**TABLA DE CONTENIDO**

[1. INTRODUCCIÓN 3](#_Toc397277607)

[2. PRESENTACIÓN GENERAL DEL ANTEPROYECTO 5](#_Toc397277608)

[2.1. TÍTULO 6](#_Toc397277609)

[2.2. PLANTEAMIENTO DEL PROBLEMA 6](#_Toc397277610)

[2.3. JUSTIFICACIÓN 7](#_Toc397277611)

[2.4. OBJETIVOS 8](#_Toc397277612)

[2.4.1. Objetivó General 8](#_Toc397277613)

[2.4.2. Objetivos Específicos 8](#_Toc397277614)

[2.5. ALCANCES Y DELIMINTACIONES 8](#_Toc397277615)

[2.5.1. Alcance 8](#_Toc397277616)

[2.5.2. Limitación y Delimitación 9](#_Toc397277617)

[2.5.2.1. Delimitación Espacial 9](#_Toc397277618)

[2.5.2.2. Delimitación Temporal 9](#_Toc397277619)

[2.5.2.3. Limitación 9](#_Toc397277620)

[3. MARCO DE REFERENCIA 9](#_Toc397277621)

[3.1. ANTECEDENTES 10](#_Toc397277622)

[3.2. MARCO TEÓRICO 10](#_Toc397277623)

[3.2.1. METODOLOGIAS AGILES 11](#_Toc397277624)

[3.2.2. Control de Versiones 11](#_Toc397277625)

[3.2.3. Servidor 11](#_Toc397277626)

[3.2.4. Protocolo de transferencia de archivos (FTP) 12](#_Toc397277627)

[3.2.5. CPanel 13](#_Toc397277628)

[3.2.6. Base de Datos Relacional 14](#_Toc397277629)

[3.2.7. Base de Datos No relacionales (NoSQL) 15](#_Toc397277630)

[3.3. MARCO CONCEPTUAL 16](#_Toc397277631)

[3.3.1. SandBox 17](#_Toc397277632)

[3.3.2. PHP (Pre – procesador de Hipertexto) 17](#_Toc397277633)

[3.3.3. Python 17](#_Toc397277634)

[3.3.4. .NET 17](#_Toc397277635)

[3.3.5. JSP – JavaServer Pages 17](#_Toc397277636)

[3.3.6. PostgreSQL 17](#_Toc397277637)

[3.3.7. Mysql 17](#_Toc397277638)

[3.3.8. MongoDB 18](#_Toc397277639)

[3.3.9. Git 18](#_Toc397277640)

[3.3.10. Tomcat 18](#_Toc397277641)

[3.4. MARCO LEGAL 18](#_Toc397277642)

[4. DISEÑO METODOLOGICO 19](#_Toc397277643)

[4.1. TIPO DE INVESTIGACIÓN 20](#_Toc397277644)

[4.2. FUENTES DE INFORMACIÓN 20](#_Toc397277645)

[4.3. TÉCNICAS Y PROCEDIMIENTO PARA LA RECOLECCIÓN DE INFORMACIÓN 20](#_Toc397277646)

[4.3.1. Técnicas 21](#_Toc397277647)

[4.3.2. Método 21](#_Toc397277648)

[4.3.3. Procedimiento 21](#_Toc397277649)

[5. CRONOGRAMA 21](#_Toc397277650)

[6. PRESUPUESTO 22](#_Toc397277651)

[7. RESULTADOS ESPERADOS 24](#_Toc397277652)

[BIBLIOGRAFIA 26](#_Toc397277653)

**LISTA DE TABLAS**

[Tabla 1. Cronograma de actividades. 22](#_Toc397271855)

[Tabla 2. Presupuesto global del trabajo de grado. 22](#_Toc397271856)

[Tabla 3. Descripción de los gastos de personal. 23](#_Toc397271857)

[Tabla 4. Descripción de materiales y suministros. 23](#_Toc397271858)

[Tabla 5. Descripción de equipos. 23](#_Toc397271859)

# INTRODUCCIÓN

Para la presentación de proyectos para el Departamento de Ingeniería de Sistemas (DISIS) de la UFPS relacionados con el desarrollo de software podría decirse que era común encontrarse con los siguientes escenarios:

1. El estudiante desarrollaba su proyecto en su computador personal. La ventaja de este escenario para el estudiante es la fácil administración y configuración de su entorno de trabajo. Sin embargo, en caso de presentarse alguna falla en el equipo personal se corre el riesgo de perder la información del proyecto; adicionalmente, si el proyecto requiere del despliegue en servidores de la Universidad, generalmente se presentan grandes inconvenientes para la configuración del mismo en el nuevo ambiente.
2. El estudiante utiliza los servicios de préstamo de computadores suministrados por la DISIS. Este escenario es adecuado para estudiantes que con los recursos para adquirir un computador personal adecuado para la elaboración del proyecto. Sin embargo, el préstamo de los recursos para los estudiantes es limitado y generalmente en los proyectos de grado de Ingeniería de Sistemas se requiere dedicar un tiempo considerable para la elaboración del mismo.

Teniendo en cuenta lo anterior el departamento de ingeniería de sistemas promovió un proyecto que tenía como idea facilitar la construcción de software por parte de los estudiantes, mediante con la creación de una herramienta intuitiva denominada Sandbox.

El Sandbox es un sistema que integra un conjunto de aplicaciones y políticas orientadas en el desarrollador de aplicaciones, facilitando la administran de los proyectos, este método se comenzó a implementar en el segundo semestre del 2011 en la carrera de Ingeniería de Sistemas de la UFPS; de este modo los estudiantes obtienen un ambiente confiable para realizar las pruebas de sus estos.

En la actualidad la lista de tecnologías utilizadas para el desarrollo es muy intensa pero dada la necesidad de prestar el mejor servicio se integrar tecnologías para el desarrollo tales como: PHP y JSP, del mismo modo para el almacenamiento de información se utilizaran administradores de base de datos relacionales como Postgresql y mysql.

Al comienzo de la implementación de este sistemas se generaron una serie de inconvenientes que al transcurrir del tiempo se ha tenido la necesidad de optimizar. Como alternativa de esta optimización se lleva a la construcción de un frontend (interfaz) que realiza la visualización de los elementos de los usuarios en el navegador y backend (motor) que es el administrador del sitio con sus respectivos sistemas, este conjunto de conceptos garantiza una mejora del sistema. No obstante a este conjunto de implementaciones el sistema no deja de presentar un número de inconvenientes y necesidades tales como;

* la creación de cuentas de usuario dinámicamente.
* Problemas de usabilidad y escalabilidad en el administrador FTP y War.

A continuación se presenta una solución con justificación, alcances, delimitaciones y cronograma para mejorar el sistema Sandbox, de este modo la integración de nuevas herramientas tales como Python, .Net y MongolDB.

.

# PRESENTACIÓN GENERAL DEL ANTEPROYECTO

1. TÍTULO

Construcción de un frontend y backend para gestión de aplicaciones web en la nube para el programa de Ingeniería de Sistemas de la Universidad Francisco de Paula de Santander.

1. PLANTEAMIENTO DEL PROBLEMA

Hace unos años en las materias de ingeniería de software se llevaba a cabo la sustentación de proyectos en unos servidores en el centro de cómputo de la Universidad, siendo estos solamente asequibles desde estas instalaciones, creando inconvenientes a la hora de desplegarlos para la sustentación, en algunos momentos se creaban errores de versiones de las tecnologías utilizadas, así mismo como problemas de acceso a las base de datos, ya que se contaba con un número limitado de equipos y de monitores en esta área para la colaboración en el proceso.

Esto hizo que a la hora de sustentar los estudiantes se demoraran horas adaptando sus proyectos a los servidores asignados para dicha tarea y en ocasiones haciendo que estas actividades fuera una pérdida de tiempo para el estudiante y el docentes.

En la primera fase se planteó una aplicación Web que permitiera realizar estas labores remotamente, este fue el primer paso para la construcción de la plataforma Sandbox de la UFPS. Al tiempo de ser puesta en funcionamiento esta presento problemas de automatización los cuales en algunos casos llevaron a que se realizara la misma gestión que se venía realizando en el centro de cómputo.

En la actualidad la plataforma ha tenido una serie de cambios que han ayudado a mejorar, pero esto no ha solucionado del todo problemas series de automatización a la hora de gestionar cuentas de usuario, realizar mantenimientos en la plataforma por parte de los administradores.

Problema de investigación

¿Por qué se debe Desarrollar un panel de control genérico de despliegue de proyectos de software en entorno web, para facilitar el desarrollo de proyectos por parte de los estudiantes del plan de estudio de ingeniería de sistemas de la UFPS mediante el uso de una GUI amigable para el usuario?

Preguntas de investigación

* ¿Cómo están estructuradas las arquitecturas de software existente en los servidores y procesos de administración de la plataforma Sandbox?
* ¿Cómo se podría integrar tecnologías de diferentes proveedores en la plataforma Sandbox?
* ¿El backend y el frontend existente en la plataforma permitiría la integración de las nuevas tecnologías necesarias para el funcionamiento de la plataforma y en tal caso de que no que metodología de desarrollo se podría implementar?
* ¿Qué estrategia de prueba ágil de software se podría implementar para garantizar el correcto funcionamiento de las aplicaciones administrativas en la plataforma Sandbox?

1. JUSTIFICACIÓN

Los proyectos de semestre son una parte vital para los estudiantes de ingeniería de sistemas, porque es en esta práctica en la que realmente se aprende a conceptualizar lo teórico con lo práctico, haciendo de esta manera más completa y competitiva la Ingeniería de Sistemas de la U.F.P.S.

El concepto de Sandbox nace como una alternativa en la que los estudiantes pudiera desplegar sus aplicaciones y administrarlas sin tener que depender de turnos en salas y administradores de servidores, es de esta manera que surge la idea de crear una plataforma para la administración de estas tareas en un entorno web el cual esté disponible todo el semestre, haciendo posible que el estudiante realice un desarrollo de sus proyectos utilizándola e ir solucionando los inconvenientes que se vayan presentando en el desarrollo.

No obstante aunque se realizó este cambio trascendental en la forma de presentar los proyectos nació un nuevo inconveniente enfatizado en la confianza de esos servicio a la hora de utilizarlo, tales como problemas a la hora de desplegar las aplicaciones, la falta de integración en las cuentas de usuario para el uso de los diferentes servicios y poco soporte ocasionando que la plataforma no cumpliera con él. Esto forzando a que los estudiantes presentaran sus proyectos nuevamente en servidores locales.

El por qué? Para qué? Quienes se benefician? Que impacto? La viabilidad y los resultados que se quieren tener?

1. OBJETIVOS
2. Objetivó General

Desarrollar un panel de control genérico de despliegue de proyectos de software en entorno web, para facilitar el desarrollo de proyectos por parte de los estudiantes del plan de estudio de ingeniería de sistemas de la UFPS mediante el uso de una GUI amigable para el usuario.

1. Objetivos Específicos

* Diagnosticar las arquitecturas de software existente en los servidores y procesos de administración de la plataforma Sandbox.
* Integrar las nuevas tecnologías y servicios tales como .Net, Python y MongoDB a la plataforma Sandbox.
* Elaborar un Backend y Frontend utilizando JSP, Html5, CSS3 y Jquery que permita la integración de las nuevas herramientas con las ya existentes utilizando la metodología de desarrollo necesaria.
* Elaborar una estrategia de prueba ágil de software para garantizar el correcto funcionamiento de las aplicaciones administrativas en la plataforma Sandbox.

1. ALCANCES Y DELIMINTACIONES
2. Alcance

El alcance para la construcción de la herramienta web de administración de los servicios del sistema Sandbox está dado por las siguientes actividades:

* Construcción de una herramienta Web que administre (ampliar) los siguientes servicios: PHP, HTML, python, FTP, Aplicaciones WAR, .NET y github.
* Creación de los protocolos de seguridad para la administración de las bases de datos en Postgresql, Mysql y MongolDB.
* Implantación de sistemas de login y registro de usuarios.
* La realización de manuales para usuarios y administradores del sistema.
* Implementación de procedimientos de instalación y actualización del sistema.

1. Limitación y Delimitación
2. Delimitación Espacial

El proyecto se desarrollará para el Plan de Estudio de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, ubicado en el 4 piso del edificio de Aula Sur en la Avenida Gran Colombia No. 12E – 96B Barrio Colsag.

1. Delimitación Temporal

El desarrollo del proyecto se dará en un tiempo de 8 meses que serán distribuidos en las diferentes etapas del proyecto.

1. Limitación

* La seguridad y robustez de los entornos Linux donde se implementa el sistema, impiden en ocasiones la correcta utilización de los servicios por parte de los usuarios.
* La falta de documentación existente para el desarrollo de sistemas Sandbox, esto impide conseguir información segura para implementar fragmentos de código en el servidor que ejecute procesos del sistema para la solución de actividades del proyectos.
* Restricciones propias de los servicios de Microsoft para el despliegue de aplicaciones .Net en sus servidores, podría ocacionar alta dependencia del sistema a configuraciones del servicio de host.

# MARCO DE REFERENCIA

1. ANTECEDENTES

* Sandboxie. es un software de tipo SandBox para Microsoft Windows, parecido a un sistema de virtualización por aislamiento. Este freeware crea un contenedor dentro del cual los programas de aplicación se ejecutan de forma segura. Sandboxie Holdings LLC – 2004. [[1](#Txo07)]
* Google Sandbox. Es un efecto observado, que postula un sistema de filtrado especial de Google en su algoritmo, que hace que las páginas web de nueva creación no aparezcan, o aparezcan en posiciones muy inferiores, en los resultados de búsqueda, aún estando muy especializadas para ciertas palabras clave, o teniendo muchos enlaces entrantes. La explicación teórica supone que Google predice que no es normal que un sitio de reciente creación, tenga muchos enlaces entrantes en sus primeras semanas de vida, por lo que debe ser automáticamente retirado de los primeros puestos de las búsquedas. De esta manera, Google parece establecer un periodo de pruebas para comprobar si el sitio web puede o no, conseguir más enlaces y relevancia, de manera normal. Google – 2004. [[2](#Mur09)]
* Universidad de los Andes, Laboratorio de Ingeniería de Sistemas y Computación, la cual cuenta con un sistema de servidores para la administración de proyectos informáticos orientado a estudiantes con tesis o proyectos informáticos. [[3](#And13)]
* Caracterización de servidores Web de ámbito académico, realizado por: Vicente José Castaño Díaz, cuyo objetivo tuvo “Comprobar si en la actualidad se cumplen los invariantes establecidos con anterioridad por otros autores en la caracterización de servidores Web de ámbito académico.” y “Comprobar si es posible realizar un estudio de estas características haciendo uso de las herramientas software disponibles en la actualidad.” Cuya conclusión fue: “Los datos de otros invariantes reflejan diferencias significativas con respecto a estudios anteriores, algunos de ellos recientes. Habría que hacer una revisión de los invariantes para poder caracterizar de manera efectiva las nuevas tendencias en Internet, como por ejemplo portales de tipo YouTube o aplicaciones Second Life”. [[4](#Cas08)]

1. MARCO TEÓRICO
2. METODOLOGIAS AGILES

Refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos autos organizados y multidisciplinarios. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en lapsos cortos. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, sino que la meta es tener una «demo» (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto. [[5](#Dan07)]

Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos ágiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (bullpen en inglés). La oficina debe incluir revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica. [[5](#Dan07)]

1. Control de Versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra dicho producto en un momento dado de su desarrollo o modificación. Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o SVC (del inglés System Version Control). [[6](#Asc04)]

1. Servidor

En informática, un servidor es un nodo que forma parte de una red, provee servicios a otros nodos denominados clientes. También se suele denominar con la palabra servidor a:

Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

Una computadora en la que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicaciones llamadas clientes, tanto si se trata de un ordenador central (mainframe), un miniordenador, una computadora personal, una PDA o un sistema embebido; sin embargo, hay computadoras destinadas únicamente a proveer los servicios de estos programas: estos son los servidores por antonomasia. [[7](#Tei02)]

Un servidor no es necesariamente una máquina de última generación de grandes proporciones, no es necesariamente un superordenador; un servidor puede ser desde una computadora vieja, hasta una máquina sumamente potente (ej.: servidores web, bases de datos grandes, etc. Procesadores especiales y hasta varios terabytes de memoria). Todo esto depende del uso que se le dé al servidor. Si usted lo desea, puede convertir al equipo desde el cual usted está leyendo esto en un servidor instalando un programa que trabaje por la red y a la que los usuarios de su red ingresen a través de un programa de servidor web como Apache. [[7](#Tei02)]

1. Protocolo de transferencia de archivos (FTP)

El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos. [[8](#cm002)]

Para solucionar este problema son de gran utilidad aplicaciones como scp y sftp, incluidas en el paquete SSH, que permiten transferir archivos pero cifrando todo el tráfico. [[8](#cm002)]

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de backup (copia de seguridad) de los archivos importantes que pueda tener una empresa. Para ello, existen protocolos de comunicación FTP para que los datos se transmitan cifrados, como el SFTP. [[8](#cm002)]

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Microsoft Windows, DOS, GNU/Linux y Unix. Sin embargo, hay disponibles clientes con opciones añadidas e interfaz gráfica. Aunque muchos navegadores tienen ya integrado FTP, es más confiable a la hora de conectarse con servidores FTP no anónimos utilizar un programa cliente. [[8](#cm002)]

Un «cliente FTP basado en Web» no es más que un cliente FTP al cual podemos acceder a través de nuestro navegador web sin necesidad de tener otra aplicación para ello. El usuario accede a un servidor web (HTTP) que lista los contenidos de un servidor FTP. El usuario se conecta mediante HTTP a un servidor web, y el servidor web se conecta mediante FTP al servidor FTP. El servidor web actúa de intermediario haciendo pasar la información desde el servidor FTP en los puertos 20 y 21 hacia el puerto 80 HTTP que ve el usuario. [[8](#cm002)]

En modo Activo, el servidor siempre crea el canal de datos en su puerto 20, mientras que en el lado del cliente el canal de datos se asocia a un puerto aleatorio mayor que el 1024. Para ello, el cliente manda un comando PORT al servidor por el canal de control indicándole ese número de puerto, de manera que el servidor pueda abrirle una conexión de datos por donde se transferirán los archivos y los listados, en el puerto especificado. [[8](#cm002)]

Lo anterior tiene un grave problema de seguridad, y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión de entrada en un puerto superior al 1024, con los problemas que ello implica si tenemos el equipo conectado a una red insegura como Internet. De hecho, los cortafuegos que se instalen en el equipo para evitar ataques seguramente rechazarán esas conexiones aleatorias. Para solucionar esto se desarrolló el modo pasivo. [[8](#cm002)]

Cuando el cliente envía un comando PASV sobre el canal de control, el servidor FTP le indica por el canal de control, el puerto (mayor a 1023 del servidor. Ejemplo: 2040) al que debe conectarse el cliente. El cliente inicia una conexión desde el puerto siguiente al puerto de control (Ejemplo: 1036) hacia el puerto del servidor especificado anteriormente (Ejemplo: 2040). [[8](#cm002)]

1. CPanel

Es una herramienta de administración basada en tecnologías web para administrar sitios de manera fácil, con una interfaz limpia. Se trata de un software no libre disponible para un gran número de distribuciones de Linux que soporten RPM, como SuSE, Fedora, Mandriva, etc. El soporte preliminar es para Debian, se debió solicitar que saliera del estado beta por varios años, y para el cual no posee soporte. A cPanel se accede por los puertos 2082 y 2083 (para versiones de SSL). La autenticación es por HTTP o una página de ingreso. [[9](#Sup06)]

Se diseñó para el uso comercial de servicios de alojamiento web, es por esto que la compañía no lo ofrece con licencia de uso personal. Sin embargo, los dueños de organizaciones sin fines de lucro, como instituciones educacionales y caridad pueden solicitar una licencia sin costo. [[9](#Sup06)]

1. Base de Datos Relacional

Una Base de Datos Relacional, es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: "Modelo Relacional". [[10](#Str05)]

Elementos y características:

* **Relaciones Base y Derivadas:** en una base de datos relacional, todos los datos se almacenan y se accede a ellos por medio de relaciones. Las relaciones que almacenan datos son llamadas "relaciones base" y su implementación es llamada "tabla". Otras relaciones no almacenan datos, pero son calculadas al aplicar operaciones relacionales. Estas relaciones son llamadas "relaciones derivadas" y su implementación es llamada "vista" o "consulta". Las relaciones derivadas son convenientes ya que expresan información de varias relaciones actuando como si fuera una sola. [[10](#Str05)]
* **Restricciones**: una restricción es una limitación que obliga el cumplimiento de ciertas condiciones en la base de datos. Algunas no son determinadas por los usuarios, sino que son inherentemente definidas por el simple hecho de que la base de datos sea relacional. Algunas otras restricciones las puede definir el usuario, por ejemplo, usar un campo con valores enteros entre 1 y 10. [[10](#Str05)]

Las restricciones proveen un método de implementar reglas en la base de datos. Las restricciones limitan los datos que pueden ser almacenados en las tablas. Usualmente se definen usando expresiones que dan como resultado un valor booleano, indicando si los datos satisfacen la restricción o no. [[10](#Str05)]

Las restricciones no son parte formal del modelo relacional, pero son incluidas porque juegan el rol de organizar mejor los datos. Las restricciones son muy discutidas junto con los conceptos relacionales. [[10](#Str05)]

* **Dominios:** un dominio describe un conjunto de posibles valores para cierto atributo. Como un dominio restringe los valores del atributo, puede ser considerado como una restricción. Matemáticamente, atribuir un dominio a un atributo significa "todos los valores de este atributo deben ser elementos del conjunto especificado". [[10](#Str05)]

Distintos tipos de dominios son: enteros, cadenas de texto, fecha, no procedurales, etc. [[10](#Str05)]

Cada tabla puede tener uno o más campos cuyos valores identifican de forma única cada registro de dicha tabla, es decir, no pueden existir dos o más registros diferentes cuyos valores en dichos campos sean idénticos. Este conjunto de campos se llama clave única. Pueden existir varias claves únicas en una determinada tabla, y a cada una de éstas suele llamársele candidata a clave primaria. [[10](#Str05)]

* **Clave primaria:** una clave primaria es una clave única elegida entre todas las candidatas que define unívocamente a todos los demás atributos de la tabla, para especificar los datos que serán relacionados con las demás tablas. La forma de hacer esto es por medio de claves foráneas. [[10](#Str05)]
* **Clave foránea:** una clave foránea es una referencia a una clave en otra tabla, determina la relación existente en dos tablas. Las claves foráneas no necesitan ser claves únicas en la tabla donde están y sí a donde están referenciadas. [[10](#Str05)]
* **Clave índice**: Las claves índices surgen con la necesidad de tener un acceso más rápido a los datos. Los índices pueden ser creados con cualquier combinación de campos de una tabla. Las consultas que filtran registros por medio de estos campos, pueden encontrar los registros de forma no secuencial usando la clave índice. [[10](#Str05)]

Las bases de datos relacionales incluyen múltiples técnicas de ordenamiento, cada una de ellas es óptima para cierta distribución de datos y tamaño de la relación. [[10](#Str05)]

Los índices generalmente no se consideran parte de la base de datos, pues son un detalle agregado. Sin embargo, las claves índices son desarrolladas por el mismo grupo de programadores que las otras partes de la base de datos. [[10](#Str05)]

* **Procedimientos almacenados:** un procedimiento almacenado es código ejecutable que se asocia y se almacena con la base de datos. Los procedimientos almacenados usualmente recogen y personalizan operaciones comunes, como insertar un registro dentro de una tabla, recopilar información estadística, o encapsular cálculos complejos. Son frecuentemente usados por un API por seguridad o simplicidad. [[10](#Str05)]

Los procedimientos almacenados no son parte del modelo relacional, pero todas las implementaciones comerciales los incluyen. [[10](#Str05)]

1. Base de Datos No relacionales (NoSQL)

Típicamente las bases de datos relacionales modernas han mostrado poca eficiencia en determinadas aplicaciones que usan los datos de forma intensiva, incluyendo el indexado de un gran número de documentos, la presentación de páginas en sitios que tienen gran tráfico, y en sitios de streaming audiovisual. Las implementaciones típicas de RDBMS se han afinado o bien para una cantidad pequeña pero frecuente de lecturas y escrituras o para un gran conjunto de transacciones que tiene pocos accesos de escritura. Por otro lado NoSQL puede servir gran cantidad de carga de lecturas y escrituras. [[11](#ElP10)]

Implementaciones de NoSQL usadas en el mundo real incluyen los 3TB de los marcadores verdes de Digg (indicados para señalar las historias votadas por otros en la red social; aunque duró menos de 3 meses y fue abandonado). Los 6 TB de la base de datos del “ENSEMBLE” de la Comisión Europea usado en los modelos de comparación y calidad del aire, y en los 50 TB de la búsqueda de la bandeja de entrada de Facebook. [[11](#ElP10)]

Las arquitecturas NoSQL frecuentemente aportan escasas garantías de consistencia, tales como consistencia de eventos o transaccional restringida a ítems únicos de datos. Algunos sistemas, sin embargo, aportan todas las garantías de los sistemas ACID en algunas instancias añadiendo una capa intermedia (como por ejemplo, AppScale o CloudTPS). Hay dos sistemas que han sido desplegados y que aportan aislamiento snapshot para almacenamientos de columna: El sistema Percolator de Google (basado en el sistema BigTable) y el sistema transaccional de Hbase desarrollado por la universidad de Waterloo. Estos sistemas, desarrollados de forma independiente, usan conceptos similares para conseguir transacciones ACID distribuidas de múltiples filas con garantías de aislamiento snapshot para el sistema subyacente de almacenamiento en esa columna, sin sobrecarga extra en la gestión de los datos, despliegue en el sistema de middleware, ni mantenimiento introducido por la capa de middleware. [[11](#ElP10)]

Bastantes sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores, y el fallo en un servidor puede ser tolerado. [[11](#ElP10)]

Algunos defensores de NoSQL promueven interfaces simples tales como los arrays asociativos o los pares clave-valor. Otros sistemas, tales como las bases de datos nativas en XML, promueven el soporte del estándar Xquery. Los sistemas más novedosos tales como CloudTPS también soportan unión de queries. [[11](#ElP10)]

1. MARCO CONCEPTUAL
2. SandBox

El sandbox protege "en tiempo real" los servidores de datos, y hace de control preventivo de la ejecución de código fuente, datos y/o contenido, evitando unos cambios que podrían ser perjudiciales (independientemente de la intención del autor de los mismos) para un sistema, o que simplemente, podrían ser cambios de difícil reversión. [[1](#Txo07)]

1. PHP (Pre – procesador de Hipertexto)

Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. [[12](#Mav02)]

1. Python

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. [[13](#Llu08)]

1. .NET

La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. [[14](#20005)]

1. JSP – JavaServer Pages

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta. [[15](#Jor04)]

1. PostgreSQL

Es un sistema de gestión de base de datos relacional orientado a objetos y libre. [[16](#Pil04)]

1. Mysql

Es un sistema de gestión de bases de datos relacionales, multihilo y multiusuario [[17](#Spe03)]

1. MongoDB

Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto. MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. [[18](#Vic12)]

1. Git

Es un software de control de versiones, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. [[19](#Par07)]

1. Tomcat

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. [[20](#Ser05)]

1. MARCO LEGAL

* El software utilizado para este proyecto se encuentra enmarcado bajo los términos de la “GNU General Public License” publicada por la “Free Software Foundation”. [[21](#Gan111)]

Esta licencia se denomina GNU Copyleft (liberación de derechos de GNU, un juego de palabras con el termino ingles Copyright). [[21](#Gan111)]

Cubre todos los programas producidos por GNU un acrónimo compuesto también de las palabras: GNU no es Unix y por la Free Software Foundation (Fundación de Software Gratuito). Esta licencia permite a los desarrolladores crear programas para el público en general. [[21](#Gan111)]

* Ya que el sistema es un prestador de servicios todos sus usuario estarán regidos por la ley 1273 de 2009 estando está relacionada con los delitos informáticos y la protección de la información y de los datos, la cual castiga con penas de prisión de hasta 120 meses y multas de hasta 1500 salarios mínimos legales mensuales vigentes. [[21](#Gan111)]
* Tomando en cuenta que el sistema está orientado principalmente a estudiantes de la UFPS, en el caso de conductas que desmeriten el nombre de la institución estas será castigadas por la institución bajo las normas y políticas amparadas por el estatuto estudiantil de dicha institución. [[21](#Gan111)]

# DISEÑO METODOLOGICO

1. TIPO DE INVESTIGACIÓN

Nuestra investigación se basa en encontrar herramientas o módulos que nos sirvan para desarrollar aplicaciones que estén orientadas a la administración del servidor Sandboxya que este como se ha explicado anteriormente había presentado una serie de dificultades a la hora de ejecutar su plataforma.

Este motivo es el que hace que nuestra investigación se desarrolle en un tipo de investigación aplicada ya que nuestro resultado dará una posible solución a esta problemática; así mismo se puede observar que la parte de análisis de esta misma se llevara a cabo bajo una filosofía investigativa tipo documental, esto hace que nuestra investigación sea de tipo documental y aplicada.

1. FUENTES DE INFORMACIÓN

* **Fuentes de información primario:** esta consiste en la recopilación de la experiencia de los administradores del sistema y así mismo utilizarla para evitar posibles inconvenientes pasados.

También tenemos que recopilar información de los usuarios que han ejecutado este servicio, como los estudiantes del Plan de Estudio de Ingeniería de Sistemas de la UFPS.

* **Fuentes de información secundaría:** para este ítem nos centramos en la utilización de la documentación ya existente para las aplicaciones a utilizar y en otras que se vayan explorando durante el proceso de investigación.

1. TÉCNICAS Y PROCEDIMIENTO PARA LA RECOLECCIÓN DE INFORMACIÓN
2. Técnicas

El tipo de técnica para realizar la recolección de los datos es de Observación ordinaria y/o participante, ya que nuestro medio de investigación es mucho más cualitativa participativa que cuantitativa.

1. Método

En nuestra investigación nos centramos en realizar una observación de los procesos que se ejecutan en el servidor para poder entender mejor como enfocarlos en una aplicación web.

1. Procedimiento

A continuación se realizara un listado de tareas para ejecutar esta investigación:

1. Conocer las actividades diseñadas para la ejecución del servidor.
2. Familiarizarse con las aplicaciones instaladas en el servidor para su administración.
3. Identificar los posibles procesos problemáticos en la administración del servidor.
4. Integrar las nuevas tecnologías necesarias para el correcto funcionamiento del servidor.
5. Mejorar las aplicaciones ya existentes en el servidor, tanto funcionalmente como visualmente.
6. Investigar sobre los componentes de FTP y administración de War ya existentes orientados a la administración de servidores.
7. Analizar las nuevas tecnologías integradas a los servidores.
8. Basándonos en el análisis y diseño de software, crear una serie de modelos de datos y objetos que nos permitan desarrollar una herramienta web acorde a nuestras necesidades.
9. Reunir los Product Owner, colocarlos en Product Backlog para diseñar los Sprint, todo esto según el ciclo de vida SCRUM.
10. Ejecutar los Sprints necesarios.
11. Diseñar un plan de pruebas para las aplicaciones en el servidor, orientado a medir su tasa de efectividad y la identificación de posibles errores con SCRUM.
12. Documentar los procesos de instalación y corrección de errores del servidor.
13. Crear un manual de usuarios para la utilización de las aplicaciones del servidor.

# CRONOGRAMA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Objetivos  específicos | actividades | Meses y semanas | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Mes 1 | | | | Mes 2 | | | | Mes 3 | | | | Mes 4 | | | | Mes 5 | | | | Mes 6 | | | | Mes 7 | | | | Mes 8 | | | |
| S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 | S1 | S2 | S3 | S4 |
| 1 | a1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | a4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | a8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a  10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | a 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a  12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| a  13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Tabla 1. Cronograma de actividades.

# PRESUPUESTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PRESUPUESTO GLOBAL DEL TRABAJO DE GRADO** | | | | |
| **RUBROS** | **FUNETE DE FINANCIACION** | | | **TOTAL** |
|  | **Estudiante** | **UFPS** | **Externa** |  |
| Personal |  | $ 2’080.000 |  | $ 2’080.000 |
| Materiales | $ 127.500 |  |  | $ 127.500 |
| Equipos | $ 800.000 | $3’400.000 |  | $ 4’200.000 |
| **Total** | $ 927.500 | $ 5’480.000 |  | $ 6’307.500 |

Tabla 2. Presupuesto global del trabajo de grado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DESCRIPCIÓN DE LOS GASTOS DE PERSONAL** | | | | | | |
| **Nombre de estudiantes** | **Función en el trabajo** | **Dedicación Hora / Semana** | **FUENTE DE FINANCIACION** | | | **TOTAL** |
| **Estudiante** | **UFPS** | **Externa** |
| Fernando José Torres Bermúdez | Administrador del servidor Sandbox | 40 Horas |  | $ 2’080.000 |  | $ 2’080.000 |
| **Total** |  |  |  | $ 2’080.000 |  | $ 2’080.000 |

Tabla 3. Descripción de los gastos de personal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DESCRIPCION DE MATERIALES Y SUMINISTROS** | | | | |
| **MATERIAL** | **FUENTE DE FINANCIACION** | | | **TOTAL** |
| **Estudiante** | **UFPS** | **Externa** |
| Resma tam. carta | $ 40.0000 |  |  | $ 40.000 |
| Tinta impresora | $ 80.000 |  |  | $ 80.000 |
| Carpetas de presentación | $ 7.500 |  |  | $ 7.500 |
| **Total** | $ 127.500 |  |  | $ 127.500 |

Tabla 4. Descripción de materiales y suministros.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DESCRIPCION DE EQUIPOS** | | | | |
| **DESCRIPCION DE EQUIPOS** | **FUENTE DE FINANCIACION** | | | **TOTAL** |
| **Estudiante** | **UFPS** | **Externa** |
| Mac Pro |  | $ 3’400.000 |  | $ 3’400.000 |
| Inspirion N4010 | $800.000 |  |  | $ 800.000 |
| **Total** | $800.000 | $ 3’400.000 |  | $ 4’200.000 |

Tabla 5. Descripción de equipos.

# RESULTADOS ESPERADOS

Al terminar el proyecto se espera obtener un software para la administración y gestión de servidores de aplicación en Php, JSP, Python y .NET, que cumpla con las buenas prácticas en el desarrollo de software y así mismo brindar la mejor experiencia a sus usuarios tomando en cuenta los siguientes puntos:

1. Que su GUI sea funcional y amigable.
2. Disponibilidad de herramientas para la ejecución de proyectos de software.
3. Fácil administración y mantenimiento.

Estos puntos resumen el objetivo de este proyecto de grado el cual entregara una aplicación tipo CPANEL para facilitar la entrega de proyectos de software del plan de estudios de ingeniería de software.

BIBLIOGRAFIA

x

|  |  |
| --- | --- |
| [1] | Txon. (2007, June) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Sandboxie> |
| [2] | Muro Bot. (2009, Feb.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Google_Sandbox> |
| [3] | Universidad de los Andes. (2013, Jan.) Universidad de los Andes. [Online]. <http://admonsis.uniandes.edu.co/isclabs/index.php?option=com_content&view=article&id=60&Itemid=18> |
| [4] | Vicente José Castaño Díaz, "Caraterización de los servidores Web de ámbito Académico," in *Caraterización de los servidores Web de ámbito Académico*, Ana  Pont  Sanjuan and José Antonio Gil Salinas, Eds. Valencia, España: Universidad Politecnica, 2008, ch. 11, p. 62. |
| [5] | Danieltellez. (2007, Oct.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software> |
| [6] | Ascánder. (2004, Nov.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Control_de_versiones> |
| [7] | Tei. (2002, Feb.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Servidor> |
| [8] | cm05840.telecable.es. (2002, Mar.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/File_Transfer_Protocol> |
| [9] | Superzerocool. (2006, Mar.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/CPanel> |
| [10] | Str0b0s. (2005, Apr.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Base_de_datos_relacional> |
| [11] | El pantera. (2010, Mar.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/NoSQL> |
| [12] | Maveric149. (2002, Oct.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/PHP> |
| [13] | Lluvia. (2008, Feb.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Python> |
| [14] | 200.92.213.173. (2005, Apr.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Microsoft_.NET> |
| [15] | JorgeGG. (2004, Mar.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/JavaServer_Pages> |
| [16] | Pilaf. (2004, May) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/PostgreSQL> |
| [17] | SpeedyGonzalez. (2003, Oct.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/MySQL> |
| [18] | Victormigh. (2012, Mar.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/MongoDB> |
| [19] | Paradoja. (2007, Oct.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Git> |
| [20] | Sergio Vares. (2005, Feb.) Wikipedia. [Online]. <http://es.wikipedia.org/wiki/Tomcat> |
| [21] | Isabella Gandini, Andres Isaza, and Alejandro Delgado. (2011, Oct.) Delta Asesores. [Online]. <http://www.deltaasesores.com/articulos/autores-invitados/otros/3576-ley-de-delitos-informaticos-en-colombia> |

x