**Arquitectura Cliente-Servidor**

(esto aqui, de cliente servidor hace falta refinar un poco)  
La arquitectura cliente-servidor es una arquitectura de procesamiento cooperativo entre computadores que está conformada por dos (2) componentes principales: clientes y servidores. Los clientes se encargan de iniciar la comunicación con el servidor a través de la red, emitiendo solicitudes de recursos o peticiones de servicios, como por ejemplo consultar una información o solicitar un archivo, y el servidor se encarga de atender las solicitudes realizadas por los clientes, respondiendo con el recurso o la información solicitada (Olivares, 2012).  
El servidor suele estar ubicado en un equipo con grandes capacidades de almacenamiento y procesamiento, para poder atender de forma eficiente las peticiones realizadas por los clientes. Existen distintos tipos de servidores como los servidores web, que almacenan principalmente documentos HTML (HyperText Markup Language, Lenguaje de Marcas de Hipertexto), los servidores de base de datos, que dan servicios de almacenamiento y gestión de bases de datos a sus clientes, los servidores del correo, que almacenan, envían, reciben y realizan todas las operaciones relacionadas con el correo electrónico de sus clientes, entre otros tipos de servidores.

En la **Figura 1:**

se puede observar la representación de un cliente-servidor:

Algunas de las características principales de esta arquitectura son:

• El cliente mantiene un papel activo en la comunicación, debido a que inicia la comunicación a través de solicitudes.  
• El cliente espera y recibe respuestas del servidor.  
• Un cliente puede comunicarse con más de un servidor de manera concurrente.  
• El cliente es el componente del sistema que interactúa de manera directa con el usuario final, a través de una interfaz gráfica.  
• El servidor permanece en espera de solicitudes, por lo tanto se considera que tiene un papel pasivo en la comunicación.  
• Un servidor puede recibir y mantener la conexión con diversos clientes al mismo tiempo, sin embargo el número máximo de conexiones posibles es limitado.

**Funcionamiento de una aplicación web**  
Las aplicaciones web se basan en la arquitectura cliente-servidor, explicada anteriormente, donde el cliente es un navegador web y el servidor es un servidor web que almacena las páginas de la aplicación.

El funcionamiento de una aplicación web es sencillo, el cliente emite una petición de una página web estática o dinámica al servidor web, y éste devuelve la página solicitada. Una página web estática, es aquélla que no cambia cuando un usuario la solicita: el servidor web envía la página al navegador web solicitante sin modificarla.

Por el contrario, una página web dinámica es resuelta y adaptada a los datos del cliente en específico, ya que tiene líneas de código no determinadas, y que son resueltas por el servidor según la información almacenada del usuario en la sesión actual; el servidor web puede transferir la página a un software especial, a una librería o a un módulo externo, generalmente llamado servidor de aplicaciones, que se encarga de leer el código de la página y generará en función de las instrucciones del código dinámico un contenido específico para el usuario.

El resultado tiende a ser una página estática (código HTML puro) que el servidor de aplicaciones devuelve al servidor web, que a su vez la envía al navegador solicitante, El servidor de aplicaciones también permite trabajar con recursos del lado del servidor, como las bases de datos. Sin embargo, el servidor de aplicaciones no puede comunicarse con la base de datos directamente, necesita de un controlador que actúa de intermediario para establecer la comunicación.

**En la Figura 2**. se puede observar el proceso para solicitar una página web estática al servidor web y en la

**Figura 2.1** el proceso para solicitar una página web dinámica al servidor de aplicaciones con datos extraídos de una base de datos.  
Aunque los servidores web están principalmente diseñados para presentar contenido estático, hoy en día la mayoría de éstos tienen complementos para soportar lenguajes de scripting (guiones) como Perl, PHP, ASP ref[2,3,4], entre otros, pudiendo servir contenido dinámico sin necesidad de comunicarse con un servidor de aplicaciones. También, muchos de los servidores de aplicaciones de la actualidad son capaces de hacer lo que un servidor web es capaz de hacer, es decir, que pueden servir tanto contenido dinámico como estático.

**Ventajas**

Algunas de las ventajas de las aplicaciones web son:

• Las aplicaciones web pueden ser utilizadas por múltiples usuarios al mismo tiempo.

• No necesitan ser descargadas, instaladas y configuradas. Se acceden vía en línea y están listas para trabajar sin importar cuál es su configuración o su hardware.

• Una aplicación web se puede ejecutar en múltiples plataformas (hardware o sistema operativo), ya que sólo se necesita de un navegador web para cada plataforma.

• Todos los cambios de interfaz o de funcionalidad que se deseen hacer en la aplicación web, se realizan en el código que reside en el servidor web y no en cada uno de los clientes.

• Se evitan problemas de inconsistencia de actualizaciones, ya que no existen clientes con distintas versiones de la aplicación.

**2.4.3. Desventajas**

Algunas desventajas son:

• Los múltiples accesos realizados en forma simultánea pueden degradar el rendimiento de las aplicaciones, debido a la sobrecarga del servidor donde se encuentran alojadas.

• Como el navegador es la herramienta utilizada para ejecutar las aplicaciones web, éstas dependen de la configuración del mismo, para que puedan trabajar de forma óptima, ya sea en cuanto a las Cookies, JavaScript o archivos Flash.

• Si el servidor donde está almacenada la aplicación web y la información no tiene los niveles de seguridad necesarios para resguardar los datos, éstas pueden ser vulnerables a los ataques de agentes maliciosos.

**Patrón MVC**

El Modelo Vista Controlador (MVC), es un patrón de diseño muy utilizado para el desarrollo de aplicaciones web, el cual tiene como idea fundamental separar los componentes principales de las aplicaciones, como lo son el sistema de gestión de base dedatos, las interfaces de usuario y las tareas de manejar los eventos que se produzcan.

A continuación se explican las partes del patrón MVC:

• Modelo: es la capa donde se trabaja con los datos, por tanto contiene mecanismos para acceder a la información. Cabe mencionar que en esta capa es habitual utilizar un ORM (Object Relational Mapping, Mapeo Objeto Relacional), que permita trabajar con abstracción de bases de datos y persistencia en objetos, en vez de usar directamente sentencias SQL (Structured Query Language, Lenguaje de Sentencias Estructuradas), que suele depender del motor de base de datos con el que se esté trabajando.

Principales responsabilidades:

o Acceder a la capa de almacenamiento de datos. o Definir las reglas de negocio (lógica del negocio). Ejemplo: “Si la persona X

tiene el privilegio Y, no puede ver la información Z”.

• Vista: es la interfaz de usuario que permite mostrar los datos del modelo e interactuar con ellos mediante acciones o eventos.

Principales responsabilidades:

o Recibir datos provenientes del modelo y mostrarlos al usuario. o Tener registro de su controlador asociado.

• 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.

Principales responsabilidades:

o Recibir los eventos de entrada que se realizan en la vista, como un clic, pulsar un botón, entre otros. o Contener las reglas de gestión de eventos (lógica de la aplicación). Ejemplo: “Si Evento Z, entonces Acción W”. Estas acciones pueden suponer peticiones al modelo o a las vistas.

**Flujo de control del patrón MVC**

Aunque se pueden encontrar diferentes implementaciones del MVC, el flujo de control más frecuente de este patrón es el siguiente:

**Figura 3:**

1. El usuario interactúa con la vista realizando alguna acción, por ejemplo, pulsar un botón.

2. El controlador recibe la notificación de la acción del usuario y gestiona el evento,normalmente a través de un gestor de eventos.

3. El controlador hace la llamada al modelo, donde le pide visualizar o actualizar los datos.

4. El modelo interactúa con la base de datos, ya sea en forma directa o con una capa de abstracción, y realiza la consulta o la actualización de los datos.

5. El modelo le envía al controlador la información obtenida o le notifica de los cambios realizados.

6. El controlador le manda la respuesta a la vista y ésta reacciona en función de la

respuesta que recibe (actualizar la interfaz, abrir un enlace, etc.)

**Ventajas del MVC**

El patrón MVC para el desarrollo de aplicaciones web presenta las siguientes ventajas:

• Disminuye el acoplamiento entre las partes de la aplicación, ya que las vistas se desacoplan del modelo y el modelo se desacopla de la forma en que se muestra e ingresa los datos.

• Aumenta la cohesión, ya que cada elemento del patrón está altamente especializado en su tarea.

• Facilita el desarrollo de la misma aplicación en distintos dispositivos o canales.

**Figura 4 - (Flujo de control del patrón MVC)**

• Permite una mayor claridad de diseño.

• Facilita el mantenimiento.

• Aumenta la escalabilidad.

**Tecnologías del lado del cliente**

**JavaScript**  
JavaScript (JS) es un lenguaje de scripting, interpretado, diseñado para la manipulación de eventos y objetos contenidos en una página web, de forma local y sin necesidad de transmisión alguna por la red.

JavaScript permite crear pequeños programas llamados script, que pueden acompañar un documento HTML o estar contenidos en su interior. Dicho programa se ejecuta en el navegador web cuando se carga el documento, o cuando se produce algún evento, como por ejemplo la activación de un enlace por parte del usuario (España, 2003).

Gracias a los scripts se pueden realizar tareas como:

• Validación de datos introducidos en un formulario.  
• Mensajes de alerta.  
• Algunos cálculos simples.  
• Control de eventos que se producen en la página: un elemento toma el foco, un elemento es activado con el ratón, etc.

La ventaja de poder de realizar tareas de este tipo en el navegador web, es que por ejemplo si el usuario no rellena correctamente un formulario, no necesita esperar mucho tiempo hasta que el servidor vuelva a mostrar el formulario indicando los errores existentes.

Para insertar código JavaScript en un documento HTML se utiliza la etiqueta <script> de la siguiente manera:

***<script language=”JavaScript”>  
 <!--<instrucciones> -->***

***</script>***  
Actualmente existe una gran cantidad de bibliotecas basadas en JavaScript, siendo jQuery una de las más utilizadas, ya que ésta posee una gran cantidad de funciones para simplificar la manera de interactuar con los documentos HTML, manipular documentos a través del árbol DOM (Document Object Model, Modelo de Objetos del Documento), gestionar eventos, animar y agregar interacción a las páginas web con la técnica AJAX (Asynchronous JavaScript And XM, JavaScript Asíncrono y XML).

La técnica AJAX tiene como objetivo crear páginas web dinámicas, mediante la realización de solicitudes asíncronas hacia el servidor, es decir sin la participación directa del usuario, las cuales permiten actualizar ciertas secciones de la página sin necesidad de recargarla por completo. En ambientes donde no se implementa esta técnica, cualquier actualización no será visible hasta tanto se recargue por completo el contenido de la página (W3Schools, 2015).

**Tecnologías del lado del servidor**

Las tecnologías del lado del servidor permiten construir la lógica de negocio de la aplicación web. Esto implica el procesamiento de las solicitudes que se realizan desde el navegador web así como también la comunicación con el sistema encargado del manejo de datos.

Entre las tecnologías del lado del servidor que se explican a continuación están: el lenguaje de programación Ruby, el marco de trabajo Ruby on Rails, el sistema manejador de base de datos PostgreSQL y el Protocolo Ligero/Simplificado de Acceso a Directorios.

**Directorio y servicio de directorio**

**Directorio**

Un directorio es un repositorio único para la información relativa a los usuarios y recursos de una organización, el cual está organizado de manera lógica y jerárquica. Por ejemplo, en un directorio se puede almacenar información acerca de las cuentas de usuario de las personas de una organización (nombres, contraseñas, números de teléfono, etc.) (EcuRed, 2012).

Generalmente se describe un directorio como una base de datos, pero es una base de datos especializada cuyas características la apartan de una base de datos relacional de propósito general (Calzada, 2001).

Algunas de las características que diferencian un directorio de una base de datos relacional son (Calzada, 2001):

• Los directorios están optimizados para accesos en lectura, frente a las bases de datos convencionales, que se encuentran optimizadas para lectura y escritura.

• Los directorios están optimizados para almacenar información relativamente estática, por lo que no son recomendables para almacenar datos que cambian con frecuencia.

• Los directorios no soportan transacciones. Las transacciones son operaciones de base de datos que permiten controlar la ejecución de una operación compleja, de modo que dicha operación se completa totalmente o no se ejecuta en absoluto. Las bases de datos convencionales implementan esta funcionalidad, a costa de hacer su implementación más compleja. Pero el tipo de información que se almacena generalmente en el directorio no requiere una consistencia estricta y se considera aceptable que el número de teléfono de una persona no esté actualizado de forma temporal.

• El diseño del directorio es bastante diferente del diseño de una base de datos relacional. En las bases de datos se tiende a diseñar un modelo de datos para asuntos de negocios y los requisitos de los procesos, el cliente, el servicio y el administrador. Con los directorios, lo que se hace es colocar la información en un repositorio común para muchos usos y usuarios. Su diseño y esquema de la información deben ser desarrollados conforme a lo que está representando, a objetos en la vida real. En la mayoría de los casos, estos objetos representan los usuarios, agendas, listas, preferencias, derechos, productos y servicios, dispositivos, perfiles, políticas, números de teléfono, rutas, etc. Además, se debe considerar también los aspectos operacionales de diseño, en vista del funcionamiento y de escala.

**Servicio de directorio**

Un servicio de directorio es una aplicación o un conjunto de aplicaciones que proporciona métodos para almacenar los datos de un directorio y a su vez ponerlos a disposición de los administradores y los usuarios que se encuentren en la misma red (EcuRed, 2012).

Un servicio de directorio está altamente optimizado para lecturas y proporciona alternativas avanzadas de búsqueda en los diferentes atributos que se puedan asociar a los objetos de un directorio.

**Protocolo Ligero/Simplificado de Acceso a Directorios**

Las aplicaciones que deseen acceder a un directorio, deben utilizar el Protocolo Ligero/Simplificado de Acceso a Directorios, mejor conocido como LDAP (Lightweight Directory Access Protocol), el cual es un protocolo de tipo cliente-servidor que permite establecer la comunicación entre la aplicación y el servicio de directorio para que ésta pueda acceder o modificar la información del repositorio.

LDAP le brinda al usuario la posibilidad de:

* - Conectarse al directorio.
* - Desconectarse del directorio.
* - Buscar información.
* - Comparar información.
* - Insertar entradas.
* - Cambiar entradas.
* - Eliminar entradas.

**CAPÍTULO 3 MARCO METODOLÓGICO**

En todo desarrollo de software, aplicar cierto grado de disciplina siempre ayuda a conseguir un producto final de mayor calidad. Las metodologías ágiles son las más adecuadas para proyectos donde es necesaria la flexibilidad y la adaptación a los cambios, como es el caso de las aplicaciones web, que suelen tener requerimientos variables y usualmente se desea que estén disponibles en poco tiempo.

Algunas metodologías ágiles de uso común en la actualidad son: Programación Extrema (XP), Proceso Unificado Ágil (AUP), Proceso Unificado Abierto (OpenUP), Scrum, AgilUs, entre otras.

AgilUs es una metodología ágil que se adapta bien a proyectos individuales o de equipos pequeños, se centra en el usuario y sus necesidades, e incorpora técnicas de evaluación de usabilidad rápidas, económicas y que no requieren de plataformas tecnológicas complejas. Por todas estas razones, esta metodología resultó ser la adecuada para el desarrollo de este Trabajo Especial de Grado.

En este capítulo se explica en profundidad la metodología AgilUs, analizando sus principales características y describiendo las etapas que contempla.

**Metodología AgilUs**

La metodología AgilUs de la profesora Alecia Eleonora Acosta nace como línea de investigación del Centro Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación de la Universidad Central de Venezuela. Se basa en el concepto de usabilidad, en la necesidad de desarrollar software usable. Se fundamenta en el análisis centrado en el usuario y en la participación de especialistas, con el objetivo de evolucionar el software, a fin de que éste alcance el mayor grado de usabilidad una vez culminado su desarrollo (Acosta, 2011).

AgilUs es un método de desarrollo iterativo e incremental que pone el mayor peso del desarrollo en la consecución de la usabilidad. Se centra en que la construcción de las interfaces de usuario se realice desde el comienzo del desarrollo del sistema y no al final.

Esta metodología reduce la cantidad de actividades y artefactos que se generan, propicia la participación del usuario y se adapta a los cambios que puedan surgir en el desarrollo. Está orientada al desarrollo de sistemas con alto grado de interactividad con los usuarios (front-end) y menos complejidad en la lógica de la aplicación (back-end).

AgilUs provee un conjunto de “buenas prácticas” para el desarrollo de software desde una perspectiva ágil, donde requiere involucrar al usuario de manera activa durante el proceso de desarrollo. Algunas de estas “buenas prácticas” son:

• Diseño centrado en el usuario (DCU): es un enfoque de diseño y desarrollo que se centra en los deseos, limitaciones y necesidades de los usuarios finales de un software.

• Diseño basado en prototipos: el desarrollo de software es guiado por la construcción de prototipos de alta fidelidad y la evaluación de los mismos por los usuarios y por especialistas en usabilidad.

• Desarrollo ágil, incremental e iterativo: el desarrollo es lo más simple posible, provee resultados sin necesidad de esclarecer todo los requerimientos al inicio del desarrollo y permite regresar a etapas anteriores una vez recibida la retroalimentación de las evaluaciones realizadas.

• Usabilidad como atributo de la calidad: se siguen las recomendaciones del estándar ISO/IEC 9126‐1 para la producción de software usable, ya que la usabilidad es considerada un atributo de la calidad interna y externa del software.

• Interacción continua con el usuario: la presencia constante y participativa del usuario es fundamental. La usabilidad del sistema sólo puede ser determinada por el usuario.

Acosta (2011) explica que AgilUs centra el desarrollo de software en los siguientes principios:

• Integra la Interacción Humano Computador (IHC) y la Ingeniería de Software (IS): IS e IHC son complementarias, no son disciplinas excluyentes. Un diseño centrado en el usuario impacta positivamente en la calidad del software (ISO 9126-1).

• La usabilidad debe considerarse desde el principio del desarrollo: la usabilidad aumenta la calidad del software, entonces es conveniente incluirla desde el principio en el desarrollo como uno de los requerimientos fundamentales para impactar positivamente en la calidad del producto final.

• La usabilidad determina la utilidad: un software se considera útil en la medida que pueda ser usado a fin de producir resultados, en forma eficiente, intuitiva y satisfactoria para los usuarios.

• El usuario determina la usabilidad: la usabilidad no es una propiedad abstracta. Un software sólo será considerado usable en un contexto específico y por un tipo de usuario específico. El objetivo es lograr que todos los usuarios del software encuentren usables las tareas que pueden realizar.

**Ciclo de vida**

“El ciclo de vida de AgilUs hace énfasis en la importancia del usuario y sus evaluaciones. Está basado en el desarrollo iterativo e incremental de prototipos de alta fidelidad hasta que se convierten en el producto final para entrega” (Acosta, 2011).

AgilUs está formado por cuatro (4) etapas diferentes: Requisitos, Análisis, Entrega y Prototipaje, y en cada una de ellas se incluyen actividades y artefactos poco costosos para la construcción de la usabilidad, como se muestra en la

**Figura 8**

**Etapas**

A continuación se describe las cuatro (4) etapas de AgilUs y las técnicas de evaluación de usabilidad utilizadas (Acosta, 2011):

• Requisitos: se realiza el análisis global del problema a solucionar, se estudian productos similares existentes, se genera un perfil de usuario, y se define la lista de requerimientos a desarrollar. Esta etapa es importante en el desarrollo del software,ya que un mal análisis de requerimientos traería como consecuencia un software que no cumple con las necesidades del usuario.

Las técnicas de evaluación de usabilidad (indagación) que se utilizan en esta etapa son:

• Tormenta de ideas: técnica de grupo para la generación de ideas nuevas y útiles, que permitan mediante reglas sencillas aumentar las probabilidades de innovación y originalidad.

• Encuesta: conjunto de preguntas normalizadas dirigidas a una muestra representativa de la población para obtener información estadística acerca de sus opiniones o hechos específicos.

• Cuestionario: lista de preguntas que se distribuye entre los usuarios para obtener información de sus opiniones. Este instrumento se puede utilizar en cualquier etapa del proceso de desarrollo, dependiendo de las preguntas que estén formuladas.

• Entrevista: diálogo entre un entrevistador y un entrevistado, donde el entrevistador hace una serie de preguntas al entrevistado con el fin de conocer sus opiniones e ideas.

• Evaluación de sistemas existentes: consiste en la revisión de versiones anteriores del mismo sistema, así como sistemas de la competencia o afines, con el objetivo de identificar ventajas, desventajas y problemas de usabilidad.

• Perfil de usuario: describe a los usuarios previstos del sistema, detallando sus

características más relevantes.

• Requerimientos funcionales y no Funcionales: recopilación de condiciones o

capacidades que debe cumplir el sistema.

• Análisis: se lleva a cabo el análisis de la solución a desarrollar, se emplean diagramas de casos de uso y modelo de objetos del dominio, siguiendo la notación UML, para definir las funcionalidades que tendrá el producto a desarrollar.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

• Prototipo en papel: bosquejo del sistema que se desea realizar. Resulta muy

útil cuando el cliente no tiene claros algunos puntos.

• Guía de estilo: documento que recoge normativas y patrones básicos relacionados con el aspecto de la interfaz para su aplicación en el desarrollo de nuevas pantallas dentro de su entorno.

• Modelo de casos de uso: modelo que permite especificar las funcionalidades y el comportamiento del sistema mediante su interacción con usuarios u otros sistemas.

• Modelo de objetos del dominio: modelo que describe y prioriza cada uno de los

objetos identificados en el sistema.

• Objetos de interfaz: modelo que permite comprender la parte visual de la

aplicación.

• Patrones de interacción: describen aspectos concernientes a la interfaz de usuario; están orientados a presentar soluciones apropiadas a problemas recurrentes que se les presentan a los usuarios cuando utilizan las aplicaciones interactivas.

• Prototipaje: Se implementa un prototipo rápido de la interfaz de usuario a partir de los patrones de interacción, el cual va evolucionando hasta convertirse en el producto final y se realizan evaluaciones de usabilidad apropiadas a esta etapa: evaluaciones heurísticas y listas de comprobación.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

• Prototipo rápido: técnica basada en ordenador que pretende reducir el ciclo iterativo de desarrollo. Los prototipos iterativos desarrollados podrán ser rápidamente reemplazados o modificados según los informes procedentes de otras evaluaciones a medida que se evoluciona en el desarrollo de las tareas a realizar. Existen muchas herramientas para la generación de prototipos rápidos, siendo habituales una secuencia de imágenes en Microsoft PowerPoint o Visual Basic.

• Evaluación heurística: se basa en una inspección donde especialistas en usabilidad juzgan si cada elemento de la interfaz sigue los principios de usabilidad establecidos.

• Lista de comprobación: consta de la verificación del cumplimiento de una lista de principios que el equipo de desarrollo acuerda para el diseño de la interfaz de usuario.

• Pensamiento en voz alta: técnica donde se les pide a los usuarios que expresen en voz alta sus pensamientos, sentimientos y opiniones mientras que interaccionan con el sistema o un prototipo del mismo.

• Prototipo ejecutable: técnica donde el prototipo utilizado para las pruebas se corresponde con la interfaz real en la mayor medida posible. Es utilizado para maquetar la interfaz y en él son aceptadas entradas desde ratón o teclado, tal como lo haría la interfaz real, a su vez que responde a esos eventos de idéntica forma (mostrando ventanas, mensajes de respuesta, entre otros).

• Entrega: se aplican las pruebas al sistema para certificar que la aplicación desarrollada sea un software usable y sin errores, finalmente se pone en producción la aplicación.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

• Pruebas de aceptación: pruebas que aportan datos tanto cuantitativos como cualitativos sobre usuarios reales que llevan a cabo tareas reales con el sistema. Los profesionales de la usabilidad pueden evaluar algunos aspectos de la accesibilidad utilizando protocolos normalizados de pruebas de usabilidad.

• Protocolo de preguntas: consiste en la formulación de preguntas directas al usuario acerca del sistema para detectar qué partes de la interfaz resultan obvias y que otras resultan oscuras.

**Errores comunes**

Acosta (2011) señala que al implementar AgilUs se pueden cometer algunos errores comunes que surgen a causa de la disminución deliberada o accidental de la participación del usuario en el proceso de desarrollo o de una mala comprensión del rol que debe ocupar a la hora de tomar decisiones con respecto al diseño del sistema y sus interfaces.

Uno de los errores más frecuentes es que el equipo de desarrollo determine la usabilidad del sistema sin incluir al usuario, intentando predecir o aproximar las reacciones, deseos, capacidades y carencias de éste. Otro error común es pensar que el cliente y el usuario son la misma persona. A pesar de que el cliente es el propietario del sistema, éste no debería tomar decisiones, por ejemplo en términos de usabilidad, ya que los usuarios son los que deberían decidir, indirectamente a través de sus evaluaciones y comentarios, qué se debe hacer y por qué.

**Tabla resumen**

**Resumen de la Metodología AgilUs**

**Definición**

Metodología ágil fuertemente orientada a la producción de software usable, por ello el usuario es el actor principal durante el proceso de desarrollo.

**Etapas**

• Requisitos. - Análisis. - Prototipaje. - Entrega.

**Roles** - No tiene roles.

Tiempos de entrega Al finalizar una iteración se entrega un prototipo para ser evaluado.

**Artefactos**

Etapa de Requisitos: - Tormenta de ideas. - Encuestas, entrevistas, cuestionarios. - Evaluación de sistemas existentes. - Perfiles de usuario. - Requerimientos funcionales y no funcionales.

Etapa de Análisis: - Prototipo en papel. - Guía de estilo. - Casos de uso. - Modelo de objetos del dominio. - Objetos de interfaz. - Patrones de interacción.

Etapa de Prototipaje: - Evaluación heurística. - Lista de comprobación. - Pensamiento en voz alta. - Prototipo ejecutable.

Etapa de Entrega: - Protocolo de preguntas. - Pruebas de aceptación. - Aplicación final.

**Ventajas**

• Constante interacción con el usuario durante todo el proceso de

desarrollo. - Incremento de la productividad, la satisfacción y la reducción de costos de entrenamiento y soporte del software, ya que las técnicas de evaluación de usabilidad propuestas en el método no son costosas ni requieren de infraestructura tecnológica compleja.

**Desventajas**

• No contempla en su ciclo de vida una fase de mantenimiento, lo cual dificulta su evolución y/o corrección una vez finalizado el desarrollo del producto. - Puede ocurrir una disminución de la participación del usuario en el proceso de desarrollo o puede haber una mala comprensión de su rol.

**CAPÍTULO 4 MARCO APLICATIVO**

En este capítulo se describen las y características relevantes de la aplicación y se detallan los pasos que se realizaron para el desarrollo de este Trabajo Especial de Grado, siguiendo de manera parcial los lineamientos de la metodología AgilUs.

por ahora pienso que deberíamos poner aqui, arquitectura Cliente-servidor, API restfull, ruby on rails, mysql, que es un Mailer, definir que es un servicio de reclutamiento, que tipo de datos se piden en entrevistas de trabajo, que información válida a un profesional, como certificar una persona como valida, que tipo de pruebas te hace el sistema, como sera la configuracion, roles de usuarios (empresas/vacante - reclutados/interesados), definir un glosario de terminos, de donde sacaremos la información, como filtraremos a las personas, que cosas generarán información para el sistema, que tipo de feedback daremos, cosas asi… averiguar métodos de reclutamiento comunes y enumerar sus pros y contras para colocar una buena justificación del proyecto,

claro, esto es un marco teórico por ahora, pero mientras mas completo, menos trabajo tendremos que completar para que sea válido para el trabajo de grado.

si encuentras Ideas que aporten para este segmento escribelo, metodologías herramientas, gráficos, etc, todo sirve, tecnología a utilizar, cronograma del proyecto, lo que sea, cosas para complementar asi como el ejeplo dejado por el profesor, de las metodologias de gestion de proyectos (crei que deberia ser metodologias de ‘desarrollo de proyectos….’)