Universidad de las Ciencias Informáticas

Facultad 5

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Propuesta de accesibilidad en los laboratorios virtuales para usuarios con discapacidades visuales moderadas.

Autor: Alexis Rey Mendoza

Tutor: Ing. Yasmany Alfonso Monteagudo

La Habana, junio de 2015

"Año 57 de la **Revolución**"

Declaración de Autoría

Declaro que soy el único autor de esta tesis y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____ _____

Autor: Alexis Rey Mendoza **Tutor:** Ing. Yasmany Alfonso Monteagudo

Pensamiento

"Si quieres vivir una vida feliz, átala a una meta no a una persona u objeto".

Albert Einstein

Dedicatoria

A mis padres Israel Rey y Bertha Mendoza por su apoyo incondicional, por haber creído siempre en mí, por sus consejos, por su paciencia infinita y sobre todo por el amor inmenso que siempre me han dado.

A mi bebito Arisnel que es la razón de mi vida y lo que más amo en este mundo.

A mi hermanito del alma Raddy, por ser un ejemplo para mí, por sus consejos y por jalarme las orejas cuando más lo necesitaba.

A mi beba malcriada Rossio, por su súper apoyo, amor, confianza, dedicación y por siempre estar conmigo en las buenas y en las malas. Si no fuera por ti nunca hubiera sacado las fuerzas para terminar una de las cosas más importantes de mi vida.

A mis amigos Roberto, Ronniel, Richel, Javier (El Flaco), Danny y Victor, por ayudarme a no perderme más de lo que estaba.

A mi tutor, por el apoyo y la dedicación.

Resumen

El trabajo investigativo que se presenta tiene como objetivo principal, proponer una solución que permita a las personas con debilidades visuales moderadas, la interacción con los laboratorios virtuales que se desarrollan en el Centro Entornos Interactivos 3D (VERTEX, *por sus siglas en español*). Para su cumplimiento se realizó un estudio del estado del arte sobre el tema a tratar y se analizaron las diferentes soluciones existentes para estos usuarios, que permitió elaborar y exponer la fundamentación teórica sobre la accesibilidad informática para débiles visuales, así como, las herramientas y tecnologías utilizadas. El proceso de desarrollo de la solución fue guiado por la metodología de desarrollo XP, utilizando C++ como lenguaje de programación. La construcción del sistema propuesto tiene un carácter iterativo, permitiendo la inserción de nuevas funcionalidades y la corrección de errores.

Como resultado, se desarrolló una configuración de accesibilidad adaptable a las diferentes deficiencias visuales que posean los usuarios con debilidades visuales moderadas que interactúen con los laboratorios virtuales del centro. También se demostró como con la utilización de las StyleSheet (Hojas de estilo, *en español*) se puede aplicar este tipo de configuración de forma más eficiente sin llegar a realizar cambios muy profundos en el estilo original de los laboratorios virtuales.

Palabras clave: *accesibilidad, debilidades visuales moderadas, laboratorios virtuales, menú.*

Índice

Índice de Tablas

Introducción

La visión, uno de los cinco sentidos del cuerpo humano, es el sentido más importante ya que el 80% de la información que recibimos entra a través de los ojos; no sólo las imágenes sino también todas las sensaciones que les acompañan. Por esto, es muy importante que el sistema visual sea eficaz, porque afecta el aprendizaje e incluso el comportamiento. En el caso de la lectura el 100% de la información que entra es puramente visual [1].

La visión no ocurre sólo en los ojos, los ojos son sólo la parte más externa de una compleja maquinaria. Ellos sólo se encargan de recibir la imagen de un objeto; es una labor muy importante, pero no sólo debido a ellos vemos. Las imágenes y toda la información que la rodea, siguen un proceso hasta llegar al cerebro y es allí donde procesamos, identificamos, entendemos, memorizamos, recordamos, aprendemos y respondemos a toda la información visual que recibimos [2].

Los términos de déficit visual, baja visión, visión residual, y otros, giran en torno a una reducción de la agudeza visual, debido a un proceso que afectó a la zona ocular o cerebral. De este modo, una persona con déficit visual es entendida como aquella que padece la existencia de una dificultad permanente en los ojos o en las vías de conducción del impulso visual. Esto conlleva a una disminución evidente en la capacidad visual, que constituye un obstáculo para su desarrollo, por lo que requiere una atención a sus necesidades especiales [3].

Una discapacidad visual que puede ser profunda, severa o moderada en dependencia del daño. Las discapacidades visuales son frecuentes, y aunque el número de personas ciegas es reducido, existe sin embargo un gran número con discapacidad visual moderada que necesita una educación con apoyos especializados [4].

En el campo de la informática se han desarrollado disímiles herramientas que permiten a los discapacitados visuales la interacción con los diferentes productos informáticos existentes en la actualidad, a esta facilidad que les brinda estas herramientas se les ha llamado accesibilidad informática

para débiles visuales, actualmente la mayoría de los sistemas cuentan con un herramienta que permita mejorar la accesibilidad informática de sus aplicaciones.

En la Universidad de las Ciencias Informáticas (UCI, *por sus siglas en español*) se encuentra el Centro Entornos Interactivos 3D (VERTEX, *por sus siglas en español*) actualmente cuenta con un proyecto de Laboratorios Virtuales con el fin de simular ambientes lo más cercano a la realidad que permiten que el usuario experimente artificialmente en gran cantidad de áreas del conocimiento sumergiéndolo en un mundo generado por ordenador, homologando su entorno a la realidad y logrando establecer operatividad e interacción en tiempo real entre la aplicación y el hombre.

Con el fin de llevar la experiencia de la interacción con los laboratorios virtuales a la mayor cantidad de usuario posibles se decidió proponer una solución para lograr una configuración de accesibilidad para los usuarios con discapacidad visual moderada, ya que estos representan la gran mayorías de estos discapacitados.

Estos laboratorios actualmente no cuentan con una herramienta que permita la modificación del tamaño y el estilo de la tipografía, la modificación del tamaño de los elementos de la aplicación o la configuración de un esquema de colores de alto contraste, tampoco cuentan con magnificadores de pantalla o lupas, características estas que debe tener un software para ser accesible.

Basado en lo anteriormente planteado se define como **problema a resolver**: ¿Cómo mejorar la interacción de los usuarios con discapacidades visuales moderadas, con la interfaz gráfica de los Laboratorios Virtuales?

El **objeto de estudio** de la presente investigación está dirigido a mejorar la accesibilidad a los laboratorios virtuales, para usuarios con discapacidades visuales moderadas. Mientras que el **campo de acción** está enmarcado en las diferentes características de accesibilidad que pueden ser aplicadas en los laboratorios virtuales del proyecto.

Determinándose como **objetivo general**: Proponer un menú de accesibilidad que permita la configuración de la interfaz gráfica de los laboratorios virtuales para los usuarios con deficiencias visuales moderadas.

Se precisan para dar cumplimiento al objetivo planteado de las siguientes **tareas de la investigación:**

1. Valorar las tendencias actuales a nivel mundial de los sistemas informáticos relacionados con la investigación.
2. Investigar las principales características que debe tener un software para ser accesible.
3. Proponer una solución para los procesos y funcionalidades del menú de accesibilidad.
4. Realizar pruebas de validación a la solución obtenida.

CAPÍTULO 1. Fundamentación Teórica

Introducción

El presente capítulo describe algunos conceptos relacionados con la investigación. Se realiza un análisis del estado del arte de algunos de los sistemas informáticos con funcionalidades similares a los procesos del menú Accesibilidad, dando a conocer sus características fundamentales y funcionalidades. Además, se realiza la fundamentación del uso de las tecnologías y herramientas utilizadas para el desarrollo de la solución.

1.1. Agudeza visual

Es la percepción de los detalles específicos de los objetos, personas, animales, es decir de todo lo que nos rodea, estos detalles percibidos son: color, forma, tamaño, peso y otras. Estas características son percibidas de cerca o de lejos según la funcionabilidad visual de cada persona [1].

Una agudeza de 1/10 significa que todo lo que una persona con visión normal ve a diez metros, es visible con la misma calidad para la persona con baja visión a un metro de distancia del objeto.

Una agudeza de 1/60 significa que una persona puede contar los dedos de una mano a una distancia de un metro.

Una agudeza de 1/300 significa que una persona puede ver el movimiento de una mano a una distancia de un metro.

La visión de distancia es medida de tres a seis metros. La visión de proximidad es medida a una distancia de 25 a 30 cm.

La agudeza visual se la manifiesta en cifras, con una connotación objetiva utilizada internacionalmente en forma de decimales.

Los objetivos que se persiguen al valorar la agudeza visual son:

- Información de base sobre el estado de la visión.
- Seguimiento y control de cómo evoluciona la enfermedad o alteración.

- Obtener criterios de selección para las ayudas ópticas, electrónicas o también de rehabilitación que se debe sugerir.
- Documentar la agudeza visual para fines legales y de rehabilitación.

También podemos definir agudeza visual como una medida de la capacidad del sistema visual para detectar, reconocer o resolver detalles espaciales, en un test de alto contraste y con un buen nivel de iluminación. Una persona con buena agudeza visual es capaz de apreciar detalles pequeños en una imagen. [3]

1.2. Discapacidad visual

Según la Organización Mundial de la Salud (OMS, *por sus siglas en español*), deficiencia o discapacidad es toda pérdida o anormalidad en una estructura a nivel fisiológico, anatómico o psicológico. En este sentido se puede hablar de discapacidad visual como la pérdida total o parcial del sentido de la vista.

Se trata de condición que afecta directamente la percepción de imágenes en forma total o parcial, por lo que se considera una discapacidad cuando las personas presentan una disminución en mayor o menor grado de la agudeza visual y una reducción significativa del campo visual.

Hay aproximadamente 285 millones de personas con discapacidad visual, de las cuales 39 millones tienen ceguera y 246 millones presentan baja visión, en el mundo de acuerdo con datos de la OMS [5].

Según la Clasificación Internacional de Enfermedades (CIE-10, *por sus siglas en español*), la función visual se subdivide en 4 niveles: [5]

a) **Ceguer**a: Carencia de visión o sólo percepción de luz. Imposibilidad de realizar tareas visuales.

b) **Discapacidad visual profunda:** Dificultad para realizar tareas visuales gruesas. Imposibilidad de hacer tareas que requieren visión de detalle.

c) **Discapacidad visual severa:** Posibilidad de realizar tareas visuales con inexactitudes, requiriendo adecuación de tiempo, ayudas y modificaciones.

d) **Discapacidad visual moderada:** Posibilidad de realizar tareas visuales con el empleo de ayudas especiales e iluminación adecuada similares a las que realizan las personas de visión normal.

La discapacidad visual moderada y la discapacidad visual severa se reagrupan comúnmente bajo el término baja visión; la baja visión y la ceguera representan conjuntamente el total de casos de discapacidad visual [5].

Según la Organización Mundial de la Salud incluye la clasificación de la agudeza visual y deterioro:

- La agudeza visual baja significa visión entre 20/70 y 20/400 con la mejor corrección posible, o una visual de campo de 20 grados o menos
- Ceguera se define como una agudeza visual peor de 20/400, con la mejor corrección posible, o un campo visual de 10 grados o menos
- Agudeza visual de 20/70 a 20/400 (inclusive) se considera moderado deterioro visual o baja visión.

1.3. Accesibilidad

Es el grado que algo posee para ser usado o accedido por las personas sin que existan barreras que se lo impidan producto al tipo de limitación física o técnica que estos presenten [1]. La eliminación de estos obstáculos ha permitido a los discapacitados actuar como cualquier otro ciudadano en la sociedad.

También se define accesibilidad como el grado en el que un objeto, lugar, servicio o producto está disponible para todas las personas, independientemente de sus capacidades técnicas, cognitivas o físicas [6].

Para poder de alguna forma eliminar estas barreras de accesibilidad se hace uso de ciertas facilidades llamadas ayudas técnicas, que logran que estas personas puedan realizar las mismas acciones que una sin discapacidad; el alfabeto Braille, la comunicación mediante señas y la silla de ruedas son ejemplo de ellas. En el contexto informático, estas ayudas técnicas incluyen las tipografías de alto contraste o gran tamaño, magnificadores de pantalla, lectores o revisores de pantalla, programas de reconocimiento de voz, teclados adaptados y otros dispositivos apuntadores de entrada de información [7].

El concepto de accesibilidad, por lo tanto, se utiliza para nombrar al grado o nivel en el que cualquier ser humano, más allá de su condición física o de sus facultades cognitivas, puede usar una cosa, disfrutar de un servicio o hacer uso de una infraestructura.

1.4. Características de un software accesible

Las cuatro características fundamentales que debe tener un software para que sea accesible a personas con discapacidades visuales moderadas son [10]:

- Permitir la modificación del tamaño y el estilo de la tipografía.
- Permitir la modificación del tamaño de los elementos de la aplicación.
- Permitir configurar un esquema de colores de alto contraste.
- Magnificadores de pantalla. Es un programa software que amplía un área de la pantalla haciendo la función de lupa.

1.5. Soluciones existentes

Actualmente existen múltiples sistemas que cuentan con herramientas que permiten mejorar la accesibilidad. Este tipo de herramientas son más usadas en la web, ya que la mayoría de la

información que se brinda actualmente, se hace mediante internet, de esta forma se ha convertido en la principal fuente de información a nivel mundial [9].

Las herramientas de accesibilidad también son un componente fundamental de cada sistema operativo basado en aplicaciones de escritorio, ya que facilita el uso de estos por los débiles visuales, por ejemplo, el sistema operativo Windows de la compañía Microsoft; este sistema resuelve la accesibilidad de una manera sencilla y de fácil uso para cualquier usuario con debilidades visuales, mediante los cinco requisitos de accesibilidad de certificación para Windows. Según estos requisitos, una aplicación accesible: [10]

- Admite las configuraciones de tamaño, color, fuente y entrada del panel de control. La barra de menú, la barra de título, los bordes y la barra de estado cambian todos automáticamente de tamaño cuando el usuario cambia la configuración del panel de control. En esta aplicación no es necesario hacer más cambios en los controles ni en el código.
- Admite el modo de alto contraste.
- Proporcionar acceso mediante teclado documentado a todas las funciones.
- Expone la ubicación del foco del teclado de forma visual y mediante programación.
- Evita ofrecer información importante únicamente por medio de sonido.

1.6. Metodologías de desarrollo de software

Imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Una metodología es el conjunto de fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de un sistema. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo [11]. Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y robustas o tradicionales.

- ✓ Las metodologías ágiles se caracterizan por hacer énfasis en la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde clientes desarrolladores y desarrolladores trabajan constantemente juntos, estableciéndose así una estrecha

comunicación y son capaces de responder a los cambios que puedan surgir a lo largo del proyecto. Se puede hacer mención dentro de las metodologías ágiles a: XP (*por sus siglas en inglés* Extreme Programming), Scrum y Crystal Methodologies.

- ✓ Las metodologías robustas o tradicionales están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo. Entre las metodologías robustas se encuentran: MSF (*por sus siglas en inglés* Microsoft Solution Framework), MÉTRICA 3 y RUP (*por sus siglas en inglés* Rational Unified Process).

1.6.1. Selección de la metodología

Metodología XP

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo, equipo pequeño y cuyo plazo de entrega es inminente. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto [12].

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Las características fundamentales de XP son:

- Pocos artefactos.
- Pocos Roles.
- Cliente es parte del equipo de desarrollo.
- Aplicable en grupos pequeños, menos de 10 integrantes y trabajando en el mismo sitio.

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas.
- Programación en parejas: Permite una mayor calidad del código escrito de esta manera es revisado y discutido mientras se escribe.
- El cliente forma parte del equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar la legibilidad y el mantenimiento del mismo.
- Propiedad del código compartida.
- Simplicidad en el código [2].

A continuación se realiza una comparación entre las metodologías XP (ágil) y la RUP (robusta) donde se puede apreciar las ventajas que se tiene al usar la metodología XP en este tipo de proyecto.

Comparación entre las metodologías XP y RUP

	XP	**RUP**
Tamaño de los equipos.	Proyectos y equipos pequeños.	Proyectos y equipos grandes.
Obtención de los requisitos.	Historias de usuarios.	Casos de usos.
Carga de trabajo.	Proceso ligero.	Proceso pesado.
Relación con el cliente.	Comunicación fluida con el cliente.	Se presentan los artefactos al final de una fase.
Tipo de desarrollo	Iterativo e incremental.	Iterativo e incremental.

Tabla 1. Diferencias entre las metodologías XP y RUP

La metodología XP comparada con otras metodologías ágiles es una de las mejores según una serie de aspectos tratados. XP es una de las más documentadas, con una amplia comunidad y certificada, además de ser la más utilizada por las empresas en proyectos de software [13].

Teniendo en cuenta que se desea agregar este menú de accesibilidad a un sistema ya terminado y las condiciones actuales de desarrollo requieren entregas de productos funcionales en cortos períodos de tiempo, se propone el uso de las metodologías ágiles para el desarrollo del menú.

Por las razones antes expuestas la metodología XP es la seleccionada para guiar el proceso de desarrollo de este menú.

1.7. Tecnologías y herramientas para el desarrollo del menú de accesibilidad

1.7.1. Qt SDK v2010.5

Es un Entorno Integrado de Desarrollo (IDE, *por sus siglas en inglés*) creado por Trolltech, multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea rápido sencillo.

Sus principales características son:

- Posee un avanzado editor de código C++.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL (Structured Query Language*, por sus siglas en inglés*), así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma

unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales [13] [14].

1.7.2. Lenguaje C++

C++ es un lenguaje imperativo orientado a objetos derivado del C. En realidad un subconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción [15].

1.7.3. Herramienta Case

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación [16]. Entre sus principales características podemos encontrar:

- Soporta aplicaciones Web.
- Es un producto de calidad.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Se estará haciendo uso de Visual Paradigm for UML (Enterprise Edition) en su versión 8.0.

1.7.4. Utilización de StyleSheet

StyleSheet (Hoja de estilo, *en español*) permite tratar la aplicación como un documento HTML, facilitando el uso de las propiedades de cada componente visual de la aplicación. De esta forma se pueden alterar las características de un mismo tipo de componente a la vez [17].

Las ventajas de utilizar StyleSheet son:

- Control centralizado de la presentación visual con lo que se agiliza de forma considerable la actualización de la misma.
- Permiten a los usuarios especificar su propia hoja de estilo local que será aplicada, con lo que aumenta considerablemente la accesibilidad.
- Se puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.

StyleSheet al igual que CSS, proporciona tres caminos diferentes para aplicar las reglas de estilo a una página Web o a una aplicación:

Hoja de estilo externa: es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código. Esta es la manera de programar más potente, porque separa completamente las reglas de formateo para la aplicación de la estructura básica de la aplicación.

Hoja de estilo interna: es una hoja de estilo que está incrustada dentro de la aplicación. En general, la única vez que se usa una hoja de estilo interna, es cuando se quiere proporcionar alguna característica especifica en un simple fichero.

Estilo en línea: que es un método para insertar el lenguaje de estilo, directamente dentro. Esta manera de proceder no es totalmente adecuada. El incrustar la descripción del formateo a nivel de código se convierte en una tarea larga, tediosa y poco elegante de resolver el problema de la programación. Este modo de trabajo se podría usar de manera ocasional si se pretende aplicar un formateo con prisa, al vuelo [17].

Conclusiones del capítulo

En este capítulo se analizaron diferentes aspectos conceptuales referentes a la discapacidad visual y a la accesibilidad, los diferentes tipos de discapacidad visual existentes, enfocando nuestra investigación en la discapacidad visual moderada. Se describen las principales características que debe tener un software para ser accesible; las tecnologías y las herramientas empleadas en el proceso de desarrollo del menú, seleccionando la Metodología XP para el desarrollo que guía la construcción de la solución final.

CAPÍTULO 2. Características del sistema

Introducción

Durante el desarrollo de este capítulo se realiza la descripción de la propuesta de solución de este trabajo, para lo que se desarrollan las actividades de análisis y especificación de requisitos, se define la arquitectura que se empleará en el desarrollo del menú y se describen las necesidades del sistema mediante historias de usuarios.

2.1. Propuesta de solución

Como propuesta de solución se presenta la creación del menú de accesibilidad que pueda integrarse a los laboratorios virtuales que se desarrollan en el Centro Entornos Interactivos 3D (VERTEX, *por sus siglas en español*). Este menú propone un conjunto de funcionalidades que hacen accesible la presentación de la información a usuarios con discapacidades visuales moderadas, ya que permitirá que el usuario configure la aplicación de manera que se ajuste a su nivel de deficiencia visual.

2.2. Exploración

La metodología XP plantea, que en esta fase los clientes, grandes rasgos, definen a las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo, el equipo de desarrollo se familiariza con las prácticas, tecnologías y herramientas que se utilizarán en el proyecto. Esta fase se recomienda que sea de 2 ó 3 semanas hasta pocos meses, todo dependiendo del tamaño y complejidad de la solución que se va a desarrollar, así como del conocimiento que tengan los programadores de la tecnología que se utilizará [18].

2.2.1. Historias de usuario

La historia de usuario se utiliza para describir cada una de las características que va a tener el sistema, son las acciones que se deben implementar. Cada una es implementada en una

determinada iteración del proyecto, de las cuales también se hace referencia a los puntos estimados que son los que definen el tiempo de duración de las historias de usuario a implementar.

Se definió la siguiente estructura para obtener los datos de las historias de usuarios determinadas:

Historia de Usuario	
Nombre:	
Código:	**Referencia:**
Iteración:	
Prioridad:	**Puntos Estimados:**
Descripción:	
Observaciones:	

Tabla 2. Estructura de una historia de usuario

- **Nombre**: nombre que identifica la historia de usuario.
- **Código**: código que identifica la historia de usuario sería HU- Número Consecutivo.
- **Referencia**: se hace referencia a los requisitos que tiene en cuenta la historia de usuario.
- **Iteración Asignada**: que iteración se desarrollará. (Según su importancia).
- **Prioridad**: prioridad puede ser Alta, Media o Baja. (Según Cliente).
- **Puntos Estimados**: tiempo en semanas que se le asignará. (Estimado).
- **Descripción**: breve descripción del proceso que define la historia de usuario.
- **Observaciones**: alguna acotación importante de señalar acerca de la historia de usuario.

Historia de Usuario	
Nombre: Configurar fuente	
Código: HU 1	**Referencia:** RF 1
Iteración: 1	
Prioridad: Alta	**Puntos Estimados:** 2

Descripción: La aplicación debe permitir al usuario seleccionar el tipo, tamaño y color de la fuente que usara en toda la aplicación.
Observaciones: El usuario debe ir probando las diferentes combinaciones de que se hacen entre el tipo, tamaño y color de la fuente para lograr visualizar el contenido de la ventana en correspondencia con la debilidad visual que posea.

HU 1. Configurar fuente

Historia de Usuario	
Nombre: Configurar contraste.	
Código: HU 2	**Referencia:** RF 2
Iteración: 2	
Prioridad: Alta	**Puntos Estimados:** 1

Descripción: La aplicación debe permitir al usuario seleccionar de contraste en que se presenta toda la aplicación.
Observaciones: El usuario debe ir probando las diferentes combinaciones de que se hacen entre el color del texto y el nivel de contraste de toda la aplicación.
Prototipo de la interfaz: 

HU 2. Configurar contraste

Historia de Usuario	
Nombre: Configurar tamaño de los elementos	
Código: HU 3	**Referencia:** RF 3
Iteración: 3	
Prioridad: Alta	**Puntos Estimados:** 1

Descripción: La aplicación debe permitir al usuario seleccionar el tamaño de los elementos presentados en la aplicación.
Observaciones: El usuario selecciona el tamaño que desea que tomen los elementos que se encuentran en la aplicación, estos elementos pueden ser principalmente los botones.

HU 3. Configurar tamaño de los elementos

Historia de Usuario	
Nombre: Magnificador de pantalla o lupa.	
Código: HU 4	**Referencia:** RF 3
Iteración: 3	
Prioridad: Alta	**Puntos Estimados:** 1
Descripción: La aplicación debe permitir al usuario seleccionar la herramienta lupa.	
Observaciones: El usuario selecciona el magnificador de pantalla o lupa que multiplica por dos el tamaño de los elementos que se encuentren en el mismo punto que se encuentra el puntero.	
Prototipo de la interfaz:	

HU 4. Magnificador de pantalla o lupa

2.2.2. Diseño de Casos de Pruebas

Uno de los pilares de la metodología XP es el proceso de pruebas que anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones [19].

Pruebas de aceptación

Las pruebas de aceptación están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente final. Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. La s pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación [20].

El diseño de las pruebas antes mencionadas, así como los resultados obtenidos luego de su aplicación al sistema se encuentran detallados en el capítulo 3.

2.3. Planificación y Entrega

La planificación es la fase donde el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas [19].

En esta fase el cliente establece la prioridad de cada HU y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debe obtenerse en no más de tres meses.

2.3.1. Plan de Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y a continuación, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el

cliente. Esta fase dura unos pocos días. Se expresa utilizando como unidad medida el punto, el cual se considera como 2 días de trabajo en programación real. [20]

Estimación de esfuerzo por Historias de Usuario:

No	Historia de Usuario	Puntos de Estimación
1	Configurar fuente	3
2	Configurar contraste	3
3	Configurar tamaño de los elementos	4
4	Magnificador de pantalla o lupa.	6

Tabla 3. Estimación de esfuerzo por Historia de Usuario

2.3.2. Iteraciones

En esta fase se definieron dos iteraciones para la realización del sistema:

Iteración 1

Las historias de usuario que presentan mayor prioridad serán implementadas en esta primera iteración, dando al sistema las primeras funcionalidades, configurar fuente y configurar contraste.

Iteración 2

En la segunda iteración se implementará la historia de usuario configurar tamaño de los elementos.

Iteración 3

En la tercera iteración se implementará la historia de usuario magnificador de pantalla o lupa.

Plan de duración de las iteraciones

Cantidad de	Orden de las Historia de Usuario a implementar	Duración total de las

Iteraciones		iteraciones
1	Configurar fuente	1 semanas
1	Configurar contraste	2 semanas
2	Configurar tamaño de los elementos	1 semana
3	Magnificador de pantalla o lupa	2 semanas

Tabla 4. Plan de duración de las iteraciones

Plan de Entregas

Historia de Usuario	Final 1ra iteración	Final 2ra iteración	Final 3ra iteración
Configurar fuente	10/2/2015		
Configurar contraste	19/2/2015		
Configurar tamaño de los elementos		4/3/2015	
Magnificador de pantalla o lupa			20/3/2015

Tabla 5. Plan de entregas

2.3.3. Plan de Tareas

Se divide cada historia de usuario en varias tareas que serán desarrolladas para dar cumplimiento a la misma, de esta forma se facilita el trabajo de implementación de los programadores.

Historias de Usuario	Tareas
Configurar fuente	1. Definir y crear los componentes visuales necesarios para configurar la fuente toda la aplicación. 2. Definir los estilos de letras que serán

	seleccionados.
Configurar contraste	1. Definir y crear los componentes visuales necesarios para configurar el contraste de la aplicación de forma sencilla. 2. Definir las combinaciones de colores que son más efectivas para la lectura.
Configurar tamaño de los elementos	1 Definir y crear los componentes visuales necesarios para configurar el tamaño de los elementos de la aplicación.
Magnificador de pantalla o lupa	1. Definir el tamaño del cuadro del magnificador de pantalla o lupa en el que será mostrado la parte de la aplicación en el que se encuentre el cursor en ese momento.

Tabla 6. Plan de tareas

2.4. Diseño del Sistema

Para el diseño de aplicaciones informáticas la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando el lenguaje UML. En su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

Todo el menú contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos, la arquitectura de las aplicaciones dependen de cómo está distribuido este código.

La arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software. En este modelo, una aplicación se convierte en un conjunto

de servicios de usuario, negocios y datos que satisface las necesidades de los procesos de negocios. Cada una de estas capas se pueden entender como un todo, sin considerar las otras y eliminando cualquier dependencia entre ellas.

2.4.1. Capa de presentación

Esta capa es la que ve el usuario, también se la denomina "capa de usuario". Presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso, realiza un filtrado previo para comprobar que no hay errores de formato. Esta capa se comunica únicamente con la capa de negocio [21]. En el caso de la aplicacion sera la ventana de configuracion de accesibilidad, donde se les presenta a los usuarios las diferentes configuraciones que pueden seleccionar para lograr un nivel de accesibilidad acorde a su discapacidad visual.

2.4.2. Capa de negocio

Es en esta capa donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio e incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. La capa de presentación se comunica con esta, para acceder a las solicitudes y presentar los resultados, y a su vez, la capa de negocio accede a la capa de acceso a datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él, en el caso de que se utilice un gestor de base de datos [21].

2.4.3. Capa de datos

Es donde se almacena de forma persistente toda la información necesaria para facilitar satisfacer las peticiones de los usuarios [21]. En este caso la información se almacena directamente en la hoja de estilos de la propia aplicación, allí es donde estará toda la información que se va a administrar en esta capa.

2.4.4. Patrones de Diseño

La arquitectura es necesaria para comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema. Para definirla es necesario seleccionar y combinar patrones.

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño; facilitan la reusabilidad, extensibilidad y mantenimiento. Entre los elementos de un patrón se encuentran: Nombre (describe el problema de diseño), Problema (describe cuándo aplicar el patrón) y Solución (describe los elementos que componen el diseño, sus relaciones, responsabilidades y colaboración). Una forma de clasificar los patrones de diseño es según su propósito: [20]

- ✓ De creación: conciernen al proceso de creación de objetos.
- ✓ De estructura: tratan la composición de clases y/o objetos.
- ✓ De comportamiento: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

Esta clasificación se asocia con el nombre bajo el cual se agrupan: GoF, denominados así pues su determinación se atribuye a los que comúnmente se conoce como Gang of Four (GoF, en español "pandilla de los cuatro") formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos [21].

A continuación se presenta los patrones de diseño GoF que fueron seleccionados para utilizar como parte de la solución:

Singleton: Su propósito es asegurar que una clase tenga una única instancia y proporcionar un punto de acceso global a dicha instancia en este caso esa función la realiza la clase “MainController”. De esta forma se logra crear una variable global, que hace el objeto accesible y para que sea posible instanciarla sólo una vez. En consecuencia hay un acceso controlado a dicha única instancia, y se gestionan de manera centralizada los recursos utilizados.

Decorator: Añade capacidad de desplazamiento a una vista. Para este caso facilita los cambios dinámicos en la apariencia de la aplicación cuando se aplica la configuración de alto contraste para mejorar la lectura de los discapacitados visuales.

Observer: Se utiliza para definir una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. En esta solución permite mantener consistencia entre objetos relacionados sin aumentar el acoplamiento entre clases.

2.4.5. Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Para poder diseñar el sistema como un equipo se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño.

Clase	
Descripción:	
Responsabilidades	
Descripción (Método):	**Colaborador:**

Tabla 7. Prototipo de tarjeta CRC

Clase: es cualquier evento que realizara una serie de acciones que se le especificara para obtener, modificar, eliminar, adicionar datos y otras según los resultados que se quieran obtener.

Responsabilidades: de una clase es la lógica del negocio asociada y las que realizan sus atributos y métodos.

Colaboradores: son el resto de las clases con las que trabaja para llevar a cabo sus responsabilidades.

El sistema contiene una serie de componentes lógicos, a continuación se presentan las respectivas tarjetas CRC pertenecientes a las clases de los mismos:

MainController	
Descripción: Clase encargada de controlar los eventos de la barra de menú.	
Responsabilidades	
Descripción (Método):	**Colaborador:**
void showConfigurationDialog(): Muestra el dialogo de configuración de temas del laboratorio virtual.	

Tarjeta CRC. 1 MainController

ConfigurationDialog	
Descripción: Vista para la configuración de temas del laboratorio virtual.	
Responsabilidades	
Descripción (Método):	**Colaborador:**
void on_ButtonOk_Clicked(): Método encargado de aplicar la configuración realizada por el usuario.	

void on_ButtonCancel_Clicked(): Método encargado de ocultar la vista de configuración.	

Tarjeta CRC. 2 ConfigurationDialog

ThemeLoader	
Descripción: Clase encargada de procesar el tema de la aplicación para aplicar el tamaño de fuente escogido por el usuario.	
Responsabilidades	
Descripción (Método):	**Colaborador:**
void applyTheme(QString ,int): Método encargado de aplicar un tema determinado a la aplicación.	

Tarjeta CRC. 3 ThemeLoader

ConfigurationLoader	
Descripción: Se encarga de leer y persistir el tema y el tamaño de letra escogido por el usuario desde ficheros (.ini) para ser utilizado por defecto cada vez que la aplicación sea iniciada.	
Responsabilidades	
Descripción (Método):	**Colaborador:**
void saveConfiguration(QString,int): Guarda en el fichero de configuración el nombre del tema y el	

tamaño de letra por defecto de la aplicación.	
void loadConfiguration(QString &, int &): Carga desde el fichero de configuración el nombre del tema y el tamaño de letra por defecto de la aplicación.	

Tarjeta CRC. 4 ConfigurationLoader

2.5. Requisitos funcionales

Los requisitos funcionales escriben lo que el sistema o el software deben hacer. La funcionalidad es la capacidad útil proporcionada por uno o más componentes de un sistema. En algunos casos, los requisitos funcionales también pueden declarar explícitamente lo que el sistema no debe hacer. La aplicación debe permitir alterar las características de cada uno de los componentes que aparecen en la ventana, así como debe permitir modificar el contraste, colores y tamaño de fuente de todo el contenido de la aplicación, esto se debe lograr mediante un menú que contenga controles para cada una de los tipos de modificaciones.

El sistema debe permitir:

RF 1 Configurar fuente.

RF 1.1 Seleccionar el tipo de fuente.

RF 1.2 Seleccionar el tamaño de la fuente.

RF 1.3 Seleccionar el color de la fuente.

RF 2 Configurar el contraste de la ventana.

RF 2.1 Seleccionar el nivel de contraste de la ventana.

RF 2.1 Seleccionar el color de fondo.

RF 3 Configurar el tamaño de la ventana.

RF 3.1 Seleccionar el tamaño de la ventana.

FR 4 Magnificador de pantalla o lupa.

RF 4.1 Seleccionar el magnificador de pantalla o lupa.

2.6. Requisitos no funcionales

2.6.1. Usabilidad

El menú debe permitir el fácil acceso a las diferentes funcionalidades del sistema. El lenguaje utilizado en el desarrollo del menú debe adaptarse a los términos utilizados por los clientes de los laboratorios virtuales.

2.6.2. Hardware

Para el desarrollo: PC Intel Pentium 4 o superior, CPU 2.0 GHZ o superior, 1 Gb RAM o más.

Conclusiones del capítulo

En este capítulo describieron los requisitos funcionales y no funcionales del sistema, se describen las tecnologías y las herramientas a empleadas en el proceso de desarrollo del menú, la metodología que guía la construcción de la solución y la arquitectura a emplear en el sistema, la cual posibilita que el menú sea fácilmente corregible a integrable con otros sistemas.

CAPÍTULO 3. Implementación y prueba

Introducción

En el presente capítulo se lleva a cabo la descripción de las tareas de implementación generadas por cada historia de usuario realizadas en las iteraciones propuestas, obteniendo en cada una de ellas una versión del producto, así como las pruebas de aceptación efectuadas sobre el menú.

3.1. Implementación

3.1.1. Historias de Usuario implementadas en la primera iteración

En esta iteración se desarrollan las historias de usuario que presentan mayor prioridad, dando al sistema las primeras funcionalidades, estas son: configurar fuente y configurar contraste.

Historias de Usuarios	Estimación	Real
Configurar fuente	**2**	**2**
Configurar contraste	**2**	**2**

Tareas implementadas de la historia de usuario “Configurar fuente”

Número tarea: 1	**Historia de Usuario 1**
Nombre tarea: Definición y desarrollo de la interfaz configurar fuente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/2/2015	**Fecha fin:** 10/2/2015

Programador responsable: Alexis Rey Mendoza
Descripción: Se realiza el diseño de componentes que permitirán modificar los valores del tamaño de la fuente en toda la aplicación. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.

Número tarea: 2	**Historia de Usuario 1**
Nombre tarea: Validación de la entrada de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 10/2/2015	**Fecha fin:** 10/2/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se realiza la validación de datos, asegurando así el correcto funcionamiento de la aplicación. Para ello se tiene presente los tipos de datos soportados y los posibles a entrar por los usuarios, así como los campos obligatorios a llenar.	

Número tarea: 3	**Historia de Usuario 1**
Nombre tarea: Prueba de la funcionalidad configurar fuente.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 11/2/2015	**Fecha fin:** 11/2/2015

Programador responsable: Alexis Rey Mendoza
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.

Tareas implementadas de la historia de usuario “Configurar contraste”

Número tarea: 1	**Historia de Usuario 2**
Nombre tarea: Definición y desarrollo de la interfaz configurar contraste.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 19/2/2015	**Fecha fin:** 19/2/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se realiza el diseño de componentes que permitirán modificar los valores del nivel de contraste deseado en la aplicación. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Número tarea: 2	**Historia de Usuario 2**
Nombre tarea: Validación de la entrada de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 19/2/2015	**Fecha fin:** 19/2/2015

Programador responsable: Alexis Rey Mendoza
Descripción: Se realiza la validación de datos, asegurando así el correcto funcionamiento de la aplicación. Para ello se tiene presente los tipos de datos soportados y los posibles a entrar por los usuarios, así como los campos obligatorios a llenar.

Número tarea: 3	**Historia de Usuario 2**
Nombre tarea: Prueba de la funcionalidad configurar contraste.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/2/2015	**Fecha fin:** 20/2/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.	

3.1.2. Historias de Usuario implementadas en la segunda iteración

Historias de Usuarios	**Estimación**	**Real**
Configurar tamaño de los elementos	**2**	**2**

Tareas implementadas de la historia de usuario “Configurar tamaño de los elementos”.

Número tarea: 1	**Historia de Usuario 3**
Nombre tarea: Definición y desarrollo de la interfaz configurar tamaño de los elementos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 4/3/2015	**Fecha fin:** 4/3/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se realiza el diseño de componentes que permitirán modificar los elementos de la aplicación. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Número tarea: 2	**Historia de Usuario 3**
Nombre tarea: Validación de la entrada de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 5/3/2015	**Fecha fin:** 5/3/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se realiza la validación de datos, asegurando así el correcto funcionamiento de la aplicación. Para ello se tiene presente los tipos de datos soportados y los posibles a entrar por los usuarios, así como los campos obligatorios a llenar.	

Número tarea: 3	**Historia de Usuario 3**
Nombre tarea: Prueba de la funcionalidad configurar tamaño de los elementos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 6/3/2015	**Fecha fin:** 6/3/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.	

3.1.3. Historias de Usuario implementadas en la tercera iteración

Historias de Usuarios	**Estimación**	**Real**
Magnificador de pantalla o lupa.	**2**	**2**

Tareas implementadas de la historia de usuario “Magnificador de pantalla o lupa”.

Número tarea: 1	**Historia de Usuario 4**
Nombre tarea: Definición y desarrollo de la interfaz del magnificador de pantalla o lupa.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 21/3/2015	**Fecha fin:** 21/3/2015

Programador responsable: Alexis Rey Mendoza
Descripción: Se realiza el diseño de componentes que permitirán activar o desactivar la opción del magnificador de pantalla o lupa.

Número tarea: 2	**Historia de Usuario 4**
Nombre tarea: Validación de la entrada de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 22/3/2015	**Fecha fin:** 22/3/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se realiza la validación de datos, asegurando así el correcto funcionamiento de la aplicación.	

Número tarea: 3	**Historia de Usuario 4**
Nombre tarea: Prueba de la funcionalidad del magnificador de pantalla o lupa.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 22/3/2015	**Fecha fin:** 22/3/2015
Programador responsable: Alexis Rey Mendoza	
Descripción: Se le realizan una serie de pruebas funcionales a la aplicación, con la finalidad de obtener posibles fallas al realizar esta tarea.	

3.2. Pruebas

El desarrollo de software implica la realización de una serie de actividades entre las que se encuentra la de pruebas, siendo esta una actividad que garantiza la calidad del producto desarrollado. La metodología XP da la posibilidad de probar constantemente tanto como sea posible, siendo las pruebas uno de sus pilares fundamentales. Permitiendo aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo de corrección del error. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

El sistema de programación extrema XP divide las pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Las pruebas unitarias realizan un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. Estas nos ayudan a tener los productos completamente seguros, pero supone un estudio demasiado absoluto, que ampliaría los planes de desarrollo del menú, por lo que solo se realizará un estudio de las pruebas funcionales del sistema.

3.2.1. Prueba de aceptación.

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Son consideradas como “pruebas de caja negra” (*“Black box system tests”*). Se definen para cada historia de usuario y los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de

que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación [23].

Estas pruebas tienen una importancia crítica para el éxito de una iteración. Por lo tanto el encargado de las pruebas (tester) debe tenerlas listas lo antes posible a partir del comienzo de la iteración, y lograr que el cliente las apruebe para poder presentárselas cuanto antes al equipo de desarrollo [24].

Prueba de aceptación al sistema

Prueba de aceptación	
Código: HU1-P1	**HU: 1**
Nombre: Configurar fuente	
Descripción: La aplicación debe permitir al usuario seleccionar el tipo, tamaño y color de la fuente que usara en toda la aplicación.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: Se selecciona de las opciones proporcionadas por la aplicación. El usuario debe ir probando las diferentes combinaciones de que se hacen entre el tipo, tamaño y color de la fuente para lograr visualizar el contenido de la ventana en correspondencia con la debilidad visual que posea.	
Resultado esperado: La fuente de toda la aplicación debe tomar la configuración realizada por el usuario.	
Resultado obtenido:	

La aplicación toma la configuración realizada por el usuario.
Evaluación de la prueba: Satisfactoria.

Prueba de aceptación 1

Prueba de aceptación	
Código: HU2-P2	**HU: 2**
Nombre: Configurar contraste	
Descripción: La aplicación debe permitir al usuario seleccionar de contraste en que se presenta toda la aplicación.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: Se selecciona de las opciones proporcionadas por la aplicación. El usuario debe ir probando las diferentes combinaciones de que se hacen entre el color del texto y el nivel de contraste de toda la aplicación.	
Resultado esperado: La presentación de la aplicación debe tomar el nivel de contraste deseado por el usuario.	
Resultado obtenido: La aplicación toma la configuración realizada por el usuario.	

Evaluación de la prueba:
Satisfactoria.

Prueba de aceptación 2

Prueba de aceptación	
Código: HU3-P3	**HU: 3**
Nombre: Configurar tamaño de los elementos	
Descripción: La aplicación debe permitir al usuario seleccionar el tamaño de los elementos presentados en la aplicación.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: Se selecciona de las opciones proporcionadas por la aplicación. El usuario selecciona el tamaño que desea que tomen los elementos que se encuentran en la aplicación, estos elementos pueden ser principalmente los botones.	
Resultado esperado: Los elementos de la aplicación debe tomar el tamaño deseado por el usuario.	
Resultado obtenido: La aplicación toma la configuración realizada por el usuario.	
Evaluación de la prueba: Satisfactoria.	

Prueba de aceptación 3

Prueba de aceptación	
Código: HU4-P4	**HU: 4**
Nombre: Magnificador de pantalla o lupa.	
Descripción: La aplicación debe permitir al usuario seleccionar el magnificador de pantalla o lupa.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: Se selecciona de las opciones proporcionadas por la aplicación. El usuario selecciona la opción “activar lupa”.	
Resultado esperado: Los elementos que se encuentren en la misma posición en la que se encuentre el puntero deberán mostrarse en el cuadro de la lupa al doble de su tamaño.	
Resultado obtenido: El magnificador de pantalla muestra los elementos en el cuadro de la lupa al doble de su tamaño.	
Evaluación de la prueba: Satisfactoria.	

Prueba de aceptación 4

Conclusiones del capítulo

En este capítulo se explicaron los procesos realizados para la implementación de la solución propuesta. Fueron expuestos y explicados los métodos de pruebas que se utilizaron a la aplicación. Se mostró el flujo de las tareas para cada una de las funcionalidades desarrolladas. Con la finalización de este capítulo se da por terminada la propuesta de solución de la aplicación a implementar.

Conclusiones Generales

Al culminar la presente investigación, propuesta de accesibilidad a los laboratorios virtuales para usuarios con discapacidades visuales moderadas se pudo concluir que:

- Con el estudio del estado del arte se determinó el tipo de discapacidad a la que el módulo a desarrollar iba a dar soporte.
- Se desarrolló un módulo que permite cambiar la hoja de estilos de los laboratorios virtuales con el fin de adaptar la interfaz visual del mismo al usuario con discapacidad visual moderada.

Recomendaciones

Durante todo el desarrollo de la propuesta de solución siempre surgen nuevas ideas sobre cómo mejorar lo que satisface la problemática planteada para el presente trabajo de diploma. Estas ideas no siempre son posibles realizarlas. Es por esto que se recomienda:

- Brindar a los usuarios otras combinaciones de colores a seleccionar para aplicar la configuración de alto contraste.
- Proporcionar acceso mediante combinaciones de teclas a todas las funciones.
- Ofrecer información importante por medio de sonido.

.

Referencias Bibliográficas

1. **Segovia, Claudio.** Accesibilidad e Internet. [En línea] [Citado el: 1 de Abril de 2015.] http://www.archena.es/files/accesibilidad_e_internet.pdf.

2. **Marca Huallpara, Michael Hugo y Quisbert Limachi, Nancy Susana.** *ANALISIS Y DISEÑO DE SISTEMAS II.* pág. 2.

3. **López Nicot, Leslier.** *Propuesta de Guía para lograr la Accesibilidad Web en los Proyectos Productivos.* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2008.

4. **Penadés, P.L.y.M.C.** *Metodologías ágiles para el desarrollo de software.* s.l. : Extreme Programming (XP).

5. **Sanchez, M.A.M.** *Metodologías De Desarrollo De Software.* 2004.

6. **Pressman, Roger S.** *Ingenieria de Software.* 2005.

7. **Ramírez, Y.** *Arquitectura y Patrones de Diseño. En Tema I. Culminación de la Fase de Elaboración.*

8. **Reynoso, C. y. K., Nicolas.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. . nº .*

9. **J. J. Gutiérrez, M.J.E., M. Mejías, J. Torres.** *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.*

10. **Juan Pablo Cassinelli, R.S., Dayvis Malfará, Diego Cukerman, Fernando Cócaro.** *Testing en eXtreme Programming.*

11. **Salud y Medicinas.** Discapacidad Visual. [En línea] [Citado el: 2 de 5 de 2015.] http://www.saludymedicinas.com.mx/centros-de-salud/visual/temas-relacionados/discapacidad-visual.html.

12. **Beck, K.** Extreme Programming Explained. 1999.

13. **Joskowicz, J.** Reglas y Prácticas en eXtreme Programming. 2008.

14. **Explorando el Mundo de la Visión.** ¿Qué es la visión? [En línea] [Citado el: 8 de 5 de 2015.] http://rosavision.blogspot.com/2008/03/qu-es-la-visin-su-importancia-what-is.html.

15. **Accesibilidad Informática.** Accesibilidad Informática. [En línea] [Citado el: 22 de 5 de 2015.] http://www.menela.org/revista/maremagnum%2011/castellano%2011/a11.21.pdf.

16. **Agudeza Visual.** ***Agudeza Visual.*** **[En línea] [Citado el: 25 de Febrero de 2015.] http://www.objetivoemetropia.com/comun/pdf/Agudeza_visual_esp.pdf.**

17. AUREA, Estándares web y Accesibilidad. [En línea] [Citado el: 22 de 5 de 2015.] http://aurea.es/2008/05/25/tipos-de-discapacidades-y-accesibilidad-web/.

18. Axon. ***Axon.*** **[En línea] [Citado el: 25 de Febrero de 2015.] http://media.axon.es/pdf/80824.pdf.**

19. Centro de Informática Médica. [En línea] [Citado el: 7 de Mayo de 2015.] http://gespro.cesim.prod.uci.cu/.

20. Clinica Ocular. La vista el sentido más importante. [En línea] [Citado el: 8 de 5 de 2015.] http://clinicaocularmarcos.wordpress.com/2012/04/10/la-vista-el-sentido-mas-importante/.

21. Manual C++. ***Manual C++.*** **[En línea] [Citado el: 24 de Abril de 2015.] http://mat21.etsii.upm.es/ayudainf/aprendainf/Cpp/manualcpp.pdf.**

22. Manual Qt Creator. [En línea] [Citado el: 22 de 5 de 2015.] http://sites.google.com/site/freeunomas/manual-qt-creator.

23. Qt Creator. Pixelcoblog.com. [En línea] [Citado el: 22 de 5 de 2015.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.

24. W3. StyleSheet. [En línea] [Citado el: 22 de 5 de 2015.] http://www.w3.org/Style/CSS/.

25. Discapacidad Visual. Visual. [En línea] [Citado el: 23 de Febrero de 2015.] http://eduespecialg.efemerides.ec/1/visual.pdf.

26. Visual Studio. ***Visual Studio.*** **[En línea] [Citado el: 24 de Abril de 2015.] http://msdn2.microsoft.com/es-es/vstudio/default.aspx.**

27. Corporation, Microsoft. Ayuda de accesibilidad de Windows. [En línea] [Citado el: 21 de 5 de 2015.] http://msdn.microsoft.com/es-es/library/cb35a5fw%28v=vs.80%29.aspx.

Glosario de Términos

API: es una interfaz de programación de aplicaciones o API (*Application Programing Interface, por sus siglas en inglés*) creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación como un entorno de desarrollo integrado o IDE (*Integrated Development Environment, por sus siglas en inglés*).

CSS (Hojas de estilo en cascada): Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (*World Wide Web Consortium, por sus siglas en inglés*) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

Framework: arquitectura reusable que brinda la estructura genérica y de comportamiento para un grupo de abstracciones de software. Incluyendo la infraestructura y los mecanismos para ejecutar interacciones entre los componentes abstractos y las implementaciones libres de estos. En sentido general es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Su diferencia fundamental con las bibliotecas de código es que emplea un flujo invertido de control entre él y sus clientes.

HTML (Lenguaje de marcado de hipertexto): Es el lenguaje de marcado predominante para la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. Describe la apariencia de un documento, e incluye un script, el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

Metodología: define el cómo trabajar eficientemente evitando las problemáticas que conllevan al fracaso de muchos proyectos de software. El objetivo fundamental de una metodología es aumentar la calidad del software, haciendo énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo “producir lo esperado en el tiempo esperado y con el coste esperado”.

SDK: kit de desarrollo de software (*Software Development Kit, por sus siglas en inglés***)** es un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, *frameworks*, plataformas de hardware, computadoras, videoconsolas y sistemas operativos.

SQL: el lenguaje de consulta estructurado o SQL (Structured Query Language*, por sus siglas en inglés*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

XML (Lenguaje de marcas extensible por sus siglas en inglés): Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.