



Universidad Autónoma de Campeche

Facultad de Ingeniería

Semestre:7 Grupo: "A"

Proyecto: Diseño e Implementación del Sistema de
Convenios de la Facultad de Ingeniería

Alumnos:

Nombre:

Matrícula:

Cesar Alberto Zapata Leon.	53968
Israel Isaías Moo Chable.	53969
Oscar Fabian Romero Hernandez.	46809
Jeanette Guadalupe Che Trejo.	46969
Sarahi Guadalupe Piña Pacheco.	47449
Manuel Alejandro Mijangos Ortiz.	44432

Fecha de Entrega: 27/11/2018

Introducción	2
Objetivo General	3
Objetivos Específicos	3
Requerimientos de Usuario	4
Requerimientos Funcionales	4
Requerimientos no Funcionales	4
Historias de usuarios	5
Modelo de Arquitectura de la Aplicación.	11
Modo de uso	12
Modelo	12
Vista	12
Controlador	12
Arquitectura Cliente-Servidor	13
Diseño de la Interfaz	14
Descripción de Componentes de la Aplicación.	15
Lenguajes de Programación Utilizados	16
Convenciones de Codificación	17
Nombres	17
Corchetes e indentación	17
Herramientas para la Construcción	18
Prácticas para la Administración de la Configuración	19
Proceso de control de cambios	19
Herramientas para el control de versiones:	19
Git y Github.	19
Procedimiento de Integración	21
Estrategias Para Sistematizar las Pruebas del Proyecto	23
Plan de Pruebas Funcionales	23
Pruebas funcionales	24
Informe de Ejecución de Pruebas Funcionales	27
Referencias	27

Introducción

En la Universidad Autónoma de Campeche, actualmente no existe un sistema de información que realice el seguimiento a los productos emanados de convenios celebrados con diversas instituciones, esto reduce que se promuevan mecanismos que impulsen los beneficios de los mismos y a los programas educativos.

Al no existir un monitoreo de los convenios, surgen las siguientes problemáticas:

- Al celebrarse convenios con instituciones interesadas en generar trabajos conjuntos o proyectos de investigación sin que se concreten, se convierten sólo en una “carta de intención” que termina no impactando favorablemente a ningún programa educativo.
- Por otro lado, hay convenios que generan productos académicos de gran valía e impacto para indicadores de diversos programas educativos, sin que exista un registro electrónico, que facilite su acceso.
- El seguimiento de los convenios se realiza de manera manual y documental, lo que representa un cúmulo de trabajo y papeles que hacen complejo el monitoreo y cruzamiento de información.

Para solucionar las problemáticas anteriores, se propone el desarrollo de una aplicación web para automatizar y optimizar el proceso de seguimiento de indicadores, facilitará el monitoreo y cruce de información, así como la generación expedita de estadísticas que aporte información para la toma de decisiones.

El presente trabajo establece el uso de una metodología de desarrollo de software para la elaboración de la aplicación web llamada SISCONVE proporcionando métodos, herramientas y técnicas a utilizar.

Objetivo General

El presente trabajo propone el desarrollo de un sitio web que permite el seguimiento de los productos académicos, derivados de los convenios de vinculación de la Universidad Autónoma de Campeche con otras instituciones generando indicadores que miden el impacto de los convenios con los diversos programas educativos.

Objetivos Específicos

- Desarrollar una aplicación web que permite el seguimiento de los productos académicos, derivados de los convenios de vinculación de la Universidad Autónoma de Campeche con otras instituciones.
- Generar indicadores que permitan medir el impacto de los convenios con los diversos programas educativos.
- Automatizar el proceso de registro y seguimiento de los convenios de vinculación de la Facultad de Ingeniería
- Desarrollar la aplicación Web que permite consultar desde cualquier navegador o dispositivo información sobre los convenios de vinculación, de acuerdo a las diferentes credenciales de acceso.
- Generar instrumentos tecnológicos que permitan medir la vinculación de la Facultad de Ingeniería con la sociedad.

Requerimientos de Usuario

Comprender la naturaleza de los problemas puede resultar difícil, especialmente si son nuevos, tras tener conocimiento de ésto, se establecieron los requerimientos para el desarrollo del software fomentando el proceso de analizar, descubrir, documentar y verificar estos servicios en un sistema propuesto, esbozando su comportamiento externo, dividiendo sus características en funcionales y no funcionales. (Sommerville, Sawyer, & Viller, 1999). Se determinaron los dos tipos de requerimientos con relación a nuestro proyecto:

Requerimientos Funcionales

El Sitio web SISCONVE es capaz de generar múltiples indicadores a fin de realizar el seguimiento de los productos resultantes de la relación de convenios con otras facultades de forma automatizada y optimizable.

Requerimientos no Funcionales

Para nuestro sistema se proponen siete atributos de calidad como requerimientos No funcionales: (Bass, Clements, & Kazman, 2012).

- **Confidencialidad**, protegiendo accesos no autorizados y divulgación de información.
- **Integridad**, protegiendo la información de corrupción e inconsistencias.
- **Disponibilidad**, garantizando el acceso a la información a los usuarios autorizados.
- **Escalabilidad**, adaptándose a las necesidades de los usuarios, prestando servicio adecuadamente, empleando un crecimiento continuo y fluido sin perder calidad en los servicios ofrecidos.
- **Usabilidad**, siendo intuitivo, utilizando una interfaz sencilla y atractiva para el usuario.
- **Soportabilidad**, facilitando el acceso a la información sobre el manejo, mantenimiento y actualización del software
- **Compatibilidad**, siendo operable en todos los sistemas operativos del mercado, así como en los navegadores existentes.

Historias de usuarios

Las historias de usuario funcionan para identificar, definir y planificar las especificaciones de requisitos en el desarrollo de un proyecto, se consideran instrumentos para el levantamiento de requerimientos de un software que ha surgido con la aparición de los nuevos marcos de trabajo de desarrollo ágil.

Las historias de usuario abarcan una serie de propiedades concretas que deben cumplir para su ejecución, estas propiedades se les conoce con las siglas INVEST según Bill Wake en su escrito Programming Explored and Refactoring.

- Independent
- Negotiable
- Valuable to users or customers
- Estimatable
- Small
- Testable

Los números empleados para medir el esfuerzo de cada historia de usuario fue basado en la serie de fibonacci, Se eligió esta serie para asegurar que las estimaciones sean más certeras.

Historia de Usuario	
Número: 1	Nombre: Graficas
Usuario: Vinculación Facultad	
Prioridad de implementación: Medio	Valor: 50
	Esfuerzo: 13
Descripción: Como Vinculación Facultativa Necesito poder generar a nivel gráfico las estadísticas de los productos generados de la facultad correspondiente Para facilitar la comprensión, comparación y análisis de la gestión de registros.	

Historia de Usuario	
Número: 2	Nombre: Evento
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80

	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Eventos Académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 3	Nombre: Patente
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Patentes únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 4	Nombre: Obras Artísticas
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar los diferentes modelos de productos de las Obras Artísticas Académicas únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 5	Nombre: Memorias Arbitrarias
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Memorias Arbitrarias académica únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 6	Nombre: Ponencia

Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Ponencias únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 7	Nombre: Visualización
Usuario: Vinculación Facultad	
Prioridad de implementación: Medio	Valor: 60
	Esfuerzo: 8
Descripción: Como Vinculación Facultativa Necesito Visualizar de manera única la gestión de registros únicamente de la facultad correspondiente Para cuidar la privacidad y seguridad de los datos en pantalla.	

Historia de Usuario	
Número: 8	Nombre: Reportes
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 8
Descripción: Como Vinculación Facultad Necesito poder generar reportes donde se pueda especificar el tipo de convenio, tipo de facultad, tipo de producto, fuente de financiamiento y la institución participante de cada uno de los productos Para producir reportes de cada producto de manera concisa y ordenada.	

Historia de Usuario	
Número: 9	Nombre: Proyecto
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Proyectos Académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 10	Nombre: Tesis
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Tesis Académicas únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 11	Nombre: Libro
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Libros Académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 12	Nombre: Prototipo
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de los prototipos académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 13	Nombre: Estancia de Investigación
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13

Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Estancia de Investigación únicamente de mi facultad Para poseer un control de los datos anexados en el software.

Historia de Usuario	
Número: 14	Nombre: Artículo
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Artículos Académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 15	Nombre: Capacitación
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Capacitación únicamente de mi Facultad Para poseer un control de los datos anexados en el software.	

Historia de Usuario	
Número: 16	Nombre: Movilidad
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de la Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de movilidad únicamente de mi Facultad Para poseer un control de los datos anexados en el software de acuerdo al programa educativo vigente.	

Historia de Usuario	
Número: 17	Nombre: Permisos

Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación Facultativa Necesito obtener permisos especiales para poder agregar, eliminar, editar y visualizar los datos registrados de la facultad correspondiente anteriormente iniciada Para gestionar los datos de manera correcta.	

Historia de Usuario	
Número: 18	Nombre: Informe Técnico
Usuario: Vinculación Facultad	
Prioridad de implementación: Alto	Valor: 80
	Esfuerzo: 13
Descripción: Como Vinculación de Facultad Necesito visualizar,editar, eliminar y agregar los diferentes modelos de productos de Informes Técnicos Académicos únicamente de mi facultad Para poseer un control de los datos anexados en el software.	

Modelo de Arquitectura de la Aplicación.

El elección del modelo de la Arquitectura se eligió de acuerdo a la funcionalidad, objetivo y adaptabilidad a nuestras necesidades del desarrollo de nuestro proyecto, integrando distintas herramientas y aplicaciones de acuerdo a la planificación hecha para su elaboración

La arquitectura con la mejor adaptabilidad en nuestros cambios de procesos en el desarrollo de nuestro proyecto, fué el patrón de diseño Modelo Vista Controlador. La arquitectura del sistema MVC (Modelo, Vista, Controlador) maneja tres archivos o scripts que interactúan entre sí para el correcto funcionamiento de las capas.

Este diseño se implementa cuando se hace uso de un gran manejo de información de datos y transacciones complejas, requiere una mejor separación de conceptos para su estructura. El patrón de diseño Modelo Vista Controlador, posee tres secciones las cuales se manejan paralelamente e independientes una de otra. Consiste en el modelo, el servidor y los clientes. Las tareas se reparten entre los proveedores de servicios (servidores), los clientes (demandantes) hacen la solicitud de servicios del servidor y este los proporciona de acuerdo a las necesidades del cliente.

Ventajas	Desventajas
<ul style="list-style-type: none"> • Separación del Modelo de la Vista 	<ul style="list-style-type: none"> • La separación de conceptos en capas agrega complejidad al sistema
<ul style="list-style-type: none"> • Capacidad de agregar múltiples representación de los mismos datos o información de manera sencilla. 	<ul style="list-style-type: none"> • La cantidad de archivos para mantener y desarrollar es extensa
<ul style="list-style-type: none"> • Facilidad de agregar nuevos tipos de datos según se requiera, ya que estas no dependen del funcionamiento de las capas. 	<ul style="list-style-type: none"> • El patrón de diseño es más complejo que usando otro modelos más sencillos
<ul style="list-style-type: none"> • Fácil mantenimiento. 	
<ul style="list-style-type: none"> • Independencia de funcionamiento. 	

Modo de uso

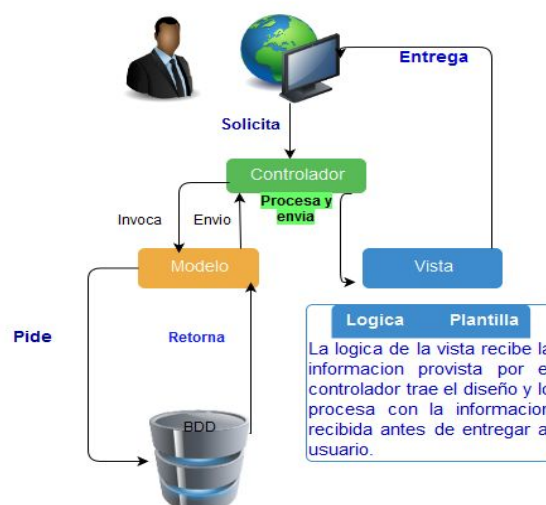


Figura No. 1. Arquitectura del sistema

Modelo

En esta capa el framework actúa directamente en la base de datos, haciendo las sentencias SQL más dinámicas, ya que cada sentencia de consulta SQL se encuentran definidas en clases propias de Codeigniter haciendo que tareas de codificación como consultar, insertar, recuperar, y modificar información sean más dinámicas y solo se necesite pocas líneas de código para su desempeño a continuación explicaremos la estructura y codificación de esta capa.

Vista

Esta es la capa donde interactúa el usuario, es la parte visual del sitio web aquí tenemos los archivos de JSP que se ejecutan conjuntamente haciendo llamados a los scripts de HTML, JavaScript, CSS, etc. Es decir todo el diseño básicamente lo encontramos en esta capa, a continuación en la siguiente tabla vamos a explicar el funcionamiento de las vistas.

Controlador

El controlador hace que interactúe la capa de la vista conjuntamente con la capa del modelo, haciendo las peticiones de envío y recepción de datos mediante el método post o haciendo el llamado a varias funciones incluidas en las clases del controlador, generando o devolviendo la lógica del negocio por medio de la vista, a continuación se describen en la siguiente tabla los scripts que intervienen en esta etapa de codificación

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de aplicaciones distribuida, la capacidad de proceso está repartida entre los clientes y los servidores, proporciona una organización y centralización en la gestión de información separando las responsabilidades, facilitando y clarificando el diseño del sistema. Esta arquitectura consta de 2 capas:

- Front/end (interactúa con el usuario), se basa en una interfaz gráfica con el usuario. Inicia solicitudes o peticiones, posee un papel activo en la comunicación, este espera y recibe la respuestas del servidor o servidores. Interactúan directamente con los usuarios finales mediante una interfaz gráfica de usuario.
- Back/end : esta capa reside en la Base de Datos, por lo tanto no es interactiva con los usuarios. El receptor de las solicitudes enviadas por el cliente se conoce como servidor, su papel es pasivo en la comunicación, después de la recepción de la solicitud, la procesan y luego envían las respuestas al cliente.

En la arquitectura en tres niveles existe un nivel intermedio. Esto significa que la arquitectura generalmente está compartida por:

- Un cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador web) para la presentación.
- El servidor de aplicaciones (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo.
- El servidor de datos, que proporciona al servidor de aplicaciones los datos que éste le solicitó.

Una arquitectura cliente-servidor es aquella con un servidor polivalente, es decir, puede responder directamente a todas las solicitudes de recursos del cliente. Sin embargo, en la arquitectura en tres niveles las aplicaciones al nivel del servidor son descentralizadas de uno a otro, es decir, cada servidor se especializa en una determinada tarea, (por ejemplo, servidor web/servidor de bases de datos).

La arquitectura en tres niveles permite:

- Un mayor grado de flexibilidad.
- Mayor seguridad, ya que la seguridad se puede definir independientemente para cada servicio y en cada nivel.
- Mejor rendimiento, ya que las tareas se comparten entre servidores.

Ventajas	
Centralización de control	Los accesos, recursos y la integralidad de los datos son controlados por el servidor, facilita su mantenimiento.
Escalabilidad	Capacidad de aumentar clientes y servidores por separado.(nodos a la red de clientes y/o servidores).
Tecnología	Aseguramiento de transacciones, facilidad de empleo y una interfaz dinámica para el usuario.
Fácil mantenimiento	Fácil reparación, reemplazos, actualizaciones e incluso traslación de un servidor a otro sin tener efectos negativos hacia los clientes(encapsulamiento).

Desventajas	
Saturación	La congestión de tráfico debido a una gran cantidad de clientes envían peticiones simultáneas al mismo servidor, lo cual requiere una gran cantidad de nodos para una mejor ancho de banda.
	Cuando el servidor está caído las peticiones de los clientes no pueden ser satisfechas.

--	--

Diseño de la Interfaz

Este diseño permite que, los requisitos levantados del sistema, se plasmen en una vista preliminar para determinar cómo está conformado el sistema, tener un punto de partida en la realización de un buen diseño que permita a los usuarios finales navegar de forma dinámica y amigable.

Descripción de Componentes de la Aplicación.

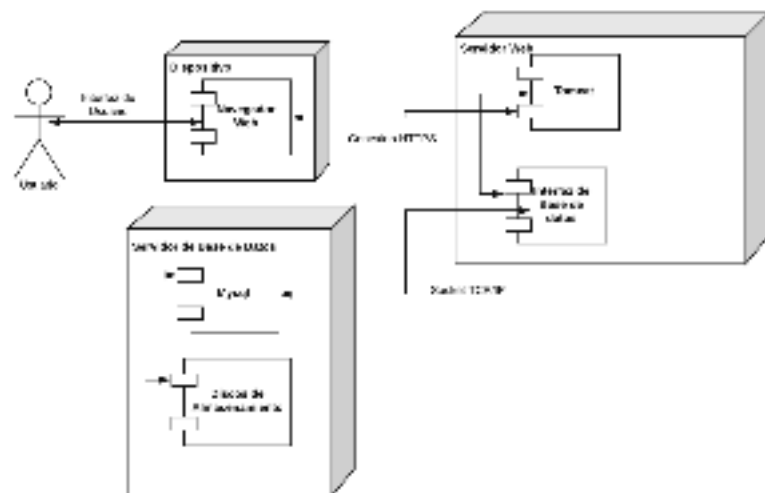


Figura No. 2. Arquitectura del sistema

En las interfaces de usuario se implementa la Interfaz Gráfica de Usuario (GUI) para que pueda ser fácilmente manipulada por el usuario haciendo que todos los elementos puedan identificarse de forma sencilla y su manejo resulte más intuitivo.

En el caso de las interfaces de hardware los usuarios pueden interactuar con el sistema por medio del teclado, raton y pantalla.

La aplicación web tiene tres componentes principales: un navegador web (o cliente), un servidor de aplicaciones web y un servidor de bases de datos.

Los clientes manejan la lógica de presentación, que controla el modo en que los usuarios interactúan con la aplicación. En el caso de nuestra aplicación se implementarían en los principales navegadores web como Google Chrome, Mozilla Firefox, Opera y Safari.

En el servidor web existe una interfaz de base de datos así como también usamos Tomcat debido al soporte de JSPs.

En la parte de servidor de base de datos donde se guardarán donde se almacenaran, recuperaran y administrarán los datos de la base de datos se emplea el gestor de base de datos relacional MySQL.

Lenguajes de Programación Utilizados

El equipo de desarrollo trabaja en los siguientes lenguajes de programación:



Debido a la practicidad y conocimiento del equipo de desarrollo además de esta usando lenguajes comunes y conocidos en el mundo de la tecnología.

Convenciones de Codificación

Nombres

En este proyecto se usa en más general para asignar nombres es la convención CamelCase en sus dos variantes:



Debido a la practicidad, simplicidad y facilidad de la notación para el trabajo del equipo de desarrollo además de una comprensión más fácil para sus revisiones futuras.

Corchetes e indentación

La indentación es algo que ayuda a darle claridad a un programa y es **INDISPENSABLE** que se haga bien. Debe hacerse con "tabs" y no con espacios en blanco. Los corchetes de un bloque if, o switch, o for, deben ir en la misma línea de la cláusula. A continuación mostramos la forma apropiada de hacerlo.

Herramientas para la Construcción

Nuestro proyecto se trabaja con base en las siguientes herramientas para la construcción dividido en tres partes:



Estas tecnologías fueron seleccionadas por su versatilidad, confianza y simplicidad en las áreas que trabajan.

Prácticas para la Administración de la Configuración

Proceso de control de cambios

En algún caso que se necesite hacer algún cambio en la aplicación informa a todos los integrantes del equipo que tendrán que manejar dicho cambio y se realiza el siguiente proceso:

























- Se realiza una solicitud de cambio, ya sea por algún cambio en los requerimientos o por algún problema detectado.
- Se clasifica y registra la solicitud del cambio.
- Se aprueba o rechaza el cambio por los responsables de hacer dicha evaluación.
- Si ha sido aprobada se evalúan los posibles efectos, esfuerzo e impactos sobre otras funciones del sistema.
- Se realiza el cambio y genera un proceso de seguimiento y control.
- Cuando se finaliza el cambio, se verifica que se han satisfecho los requerimientos modificados o solucionado los problemas detectados.
- Se notifica y registra el cambio realizado.

Herramientas para el control de versiones:

Git y Github.

Se implementa Git y Github debido a que son muy buenos para trabajar de manera colaborativa ya que permiten leer y escribir directamente sobre el repositorio y además son gratis e ilimitados para proyectos públicos, se puede tener acceso al historial de cambios, ya sea para regresar a una de esas versiones o para hacer comparaciones entre ellas, no se necesitan hacer copias de seguridad a un repositorio o servidor para proteger el código que se

escribe, solo hay que instalar Git en un equipo y crear una cuenta en Github y estará listo para usarlo, además son compatibles con los diferentes sistemas operativos.

Commits on Nov 27, 2018	<div><div>v5.7.0.0.0.3 (Productos)</div><div> fabianrh97 committed 2 hours ago</div><div> 1bbdbfb </div></div>
Commits on Nov 25, 2018	<div><div>v5.7.0.0.0.2 (Add)</div><div> fabianrh97 committed 2 days ago</div><div> cbc48d2 </div></div> <div><div>v5.7.0.0.0.1 (Colores)</div><div> fabianrh97 committed 2 days ago</div><div> a7c71ab </div></div> <div><div>v5.7 (Descarga esto, isra)</div><div> fabianrh97 committed 2 days ago</div><div> 3a17d8c </div></div>
Commits on Nov 24, 2018	<div><div>v5.6.1(proceso maqueta)</div><div> israel97 committed 4 days ago</div><div> 69f44ab </div></div>
Commits on Nov 22, 2018	<div><div>v5.6 (Navbar Funcionando con nuevos cambios del query)</div><div> fabianrh97 committed 5 days ago</div><div> 61987cf </div></div> <div><div>v5.5.2(interfaz nueva)</div><div> israel97 committed 5 days ago</div><div> 4b2bdad </div></div>
Commits on Nov 15, 2018	<div><div>v5.5.1(base y querys nuevos)</div><div> israel97 committed 12 days ago</div><div> a421693 </div></div>

Procedimiento de Integración

El proyecto fue basado en la integración continua permitiendo encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software, el equipo conoce las fases por las que va pasando el código, y el estado del software en cada momento (si compila, si pasa las pruebas, en qué entorno está cada versión, qué versión se está probando etc).

Esto evita que trabajemos de manera aislada y todo el equipo de trabajo esté en una misma sintonía .

Para ello enviamos los cambios de forma periódica a un repositorio compartido con un sistema de control de versiones como Github, Antes de cada envío, los desarrolladores ejecutan pruebas de unidad local en el código como medida de verificación adicional antes de la integración

Durante la elaboración de este proyecto se ha empleado la metodología Scrum, se empleó la herramienta iceScrum para la gestión ágil de proyectos estableciendo las historias de usuario en el software.

Después de definir una historia de usuario, se procedio a ser aceptada donde si el propietario del producto lo reconoce como una idea valiosa para el proyecto, seguidamente todas las historias aceptadas se trasladaban a la pila del producto

Luego las historias fueron ordenadas y priorizadas por el propietario del producto de acuerdo a sus necesidades y complejidad de cada actividad,luego el equipo estimó el coste aproximado de desarrollo de cada historia. Se hizo la planificación que nos llevaría a poder terminar el producto en la fecha estipulada



Por cada historia de usuario se definieron tareas, donde miembros del equipo tomaban tareas , que pasaban de estar en “pendiente” a estar en “progreso”, una vez terminadas la tareas, pasan a estar como “hechas”.

Cada día se hizo una breve reunión, donde nos preguntamos: ¿Qué se hizo ayer para ayudar a terminar el proyecto?, ¿Qué se hará mañana para ayudar a terminar el proyecto?, y ¿Qué obstáculos nos impiden lograr nuestras metas ?, estas reuniones fueron de gran ayuda.

Estrategias Para Sistematizar las Pruebas del Proyecto

- Analizar los requerimientos de desarrollo de software

Donde se tuvieron que entender los requerimientos de usuario que componen el proyecto.

- Identificar las nuevas funcionalidades a probar

A partir de lo que se haya generado en el análisis de requisitos y de las reuniones con el equipo incluye la lista de funcionalidades o características nuevas

- Identificar las funcionalidades de sistemas existentes que deben probarse

Donde se identifican las funcionalidades que ya existen que son afectadas por el desarrollo, donde se podrían encontrar Funcionalidades modificadas de cara al usuario, por ejemplo si una funcionalidad está siendo modificada agregando más pantallas o cambios a su flujo de proceso, o Funcionalidades modificadas en sus componentes internos que son funcionalidades no modificadas de cara al usuario, manteniendo la misma interfaz gráfica y flujo de procesos, sin embargo, si se modifican componentes internos que comparten con otras funcionalidades del sistema, en las capas de lógica de negocio o acceso a datos.

- Definir la estrategia de pruebas.

En nuestro caso, pruebas de caja negra

- Elaborar la planificación de las pruebas

Donde Las tareas del plan de pruebas deben estar alineadas con las habilidades y conocimientos de cada persona.

- Identificar los riesgos y definir planes de respuesta

.

Plan de Pruebas Funcionales

Caso Prueba: Agregar Artículo	
Número: 2	Fecha: 26/11/2018
Descripción: Funcionalidad del Botón “Agregar”	
Datos de Entrada: Convenios Carrera Nombre artículo Año de publicación ISBN Participante interno Participante externo Autores	Funcionalidad/Característica: Submit
	Estatus:
Resultado esperado: Que al oprimir el botón “Agregar” el Formulario sea procesado y almacenado en tablas visibles de todos los productos	

Caso Prueba: Llenado Incorrecto del Formulario	
Número: 2.1	Fecha:26/11/18
Descripción: Reacción del sistema ante un llenado incorrecto del formulario	
Datos de Entrada: Convenios Carrera Nombre artículo Año de publicación ISBN Participante interno Participante externo Autores	Funcionalidad/Característica: Form
	Estatus:
Resultado esperado: Que ante un llenado incorrecto, el sistema orille al usuario a corregir la información ingresada.	

Caso Prueba: Editar Artículo	
Número: 3	Fecha:26/11/2018
Descripción: Funcionalidad del botón “Editar”	
Datos de Entrada: Convenios Carrera Nombre artículo Año de publicación ISBN Participante interno Participante externo Autores	Funcionalidad/Característica: Submit
	Estatus:
Resultado esperado: Que al oprimir el botón “Editar” el Formulario sea procesado y despliegue un apartado de edición de artículo	

Caso Prueba: Eliminar Artículo	
Número: 4	Fecha:26/11/2018
Descripción: Funcionalidad del botón “Eliminar”	
Datos de Entrada:	Funcionalidad/Característica: Submit
	Estatus:
Resultado esperado: Que al oprimir el botón “Eliminar” el Artículo seleccionado sea eliminado de las tablas visuales	

Informe de Ejecución de Pruebas Funcionales

Cuando ejecutamos proyectos de desarrollo de software, la fase de pruebas (Software Testing) suele ser crítica, y es un momento en el cual diversos interesados (stakeholders) requieren información al minuto sobre el estado de la calidad del software que se está desarrollando.

Fecha comienzo planificada	% avance planificado	Casos planificados	Casos de prueba (Total)
Diciembre 2017	90%	68	51
Fecha de finalización planificada	% avance real	Casos con incidencia	Casos exitosos
Septiembre 2019	10%	48	3
Fecha fin pronóstico	% desviación		
20 Diciembre 2018	80%		

Situación actual de casos de prueba			
Exitosos	Con defectos	Bloqueados	Pendientes
3	2	0	39

Situación actual de defectos				
Reportados	En análisis	Descartados	En proceso	Corregidos
0	2	0	2	0

Resultados de la jornada

Casos del día	Meta diaria
12	6

Referencias

- ❖ Len Bass, Paul Clements, Rick Kazman. (1998). Software architecture in practice. Reading, Mass. Addison-Wesley Professional.
- ❖ Sommerville, I., Sawyer, P., & Viller, S. (1999). Managing process inconsistency using viewpoints. IEEE Transactions on Software Engineering, 25(6), 784-799.
- ❖ pmoinformatica.com . (2015). Introducción al Testing de Software para Principiantes . lunes, 22 de junio de 2015, de Amazon Associates LLC Sitio web: <http://www.pmoinformatica.com/2015/06/modelo-informe-pruebas-software.html>