfermedades crónicas, actividad física y cuestionario personal.

	Column Name	Data Type	Allow Nulls
▶	idNumResp	smallint	<input type="checkbox"/>
	iIdPaciente	int	<input type="checkbox"/>
	iIdRespuesta	smallint	<input type="checkbox"/>
	iCuestionario	smallint	<input type="checkbox"/>
			<input type="checkbox"/>

Figura 26. Estructura de tabla para almacenar respuestas del usuario.

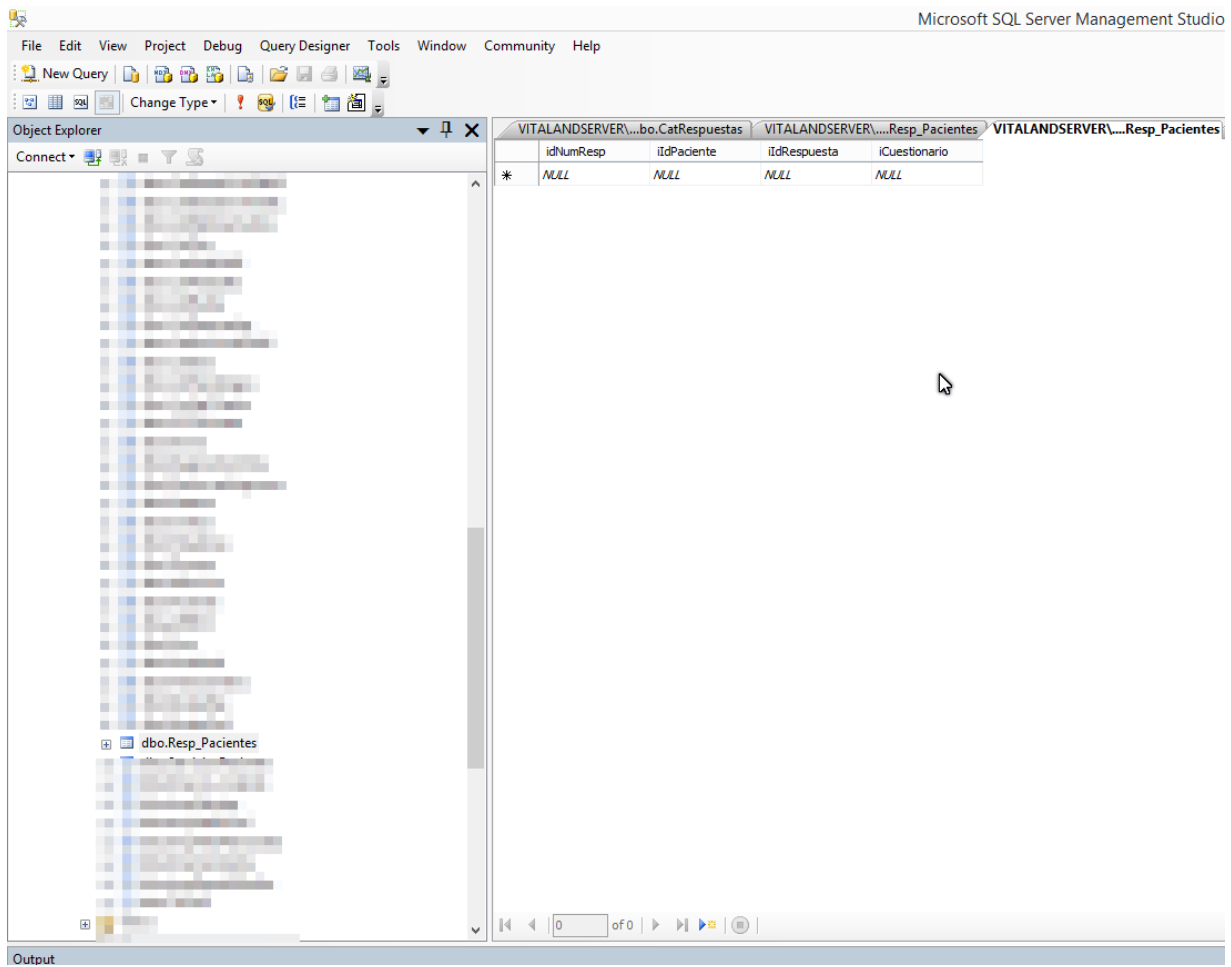


Figura 27. Tabla para guardar respuestas (sin datos).

### 3.2.2 Resultado de la codificación del cuestionario

De esta manera se analizan los componentes dentro de un panel (para evitar hacerlo de uno por uno), sirve para hacer distintas cosas, depende de la imaginación y necesidades del programador:

```
for (int i = 0; i < cuestionario.getComponents().length; i++) {
    if (cuestionario.getComponent(i) instanceof JCheckBox) {
```

```

if (((JCheckBox) cuestionario.getComponent(i)).isSelected()) {
    listaGuardar.add(respuesta);
} else {
    listaGuardar.add("Vacio");
}

```

Después de haber analizado todas las casillas seleccionadas, se guardan en una lista temporal, y después, mediante otro ciclo, se almacenan en la base de datos:

```

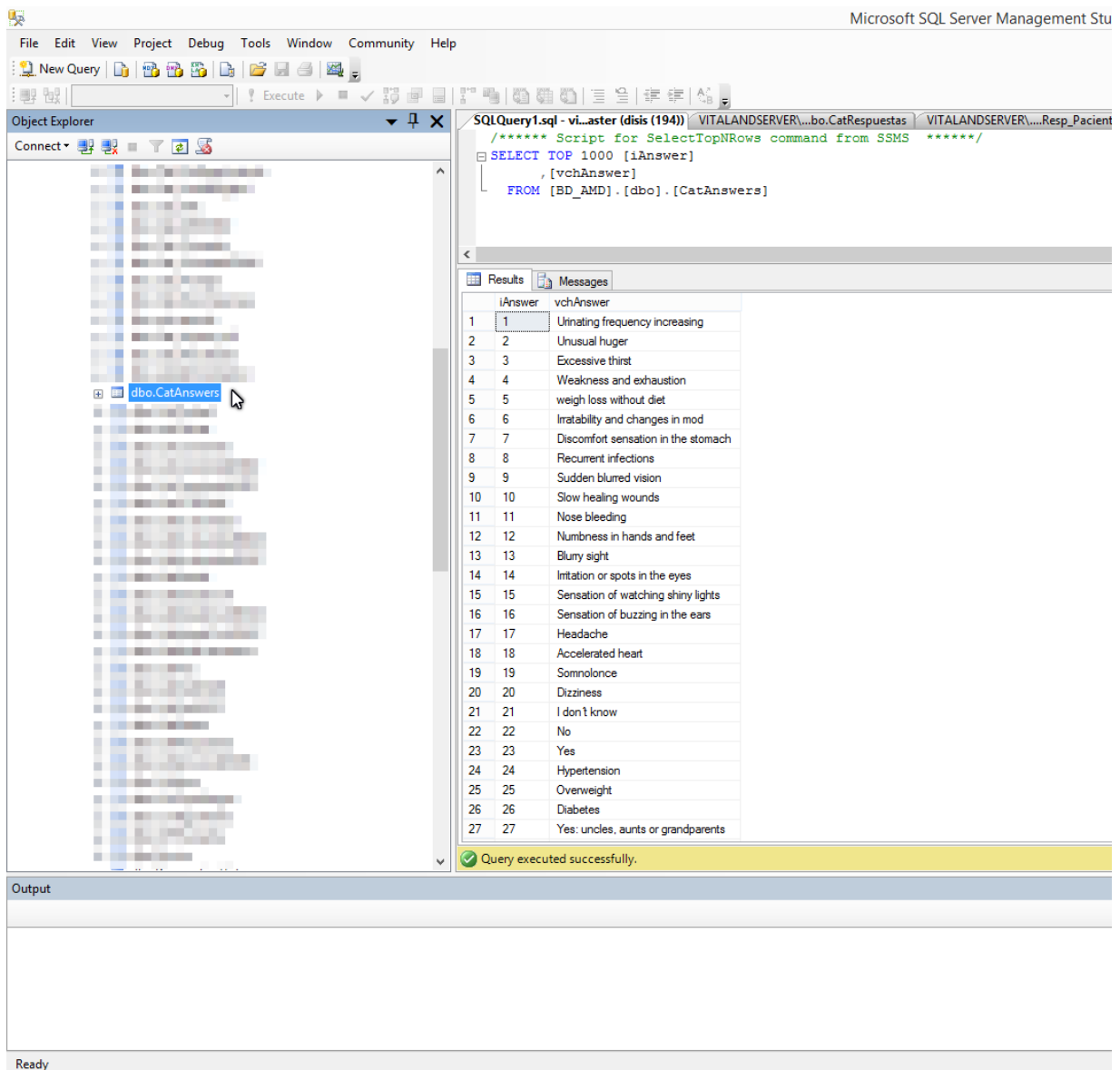
for (Object listaGuardar1 : listaGuardar) {
    try {
        switch (opcion) {
            case "nuevo":
                ultimoldResp++;
                query = "insert into Resp_Pacientes
(idNumResp,ildPaciente,ildRespuesta,iCuestionario) values('" + ultimoldResp + "','" +
idPaciente + "','" + getIdRespuesta(String.valueOf(listaGuardar1)) + "','" +
numCuestionario + "')";
                break;
            case "modificar":
                query = "update Resp_Pacientes set ildRespuesta =
'" + getIdRespuesta(String.valueOf(listaGuardar1)) + "'" where idNumResp =
'" + idsNumResp.get(contadorResp-1) + "'"; break;
        }
        ConexionDB.scSQL.executeUpdate(query);
    }
}

```

}

### 3.2.3 Resultado de la creación de tablas y código para versión en inglés

A continuación se muestra la imagen de la tabla utilizada para el almacenamiento de las respuestas del cuestionario para la versión en inglés:



Microsoft SQL Server Management Studio

File Edit View Project Debug Tools Window Community Help

New Query

Object Explorer

Connect

SQLQuery1.sql - vl...aster (disis (194)) VITALANDSERVER\...bo.CatRespuestas VITALANDSERVER\...Resp\_Pacient

```
/* Script for SelectTopNRows command from SSMS */
SELECT TOP 1000 [iAnswer]
, [vchAnswer]
FROM [BD_AMD].[dbo].[CatAnswers]
```

Results Messages

	iAnswer	vchAnswer
1	1	Urinating frequency increasing
2	2	Unusual hunger
3	3	Excessive thirst
4	4	Weakness and exhaustion
5	5	weigh loss without diet
6	6	Irritability and changes in mod
7	7	Discomfort sensation in the stomach
8	8	Recurrent infections
9	9	Sudden blurred vision
10	10	Slow healing wounds
11	11	Nose bleeding
12	12	Numbness in hands and feet
13	13	Blurry sight
14	14	Itiation or spots in the eyes
15	15	Sensation of watching shiny lights
16	16	Sensation of buzzing in the ears
17	17	Headache
18	18	Accelerated heart
19	19	Somnolence
20	20	Dizziness
21	21	I don't know
22	22	No
23	23	Yes
24	24	Hypertension
25	25	Overweight
26	26	Diabetes
27	27	Yes: uncles, aunts or grandparents

Query executed successfully.

Output

Ready

Figura 28. Tabla con respuestas en inglés.

```
if(ElegirLogueo.banderaEspañol){
    query = "select * from CatRespuestas";
}else{
    query = "select * from CatAnswers";
}
```



Logga in

Användarnamn

Lösenord

Logga in

Bakåt

	-	#	+	@	&	/	?	*	_	←
1	2	3	4	5	6	7	8	9	0	↔
q	w	e	r	t	y	u	i	o	p	↑
a	s	d	f	g	h	j	k	l	ä	ö
z	x	c	v	b	n	m				

© Copyright 2015, Vital Assistance. All rights reserved.  
Ir a Configuración de PC para activar Windows.

Figura 30. Resultado real del teclado virtual para versión sueca.

## 3.3 RESULTADOS DEL PROYECTO ATENCION VITAL V2

### 3.3.1 Propuesta de diseño

En esta parte se muestran capturas sobre el diseño propuesto para la versión 2 del sistema “Atención Vital”. Para trabajar con este diseño se implementaron herramientas de diseño como Photoshop e Illustrator, así como el programa PowerPoint para hacer la presentación formal.

A continuación se presentan algunas capturas de los sistemas de diseño en donde se llevó a cabo el diseño de las interfaces. En la primera imagen se muestran los componentes que conforman la mayor parte de las interfaces:

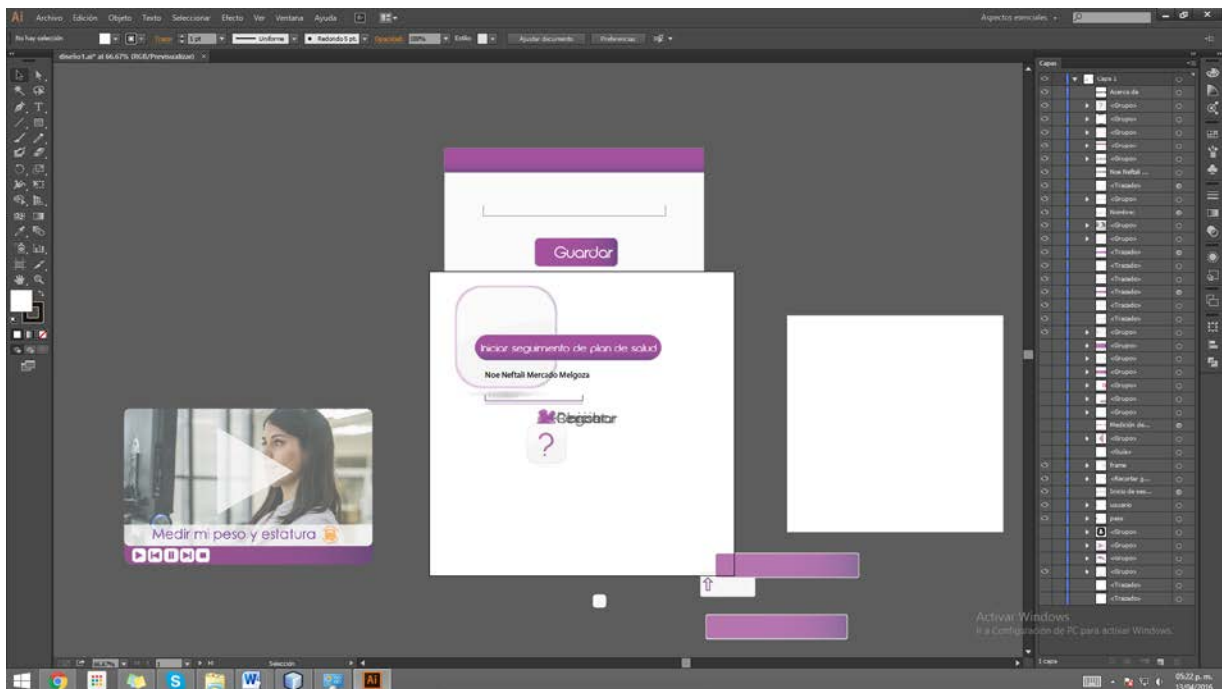


Figura 31. Creación de recursos con Illustrator.

En la siguiente imagen se muestra una parte del diseño para la nueva versión del cuestionario:

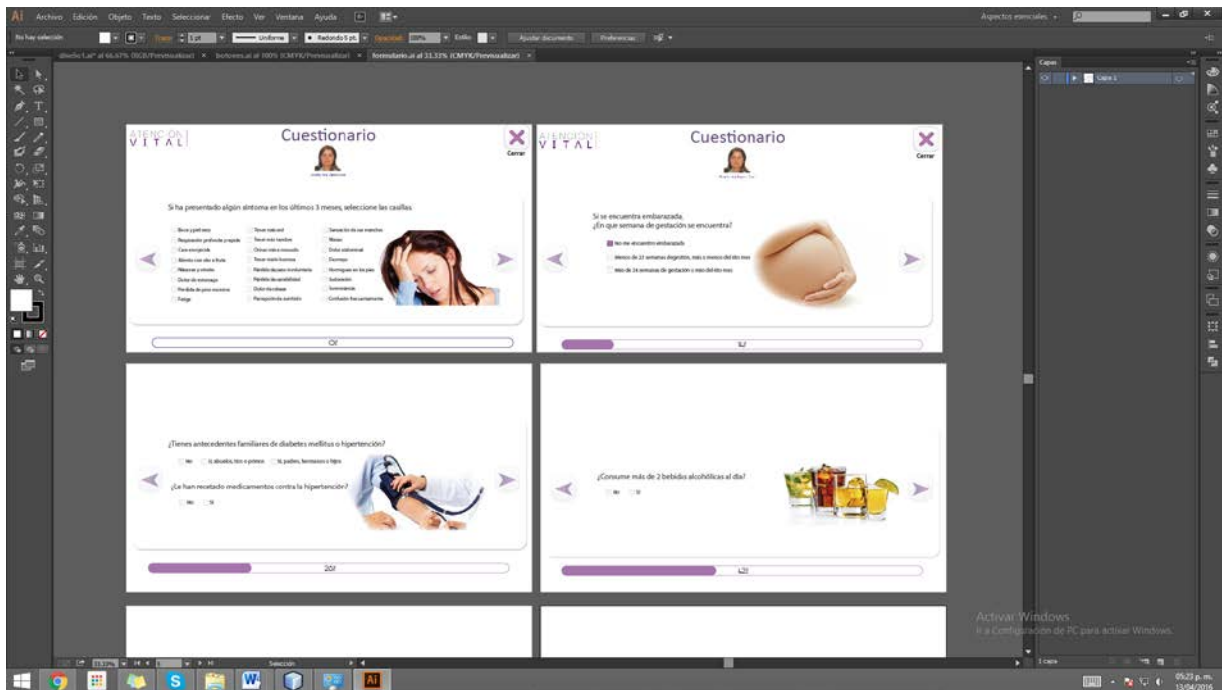
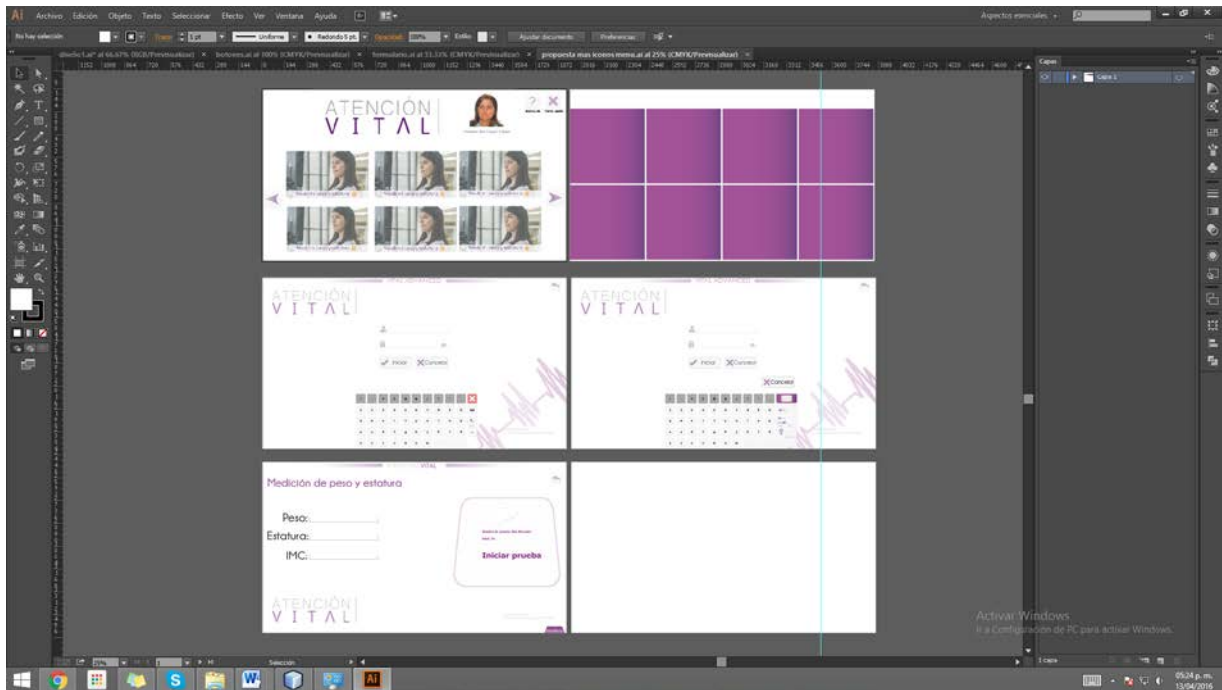
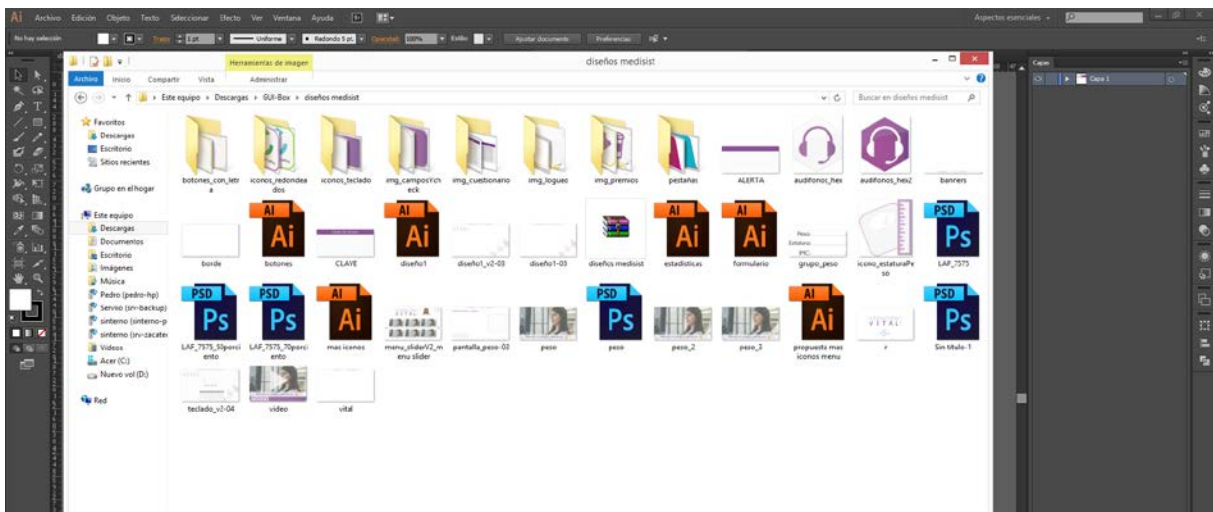


Figura 32. Diseño de cuestionario en Illustrator.

Ahora en esta imagen, se puede observar el diseño que se le dio a la interfaz de menú principal y al teclado virtual:



En la siguiente imagen se muestran los archivos creados con los programas de diseño, y se muestran los componentes que conformaran las interfaces:



Por último la imagen que muestra el diseño de uno de los banners para el acceso a alguna parte del sistema:

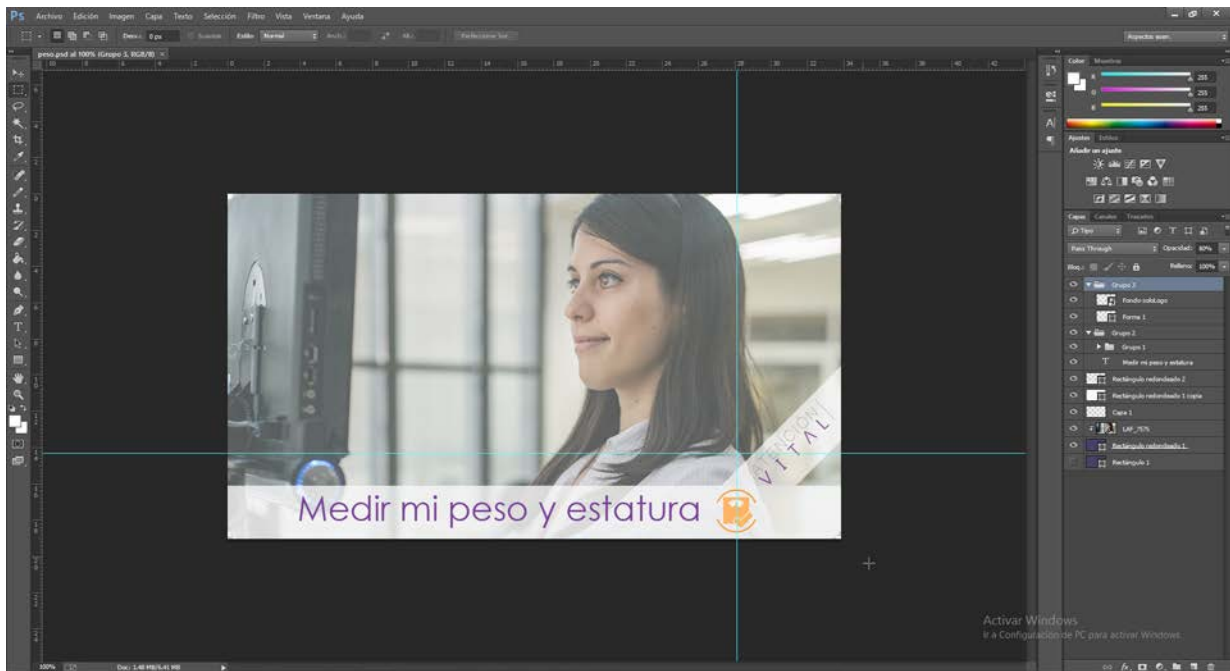


Figura 35. Diseño de banner para menú principal.

A continuación se muestran algunas capturas de la propuesta formal de diseño para la segunda versión del sistema, a la cual ya se le dio un orden de secuencia y una presentación entendible para los desarrolladores. En la ilustración 36 se puede ver el tipo de inicio de sesión:



Figura 36. Propuesta para el tipo de inicio de sesión.

En la siguiente imagen se muestra el orden de la pantalla de menú principal:

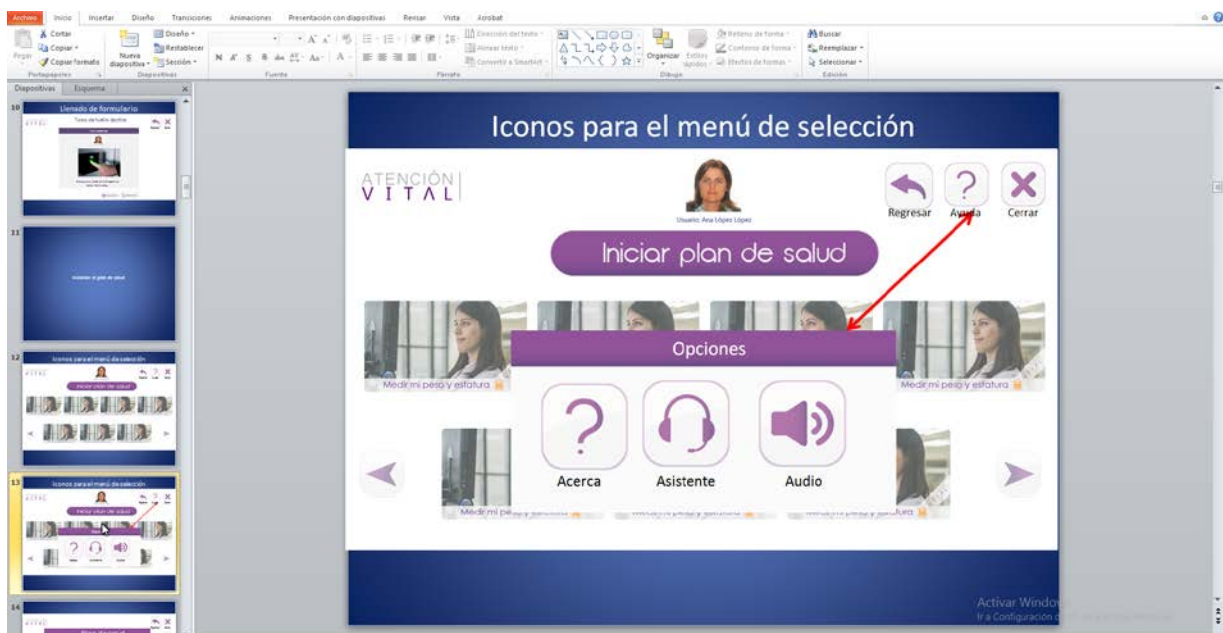


Figura 37. Propuesta de diseño para la ayuda y menú principal.

A continuación, algunas capturas para demostrar el orden y estructura de las pantallas de peso y estatura, así como de las pantallas del cuestionario y de premios para pacientes:

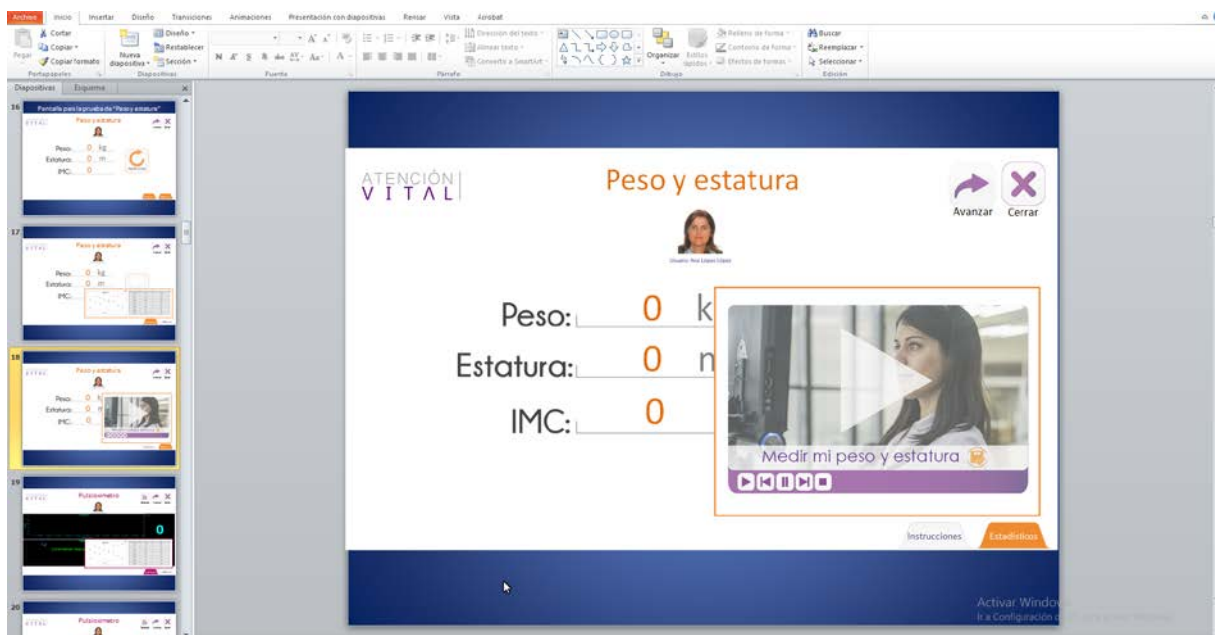


Figura 38. Propuesta de diseño para interfaz de peso y estatura.



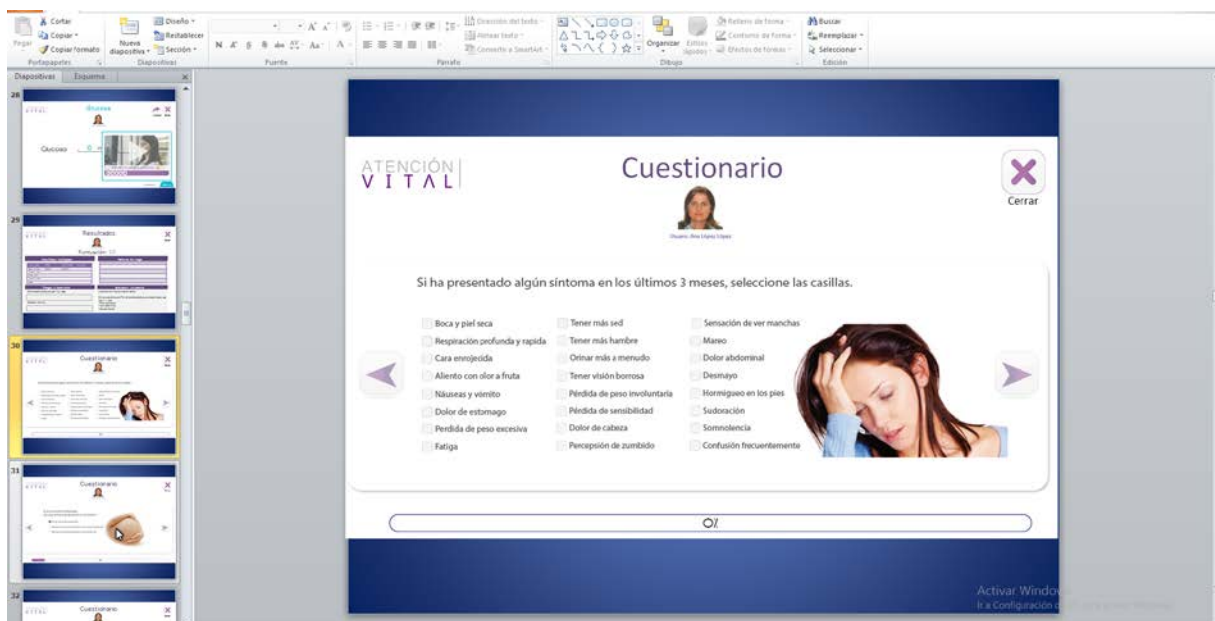


Figura 39. Propuesta de diseño para cuestionario de salud.

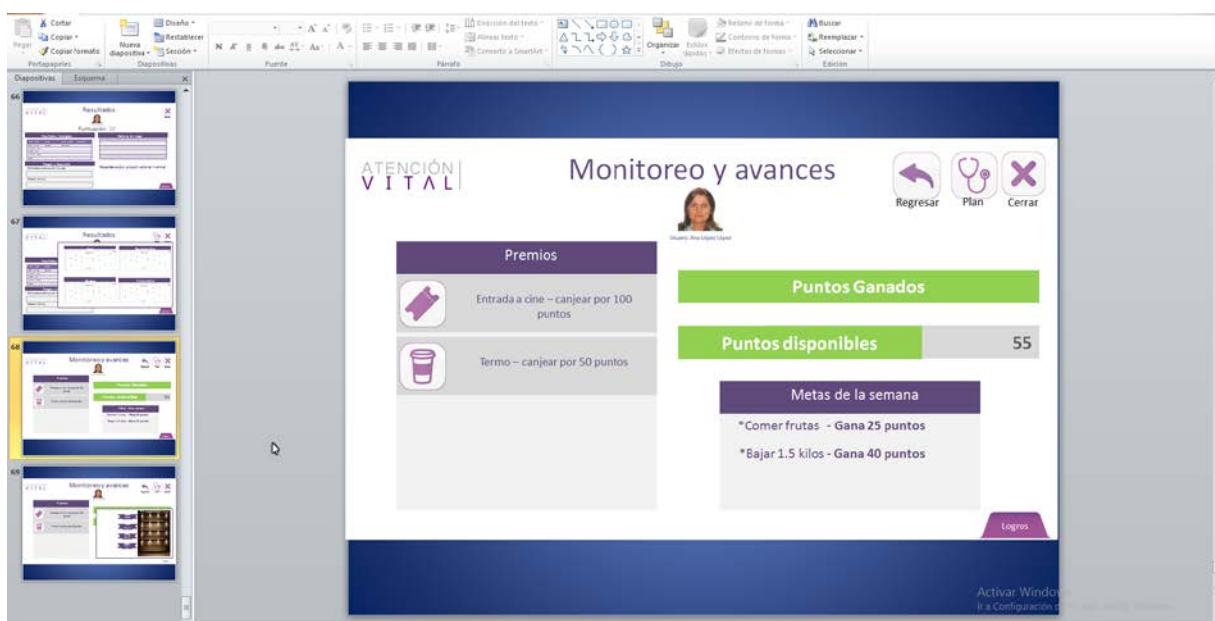


Figura 40. Propuesta de diseño para interfaz de monitoreo.



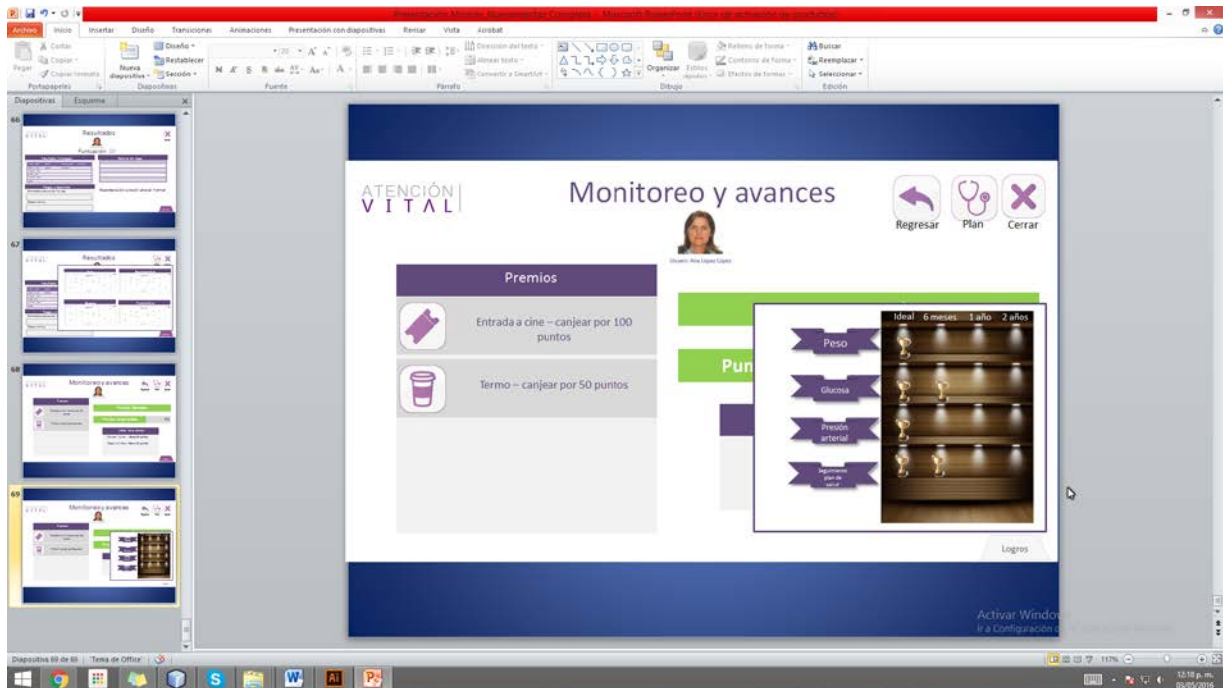


Figura 41. Propuesta de diseño para interfaz de premios.

### 3.3.2 Resultado del diseño del tipo de inicio de sesión

El siguiente resultado enlista algunas capturas previas de la vista de selección de inicio de sesión, mostrando el inicio manual (usuario y contraseña), inicio con código QR e inicio con huella dactilar:

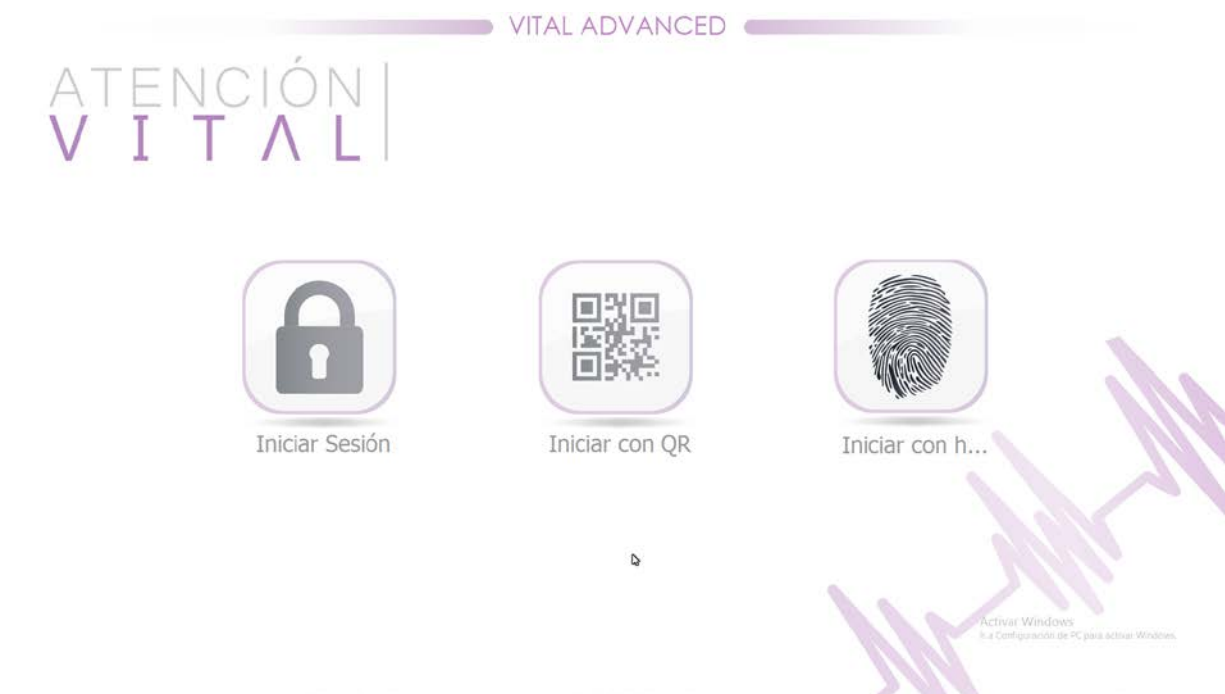


Figura 42. Vista de diseño del tipo de inicio de sesión.

Y capturas de cómo es que se desarrolló esta vista:

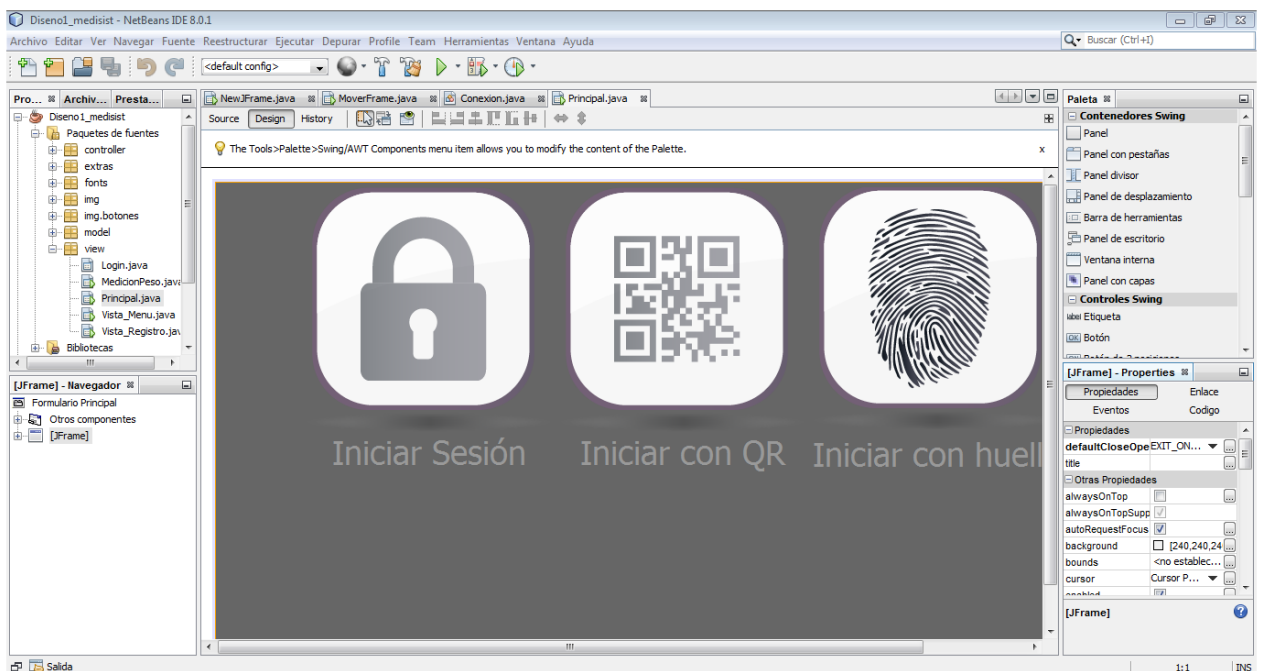


Figura 43. Diseño del tipo de inicio de sesión en NetBeans.

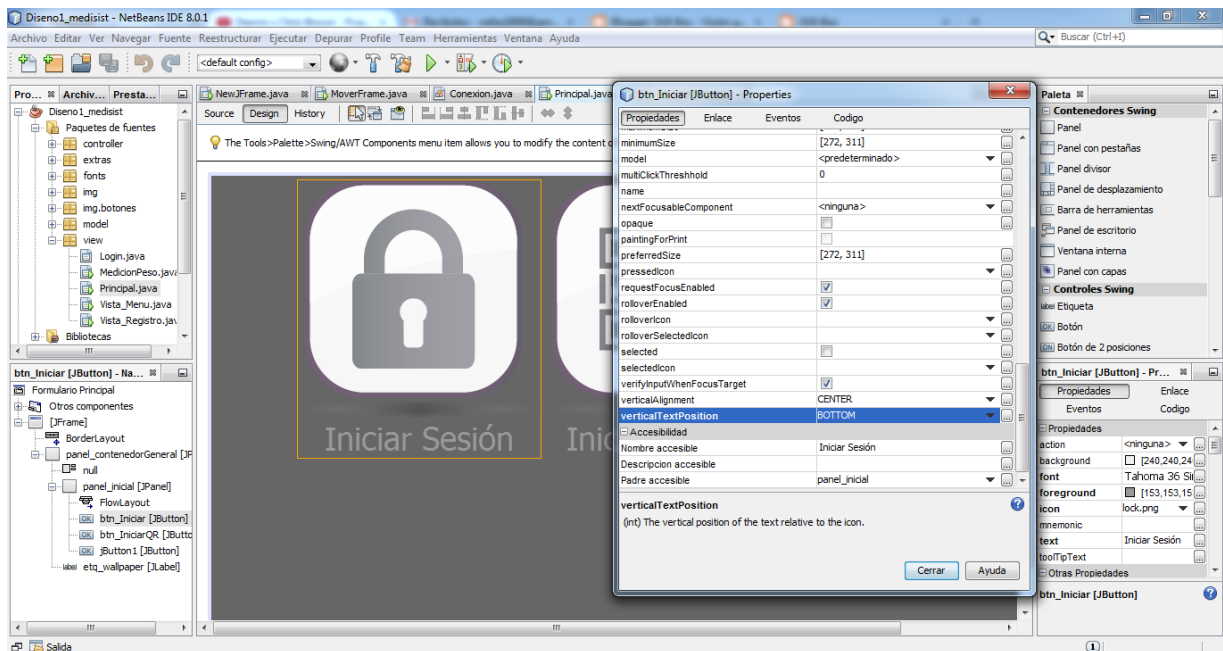


Figura 44. Personalización de botones.

Con el siguiente código lo que se consigue es obtener las medidas de la pantalla anfitriona y colocar la imagen de fondo, de manera que cubra toda la pantalla y no se distorsione:

```
private void medidas_pantalla() {
    principal.etq_wallpaper.setSize((int) principal.getWidth(), (int)
principal.getHeight());

    cabecera = new
ImageIcon(imagen.getImage()).getScaledInstance(principal.etq_wallpaper.getSize().wi
dth, principal.etq_wallpaper.getSize().height, Image.SCALE_SMOOTH));
    principal.etq_wallpaper.setIcon(cabecera);
}
```

```
}
```

Y para que los botones se muestren en una sola fila con tres columnas y que tomen un tamaño basado en porcentaje a la pantalla anfitriona:

```
private void posicionarBot(Dimension dim) {  
    posicionAl = (dim.height / 2) - (principal.btn_Iniciar.getSize().height/2);  
    int maxButtonWidth = principal.btn_Iniciar.getSize().width;  
    if(maxButtonWidth < principal.btn_IniciarQR.getSize().width)  
        maxButtonWidth = principal.btn_IniciarQR.getSize().width;  
    posicionAn = ((dim.width / 4) - maxButtonWidth)-15;  
    if(posicionAn<0)  
        posicionAn = ((dim.width / 3) - maxButtonWidth)-15;  
    int sumaTamanos = (272*3)+(posicionAn*3);  
    if(sumaTamanos>dim.width){  
        posicionAn = (int) ((dim.width / 3.1) - maxButtonWidth);  
    }  
    panel_login.setSize(dim);  
    panel_login.setLocation(0, dim.height + 1);  
}
```

### 3.3.3 Resultado de las animaciones para el tipo de inicio de sesión

A continuación se plasman algunas capturas que tratan de explicar el funcionamiento básico de las animaciones implementadas en esta parte del sistema, básicamente es mostrar y ocultar componentes mediante un sistema de movimiento para desplazar a estos mismos de manera ascendente y descendente.

El primer estado, donde todo está estático, sin ningún tipo de animación, es como se ve en la siguiente imagen:



Figura 45. Animación en estado inicial.

En la siguiente imagen se muestra una animación que se desencadena al presionar el icono del candado:



Figura 46. Animación al presionar el icono de inicio de sesión manual.

Y así es como va llevando a cabo el proceso de animación, desplazando los componentes hacia la parte inferior de la pantalla y superponiendo la nueva parte de la interfaz:



Figura 47. Animación de deslizamiento.

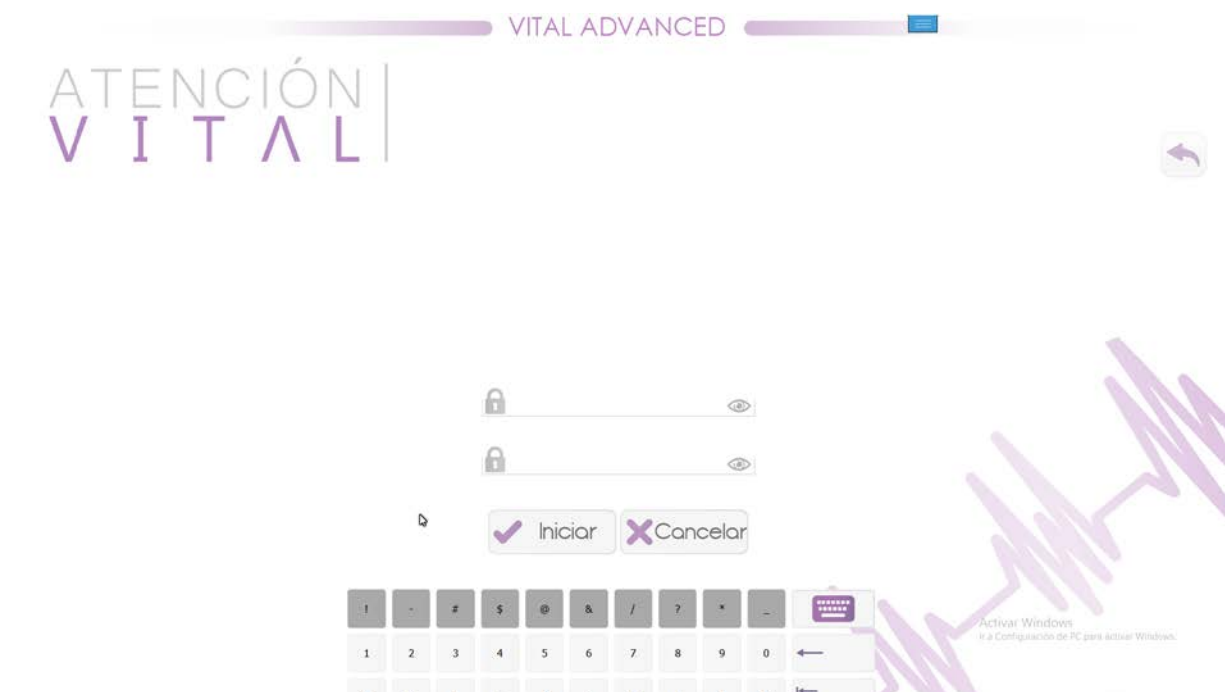


Figura 48. Animación para mostrar (subir) un nuevo panel.

Para realizar las animaciones de subir y bajar componentes se hizo uso de hilos para desplazar componentes:

```
public void subir(final JComponent elemento) {  
    SwingWorker worker = new SwingWorker() {  
        @Override  
        protected Object doInBackground() throws Exception {  
            for (int i = dim.height + 1; i >= 0; i -= 10) {  
                elemento.setLocation(0, i);  
                Thread.currentThread().sleep(1);  
            }  
            return null;  
        }  
    };  
    worker.execute();  
}
```

Y para que los botones tuvieran ese efecto de opacidad y cambio de imagen al ser presionados se utilizó un hilo y elementos de graficos 2D (Graphics 2D) para hacer el repintado de imágenes:

```
public void repintar(BufferedImage bufImg, BufferedImage original, float opacity,  
JButton boton) {  
    Graphics2D g2 = bufImg.createGraphics();
```



```

        AlphaComposite                newComposite                =
AlphaComposite.getInstance(AlphaComposite.SRC, opacity);

        g2.setComposite(newComposite);

        g2.drawImage(original, 0, 0, null);

        g2.dispose();

        icon = new ImageIcon(buflmg);

        boton.setIcon(icon);

    }

```

#### **3.3.4 Resultado del diseño del inicio de sesión manual**

Pantallas que contienen el teclado virtual y campos personalizados para el nombre de usuario y contraseña, también contiene animaciones para hacer la experiencia del usuario un poco más agradable.

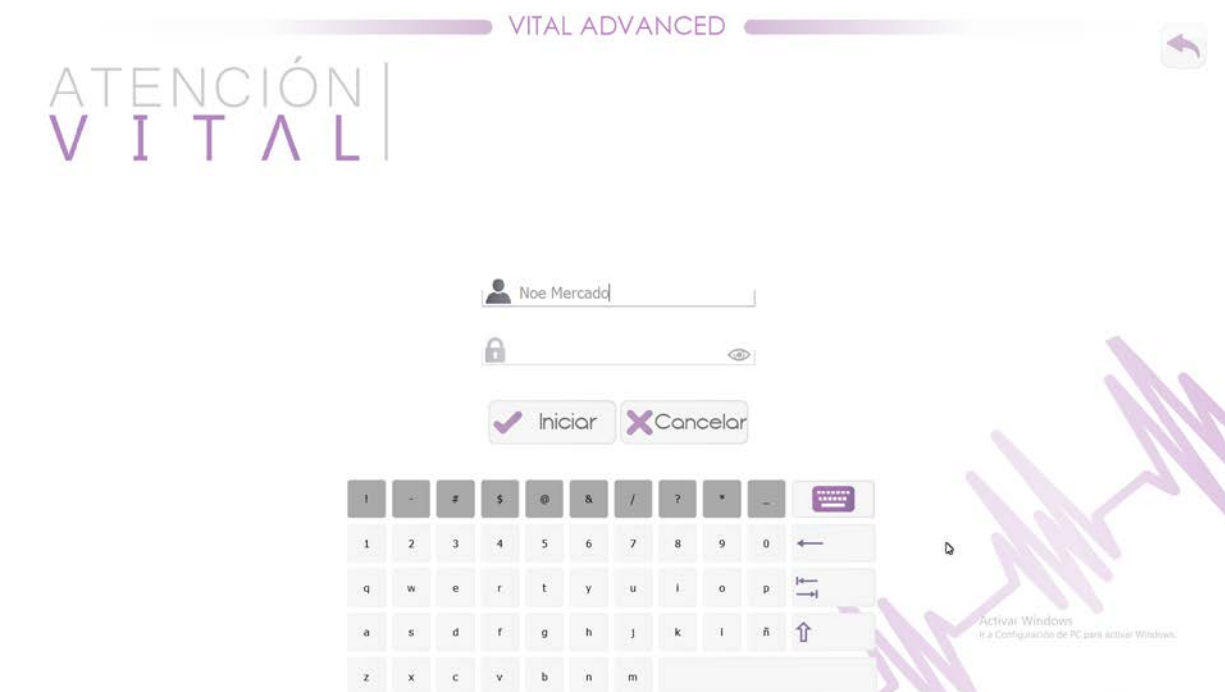


Figura 49. Vista del diseño de logueo manual.

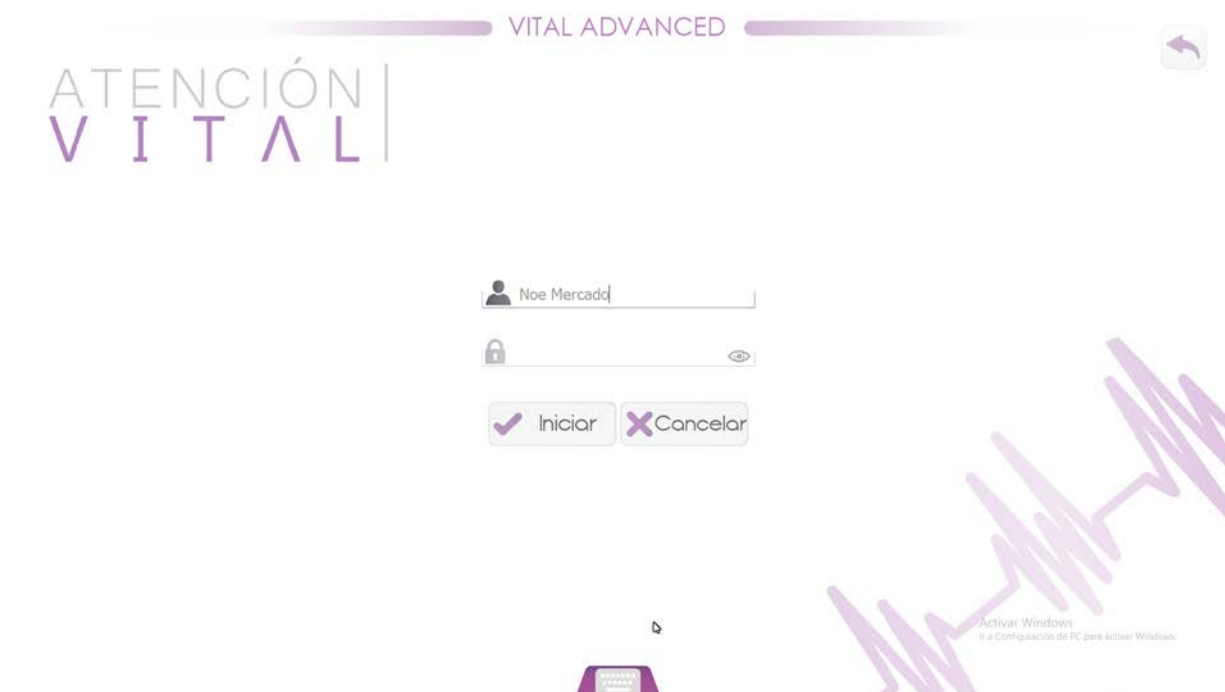


Figura 50. Teclado virtual oculto.

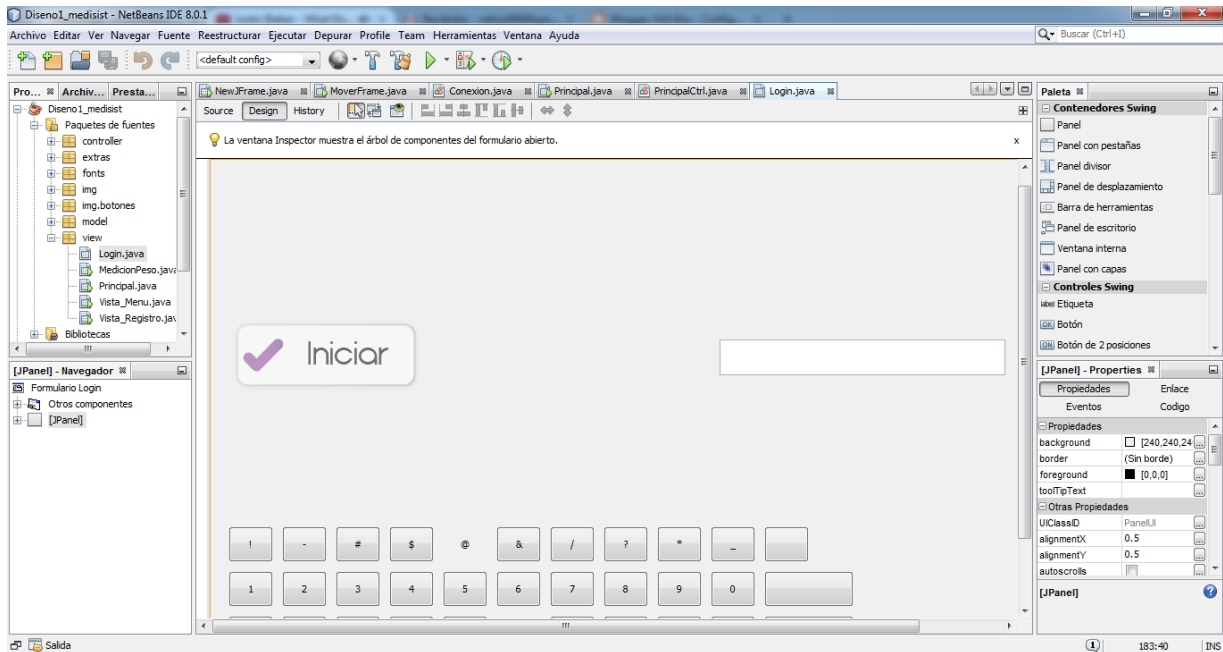


Figura 51. Diseño del logeo manual en NetBeans.

A continuación se muestra una parte del código implementado para el acomodo de los componentes:

```
public void acomodar(Dimension tamaño) {

    ancholconoUser = (ancholconoUser*porcentajePantalla)/100;

    ancholconoCandado = (ancholconoCandado*porcentajePantalla)/100;

    ancholconoOjo = (ancholconoOjo*porcentajePantalla)/100;

    login.panel_botones.setBounds(x,                                y+(porcentajeAltoCampo*4),
    porcentajeAnchoCampo, login.Btn_Accept.getSize().height+15);

}
```

Y se le dio un diseño al teclado y se cambió el tamaño de los botones de acuerdo a la resolución de la pantalla:

```
private void resizeImages() { //para el teclado

    int anchoOriginalPestana = 151;

    int altoOriginalPestana = 54;

    int nuevoAnchoTcl = (porcentajePantalla * anchoOriginalPestana) / 100;

    int nuevoAltoTcl = (int) ((double) ((porcentajePantalla * altoOriginalPestana) /
100));

    ImageIcon ic = new
ImageIcon(getClass().getResource("/img/botones/teclado.png"));

    ImageIcon imgTeclado = new
ImageIcon(ic.getImage().getScaledInstance(nuevoAnchoTcl, nuevoAltoTcl,
Image.SCALE_SMOOTH));

    int ancho = (login.panel_pie.getSize().width - nuevoAnchoTcl) / 2;

    int alto = login.panel_pie.getSize().height;

    login.Btn_showTcl.setIcon(imgTeclado);

    login.Btn_showTcl.setBounds(ancho, (alto - nuevoAltoTcl), nuevoAnchoTcl,
nuevoAltoTcl);

}
```

### 3.3.5 Resultado del diseño del menú principal

Capturas del menú principal que contiene “banners” para las funciones principales. En la siguiente imagen se muestra el diseño que se le dio en el IDE de desarrollo, el acomodo de componentes:

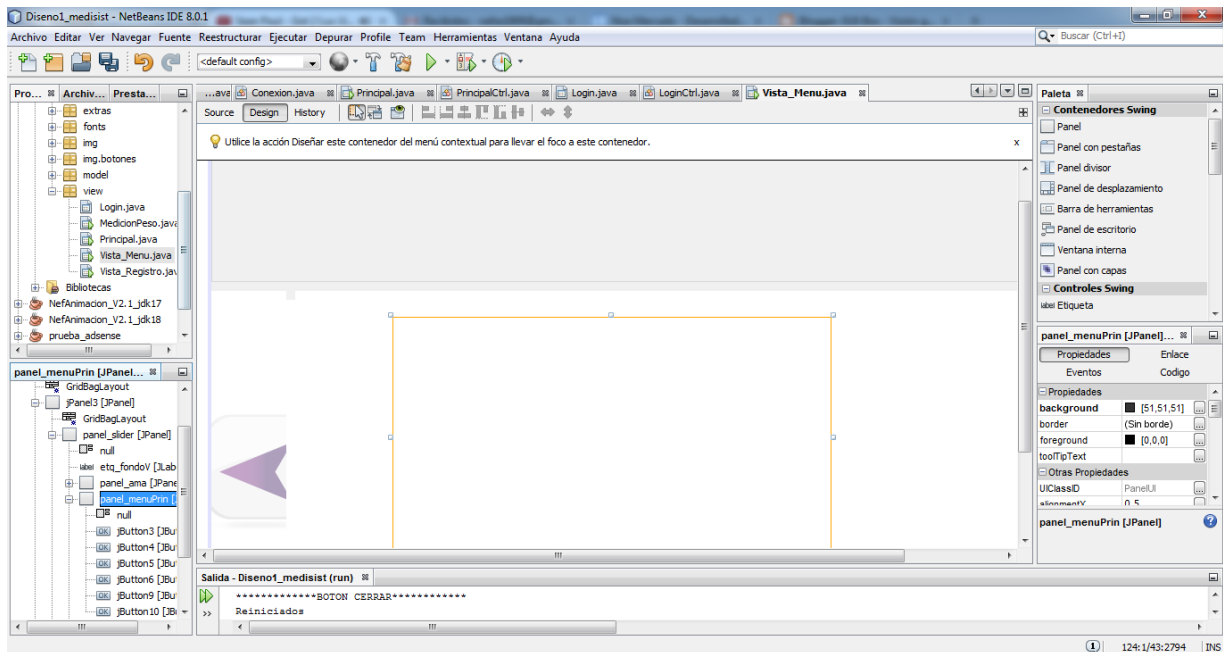


Figura 52. Diseño de menú principal en NetBeans.

En la ilustración 53 se muestra el uso del grid layout, para dar un acomodo más preciso a los componentes y evitar que se desfasen de la pantalla:

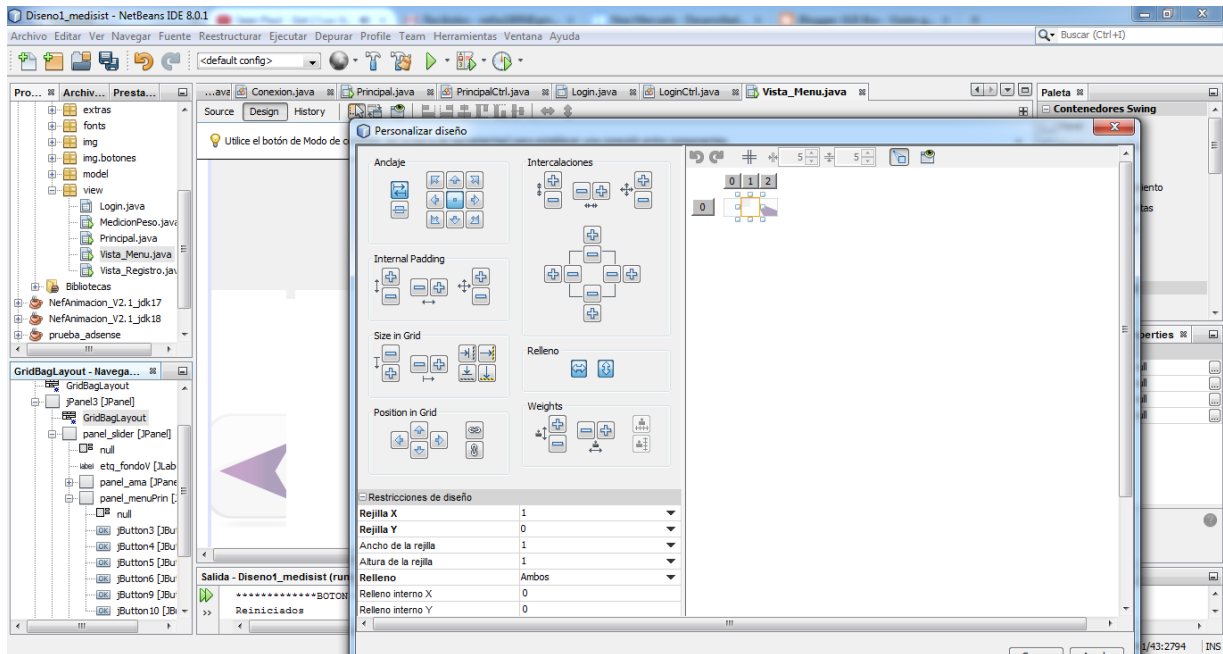


Figura 53. Aplicando el grid layout al menú principal.

En la siguiente imagen se puede observar el diseño del menú principal ya en función, donde se muestran los banners y las flechas de desplazamiento:

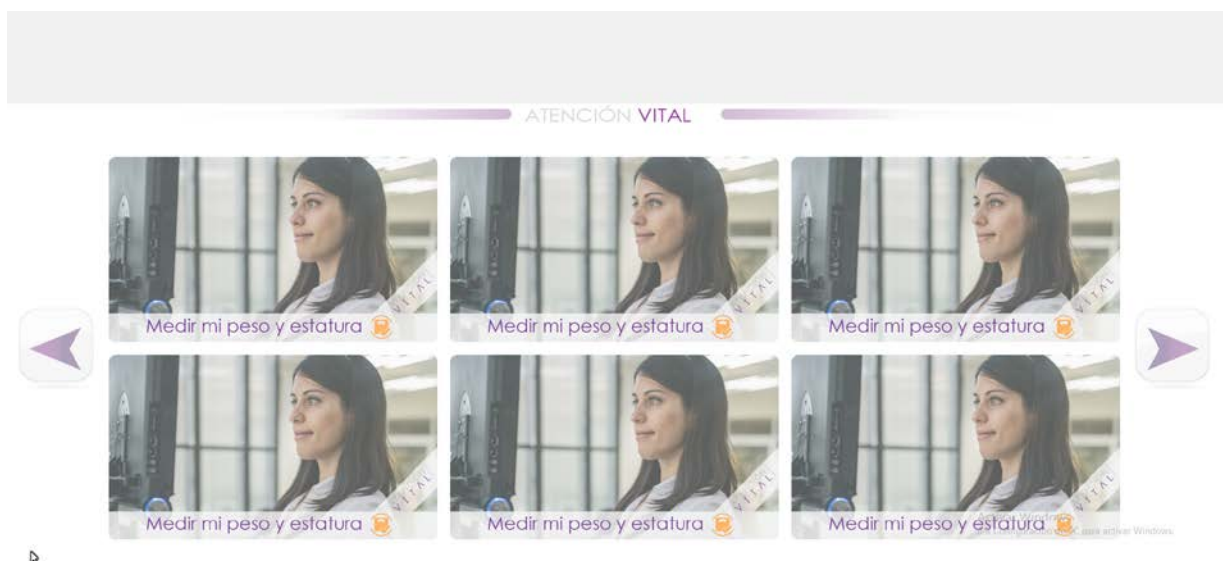


Figura 54. Vista previa del menú principal.

El código utilizado para acomodar seis banners en la pantalla principal es el siguiente:

```
private void acomodarBotones() {  
    int proporcionY = 1920 / 2;  
    double ancho2 = ancho / 2.3; //1620/2  
    double porcentaje = 1920 / ancho2;  
    double alto = 1080 / porcentaje;  
    int Centroancho = 480;  
    int Centroalto = 270;  
    int diferenciaAlto = (tamPanPri.height - (Centroalto * 3)) / 3;  
    int diferenciaAncho = (tamPanPri.width - (Centroancho * 3)) / 3;  
    jButton1.setBounds(diferenciaAncho, diferenciaAlto, Centroancho, Centroalto);  
}
```

### **3.3.6 Resultado de las animaciones para el menú principal**

El código implementado para animar esta vista, pensado para pantallas táctiles, que básicamente funcionaría como una galería de un dispositivo móvil (arrastrando los componentes con el dedo) en el evento mouse dragged:

```
private void mover2(MouseEvent evt) {
```

```

        paneles[0].setLocation((paneles[0].getLocation().x + (evt.getX() - x)),
paneles[0].getLocation().y);

        paneles[1].setLocation((paneles[1].getLocation().x + (evt.getX() - x)),
paneles[1].getLocation().y);

        paneles[2].setLocation((paneles[2].getLocation().x + (evt.getX() - x)),
paneles[2].getLocation().y);

    }

```

Esa sería la idea para implementar la interacción con una pantalla touch y el usuario. Ahora bien, también se pensó en dispositivos que aún no tienen pantalla táctil, para ello se implementaron botones de flechas para indicar las páginas anterior y siguiente. Por ejemplo para deslizar a la pantalla anterior, se utilizaría el siguiente código (dentro del evento actionPerformed):

```

panel_slider.setComponentZOrder(panel_menuPrin, 0);
Animacion.Animacion.mover_derecha(panel_menuPrin.getX(), 0, (long) retardo, salto,
panel_menuPrin);
Animacion.Animacion.mover_derecha(panel_ama.getX(), panel_slider.getWidth(),
(long) retardo, salto, panel_ama);

```

### **3.3.7 Resultado del diseño adaptivo**

Para obtener un diseño adaptable a distintas resoluciones de pantalla fue necesario implementar cálculos matemáticos para obtener porcentajes y redimensionar los



componentes. Para obtener las nuevas medidas de los componentes fue necesario hacer los siguientes pasos:

- Obtener la resolución de la pantalla (específicamente el ancho).
- Multiplicar el valor anterior \* 100 y dividirlo entre 1920 (1920 es para las resoluciones full HD, entonces se toma esa medida como base 100%).
- Multiplicar el ancho original de cualquier componente por el porcentaje obtenido en el paso anterior. Posteriormente dividir el resultado sobre 100.
- Hacer el paso anterior para obtener el porcentaje para lo alto del componente.
- Establecer los nuevos valores de ancho y alto.

Estas son algunas capturas en una resolución de 1920 \* 1080 (Full HD) y en 1366 \* 768.

Se puede ver una diferencia en los tamaños de los iconos, esto ocurre porque el tamaño de la pantalla cambia y es por eso que en la menor resolución se ven más grandes.

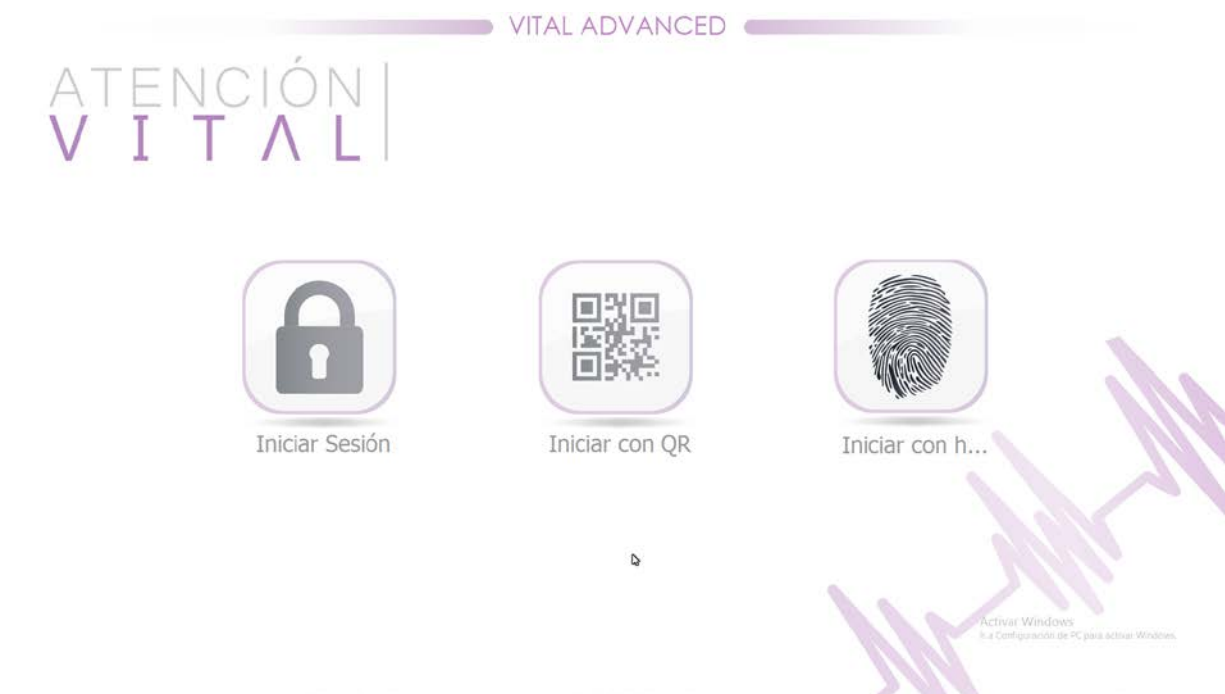


Figura 55. Elección de logueo a 1920 \* 1080.



Figura 56. Elección de logueo a 1366 \* 768.

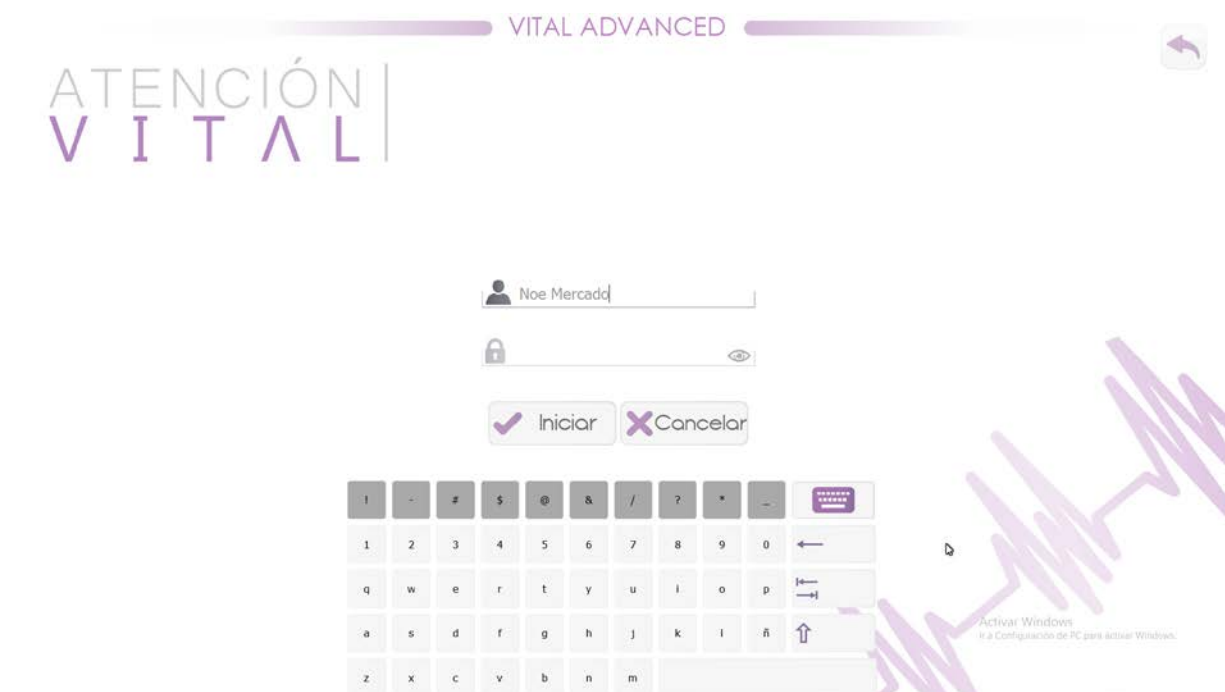


Figura 57. Logueo manual a 1920 \* 1080.

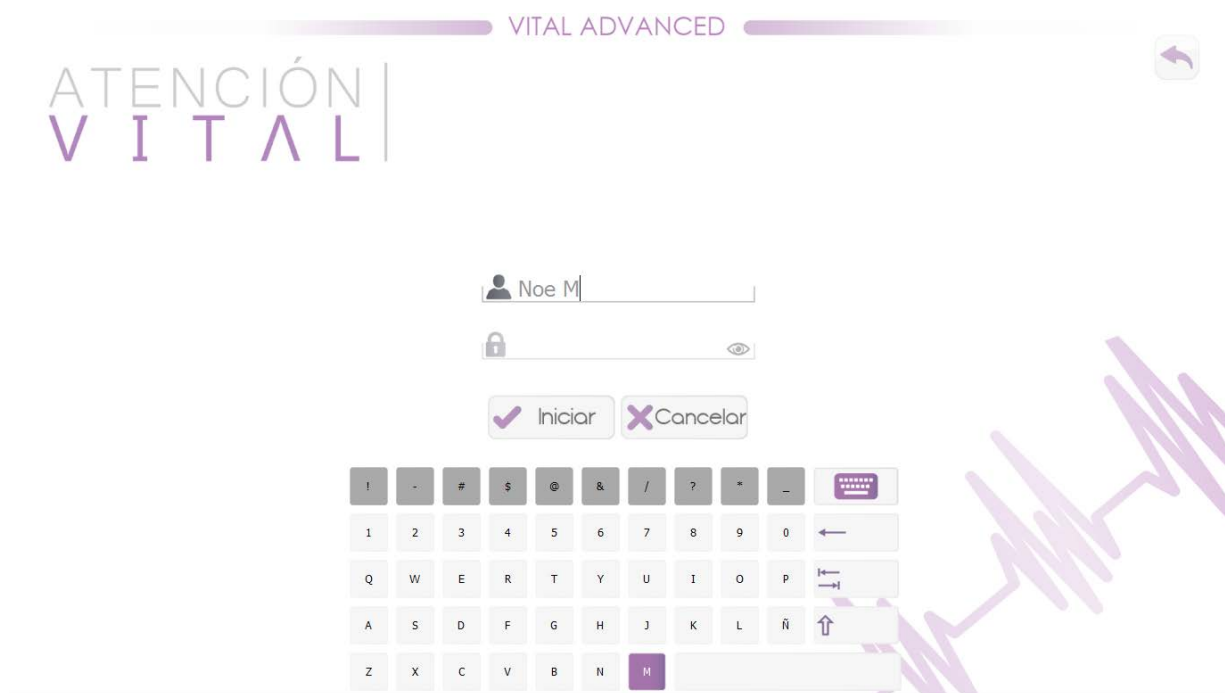


Figura 58. Logueo manual a 1366 \* 768.



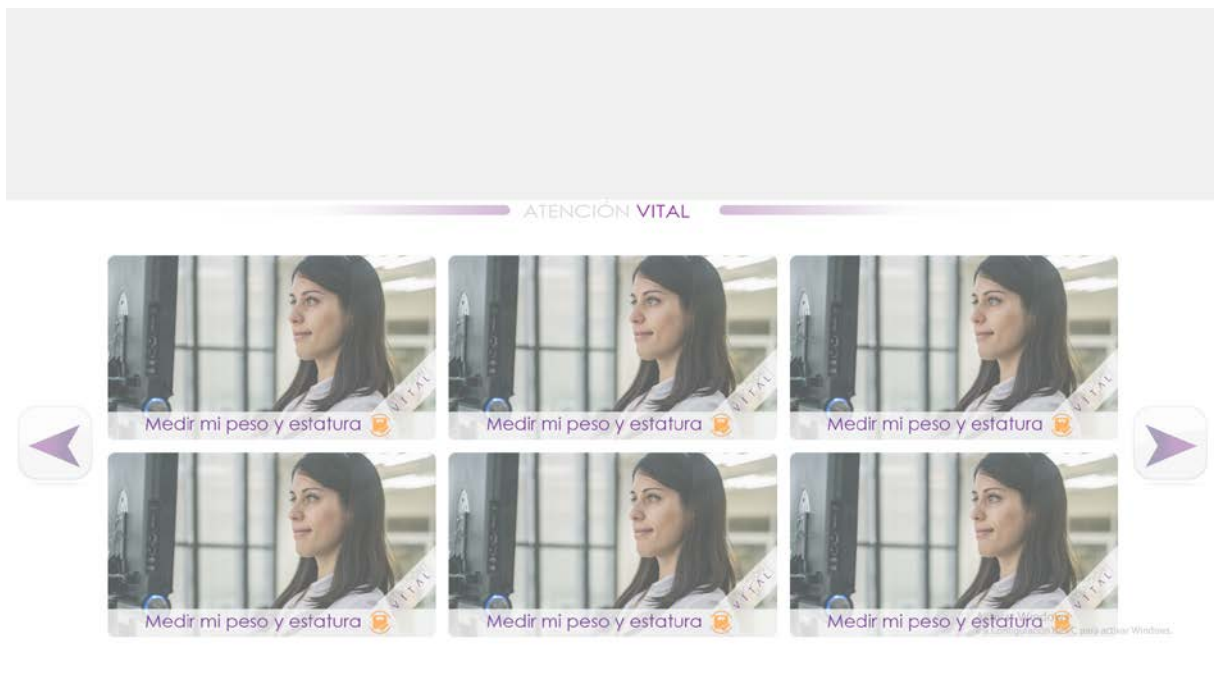


Figura 61. Menú principal a 1920 \* 1080.

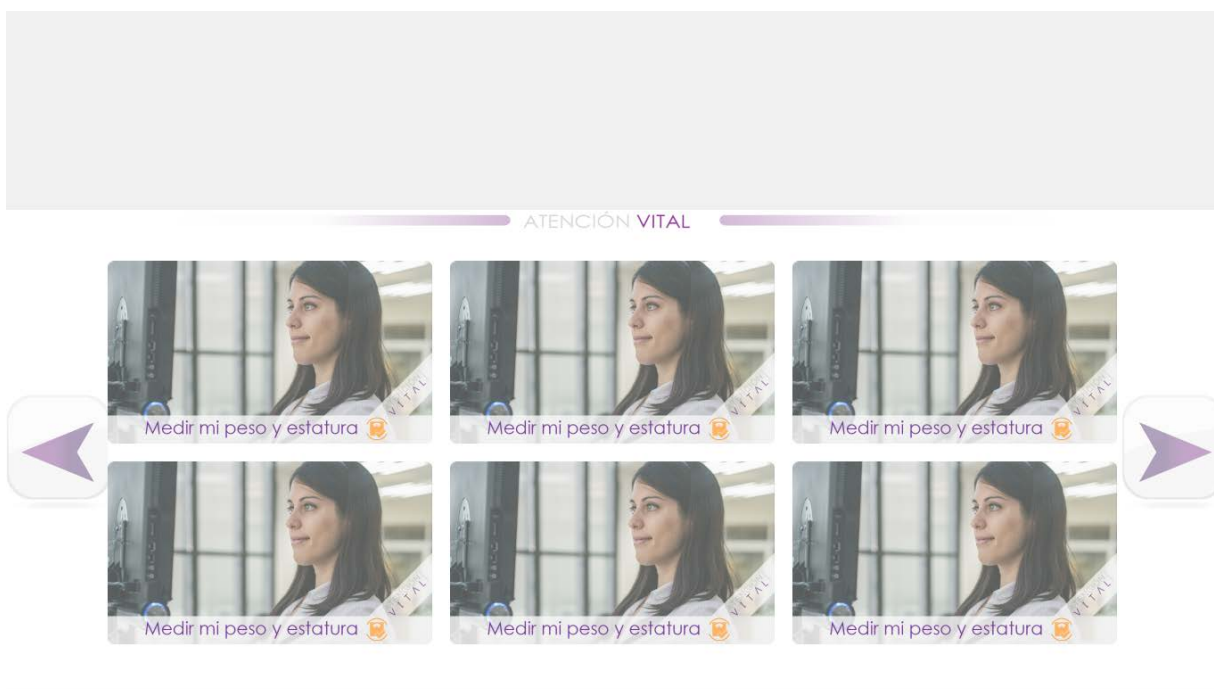


Figura 62. Menú principal a 1366 \* 768.

En algunas imágenes no se ve realmente la diferencia, solo se pueden observar pequeños detalles, como por ejemplo, en la última imagen (figura 62) se ve como un banner se superpone encima de la flecha de desplazamiento izquierda.

### 3.3.8 Resultado del uso de la arquitectura MVC

Para el inicio de la segunda versión del sistema se planeó el MVC, dejando como base estos paquetes:

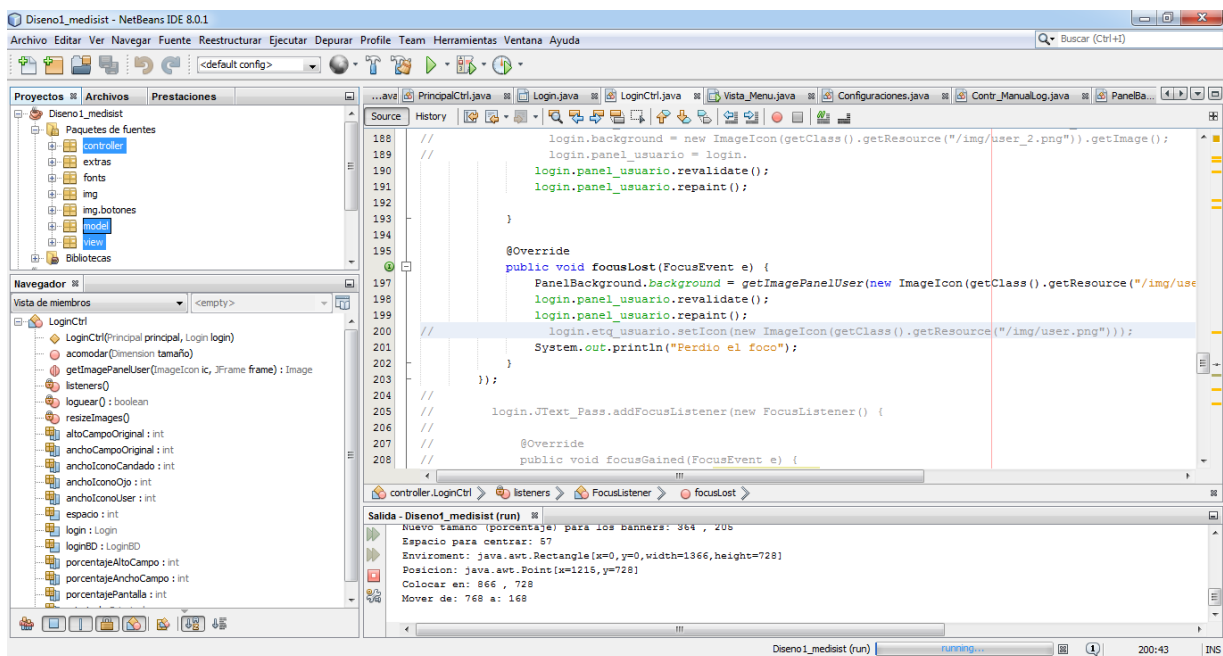


Figura 63. Estructura MVC del sistema "Atención Vital".

Los paquetes básicos son: view, model y controller. Después se implementaron más paquetes, en los cuales se almacenan tipografías, imágenes y clases extras con funcionalidades variadas.

## CONCLUSIONES

Como en todos los proyectos en los que se trabaja, ya sea para desarrollos nuevos o actualizaciones, siempre existen cosas buenas y malas que deben tomarse como experiencia, bien para implementar cosas nuevas aprendidas o para corregir detalles sucedidos y no querer que vuelvan a ocurrir.

En el periodo de residencia, hubo cosas buenas y malas con los proyectos en los que se trabajó. Una de las cosas buenas, es que se trabajó con proyectos reales un poco más complejos, en donde se implementaron cosas que no se habían visto en las prácticas escolares y esto aporta un plus al conocimiento práctico, así como una productividad mayor al querer investigar esos nuevos temas.

Por otra parte, también existen aspectos negativos que llevan a querer dejar las actividades y perder atención a las mismas. Esto puede llegar a ser una realidad cuando el ambiente laboral no es adecuado para el desarrollo del trabajador. Con el hecho de no tener comunicación entre líder y desarrollador puede hacer que las actividades se hagan mal o que se atrasen en entrega.

Dentro de las actividades realizadas en la empresa, una de las de mayor impacto fue el rediseño de las interfaces de usuario para la segunda versión de su sistema de “Atención Vital”. Se perdió muchísimo tiempo esperando una respuesta por parte de la diseñadora gráfica, que era la encargada de validar la propuesta de diseño

y agregar los cambios necesarios para mejorar la propuesta. Se llevó alrededor de un mes para poder trabajar en ello y obviamente la actividad de rediseño se vio detenida todo ese tiempo, impidiendo llevarla a cabo y tener que trabajar en otras actividades que no se tenían contempladas o que necesitaban de otros lenguajes que no podían ser cubiertos en el tiempo especificado para llevar a cabo la actividad.

Otro de los puntos que son de máxima prioridad es que, la empresa no cuenta con una documentación “formal” sobre los proyectos que tiene elaborados, simplemente las clases (las clases de lenguaje de programación) contienen comentarios para “facilitar” el entendimiento del código al usuario. Obviamente esto es servible para que se entiendan algunos puntos de la programación, pero no es aplicable para lograr entender el funcionamiento general del sistema, ni para saber cómo implementar nuevos requerimientos en un futuro, pues no existen (al menos nunca la hicieron presente los líderes de proyectos) documentos donde se expongan las historias de usuario, los diagramas de clases, el estándar de código, etc. Al no tener nada de esta documentación, los desarrolladores continúan haciendo los sistemas “a su gusto” y por más que se quiera tener un orden en los mismos, resultara muy difícil pues siempre entra y sale personal y siempre vuelve a ocurrir lo mismo.

Cabe mencionar que dentro de la empresa no se lleva a cabo un ciclo de vida específico para un proyecto, ni tampoco ninguna metodología de desarrollo. Esto hace que desarrollar un sistema se vuelva complicado, tedioso y con fallas.



A diferencia de lo aprendido en el ITSZaS, se puede decir que los proyectos desarrollados en la institución ofrecen una buena práctica para poder enfrentarse a la vida laboral. La documentación llevada a cabo y la implementación de alguna metodología ágil de desarrollo hicieron que realizar un sistema se volviera más práctico, entretenido, responsable y escalable. Estas son unas de las ventajas de que se trabajara con metodologías y documentación en los proyectos escolares, de esta manera se puede llegar a una empresa sin sentirse “perdido” en el proceso de desarrollo.

## RECOMENDACIONES

Existen un par de recomendaciones que serían buenas hacer notar tanto para empresas, proyectos, estudiantes e instituciones para que sean llevadas a cabo y poder ofrecer una mejor calidad a la hora de desarrollar un sistema.

Como primer punto, se debe tomar en cuenta que el trabajo en equipo es fundamental para este tipo de trabajos, que sin un equipo no se puede llegar a tener los objetivos deseados y que claramente son “el equipo” los que llegan al éxito al término del proyecto, es casi imposible pensar en individualidades y sobresalir.

Para la empresa Medisist, es posible que necesiten personal extra en diseño y menos personal en bases de datos, ya que para hacer interfaces, imagen corporativa, publicidad y más diseños, solo hay dos personas. Es impresionante que sean las dos personas que puedan tener más trabajo y que no puedan cubrir todas las actividades a tiempo. Por otro lado se encuentran los administradores de bases de datos, que son alrededor de 10 a 15 personas, que rara vez se encuentran todos trabajando y que pudiera ser un desperdicio de dinero y tiempo.

Otro punto de vista es para las empresas y los sistemas que desarrollan, es de suma importancia contar con personal de buena actitud, líderes que se preocupan por el equipo y que siempre están al pendiente de los proyectos y de sus trabajadores. Una recomendación sería que se llevara si o si algún tipo de metodología y

documentación de los proyectos así como contar con el personal adecuado para llegar a cumplir los objetivos del proyecto.

En cuanto a las instituciones académicas y los estudiantes, tal vez pudieran “actualizarse” para cubrir las áreas o lenguajes con mayor demanda en el mercado. Esto no quiere decir que las instituciones hagan u ofrezcan malos servicios, más bien es pensado en poder orientar a los estudiantes a adentrarse a nuevos retos, nuevos lenguajes, nuevas técnicas, para que en un futuro estén más preparados para enfrentar la vida laboral.

## FUENTES DE INFORMACIÓN

*Aprende a Programar - ¿Qué es Java?* (14 de 03 de 2016). Obtenido de Aprende a Programar - ¿Qué es Java?:

[http://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=368:ique-es-java-concepto-de-programacion-orientada-a-objetos-vs-programacion-estructurada-cu00603b&catid=68:curso-aprender-programacion-java-desde-cero&Itemid=188](http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=368:ique-es-java-concepto-de-programacion-orientada-a-objetos-vs-programacion-estructurada-cu00603b&catid=68:curso-aprender-programacion-java-desde-cero&Itemid=188)

*Desarrollo web - ¿Qué es MVC?* (15 de 03 de 2016). Obtenido de Desarrollo web

¿Qué es MVC?: <http://www.desarrolloweb.com/articulos/que-es-mvc.html>

*Web - Qué es el "responsive web design".* (16 de 03 de 2016). Obtenido de ID

Web - Qué es el "responsive web design":

[http://imasdeweb.com/index.php?pag=m\\_blog&gad=detalle\\_entrada&entry=12](http://imasdeweb.com/index.php?pag=m_blog&gad=detalle_entrada&entry=12)

*Imagen Digital UCP - ¿Qué es Photoshop?* (14 de 03 de 2016). Obtenido de Imagen

Digital UCP - ¿Qué es Photoshop?:

<https://imagendigitalucp.wordpress.com/tutoriales-photoshop/>

*JFreeChart.* (14 de 03 de 2016). Obtenido de JFreeChart:

<http://www.jfree.org/jfreechart/index.html>

*Kambrica - UI, UX, IxD: ¿Cuál es la diferencia?* (15 de 03 de 2016). Obtenido de

Kambrica - UI, UX, IxD: ¿Cuál es la diferencia?: <http://www.kambrica.com/blog/ui-ux-ixd-cual-es-la-diferencia/>

*NetBeans.* (10 de Marzo de 2016). Obtenido de NetBeans:

[https://netbeans.org/index\\_es.html](https://netbeans.org/index_es.html)

*Onnes sourceforge - Aplicaciones en capas.* (15 de 03 de 2016). Obtenido de Onnes sourceforge - Aplicaciones en capas:

<http://oness.sourceforge.net/proyecto/html/ch03s02.html>

*Oracle Java.* (14 de 03 de 2016). Obtenido de Oracle Java:

[https://www.java.com/es/download/faq/whatis\\_java.xml](https://www.java.com/es/download/faq/whatis_java.xml)

*TechTarget - ¿Qué es sql server?* (15 de 03 de 2016). Obtenido de TechTarget - ¿Qué es sql server?: <http://searchdatacenter.techtarget.com/es/definicion/SQL-Server>

*Wikipedia - Adobe Illustrator.* (14 de 03 de 2016). Obtenido de Wikipedia - Adobe Illustrator: [https://es.wikipedia.org/wiki/Adobe\\_Illustrator](https://es.wikipedia.org/wiki/Adobe_Illustrator)

## GLOSARIO

- **Interfaz de usuario:** hace referencia a “lo bonito”, “llamativo” de un sistema, es decir los botones, imágenes, tablas, etc. que componen una ventana, pantalla o una sección dentro del sistema.
- **UI:** significa User Interface (interfaz de usuario).
- **Componente:** se refiere a todo aquel elemento que se encuentra dentro de la interfaz de usuario, como pueden ser los botones, listas, tablas, textos, etc.
- **Layout:** Se le denomina así al gestor de contenido dentro de un panel, es decir que es quien acomoda los botones, tablas, imagines, etc.
- **Sorter:** conocido como clasificador de términos, permite ordenar o filtrar un conjunto de datos.
- **JOptinoPane:** es una interfaz de usuario que generalmente es utilizada para notificar errores, advertencias o progreso al usuario.
- **JComboBox:** no es nada más que una lista desplegable para que el usuario seleccione algún elemento dentro de esta lista.
- **CheckBox:** es una casilla de verificación, utilizadas comúnmente para responder a preguntas de si – no.
- **Prototipo:** se le llama así al “cascaron” de una aplicación, es decir, lleva la mínima parte de lógica, o que no funciona a detalle. Utilizado para mostrar avances de un proyecto a corto plazo.
- **Minimalista:** en diseño, se enfoca al “menos es más”, y que se puede hacer con el mínimo de detalles y que aun así será agradable a la vista.

- **Eventos:** se refiere a “acciones” que son resultados de la comunicación entre el usuario y el sistema, como presionar una tecla o dar clic con el ratón en un botón, etc.
- **Mantenible:** que el sistema se encuentre bien organizado, es decir que una modificación deberá ser de fácil implementación y que no requiere de un proceso complejo, como cambiar la resolución de algún componente y que no afecte a la lógica del sistema.
- **Vista:** hace referencia a las interfaces de usuario.
- **Panel:** es aquel componente gráfico, que contiene los botones, imágenes, tablas, etc. Que se ven en la pantalla.
- **Banner:** más conocido en el ámbito publicitario, hace referencia a un espacio con “publicidad” o imágenes que anuncian algo, dentro del software se usan banners para denotar alguna parte del sistema con mayor importancia.
- **Hilo:** se le conoce así a la plataforma virtual donde se ejecutan procesos del sistema, es decir, cuando un programa está en funcionamiento, hace uso de un “hilo” para ejecutar sus procesos, como mostrar una imagen, guardar alguna registro, reaccionar a eventos del usuario, etc.
- **IDE:** es la herramienta que ayuda al desarrollo de aplicaciones o sistemas, que hace el trabajo un poco menos laborioso pues aporta varias funcionalidades, como interfaz gráfica, correctores de sintaxis, plantillas, depuradores, etc.