****

**SISTEMA DE** **ORGANIZACIÓN AUTOMATIZADO**

**MEMORIA DE ESTADÍA PROFESIONAL**

**R E P O R T E T É C N I C O**

PARA OBTENER EL TÍTULO DE

**INGENIERIA EN GESTION Y DESARROLLO DE SOFTWARE**

**P R E S E N T A**

**BRANDON JESUS DIAZ VEGA**

**MAGDIEL ITAMAR PÉREZ HERNÁNDEZ**

**FRANCISCO ALEJANDRO RUIZ MARTINEZ**

**RAFAEL ZAMORA ROSALES**

MATERIA: “DESARROLLO WEB INTEGRAL”  
GENERACIÓN: ENERO 2023 – DICIEMBRE 2024

CUATRIMESTRE DE TÉRMINO: MAYO-AGOSTO 2024

**ÍNDICE**

[RESUMEN 4](#_Toc151676381)

[INTRODUCCIÓN. 6](#_Toc151676382)

[MISIÓN 7](#_Toc151676383)

[VISIÓN 7](#_Toc151676384)

[PROPÓSITO 7](#_Toc151676385)

[OBJETIVOS 9](#_Toc151676386)

[MARCO TEORICO 10](#_Toc151676387)

[CAPÍTULO 1 PROCESO DE DESARROLLO WEB 13](#_Toc151676388)

[1.1 Plan del proceso de desarrollo *web* 13](#_Toc151676389)

[1.2 Arquitecturas de Software 15](#_Toc151676390)

[1.2.1 Diagrama de Arquitectura. 17](#_Toc151676391)

[1.3 Pruebas de Desarrollo 21](#_Toc151676392)

[CAPITULO 2 CONTROL DE VERSIONES E INTEGRACION DE COMPONENTES 23](#_Toc151676393)

[2.1 Versionamientos de software 23](#_Toc151676394)

[2.2 Sistema de control de versiones 24](#_Toc151676395)

[2.3 Herramientas de razonamiento 25](#_Toc151676396)

[2.4 Especificación de principios de código 27](#_Toc151676397)

[2.4.1 Principios de Codificación Segura 29](#_Toc151676398)

[Certificado de Seguridad Web – Sistema de almacenamiento web 32](#_Toc151676399)

[CAPITULO 3 PRUEBAS Y LIBERACION DEL DESARROLLO WEB 34](#_Toc151676400)

[3.1 Despliegue de servicios 34](#_Toc151676401)

[3.2 Herramientas de gestión de contenedores 35](#_Toc151676402)

[3.3 Pruebas de software 37](#_Toc151676403)

[3.3.1 Técnica de caja negra 37](#_Toc151676404)

[3.4 Liberación del software 39](#_Toc151676405)

[3.4.1 Políticas y normativas aplicables 39](#_Toc151676406)

[3.4.2 Herramientas de liberación de software 41](#_Toc151676407)

[CONCLUSIONES 43](#_Toc151676408)

[LISTADO DE SIGLAS O ACRÓNIMOS 44](#_Toc151676409)

[GLOSARIO 45](#_Toc151676410)

[Referencias 46](#_Toc151676411)

# RESUMEN

El presente proyecto consiste en diseñar un Sistema de Organización Automatizado para la gestión y operatividad de los Convenios Generales, Marco y Específico que se gestionan en la Secretaría de Vinculación con apoyo de las diferentes Divisiones Académicas, mismos que son de interés tanto para personal directivo y administrativo de la Universidad Tecnológica de Tecámac.

El diseño de este proyecto será realizado con base a la problemática que hay dentro de la Secretaría de Vinculación, por tanto, dentro del mismo no hay un sistema actualizado que pueda organizar de forma específica, por fecha, hora, lugar, etc., cada uno de los convenios, aceptados, rechazados o que sigan en espera de una respuesta. Con base al control y consulta de convenios el sistema establecido por la institución no es muy clara por tanto la información dentro del sistema que maneja la Secretaría de Vinculación no es oportuna y de esta forma la atención que proporcionan las empresas es muy limitada. Por esta razón, se implementará un sistema de organización automatizado.

Al igual que una forma más específica para ingresar la información de nuevos convenios o incluso para modificar o actualizar la información de algunos que ya se encuentran en disposición por parte de las empresas.

Para la elaboración de este proyecto será fundamental identificar la información actualizada del número de convenios vigentes, el sector al que pertenecen, en que programas educativos están operando y cuántos alumnos realizas sus estadías y o Educación Dual. Este proyecto muestra los pasos a seguir para recolectar, organizar y construir el sistema utilizando *Laravel*, *Composer*, *React* y un gestor de base de Datos.

Para programar, como método extra, se aplicó el uso de formularios. Un formulario brinda una forma conveniente de escribir o mostrar una fila completa de información en un intervalo o una tabla. Se podrá observar que la entrada de datos será más sencilla con el uso de cuadro de listas de uno o más elementos de texto entre los cuales puede elegir el usuario, permitiendo solamente una elección. Imagen con botón este permite realizar una acción cuando un usuario hace clic sobre la imagen. Botones de comando también se conocen como botón de inserción, es un objeto pequeño y rectangular que se usa para realizar una acción. Por ejemplo, los botones de registro, actualizar, eliminar y cancelar que se puede configurar, filtrar datos o calcular números, ejecutando un procedimiento cuando el usuario hace clic en él.

# INTRODUCCIÓN.

La Universidad Tecnológica de Tecámac, un organismo público descentralizado del Gobierno del Estado de México, con el objetivo de formar Técnicos Superiores y Profesionales Universitarios, aptos para el desarrollo de conocimientos aplicados a la solución creativa de los problemas del sector productivo y a los requerimientos del crecimiento económico y social. La Universidad Tecnológica de Tecámac se constituye en una Institución de Educación Superior con carreras profesionales innovadoras, de alto nivel de preparación teórico-práctico y con un modelo académico de vanguardia, con tecnología que responde a las necesidades sociales y productivas. En esta Institución se imparten once programas educativos de Técnico Superior Universitario y once de nivel Licenciatura/Ingeniería se atiende cerca de 6,000 estudiantes, de los cuales, el 65% cursa estudios de Técnico Superior Universitario y el 35% restante estudios de nivel Licenciatura, en programas educativos reconocidos por su buena calidad. Es práctica constante de esta universidad actualizar sus planes y programas de estudio de acuerdo con las necesidades del sector productivo; renovar su equipamiento y mejorar sus instalaciones.

La primera estructura del organismo fue aprobada por la Secretaría de Administración del Gobierno Estatal en el mes de julio de 1996, la cual se integró por 14 unidades administrativas (una rectoría, una unidad de planeación, una contraloría interna, un abogado general, dos direcciones de división de carrera, dos direcciones de área y seis departamentos). En este contexto la Universidad Tecnológica de Tecámac, inicia sus labores docentes en septiembre de 1996, con la formación de 143 estudiantes, 100 en la carrera de Técnico Superior Universitario en Informática y 43 en la de Técnico Superior Universitario en Procesos de Producción. Trabajando en aulas provisionales y con el apoyo de seis profesores investigadores. El 17 de junio de 1997 se inició la construcción de sus propias instalaciones en una extensión de 35 hectáreas que originalmente formaban parte del Parque Estatal Ecológico, Turístico y Recreativo "Sierra Hermosa", ubicado en el kilómetro 37.5 de la carretera federal México-Pachuca.

# MISIÓN

“La Universidad Tecnológica de Tecámac, es una institución superior de educación tecnológica que profesionalmente forma de manera integral al estudiantado para mejorar su calidad y nivel de vida, con base en un modelo educativo de vanguardia y pertinencia social, al tiempo que ofrece estudios y servicios tecnológicos encaminados a la resolución de problemáticas del sector productivo y social con el objetivo de propiciar el desarrollo regional, estatal y nacional”.

# VISIÓN

“Ser una universidad de vanguardia, reconocida por su compromiso institucional con la excelencia académica, pertinente, con espíritu de servicio y mejora continua de su modelo educativo, distinguida por formar emprendedores exitosos y profesionales competitivos, generadores del cambio, reconocidos y aceptados en el mercado laboral”

# PROPÓSITO

La Universidad Tecnológica de Tecámac además de la excelencia en la formación académica, se ha preocupado por ofrecer a la juventud una educación integral que favorezca el desarrollo del ser y el convivir con los demás, a través de una sólida preparación centrada en los valores, con la finalidad de facilitar su inserción en las esferas social y productiva.

Con base al Artículo 5 de la ley que crea el organismo público descentralizado de carácter estatal denominado Universidad Tecnológica de Tecámac.

Para el cumplimiento de su objetivo, la Universidad Tecnológica de Tecámac, tendrá las siguientes atribuciones.

Adoptar la organización administrativa y académica que estime conveniente de acuerdo con los lineamientos generales previstos en esta ley.

Con base a esta ley se crea la Secretaria de Vinculación, actualmente, a cargo del M. en A. Román Velázquez Germán.

Como objetivo central esta unidad administrativa es promover una mayor vinculación de la Institución con los sectores público, privado y social, tanto nacionales como internacionales, a través de la concertación y celebración de convenios que contribuyan a elevar la calidad de las funciones académicas, de investigación, de educación continua y que ayuden al fortalecimiento del impacto institucional de la universidad en su zona de influencia.

Como funciones la Secretaria de Vinculación celebra los convenios con los sectores público, privado y social, que posibiliten el desarrollo de los programas de estadías profesionales, visitas industriales, servicio social, educación continua, intercambio académico, bolsa de trabajo e investigación y desarrollo tecnológico.

De la Secretaria de Vinculación surgen varias ramas dependientes de esta, las cuales son la Subdirección de Proyectos de Vinculación, el Departamento de Prácticas y Estadías, Departamento de Educación Continua, Departamento de Seguimiento a Egresados y el Departamento de Investigación y Desarrollo.

# OBJETIVOS

Objetivo General:

* Desarrollar e implementar un sistema de organización, control y consulta de los convenios correspondientes en los sectores público, privado y social de la Secretaría de Vinculación de la UTTEC.

Objetivos Específicos:

* Recopilar información que contribuya a la identificación de los elementos que contendrá el sistema de organización.
* Implementar la metodología necesaria para la aplicación de un Sistema *Web* que ocupara Secretaría de Vinculación.
* Detallar y documentar el procedimiento de llenado de cada uno de los convenios de la Secretaría de Vinculación.
* Facilitar la búsqueda y la consulta documental, que permitan recuperar los documentos de trámite de las unidades administrativas de la Secretaría de Vinculación.

# MARCO TEORICO

**Base de datos**

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (*DBMS)*. En conjunto, los datos y el *DBMS*, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

En el contexto de un Sistema de Organización Automatizado para la gestión de Convenios Generales, Marco y Específicos en la Secretaría de Vinculación con apoyo de las Divisiones Académicas, la aplicación de una base de datos sería fundamental para facilitar y mejorar la gestión de estos convenios.

**Almacenamiento de información**

Se crearía una base de datos centralizada donde se registrarían todos los datos relacionados con los Convenios Generales, Marco y Específicos. Esto incluiría información detallada sobre los convenios, como fechas de inicio y finalización, instituciones involucradas, divisiones académicas relacionadas, objetivos, términos y condiciones, y cualquier otro dato relevante.

En el contexto de la Secretaría de Vinculación, aplicar el almacenamiento de información implica establecer un sistema organizado y seguro para guardar todos los convenios generales, marco y específicos.

Esto se logra creando una base de datos electrónica donde se registran los detalles esenciales de cada convenio, como nombres, fechas y partes involucradas. Además, es importante categorizar estos convenios para una mejor organización. La seguridad de la información es prioritaria, por lo que se implementan medidas como contraseñas y cifrado para proteger los datos. Mantener esta base de datos actualizada con los nuevos convenios y permitir un acceso fácil y rápido para los usuarios son partes clave de este proceso. También se realizan copias de seguridad periódicas para prevenir la pérdida de datos, y se brinda capacitación a los usuarios para que utilicen eficazmente este sistema. Por último, se asegura el cumplimiento de las regulaciones y normativas relevantes en la gestión de la información.

**Acceso y consulta eficiente.**

El personal de la Secretaría y las Divisiones Académicas tendrían acceso a la base de datos a través de una interfaz amigable. Esto les permitiría buscar, consultar y actualizar la información de manera rápida y sencilla. Por ejemplo, podrían buscar todos los convenios relacionados con una división académica específica o filtrar los convenios por fecha de vencimiento.

Para lograr un acceso y consulta eficiente en un Sistema de Organización Automatizado para la gestión de Convenios Generales, Marco y Específicos en la Secretaría de Vinculación, es necesario organizar los convenios en categorías claras, implementar un motor de búsqueda eficiente, usar metadatos descriptivos, diseñar una interfaz intuitiva y mantener el sistema actualizado.

**Repositorios**

Un repositorio es un tipo de almacenamiento digital centralizado que los desarrolladores utilizan para realizar y administrar cambios en el código fuente de una aplicación, los desarrolladores tienen que almacenar y compartir carpetas.

**Convenio**

Fomentar la colaboración entre el área académica y las Direcciones de División para impulsar la firma de convenios generales de colaboración y acuerdos específicos con diferentes sectores de la sociedad, con el propósito de facilitar a los estudiantes la oportunidad de llevar a cabo estadías profesionales, visitas industriales y servicio social, al mismo tiempo que se promueve y se implementa el modelo educativo que optimiza la gestión de espacios destinados a estas actividades.

**Convenio General**

Un convenio general es un acuerdo amplio que establece los términos y condiciones generales para la colaboración entre dos o más partes. Suele ser un documento inicial que sienta las bases para futuras colaboraciones o acuerdos más específicos. En esencia, es un acuerdo marco que proporciona un marco general para la cooperación. Por ejemplo, una universidad podría firmar un convenio general con una empresa para establecer una relación de colaboración, pero los detalles específicos de proyectos o actividades se definirían en acuerdos adicionales.

Convenio Marco: Un convenio marco es un acuerdo más específico que se deriva de un convenio general. Establece las áreas de colaboración o cooperación en detalle, pero aún puede ser bastante amplio en comparación con un acuerdo específico. Por ejemplo, un convenio marco entre una universidad y una empresa podría definir las áreas de colaboración en investigación, desarrollo conjunto de proyectos, pasantías de estudiantes, pero no entraría en los detalles exactos de proyectos específicos o fechas.

**Convenio Específico**

Un convenio específico es el acuerdo más detallado y específico dentro de una colaboración más amplia. Este tipo de convenio se utiliza para establecer los términos precisos, objetivos, plazos y obligaciones de las partes involucradas en un proyecto o actividad particular. Por ejemplo, un convenio específico podría detallar un proyecto de investigación conjunto entre una universidad y una empresa, especificando los recursos necesarios, el cronograma, los resultados esperados y otros detalles clave.

# CAPÍTULO 1 PROCESO DE DESARROLLO WEB

## 1.1 Plan del proceso de desarrollo *web*

Para el desarrollo del Sistema *Web***,** se implantará un proyecto adaptativo con XP (Programación Extrema).



**Figura 1.1 Logo XP**

Esta metodología se basa en los recursos la innovación de proyecto para poder modificar en cualquier momento. La metodología del proyecto adaptativo se basa en los siguientes puntos para aplicar la metodología.

**Planificación del Juego**

En esta etapa el equipo XP trabaja en conjunto con el cliente para identificar los objetivos del proyecto y las características prioritarias que se deben desarrollar. El cliente y el equipo negocian y establecen las historias de usuario, que son descripciones de alto nivel de las funcionalidades que el software debe tener.

**Planificación de Iteraciones**

XP organiza el trabajo en iteraciones, que son ciclos de desarrollo con una duración fija (por ejemplo, 1-2 semanas). En esta etapa, el equipo selecciona las historias de usuario que se abordarán en la próxima iteración. También desglosan estas historias en tareas más pequeñas y estiman el tiempo necesario para completarlas.

**Diseño**

El equipo de desarrollo crea un diseño inicial para las características seleccionadas durante la planificación de iteraciones. El diseño se enfoca en mantener el código limpio, modular y fácil de mantener.

**Programación**

Durante esta fase, los desarrolladores escriben código para implementar las funcionalidades diseñadas. XP promueve la programación en parejas (dos programadores trabajando juntos en una tarea) para mejorar la calidad del código y el intercambio de conocimientos.

**Pruebas**

Las pruebas son una parte fundamental de XP. El equipo realiza pruebas unitarias y pruebas de aceptación para asegurarse de que el código funcione correctamente y cumpla con las historias de usuario definidas. La automatización de pruebas es altamente recomendada en XP.

**Integración Continua**

El equipo integra el código frecuentemente (varias veces al día) en un repositorio centralizado. Esto garantiza que todas las partes del software funcionen juntas sin problemas y facilita la detección temprana de problemas de integración.

**Entrega Continua**

XP promueve la entrega continua de software funcional. Después de cada iteración, el software se encuentra en un estado potencialmente entregable, lo que significa que puede ser entregado al cliente en cualquier momento. Refactorización medida que el software evoluciona, el equipo realiza refactorizaciones para mejorar la estructura del código sin cambiar su funcionalidad. Esto ayuda a mantener el código limpio y fácil de mantener a lo largo del tiempo.

**Retrospectiva**

Al final de cada iteración, el equipo realiza una retrospectiva para evaluar lo que funcionó bien y lo que se puede mejorar. Se identifican oportunidades de mejora en el proceso y se planifican acciones correctivas.

**Cierre del Juego**

Cuando se han completado todas las historias de usuario y se ha entregado el software final al cliente, se cierra el proyecto.

El proceso de desarrollo de XP es altamente colaborativo, iterativo y adaptable a los cambios. Está diseñado para satisfacer las necesidades cambiantes del cliente y proporcionar un producto de alta calidad de manera constante.

## 1.2 Arquitecturas de Software

*Laravel* es un *framework* de desarrollo web *PHP* muy popular que se destaca por su arquitectura limpia y bien estructurada, además de su amplia comunidad y comercialización.

*Laravel* sigue el patrón de diseño *MVC (*Modelo-Vista-Controlador*)*, que es ampliamente aceptado y utilizado en el desarrollo web. Esta arquitectura divide la

aplicación en tres componentes principales:

* El Modelo (que representa los datos y la lógica de negocio)
* La Vista (que se encarga de la presentación y la interfaz de usuario)
* El Controlador (que gestiona las interacciones entre el Modelo y la Vista).

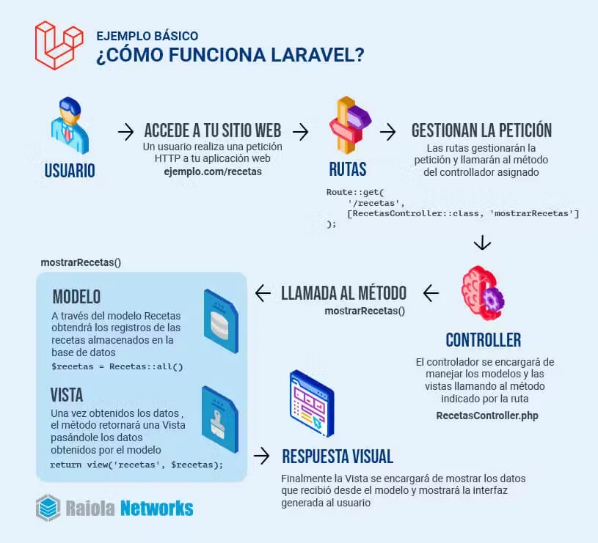
Explicar y justificar esta arquitectura es esencial, ya que ayuda a organizar el código de manera eficiente, facilita la colaboración entre desarrolladores y mejora la mantenibilidad.

*Eloquent* *ORM*: *Laravel* incluye *Eloquent*, esto permite a los desarrolladores interactuar con la base de datos utilizando objetos y consultas en lugar de *SQL* crudo. La justificación aquí es que *Eloquent* simplifica la manipulación de datos, mejora la seguridad y facilita la portabilidad de la aplicación al cambiar la base de datos subyacente. Enrutamiento claro y flexible Laravel ofrece un sistema de enrutamiento claro y fácil de usar. Los docentes pueden explicar cómo funciona el enrutamiento y cómo se asocian las rutas a controladores para manejar las solicitudes *HTTP*. Esto es fundamental para comprender cómo *Laravel* gestiona las solicitudes entrantes y cómo se dirigen a las acciones apropiadas.

Paquete de herramientas y bibliotecas: Laravel proporciona una amplia gama de

bibliotecas y herramientas para tareas comunes, como la autenticación, la gestión de caché, el envío de correo electrónico y más. Puedes justificar su uso al resaltar cómo aceleran el desarrollo y mejoran la calidad de la aplicación al aprovechar soluciones probadas.

Comunidad activa y documentación: Finalmente, puedes mencionar que *Laravel* tiene una comunidad activa de desarrolladores y una documentación completa. Esto facilita el aprendizaje y el desarrollo colaborativo, y muestra cómo Laravel se mantiene actualizado y seguro en la figura 1.1 se muestra los pasos de realizar funciona *Laravel* en la aplicación de los proyectos.



**Figura 1.2 Definición Laravel**

## 1.2.1 Diagrama de Arquitectura.

El diagrama de arquitectura de Laravel muestra los componentes necesarios que componen el *framework* a continuación se muestra lo que se establece para el diagrama.

* *Web* Navegador: El usuario interactúa con la aplicación a través de un navegador.
* Rutas: *Laravel* gestiona las rutas de la aplicación, lo que significa que decide qué controlador y método se ejecutarán en respuesta a una solicitud *HTTP* específica. Las rutas se definen en archivos de rutas.
* Controladores: Los controladores en Laravel son responsables de manejar las solicitudes del usuario y coordinar la lógica de la aplicación. Reciben datos de las rutas y, en función de esa información, realizan acciones como recuperar datos del modelo y pasarlos a las vistas para su renderización.
* Modelos: Los modelos representan la capa de acceso a la base de datos de la aplicación. *Laravel* utiliza *Eloquent*, un *ORM* (*Object*-*Relational* *Mapping*), para interactuar con la base de datos. Los modelos definen la estructura de los datos y proporcionan métodos para recuperar y almacenar datos en la base de datos.
* Vistas: Las vistas son responsables de la presentación y la interfaz de usuario. Definen cómo se muestra la información al usuario. *Laravel* utiliza el motor de plantillas *Blade* para crear vistas dinámicas.
* Base de Datos: Laravel admite múltiples sistemas de gestión de bases de datos, como *MySQL*, *PostgreSQL*, *SQLite*, etc. La capa de base de datos se encarga de almacenar y recuperar datos según lo solicitado por los modelos y controladores.
* Esta arquitectura facilita la separación de preocupaciones y la organización del código en una aplicación *Laravel*, lo que mejora la mantenibilidad y la escalabilidad del proyecto.

Laravel utiliza varios patrones de diseño en su arquitectura para lograr un desarrollo de aplicaciones web eficiente y organizado. Los patrones de diseño que son prominentes en Laravel incluyen:

* *MVC* (*Model*-*View*-*Controller*): El patrón *MVC* es fundamental en Laravel. Divide la aplicación en tres componentes principales
* Modelo: Representa la capa de acceso a la base de datos y contiene la lógica empresarial.
* Vista: Encargada de la presentación y la interfaz de usuario.
* Controlador: Gestiona las solicitudes del usuario y coordina la interacción entre el Modelo y la Vista. En *Laravel*, los controladores se utilizan para manejar rutas y responder a las peticiones *HTTP*.
* *Singleton*: *Laravel* utiliza el patrón *Singleton* para gestionar la instancia única de la aplicación durante el ciclo de vida de una solicitud *HTTP*. Esto permite compartir recursos y servicios en toda la aplicación sin crear múltiples instancias.
* *Dependency* *Injection*: *Laravel* fomenta la inversión de control y la inyección de dependencias. Utiliza contenedores de inversión de control (*IoC*) para manejar la resolución y la inyección de dependencias de manera automática, lo que facilita la administración de clases y servicios.
* Factory: Laravel utiliza el patrón Factory para crear instancias de objetos de manera eficiente, especialmente en el contexto de las pruebas unitarias y la generación de datos de prueba.
* *Facade*: Las *facades* en *Laravel* proporcionan una interfaz sencilla y unificada para acceder a componentes complejos de la aplicación, como servicios de autenticación, gestión de correo electrónico y otros. Esto simplifica la interacción con estas características y oculta la complejidad subyacente.
* *Repository*: Aunque no es un patrón de diseño clásico, *Laravel* promueve el uso de patrones de repositorio para abstraer la interacción con la base *de* datos. Esto facilita el cambio de motor de base de datos o la realización de pruebas unitarias.



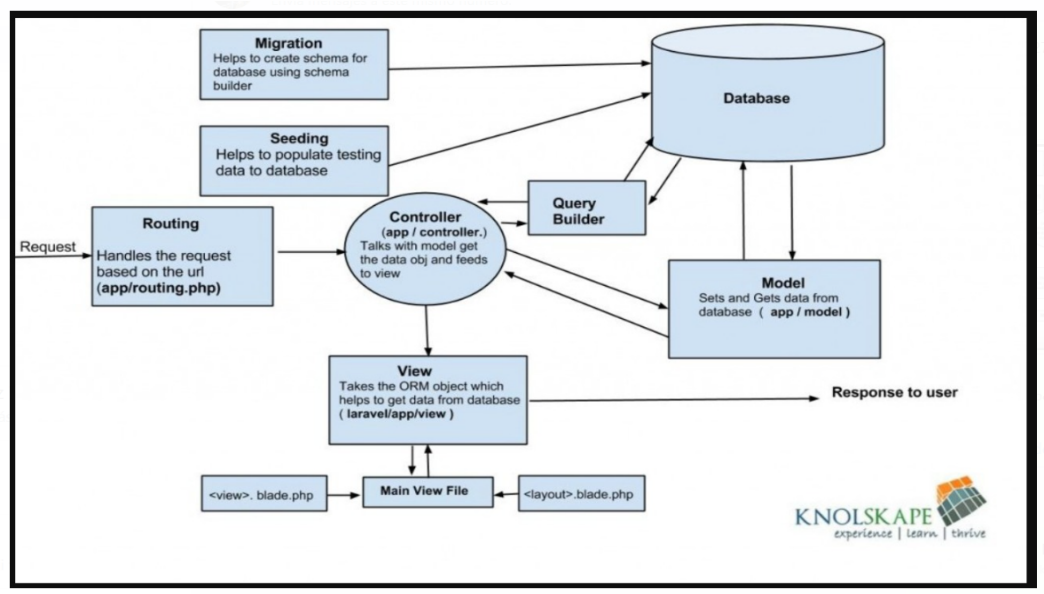
**Figura 1.3 Logo Laravel**

*React Native* es un *framework* de desarrollo de aplicaciones móviles creado por Facebook que permite a los desarrolladores crear aplicaciones móviles multiplataforma utilizando *JavaScript* y *React*, el popular marco de trabajo para construir interfaces de usuario *web*. *Native* se utiliza para desarrollar aplicaciones nativas para dispositivos *iOS* y *Android*, lo que significa que las aplicaciones resultantes tienen un rendimiento y apariencia similares a las aplicaciones desarrolladas con tecnologías nativas de cada plataforma.

*React Native* se basa en el mismo enfoque de desarrollo de componentes que *React* para la *web*. Los componentes son elementos reutilizables que encapsulan la lógica y la interfaz de usuario de la aplicación. Utiliza la biblioteca *React* para crear y gestionar estos componentes. Uno de los componentes más importantes de la arquitectura de *React Native* es el "*bridge*" o puente El *bridge* permite que el código *JavaScript* interactúe con los componentes nativos de la plataforma, como botones, vistas y módulos de hardware, a través de una interfaz de programación de aplicaciones (*API*) nativa. Además incluye una biblioteca de componentes nativos que se traducen directamente en elementos de interfaz de usuario nativos de iOS y Android. Estos componentes incluyen botones, vistas, listas, cajas de texto y muchos otros elementos comunes de las interfaces de usuario móviles en la figura 1.1 se muestra los componentes de arquitectura de Laravel con *React Native* para la realización de este proyecto.



**Figura 1.4 Logo React**



**Figura 1.5 Arquitectura Laravel**

## 1.3 Pruebas de Desarrollo

Las pruebas de software hacen referencia a una herramienta encargada de analizar la funcionalidad y el rendimiento de un sistema para mejorar la experiencia del usuario. Estas opciones tienen como base la monitorización de diversos aspectos de los equipos; de esta forma existe, por ejemplo, las llamadas pruebas de caja negra, también conocidas como *black* *box*. Se definen como una técnica de análisis de la funcionalidad de un sistema que no tiene en cuenta la estructura interna del código.

Se utilizarán las pruebas de caja negra porque son parte esencial del proceso de software desde la perspectiva de usuario y controlando y comprobando las funcionalidades de garantizar que el software que es fundamental que el cliente este satisfecho cumpliendo todos los requisitos funcionales sus requerimientos de cliente y la perspectiva de usuario para ello se realizó una breve investigación y los puntos clave de las pruebas de caja negra para el desarrollo de este dicho proyecto.

Una "caja negra" se refiere a una técnica en la que se prueba una unidad, un módulo o una aplicación sin conocer o preocuparse por su estructura interna, diseño o funcionamiento interno. En otras palabras, se trata de probar el software desde el exterior, como si fuera una "caja negra" cuyo contenido no se conoce.

En las pruebas de caja negra, el enfoque se coloca en las entradas proporcionadas y las salidas generadas por el software bajo prueba. Los detalles internos del código no son relevantes para las pruebas de caja negra. Las pruebas de caja negra se centran en evaluar si el software realiza las funciones y tareas requeridas según las especificaciones o requisitos del usuario. Los casos de prueba se diseñan para verificar que el software produce resultados esperados para una variedad de entradas.

Las pruebas de caja negra pueden incluir pruebas de funcionalidad, pruebas de aceptación, pruebas de integración, pruebas de estrés y muchas otras, dependiendo de los objetivos de las pruebas y el nivel de detalle necesario. Las pruebas de caja negra son independientes de cómo se ha implementado el software.

# CAPITULO 2 CONTROL DE VERSIONES E INTEGRACION DE COMPONENTES

## 2.1 Versionamientos de *software*

Sistema de Almacenamiento v1.0.0

* Lanzamiento inicial del Sistema de Organización Automatizado y edición de texto.

Sistema de Almacenamiento v1.1.0

* Añadida la capacidad de crear carpetas para organizar documentos.
* Se implementan funciones básicas de formato de texto: negrita, cursiva y alineación.

Sistema de Almacenamiento v1.1.1

* Corrección de errores relacionados con la carga de documentos grandes.
* Mejora en la velocidad de carga de la interfaz.

Sistema de Almacenamiento v1.2.0

* Introducción de la función de búsqueda de documentos por nombre y contenido.
* Se añade la opción de compartir documentos con otros usuarios para colaboración.

Sistema de Almacenamiento v1.2.1

* Rediseño completo de la interfaz para mejorar la usabilidad en dispositivos móviles.
* Se implementa un sistema de historial de versiones para realizar un seguimiento de los cambios en los documentos.
* Añadida la posibilidad de acceder y editar documentos sin conexión.

Sistema de Almacenamiento v2.1.1

* Solucionados problemas de sincronización específicos relacionados con la colaboración en tiempo real.
* Corrección de errores al utilizar la función de comentarios en documentos con formato complejo.

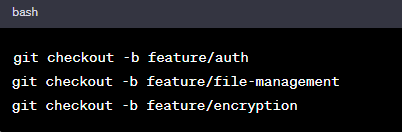
Sistema de Almacenamiento v2.1.1

* Implementación de un sistema de complementos para ampliar las capacidades del editor en línea.
* Mejoras en la velocidad de carga de documentos y la estabilidad del sistema.

## 2.2 Sistema de control de versiones

En Sistema de Organización Automatizado Utilizaremos GitHub como sistema de control de versiones para gestionar el código fuente del proyecto y colaborar en su desarrollo.

* Creación del repositorio: Se crea un repositorio en GitHub llamado "Sistema Web de almacenamiento" para almacenar el código fuente de la aplicación.
* Estructura del Proyecto: El proyecto se organiza en módulos como autenticación, gestión de archivos, cifrado, etc.
* Colaboración de Desarrolladores: Varios desarrolladores trabajan
* simultáneamente en diferentes aspectos del proyecto. Cada uno tiene su rama de desarrollo para evitar conflictos como se muestra en la figura 2.1.



**Figura 2.1 Código GitHub**

* Integración Continua (CI/CD): Se configura un sistema de integración continua para realizar pruebas automáticas y despliegues continuos en entornos de desarrollo y pruebas.
* Gestión de Problemas: Se utilizan issues de GitHub para gestionar tareas, bugs y mejoras. Cada issue se asocia con un proyecto en el tablero de proyectos de GitHub.
* Despliegue en Producción: Se utiliza GitHub Actions para automatizar el despliegue en entornos de producción después de una fusión exitosa en la rama master.
* Documentación Automática: Se utiliza GitHub Pages para generar y alojar documentación automáticamente desde el código fuente, proporcionando una referencia actualizada.

## 2.3 Herramientas de razonamiento

***Razonamiento Lógico y Cognitivo***

Característica: La aplicación utiliza algoritmos de inteligencia artificial para analizar los patrones de uso de los usuarios y proporcionar sugerencias inteligentes sobre la organización y acceso de archivos.

Ejemplo: Cuando un usuario tiende a acceder a ciertos tipos de archivos en momentos específicos, el sistema aprende estos patrones y sugiere automáticamente la organización de archivos relacionados.

***Sentido de la Planificación:***

Característica: El sistema tiene una función de planificación que permite a los usuarios programar la gestión automática de archivos, como la creación de copias de seguridad periódicas o la organización automática de documentos.

Ejemplo: Un usuario puede planificar que el Sistema de Organización Automatizado realice una copia de seguridad semanal de ciertas carpetas importantes.

***Organización Efectiva***

Característica: La aplicación organiza automáticamente los archivos en categorías basadas en su contenido y patrones de uso.

Ejemplo: Los documentos relacionados con el trabajo se agrupan en una categoría separada de los archivos personales. Además, el sistema crea automáticamente una carpeta de "Proyectos Activos" para facilitar el acceso rápido a los archivos más relevantes.

***Capacidad de Autoaprendizaje:***

Característica: Sistema de Organización Automatizado utiliza aprendizaje automático para adaptarse a las preferencias del usuario y mejorar las sugerencias y acciones automatizadas con el tiempo.

Ejemplo: Después de un período, el sistema aprende que el usuario prefiere organizar manualmente ciertos tipos de archivos y ajusta sus sugerencias en consecuencia.

***Flujos de Trabajo Personalizables:***

Característica: Los usuarios pueden crear flujos de trabajo personalizables para realizar acciones específicas, como la organización de archivos, la compartición de documentos y la creación de enlaces de acceso rápido.

Ejemplo: Un usuario configura un flujo de trabajo que organiza automáticamente los archivos descargados por tipo y elimina archivos temporales cada mes.

***Interfaz Intuitiva y Orientada a la Productividad:***

Característica: La interfaz de usuario está diseñada para facilitar la realización de tareas sin esfuerzo y para ofrecer una experiencia de usuario intuitiva.

Ejemplo: Los usuarios pueden arrastrar y soltar archivos para organizarlos, establecer reglas de organización y ver visualmente el historial de versiones de los archivos.

## 2.4 Especificación de principios de código

Para la elaboración de este proyecto se utilizó Lineamientos de Protección de Datos Personales – SecureDataVault.

Descripción: Antes de recopilar cualquier dato personal, el sistema solicitará el consentimiento informado de los usuarios.

Implementación: Se mostrará un cuadro de diálogo explicando claramente qué datos se recopilan, con qué propósito y cómo se utilizarán. Los usuarios deberán dar su consentimiento explícito antes de utilizar el servicio

***Mínima Recopilación de Datos***

* Descripción: Se recopilará la cantidad mínima de datos personales necesarios para proporcionar el servicio.
* Implementación: Solo se solicitarán los datos esenciales para el registro y la prestación del servicio, evitando la recopilación innecesaria de información personal.

***Seguridad de Datos:***

* Descripción: Se implementarán medidas de seguridad robustas para proteger los datos personales contra accesos no autorizados, pérdida o robo.
* Implementación: Uso de cifrado para datos en tránsito y en reposo, acceso restringido basado en roles, auditorías periódicas de seguridad y actualizaciones regulares para abordar vulnerabilidades conocidas.

***Transparencia en el Manejo de Datos:***

* Descripción: Los usuarios tendrán acceso a información clara sobre cómo se manejan y almacenan sus datos personales.
* Implementación: Se proporcionará una política de privacidad detallada que explique los procesos de recopilación, almacenamiento sus propios datos almacenados.

***Derecho al Olvido:***

* Descripción: Los usuarios tendrán el derecho de solicitar la eliminación de sus datos personales cuando ya no sean necesarios.
* Implementación: Se ofrecerá a los usuarios un mecanismo fácil para solicitar la eliminación de sus datos, y el sistema garantizará la eliminación segura y permanente de la información.

***Acceso y Control de Datos:***

* Descripción: Los usuarios tendrán la capacidad de acceder y controlar sus propios datos personales.
* Implementación: Se proporcionará a los usuarios un panel de control donde podrán ver, editar y eliminar sus datos. Además, podrán gestionar las configuraciones de privacidad y preferencias de notificación.

***Notificaciones de Brechas de Seguridad:***

* Descripción: En caso de una violación de seguridad que pueda afectar la privacidad de los datos, se notificará a los usuarios de manera oportuna.
* Implementación: Se establecerá un plan de respuesta a incidentes que incluya la notificación rápida y clara a los usuarios afectados en caso de una brecha de seguridad.

***Protección de Datos Sensibles:***

* Descripción: Los datos personales sensibles se tratarán con medidas de seguridad adicionales.
* Implementación: Los datos sensibles, como información financiera o datos médicos, estarán sujetos a capas adicionales de cifrado y protección, y se aplicarán políticas de acceso más estrictas.

***Revisión y Actualización de Políticas:***

* Descripción: Las políticas de privacidad y protección de datos se revisarán y actualizarán regularmente para cumplir con las normativas actuales.

## 2.4.1 Principios de Codificación Segura

***Validación de Entradas:***

* Principio: Todas las entradas de datos proporcionadas por los usuarios deben ser validadas y filtradas para prevenir ataques de inyección.
* Implementación: Se implementarán filtros de entrada y validación de datos en todas las áreas del sistema que acepten entrada del usuario, como formularios de carga de archivos, campos de búsqueda y comentarios.

***Prevención de Inyecciones SQL:***

* Principio: Se deben utilizar consultas preparadas y procedimientos almacenados para prevenir la inyección de código SQL.
* Implementación: Se utilizarán consultas parametrizadas o procedimientos almacenados en lugar de construcciones de SQL concatenadas para interactuar con la base de datos.

***Manejo Seguro de Sesiones:***

* Principio: Las sesiones deben manejarse de manera segura para evitar ataques de secuestro de sesión.
* Implementación: Se utilizarán identificadores de sesión seguros, se implementará HTTPS y se establecerán políticas de expiración de sesión adecuadas.

***Cifrado de Datos en Reposo y en Tránsito:***

* Principio: Todos los datos sensibles, tanto en reposo como en tránsito, deben cifrarse para proteger la confidencialidad.
* Implementación: Se utilizarán algoritmos de cifrado fuertes para proteger la información almacenada y se implementará HTTPS para asegurar la comunicación entre el cliente y el servidor.

***Manejo de Errores Seguro:***

* Principio: Los mensajes de error no deben revelar información sensible y deben registrarse de manera segura.
* Implementación: Se personalizarán los mensajes de error para no proporcionar detalles sensibles al usuario y se registrarán los errores de manera segura sin exponer información confidencial.

***Control de Acceso y Autorización:***

* Principio: Se implementará un sistema de control de acceso y autorización para garantizar que los usuarios solo tengan acceso a los recursos necesarios.
* Implementación: Se establecerán roles y permisos, y se validarán las solicitudes de acceso a recursos según la autorización del usuario.

***Actualizaciones y Parches:***

* Principio: Mantener el sistema actualizado con las últimas actualizaciones y parches de seguridad.
* Implementación: Se establecerá un proceso de gestión de parches para aplicar rápidamente las actualizaciones de seguridad en el sistema y sus dependencias.

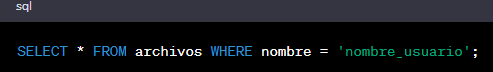
***Escaneo de Seguridad y Pruebas Penetración:***

* Principio: Se realizarán regularmente escaneos de seguridad y pruebas de penetración para identificar y abordar posibles vulnerabilidades.
* Implementación: Se contratarán servicios de escaneo de seguridad y pruebas de penetración periódicas para evaluar y mejorar continuamente la seguridad del sistema.

***Registro de Auditoría:***

* Principio: Se implementará un registro de auditoría para rastrear actividades del sistema y facilitar la detección de comportamientos inusuales.

Una vulnerabilidad común en sistemas web de almacenamiento es la "Inyección de Código SQL". Este tipo de vulnerabilidad ocurre cuando los datos proporcionados por un usuario son maliciosamente manipulados e insertados en instrucciones SQL que se ejecutan en la base de datos del sistema. Esto puede dar lugar a la exposición, modificación o eliminación no autorizada de datos almacenados en la base de datos como se muestra en la figura 2.2.



***Figura 2.2 “código mysql”***

Esta consulta siempre devolverá resultados, ya que '1'='1' es siempre verdadero, y el doble guion doble (--') comenta el resto de la consulta, ignorando cualquier restricción que pueda seguir.

Con esta inyección de SQL, un atacante podría obtener acceso no autorizado a datos sensibles, como archivos de otros usuarios o información confidencial almacenada en la base de datos. Además, podrían realizar acciones maliciosas, como la eliminación de archivos o la modificación de registros.

Para prevenir la inyección de código SQL, es crucial implementar prácticas seguras de manejo de datos, como el uso de consultas parametrizadas o declaraciones preparadas, y la validación y filtrado adecuado de las entradas del usuario.

El sistema permite a los usuarios acceder a sus propios archivos mediante la autenticación y autorización adecuadas. Cada usuario tiene un identificador único y se almacena en la base de datos junto con sus archivos.

En este escenario, existe una vulnerabilidad en la implementación de la verificación de autorización que permite a un usuario malintencionado elevar sus privilegios y acceder a los archivos de otros usuarios.

## Certificado de Seguridad Web – Sistema de almacenamiento web

***Información del Certificado:***

* Nombre del Sitio Web: Sistema de almacenamiento.com
* Emisor del Certificado: Let's Encrypt
* Tipo de Certificado: Certificado SSL/TLS de servidor
* Vigencia del Certificado: Desde 01 de enero de 2023 hasta 01 de enero de 2024

***Detalles del Certificado:***

* Asunto (Subject):
* Organización: SafeStorage Inc.
* Unidad Organizativa: Departamento de Seguridad
* Ubicación: Ciudad Segura
* Estado: Estado Seguro
* País: País Seguro

***Propósito del Certificado:***

* Este certificado se emitió con el propósito de asegurar la comunicación cifrada entre el servidor SafeStorage y los navegadores de los usuarios. Utiliza el protocolo TLS (Transport Layer Security) para garantizar la privacidad y la integridad de los datos transmitidos.

***Algoritmo de Cifrado:***

* El certificado utiliza el algoritmo de cifrado asimétrico RSA con una longitud de clave de 2048 bits para el intercambio de claves y el algoritmo de cifrado simétrico AES con una longitud de clave de 256 bits para la comunicación cifrada.

***Firma Digital:***

* La firma digital del certificado se generó utilizando el algoritmo de firma SHA-256 para garantizar la autenticidad del certificado.

***Autoridad de Certificación (CA): Let's Encrypt***

* La validez del certificado se puede verificar mediante la cadena de confianza hasta la autoridad de certificación Let's Encrypt, que es ampliamente reconocida y confiable.

Cómo Verificar el Certificado:

***Navegador Web:***

* Los usuarios pueden verificar la validez del certificado haciendo clic en el icono de candado en la barra de direcciones de su navegador. Se mostrará información detallada sobre la conexión segura, incluido el certificado.

***Herramientas en Línea de Comprobación de Certificados:***

Los administradores del sistema pueden utilizar herramientas en línea de comprobación de certificados para evaluar la configuración del certificado y su validez.

# CAPITULO 3 PRUEBAS Y LIBERACION DEL DESARROLLO WEB

## 3.1 Despliegue de servicios

Contenedores

Los contenedores son paquetes de software que incluyen todos los elementos necesarios para ejecutar tus productos en cualquier entorno. Como virtualizan el sistema operativo, se pueden ejecutar en cualquier parte, desde un centro de datos privado hasta la nube pública o en la computadora personal de un desarrollador.

Según la definición de Google Cloud:

*“Los contenedores son paquetes ligeros que incluyen el código de las aplicaciones junto con sus dependencias, como versiones concretas de entornos de ejecución de ciertos lenguajes de programación y bibliotecas indispensables para ejecutar los servicios de software.”*

Un contenedor es usado para desvincular el despliegue de una aplicación que se ejecuta ya sea en la nube, un centro de datos o una computadora, al contar con un mecanismo que empaqueta todo el programa, permite extraerse de su entorno principal.

Ventajas que los desarrolladores obtienen al utilizar contenedores:

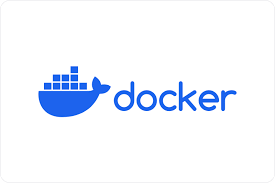
* Separación de responsabilidades
* Portabilidad de las cargas de trabajo
* Aislamiento de aplicaciones
* Desarrollo ágil
* Operaciones eficientes
* Ejecución en cualquier lugar

En el presente proyecto, se hace mención del uso del framework Laravel y la librería React para el desarrollo del Sistema de Organización Automatizado, normalmente para cargar el sistema se deben hacer ciertas modificaciones a archivos de programas que trabajan en conjunto, paqueterías y dependencias que son de suma importancia, aunque si el sistema se necesita ejecutar en otro entorno puede que sea algo complicado, con el apoyo de los contenedores el sistema se convierte en un programa portátil que podrá ejecutarse muy fácil y rápidamente.

## 3.2 Herramientas de gestión de contenedores

**Docker**

Docker engloba un conjunto de productos bajo la ‘fórmula’ plataforma como servicio que emplea la virtualización a nivel del sistema operativo para entregar software en paquetes llamados contenedores. Incluye la administración de clústeres integrada con Docker Engine, el balanceo de carga, opciones de seguridad como el cifrado y la autenticación, y actualizaciones continuas.



**Figura 3.1 Logo Docker**

**AWS**

Amazon Elastic Container Service (ECS) es un servicio de orquestación de contenedores completamente administrado que ayuda a implementar, administrar y escalar aplicaciones con contenedores. Se integra profundamente con el entorno de AWS para proporcionar una solución fácil de usar para ejecutar cargas de trabajo de contenedores en la nube y en las instalaciones con funciones de seguridad avanzada.



**Figura 3.2 Logo Amazon ECS**

**Azure Container Instances**

ACI ofrece la posibilidad de ejecutar contenedores sin la preocupación del entorno de ejecución ni del cluster donde se ejecutará el mismo. Está pensando para la ejecución de aplicaciones basadas en eventos o para trabajos de procesamiento de datos, pero también se utiliza para extender la capacidad de AKS que son las instancias de microservicios bajo demanda que se ejecutan en este servicio.

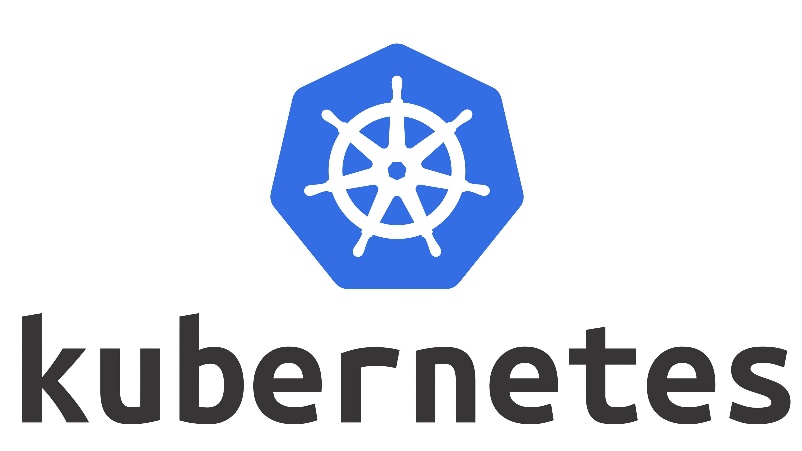


**Figura 3.3 Logo Azure**

**Google cloud con Kubernetes**

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Facilita la automatización y la configuración y tiene un ecosistema grande y rápido crecimiento.

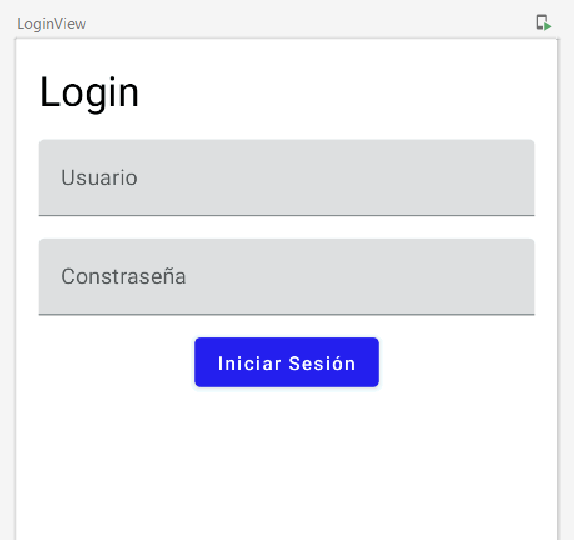
Kubernetes ofrece un entorno de administración centrado en contenedores. Orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo.



**Figura 3.4 Código GitHub**

## 3.3 Pruebas de software

## 3.3.1 Técnica de caja negra



**Figura 3.5 Inicio de sesión**

**Actividad: Iniciar sesión**

**1-. Identificar las entradas y salidas**

|  |  |
| --- | --- |
| Entradas | Salidas |
| Mover el cursor en el apartado de usuario | **Escribir tu nombre de usuario** |
| Mover el cursor en el apartado de contraseña | **Escribir tu contraseña** |
| Dar clic en el botón Iniciar Sesión | **Ingresar** |

**Tabla 1.1 Entradas y salidas**

**2-. Identificar las causas y efectos**

|  |  |
| --- | --- |
| Causas | Efectos |
| A1 Mover el cursor a el apartado usuario | **10 Permite escribir el nombre de usuario** |
| A2 Mover el cursor a el apartado contraseña | **20 Permite escribir la contraseña del usuario** |
| A3 Dar clic en el botón Iniciar Sesión | **30 Ingresa al sistema** |

**Tabla 1.2 Causas y efectos**

3-. Grafos

**Grafo 1**

**Grafo 2**

**Grafo 3**

**4-. Tabla de decisión**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grafo 1 | Grafo 2 | Grafo 3 |
| Casos de prueba | 1 | 2 | 3 |
| Causas |  |  |  |
| A1 Mover el cursor a el apartado usuario | 1 | 0 | 0 |
| A2 Mover el cursor a el apartado contraseña | 0 | 1 | 0 |
| A3 Dar clic en el botón Iniciar Sesión | 0 | 0 | 1 |
| 10 Permite escribir el nombre de usuario | 1 | 0 | 0 |
| 20 Permite escribir la contraseña del usuario | 0 | 1 | 0 |
| 30 Ingresa al sistema | 0 | 0 | 1 |

**Tabla 1.3 Tabla de decisión**

**5-. Casos de prueba**

|  |  |  |  |
| --- | --- | --- | --- |
| Casos de prueba | Datos de entrada | Resultados esperados | Resultados reales |
| 1 | **Escribir el nombre de usuario** | **Capturado** |  |
| 2 | **Escribir la contraseña de usuario** | **Capturado** |  |
| 3 | **Dar clic en el botón Iniciar Sesión** | **Ingresa al sistema** |  |

**Tabla 1.4 Casos de prueba**

## 3.4 Liberación del software

## 3.4.1 Políticas y normativas aplicables

1-. Como primer punto, es necesario que el encargado y/o autores de la liberación lean la normativa de la Universidad de Granada sobre Derechos de Propiedad Industrial e Intelectual.

2-. Todos los contribuidores deben estar de acuerdo con la liberación y aparecer en la documentación y firmar el documento oportuno.

En caso de resultados de investigación, el líder de proyecto podría decidir sobre los derechos, estando prevista la creación de una comisión para resolver (si los hubiese) las discrepancias que puedan surgir durante este proceso.

3-. La publicación en repositorios y/o liberación del código debe esperar hasta que se autorice la misma.

4-. Debe asegurarse que todo el contenido a liberar que no haga uso de librerías privativas u otros recursos sujetos a derechos de autor tales como imágenes, videos, audios, etc.

5-. En caso de que existan otras librerías o recursos en el proyecto con licencias más o menos permisivas, deben ser compatibles con la licencia que se quiere usar en la liberación del proyecto.

6-. La Oficina de Software Libre hace uso de la plataforma de registro de propiedad intelectual SafeCreative, en la cual se incluirá la información relativa a la liberación.

7-. El encargado de la liberación deberá rellenar el formulario para registrar la liberación del software.

8-. Al terminar de rellenar los datos se generará un documento PDF con dichos datos. El encargado de la liberación debe asegurarse de que este documento tiene la información correcta antes de proceder a su firma y envío.

9-. Una vez complementada y firmada, esta documentación debe ser remitida a la Oficina de Software Libre en formato digital y físico.

## 3.4.2 Herramientas de liberación de software

Para la realización de liberación del Sistema de Organización Automatizado, se llevaron acabo los siguientes procesos de desarrollo:

**Control de versiones**

Primero se estableció un sistema de control de versiones, ya que la mayoría de los procesos se tienen que realizar en el momento en el que se envían cambios al código del proyecto. Para el Sistema de Organización Automatizado se usó Git, ya que cuenta servicios en la nube, capaces de dar soporte a operaciones de todo tipo.

**Pruebas del software**

Las pruebas del software permiten realizar comprobaciones del código del proyecto de forma que se puedan detectar errores de manera temprana. Todo código deber tener sus pruebas.

Las pruebas nos ofrecen una red de seguridad que ayuda en el día a día del proyecto, permitiendo la refactorización del código sabiendo que los cambios aplicados no han roto nada, ni en el módulo que se está desarrollando ni en el resto de la aplicación.

**Git Hooks**

Por último, Git Hooks permite trabajar de manera sincronizada. Además permite automatizar las comprobaciones de estilo y las pruebas en local en el momento en el que se van a confirmar cambios al repositorio.

**Herramientas a nivel remoto**

Una vez que se envían cambios al repositorio remoto se pueden enganchar nuevos procesos automáticos con comprobaciones para asegurar la calidad del software, así como realizar el despliegue.

**GitHub Actions**

GitHub Actions permite automatizar diversos procesos una vez se han recibido cambios en el repositorio.

En este caso para el proyecto, lo que se realizar primero es el código que se envía a la rama master, lo que desencadena el despliegue en el servidor de producción los cuales trabajan con la sintaxis Yaml, por lo que se pueden realizar las siguientes acciones:

* Las pruebas unitarias
* Las pruebas de integración
* Organizar el despliegue
* Verificaciones manuales

**GitLab CI/CD**

GitLab sería el equivalente a las GitHub Actions, permite configurar los flujos de trabajo para comprobaciones de pruebas automatizadas y realizar el despliegue en los distintos entornos.

Enlace del repositorio en funcionamiento

**https://github.com/BrandonDv22345/Proyect\_ING**

# CONCLUSIONES

Magdiel Itamar Pérez Hernández

En conclusión, el diseño e implementación de un Sistema de Organización Automatizado para la gestión de Convenios Generales, Marco y Específicos en la Secretaría de Vinculación representa un avance significativo en la eficiencia operativa de la Universidad Tecnológica de Tecámac. Este sistema no solo optimiza el flujo de trabajo interno, sino que también fortalece la colaboración entre la Secretaría de Vinculación y las Divisiones Académicas. Los beneficios se extienden a todo el personal directivo y administrativo, proporcionando una plataforma centralizada para el seguimiento y la actualización de los convenios, mejorando así la transparencia y la toma de decisiones informadas. En última instancia, este proyecto demuestra el compromiso de la universidad con la innovación y la mejora continua en su búsqueda de la excelencia en la gestión de convenios y la promoción de relaciones efectivas con sus socios externos.

Brandon Jesus Diaz Vega

La implementación del Sistema de Organización Automatizado en la gestión de Convenios Generales, Marco y Específicos ha marcado un hito en la Universidad Tecnológica de Tecámac. Este avance no solo agiliza las operaciones internas, sino que también teje una red más fuerte entre la Secretaría de Vinculación y las Divisiones Académicas. Al alcanzar a todo el personal directivo y administrativo, el sistema se erige como un faro de eficiencia al proporcionar una plataforma centralizada para el monitoreo y la actualización de los convenios. Este logro no solo eleva la transparencia y la calidad en la toma de decisiones, sino que también subraya el compromiso de la universidad con la innovación y la mejora constante. En definitiva, este proyecto es un testimonio tangible del impulso de la institución hacia la excelencia en la gestión de convenios y el fortalecimiento de relaciones fructíferas con sus colaboradores externos.

Francisco Alejandro Ruiz Martinez

La puesta en marcha del Sistema de Organización Automatizado para la gestión de Convenios Generales, Marco y Específicos representa un logro significativo en la Universidad Tecnológica de Tecámac. Este avance no solo optimiza los procesos internos, sino que también consolida una conexión más sólida entre la Secretaría de Vinculación y las Divisiones Académicas. Al llegar a todo el personal directivo y administrativo, el sistema se erige como un referente de eficiencia al proporcionar una plataforma centralizada para el seguimiento y la actualización de los convenios. Este hito no solo mejora la transparencia y la calidad en la toma de decisiones, sino que también enfatiza el compromiso de la universidad con la innovación y la mejora continua. En última instancia, este proyecto es una prueba palpable del impulso de la institución hacia la excelencia en la gestión de convenios y en el fortalecimiento de relaciones productivas con sus colaboradores externos.

Rafael Zamora Rosales

La implementación del Sistema de Organización Automatizado para la gestión de Convenios Generales, Marco y Específicos marca un logro notable en la Universidad Tecnológica de Tecámac. Este avance no solo perfecciona los procedimientos internos, sino que también fortalece la conexión entre la Secretaría de Vinculación y las Divisiones Académicas. Alcanzando a todo el personal directivo y administrativo, el sistema se posiciona como un modelo de eficiencia al brindar una plataforma centralizada para el monitoreo y la actualización de los convenios. Este logro no solo eleva la transparencia y la calidad en la toma de decisiones, sino que también resalta el compromiso de la universidad con la innovación y la mejora constante. En resumen, este proyecto constituye una evidencia tangible del impulso de la institución hacia la excelencia en la gestión de convenios y en el fortalecimiento de relaciones fructíferas con sus colaboradores externos.

# LISTADO DE SIGLAS O ACRÓNIMOS

UTTEC Universidad Tecnológica de Tecámac

DBMS Sistema de Gestión de Bases de Datos

XP Programación Extrema

PHP Preprocesador de Hipertexto

MVC Modelo Vista Controlador

ORM Mapeo de Objetos Relacional

SQL Lenguaje de Consulta Estructurada

HTTPS Protocolo de Transferencia de Hipertexto Seguro

IoC Contenedores de Inversión de Control

API Programación de Aplicaciones

CI/CD Integración Continua/Implementación Continua

SSL/TLS Seguridad de Capa de Sockets Seguros/Capa De Transporte

RSA Algoritmo de Cifrado Asimétrico

AES Algoritmo de Cifrado Simétrico

CA Autoridad de Certificación

ECS Servicio de Contenedor Elástico

# GLOSARIO

Base de datos: es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS).

Repositorios: un repositorio es un tipo de almacenamiento digital centralizado que los desarrolladores utilizan para realizar y administrar cambios en el código fuente de una aplicación, los desarrolladores tienen que almacenar y compartir carpetas.

Convenio: acuerdo entre dos o más personas o entidades sobre un asunto.

Convenio General: un convenio general es un acuerdo amplio que establece los términos y condiciones generales para la colaboración entre dos o más partes. Suele ser un documento inicial que sienta las bases para futuras colaboraciones o acuerdos más específicos.

Convenio Específico: un convenio específico es el acuerdo más detallado y específico dentro de una colaboración más amplia. Este tipo de convenio se utiliza para establecer los términos precisos, objetivos, plazos y obligaciones de las partes involucradas en un proyecto o actividad particular.

Arquitecturas de Software: las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.

Framework: un framework es un conjunto de reglas y convenciones que se usan para desarrollar software de manera más eficiente y rápida. Estos marcos de trabajo se emplean para ahorrar tiempo y esfuerzo en el desarrollo de aplicaciones, ya que proporcionan una estructura básica que se puede utilizar como punto de partida.

Modelo Vista Controlador: es un patrón en el diseño de software combinado utilizado para implementar interfaces de usuario, datos y lógica de control.

Técnica caja negra: se refiere a una técnica en la que se prueba una unidad, un módulo o una aplicación sin conocer o preocuparse por su estructura interna, diseño o funcionamiento interno. En otras palabras, se trata de probar el software desde el exterior, como si fuera una "caja negra" cuyo contenido no se conoce.

Versionamientos de software: se refiere a una opción que permite identificar de forma unívoca las nuevas funciones, actualizaciones y problemas corregidos.

Parches: un parche es una pieza de software que se puede descargar de forma adicional en un paquete de software, y que se instala en el software. Estos softwares se usan para hacer mejoras, actualizaciones, reparar errores o añadir una nueva funcionalidad.

Contenedores: Son paquetes de software que incluyen todos los elementos necesarios para ejecutar tus productos en cualquier entorno.

# Referencias

Avast. (2023). *Avast*. Obtenido de Avast: https://www.avast.com/es-es/c-what-is-an-ssl-certificate#:~:text=Un%20certificado%20SSL%20es%20un,el%20sitio%20web%20que%20visita.

CESUMA. (2023). *CESUMA*. Obtenido de CESUMA: https://www.cesuma.mx/blog/que-es-el-framework.html

Cloud, G. (2023). *Google Cloud* . Obtenido de Google Cloud : https://cloud.google.com/learn/what-are-containers?hl=es#:~:text=Los%20contenedores%20son%20paquetes%20de,port%C3%A1til%20personal%20de%20un%20desarrollador.

economipedia. (1 de Enero de 2021). *economipedia*. Obtenido de economipedia: https://economipedia.com/definiciones/convenio.html

GITHUD\_BRANDON. (s.f.). *github*. Obtenido de https://github.com/BrandonDv22345/Proyect\_ING.git

Hostinger. (2023). *Hostinger*. Obtenido de Hostinger: https://www.hostinger.mx/tutoriales/que-es-react

OpenWebinars. (2023). *OpenWebinars*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/

Oracle. (2023). *Oracle*. Obtenido de Oracle: https://www.oracle.com/mx/database/what-is-database/

PoliScience. (2023). *PoliScience*. Obtenido de PoliScience: https://poliscience.blogs.upv.es/acceso-abierto/repositorios-2/

revistabyte. (2023). *revistabyte*. Obtenido de revistabyte: https://revistabyte.es/comparativa/comparativa-10-contenedores/

SYDLE. (2023). *SYDLE*. Obtenido de SYDLE: https://www.sydle.com/es/blog/extreme-programming-602ee205da4d096809438c9c

WatchGuard. (2023). *WatchGuard*. Obtenido de WatchGuard: https://www.watchguard.com/es/wgrd-news/blog/consejos-para-practicar-la-codificacion-segura-y-ser-un-programador-responsable-1#:~:text=La%20codificaci%C3%B3n%20segura%20facilita%20a,que%20ayudan%20a%20prevenir%20ciberataques.

ZAPTEST. (2023). *ZAPTEST*. Obtenido de ZAPTEST: https://www.zaptest.com/es/pruebas-de-caja-negra-que-son-tipos-procesos-enfoques-herramientas-y-mucho-mas#:~:text=Las%20pruebas%20de%20caja%20negra%20consisten%20en%20probar%20un%20sistema,dise%C3%B1o%20que%20rodean%20al%20software.