**UNIVERSIDAD TECNOLÓGICA**

**EMILIANO ZAPATA DEL ESTADO DE MORELOS**

**DIVISIÓN ACADÉMICA DE TECNOLOGÍAS DE LA INFORMACIÓN Y DISEÑO**

**SISTEMA DE GESTIÓN DE CAPACITACIONES MÓDULO DE REPORTES PARA EL DEPTO. DE DESARROLLO ACADEMICO**

**REPORTE DE ESTADÍA**

**QUE PARA OBTENER EL GRADO DE:**

**TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE LA INFORMACIÓN**

**ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA**

|  |  |  |
| --- | --- | --- |
| **PRESENTA:**  **JOSE EMILIO ENRIQUEZ TORRES** | | |
| ASESORA EMPRESARIAL |  | ASESOR UNIVERSITARIO |
| **DRA. MARTHA FABIOLA WENCES DIAZ** |  | **M. T. I. MAURO BAHENA CASTRO** |

**EMILIANO ZAPATA, MOR., SEPTIEMBRE DE 2023**

|  |
| --- |
| **CONTENIDO** |

Índice de figuras

Índice de tablas

Agradecimientos

Resumen

Summary

[CAPÍTULO 1. DESCRIPCIÓN DEL PROYECTO 8](#_Toc139820096)

[1.1 Datos generales de la empresa 8](#_Toc139820097)

[1.2 Antecedentes del proyecto 9](#_Toc139820098)

[1.3 Objetivo general 9](#_Toc139820099)

[1.4 Objetivos específicos 9](#_Toc139820100)

[1.5 Justificación 10](#_Toc139820101)

[1.6 Alcances 10](#_Toc139820102)

[1.7 Restricciones 11](#_Toc139820103)

[CAPÍTULO 2. MARCO DE REFERENCIA 12](#_Toc139820104)

[2.1 Conceptos, metodologías y herramientas 12](#_Toc139820105)

[2.1.1 Conceptos y principios de la programación y POO 12](#_Toc139820106)

[2.1.2 Clean Architecture y conceptos y principios de SOLID 14](#_Toc139820107)

[2.1.3 Metodología de Scrum 16](#_Toc139820108)

[2.1.4 Conceptos de DMS 18](#_Toc139820109)

[2.1.5 Herramientas de desarrollo 21](#_Toc139820110)

[2.2 Propuesta de solución 24](#_Toc139820111)

[CAPÍTULO 3. DESARROLLO 26](#_Toc139820112)

[3.1 Inicio 26](#_Toc139820113)

[3.2 Planeación 26](#_Toc139820114)

[3.3 Ejecución 27](#_Toc139820115)

[3.4 Control 27](#_Toc139820116)

[3.5 Cierre 27](#_Toc139820117)

[CAPÍTULO 4. CONCLUSIONES 28](#_Toc139820118)

[4.1 Cumplimiento de objetivos 28](#_Toc139820119)

[4.2 Resultados 28](#_Toc139820120)

[4.3 Contribuciones 28](#_Toc139820121)

**REFERENCIAS**

**ANEXOS**

|  |
| --- |
| **ÍNDICE DE FIGURAS** |

[Figura 1.1Logotipo de la empresa 8](#_Toc103865183)

|  |
| --- |
| **ÍNDICE DE TABLAS** |

[Tabla 3.1 Involucrados y sus funciones 13](#_Toc103865184)

|  |
| --- |
| **AGRADECIMIENTOS** |

Texto libre escrito por el estudiante.

|  |
| --- |
| **RESUMEN** |

El texto del resumen debe redactarse para captar la atención del lector y describir brevemente lo que encontrará dentro del documento. Debe responder a las preguntas:

¿qué se hizo?, ¿cómo se hizo? y ¿qué impacto tuvo en la empresa?

De igual forma, debe incluir la estructura del documento descrita por capítulos.

|  |
| --- |
| **SUMMARY** |

Debe evitarse la traducción directa realizada por una herramienta tecnológica. Por ejemplo: Google Translator. Solicitar al alumno que sea revisado por un profesor de inglés en los horarios de asesoría.

|  |
| --- |
| DESCRIPCIÓN DEL PROYECTO |

## Datos generales de la empresa

La Universidad Tecnológica Emiliano Zapata (UTEZ) es una Universidad que ofrece planes de estudio para formar Técnicas y Técnicos Superiores Universitarios e Ingenieros e Ingenieras generando profesionistas competitivos, con altos niveles tecnológicos y sentido humanista para dar respuesta a las necesidades de los sectores social y productivo de la región y del país. El Centro de Desarrollo de Software de la Universidad Tecnológica Emiliano Zapata (CDS-UTEZ), es un centro de entrenamiento cuyo objetivo principal es la de formar estudiantes con habilidades de desarrollo y gestión de proyectos de TI, con apego a las necesidades de los clientes.



Figura .Logotipo de la empresa

CDS-UTEZ se encuentra ubicada en la Av. Universidad Tecnológica No. 1. Col. Palo Escrito, C. P. 62760, Emiliano Zapata, Morelos. La UTEZ cuenta con aproximadamente 42 profesores de tiempo completo, 226 de asignatura, 3 700 estudiantes de Técnico Superior Universitario y 1500 en Licenciatura e Ingeniería.

Este proyecto se realizará bajo la tutela de M.T.I. Mauro Bahena Castro que actualmente desempeña el cargo de administrador de recursos humanos y ambiente de trabajo y como Responsable de la Administración del Proyecto Específico (RAPE), y de Ángel Yazveck Alcocer Durán como Responsable del Desarrollo (RD) del proyecto descrito en este reporte.

## Antecedentes del proyecto

Actualmente el encargado del Departamento de Desarrollo Académico de la universidad lleva la gestión de las capacitaciones por medio de archivos Excel, lo cual se hace complejo de leer, así mismo los reportes que le solicitan genera una carga de trabajo y una dedicación de horas muy grandes. Por lo tanto, al contar con un sistema que le reúna la información y genere automáticamente los reportes, podrá agilizar su proceso.

## Objetivo general

Administrar de manera ágil y eficiente las capacitaciones aplicadas a los docentes de la UTEZ, a través de la generación de reportes, utilizando una aplicación web.

## Objetivos específicos

1. Diseñar la base de datos que utilizará la aplicación y definir los métodos de acceso a datos mediante la implementación de servicios Web REST.
2. Mantener una estructura del sistema basada en Clean Architecture, donde se priorice la separación de capas y dependencia hacia el dominio para lograr un sistema modular y escalable.
3. Implementar los módulos para la gestión de las capacitaciones, docentes, áreas gestoras y planes de desarrollo docente con el objetivo de generar reportes más precisos.
4. Implementar funciones para la carga masiva de datos de las entidades de capacitaciones y docentes mediante archivos Excel con una plantilla específica para el ágil registro y actualización de la información en el sistema.

## Justificación

El proceso actual de administración de capacitaciones mediante archivos Excel es complejo y dificulta la lectura y seguimiento de la información, la aplicación web permitirá agilizar y centralizar la gestión de las capacitaciones, mejorando la eficiencia y optimizando la gestión de la información, a su vez facilitará la gestión de reportes personalizados, brindando información específica según los criterios de filtrado establecidos permitiendo obtener datos relevantes según la necesidad.

## Alcances

* El sistema no contara con una gestión de diversos usuarios o roles, su uso está dirigido específicamente para el departamento de desarrollo académico.
* El sistema gestionara la información de los catálogos de plan de desarrollo docentes, capacitaciones parte 1 y 2, docentes, áreas gestoras, programas educativos y planes de desarrollo docente.
* Los módulos de capacitaciones y docentes contaran con una función de registros de una carga masiva por medio de archivos Excel con una plantilla específica.
* El sistema brindara la posibilidad de la recuperación de contraseña a través de una solicitud de recuperación de contraseña.
* Permitir a la persona responsable del proceso “Departamento de Desarrollo Académico” obtener información concreta y resumida de la base de datos.
* El sistema no gestionara ni administrara las constancias que reciban los docentes después de una capacitación.
* El sistema brindara diversas formas de filtrado de información para la generación de reportes en formato de archivo PDF.
* Brindar al usuario la posibilidad de registrar, visualizar y modificar la información de los catálogos existentes en la plataforma.
* Brindar al usuario la capacitad de generar reportes de los catálogos de Docentes y capacitaciones.

## Restricciones

* El cliente cuenta con la infraestructura necesaria para la puesta en marcha de la aplicación web.
* El cliente brindará asesoría para la resolución de dudas sobre los procesos a automatizar mediante el sistema.

|  |
| --- |
| MARCO DE REFERENCIA |

## Conceptos, metodologías y herramientas

Para el completo entendimiento del proyecto es útil comprender bajo que tecnologías, principios y procesos fue desarrollado. En base a lo recién mencionado, en este capítulo se hará exposición de los conceptos necesarios para el desarrollo inicial del proyecto.

### Conceptos y principios de la programación y POO

**Programación**

El objetivo de un programa se puede definir como un proceso que se realiza sobre ciertos datos con el fin de obtener un resultado en específico. Entonces la programación se podría entender como el momento en el que se define para qué sirve el proceso a realizar, qué datos se usarán, qué resultados se obtendrán y cómo se llevará a cabo ese proceso. Para poder proporcionarle las instrucciones a la computadora sobre cómo debe hacerlo a través de un lenguaje de programación.

**POO**

La Programación Orientada a Objetos (POO) es una técnica en la que se intenta disminuir el coste del programa aumentando la eficiencia mediante el uso de “objetos”, que se definen como el uso de entidades con atributos y formas de operar sobre los datos de forma específica mediante métodos.

**Polimorfismo**

Uno de los principios fundamentales de POO. Se entiende al polimorfismo como una cualidad que tienen los objetos de poder responder de diferente manera al mismo mensaje. Esta característica facilita el poder asignarles nombre a algunos métodos, ya que todas las clases derivadas de la original afectara de manera similar, pero podrán modificar su manera de responder a ella.

Esta capacidad de respuesta variable del polimorfismo se basa en la relación de herencia entre clases y la sobreescritura de métodos.

Al utilizar polimorfismo, se pueden agregar nuevas clases derivadas en el futuro sin afectar el funcionamiento de las clases existentes, lo que facilita la incorporación de nuevas funcionalidades y la adaptación a cambios en los requisitos del sistema.

**Herencia**

La herencia como un principio fundamental de POO, se puede comprender fácilmente cuando se compara con la vida cotidiana cuando un padre transmite a sus hijos algo de él. En la programación se observa como la sucesión de atributos y métodos de una clase padre o base a otra derivada de la misma, haciendo que varias clases puedan tener un comportamiento similar.

**Encapsulamiento**

Este es el principio que consiste en ocultar los detalles de una clase y protegerlos mediante modificadores de acceso (public, private y protected) y solo dejar expuesta una interfaz publica para interactuar. Esto permite una mejor integridad de los datos y mejor control sobre la manipulación de los mismos.

### Clean Architecture y conceptos y principios de SOLID

**Clean Architecture**

Es un diseño de arquitectura de software propuesto por Robert C. Martin. Es un conjunto de principios y prácticas para la creación de sistemas flexibles, mantenibles siguiendo los principios de SOLID que son principios de diseño de software que promueven el modularidad, la flexibilidad, entre otras cosas.

**Principio de Responsabilidad Única (Single Responsability Principle)**

Establece que una clase debe tener una sola única responsabilidad o función en el sistema o razón para cambiar. Explicando que cada clase debe estar enfocada en realizar solo una tarea favoreciendo el modularidad y cohesión en el código. Esto porque si una clase tiene más de una responsabilidad se vuelve más difícil de entender, mantener y reutilizar.

**Principio de Abierto/Cerrado (Open/Closed Principle)**

Este principio indica que las entidades en el sistema como lo son las clases, módulos y funciones deben estar abiertas para cualquier extensión, pero cerradas para modificaciones. En resumen, promueve que el código este abierto a crecer sin la necesidad de modificar el código ya existente a través de nuevos módulos.

**Principio de Sustitución de Liskov (Liskov Substitution Principle)**

Este es un principio fundamental para la programación orientada a objetos, basándose en los conceptos de herencia y el polimorfismo, establece que los objetos de una clase deben poder ser sustituibles por sus clases derivadas sin que se altere el comportamiento esperado del programa. Esto significa que, si hay un objeto B que es derivado de un objeto A, el objeto B puede usarse como un objeto A sin ningún problema, esto a partir de que cualquier método o atributo que exista en la clase base puede ser invocado en cualquiera de sus objetos descendientes y los resultados deberán ser coherentes y conforme a los contratos ya establecidos.

**Principio de Segregación de Interfaz (Interface Segregartion Principle)**

Este principio establece que el cliente no debe depender de interfaces que no necesita utilizar. Las interfaces deberán ser lo más específicas posibles, intentando ofrecer solo los métodos y funcionalidades necesarias promoviendo la idea de dividir las interfaces que sean muy grandes y poco cohesivas en unas más pequeñas evitando la sobrecarga y dependencia innecesaria de funcionalidades que no se utilizan, reduciendo la complejidad en el diseño del sistema. Esto proporciona una mayor claridad en la comunicación entre componentes, ya que cada uno puede enfocarse en las operaciones que necesita realizar y no en aquellas que son irrelevantes para su contexto.

**Principio de Inversión de Dependencia (Dependency Invercion Principle)**

El principio de DIP establece que los módulos de alto nivel no deberán ser dependientes de los de bajo nivel ni viceversa, ambos deben de depender de abstracciones, promoviendo las clases abstractas e interfaces en lugar de otras implementaciones concretas. Además, las abstracciones no deberán de depender de detalles específicos sino al revés provocando una arquitectura más sólida, flexible y facilitando la introducción de nuevos componentes, la reutilización de código y las pruebas al sistema.

### Metodología de Scrum

**Marco de trabajo Ágil**

Scrum es una estructura para el trabajo ágil, en la actualidad es altamente empleado en el desarrollo de software, sistemas y proyectos de gestión. Se centra principalmente en la entrega iterativa, colaboración entre los integrantes del equipo y una adaptación continúa, permitiendo responder de manera más flexible a cambios en requisitos y entregas de avance de una forma incremental.

Scrum se basa en tres pilares fundamentales:

Transparencia: Que todo tipo de información que se vaya generando sobre el proyecto sea clara, visible y comprensible para todos los miembros del equipo, algo que fomenta la confianza y facilita la toma de decisiones.

Inspección: Los equipos deben de inspeccionar su trabajo, el progreso y los resultados obtenidos. Mediante la revisión continua de la calidad de un producto, se pueden identificar más oportunidades de mejora y poder tomar medidas para las desviaciones.

Adaptación: Se fomenta la adaptación continua gracias a los aprendizajes adquiridos durante el desarrollo del proyecto para poder dar respuesta a cualquier tipo de cambio que surja. Esto es de suma importancia, ya que permite que el equipo pueda ajustarse ágilmente a algún nuevo lineamiento.

**Roles y responsabilidades**

A las personas que participan bajo la metodología de Scrum se les asigna un rol especifico que van a desempeñar durante el desarrollo del proyecto, cada rol tiene sus propias responsabilidades.

Los 3 roles principales dentro de Scrum son:

Scrum Master: Actúa como facilitador y defensor del proceso Scrum, asegurándose que se implemente correctamente y que todos los miembros sigan las prácticas y principios de Scrum. Otorga guía y apoyo a el Equipo de Desarrollo y al Product Owner durante la planificación, las estimaciones, seguimiento y en las mejoras continuas. También se encarga de eliminar cualquier obstáculo que pueda afectar la productividad, fomentar la colaboración entre los miembros del equipo y la comunicación afectiva. Otra de sus responsabilidades es asegurarse que se lleven a cabo las reuniones Scrum, como lo son las diarias de seguimiento, las de revisión y las de retrospectiva.

Product Owner: Comúnmente se le ve como la persona encargada de la gestión de los requisitos y del Product Backlog, y es cierto, pero su papel es fundamental durante el desarrollo con Scrum. El Product Owner se encarga de mantener el Backlog del producto bien estructurado, detallado y priorizado. Entiende perfectamente cual es la meta que el producto debe alcanzar en todo momento, siendo capaz de describirlo y transmitirlo de manera efectiva. Tiene la autoridad de tomar cualquier decisión que afecte al producto y actúa como portavoz del cliente, transmitiendo sus necesidades y demandas al equipo de desarrollo, pero trabajando en estrecha colaboración con ellos, garantizando que el producto final cumpla con las expectativas del cliente.

Equipo de Desarrollo: es un grupo de entre 3 y 9 profesionales, el cual su trabajo principal como su nombre lo dice, es el desarrollo del producto, trabajando de manera autónoma y colaborativa para entregar incrementos de alta calidad. Son responsables de estimar su propio esfuerzo y tiempo para entregar un incremento de calidad y completo al final de cada sprint. El equipo participa en reuniones diarias de seguimiento, reuniones de revisión y de retrospectiva al final de cada sprint. Estas reuniones proporcionan retroalimentación para mejorar continuamente el proceso de desarrollo del equipo.

**Artefactos**

1. Product Backlog: De forma muy resumida se puede considerar al producto backlog como una lista de todas las funcionalidades, requisitos, casos de uso, mejoras y correcciones que deben realizarse al producto. No es necesario que al inicio este sea una lista completa y larga; puede comenzar con dos o tres requerimientos de alta prioridad e ir creciendo a medida que se comprenden mejor las necesidades del producto. Este es gestionado exclusivamente por el Product Owner.

2. Sprint Backlog: Se trata de elementos con los cuales se va a estar trabajando durante la etapa del sprint. Estos elementos son elegidos por el equipo de desarrollo y será gestionado por ellos mismos. Su propósito es mantener a todo el equipo informado y actualizado durante toda la iteración aquellos elementos que aún no han comenzado a desarrollarse, los que están en proceso y cuales están completamente terminados.

3. Incremento: Es el resultado del sprint. Consiste en elementos del Product Backlog que han sido completados durante el Sprint. El incremento es un elemento tangible y entregable, con características que lo hacen listo para entrega o uso.

### Conceptos de DMS

**DMS**

Conocida también como Desarrollo en Sistemas de Información, es un enfoque estructurado y sistemático para el desarrollo de proyectos de software. Su objetivo principal es garantizar la entrega exitosa de sistemas de información que cumplan con los requerimientos del cliente, siendo funcionales, confiables de calidad. Para lograr esto, usa una serie de fases y actividades que permiten planificar, analizar, diseñar, implementar y mantener estos sistemas de manera efectiva a partir de un proceso secuencial que abarca desde la identificación de requerimientos hasta la liberación del sistema y su mantenimiento.

**Requisitos**

Es el proceso de recopilación y análisis inicial de los requisitos del sistema que se va a construir. Se identifican y se documentan las necesidades funcionales y restricciones del sistema, estableciendo así las bases para el diseño posterior.

La recopilación de requisitos implica la comunicación con los clientes para comprender sus necesidades y expectativas. Estos requisitos pueden incluir funcionalidades específicas, restricciones de rendimiento, seguridad, entre otros aspectos relevantes para el desarrollo del sistema.

**Análisis y Diseño**

Esta fase se centra en la comprensión de los requisitos recopiladas y en la creación de una solución técnica que cumpla con estos mismos. Esta fase se divide en dos etapas:

Análisis: En esta etapa implica el estudio y comprensión de los requisitos del sistema, con el objetivo de tener los suficientemente claras las necesidades de los usuarios y los procesos involucrados. Utilizando una variedad de técnicas y herramientas para modelar el sistema, como lo son diagramas de flujo, diagramas de casos de uso, entrevistas, entre otros.

Diseño: Aquí se traducen todos los requisitos en soluciones técnicas, definiendo los componentes del sistema, su estructura, las interfaces y las estrategias de implementación. Se elaboran los diseños de las bases de datos y algoritmos necesarios, si corresponde.

**Programación**

Se implementa el diseño creado en la fase anterior, utilizando un lenguaje de programación especifico. Los desarrolladores traducen las especificaciones, requisitos y diseño en código, siguiendo las mejores prácticas de codificación y las herramientas adecuadas.

Este desarrollo en sistemas implica la escritura y la codificación de programas informáticos que permitirán automatizar los procesos y funcionalidades necesarios para el sistema. Aquí es donde los desarrolladores utilizan lenguajes de programación para crear el software que cumpla con los objetivos establecidos en las etapas anteriores del ciclo de vida del desarrollo de sistemas de información.

**Pruebas**

En la fase de pruebas se busca verificar y validar el software desarrollado en la fase de programación. Las pruebas llegan a abarcar distintos niveles o métodos, como lo

son las pruebas unitarias, pruebas de integración, pruebas de sistema, de seguridad, pruebas de estrés y pruebas de aceptación. El principal objetivo de esta etapa es identificar y corregir errores y defectos que el software tenga antes de su lanzamiento y que cumpla con los requisitos establecidos.

**Liberación**

Una vez el proceso de pruebas a concluido corrigiendo todos los errores identificados, se produce la fase de liberación. Aquí se prepara el software para su instalación en un entorno de producción. Esto implica que sea empaquetado y distribuido en un formato para su adecuada instalación, documentando los pasos y toda la configuración aplicada. También puede llegar a incluir la realización de algunas pruebas ya en el entorno de producción antes de que este disponible para los usuarios finales.

### Herramientas de desarrollo

**TypeScript**

Es un lenguaje de programación basado en JavaSript, agregando varias características adicionales mejorando la estructura y mantenibilidad de los programas desarrollados con él. Agrega la opción de agregar tipos estáticos a variables y elementos, brindando verificación de tipos en tiempo de compilación para la detección de errores antes de ejecutar el código.

TypeScript también admite la programación orientada a objetos y proporciona conceptos como clases, interfaces y herencia. Además, se integra bien con diferentes herramientas de desarrollo, lo que facilita su adopción en proyectos existentes.

**Node.js**

Es una plataforma de desarrollo de aplicaciones web escalables y de alto rendimiento del lado del servidor que utiliza JavaScript. Este se ejecuta fuera del navegador y se basa en un modelo E/S sin bloqueo y basado en eventos. Esto significa que las operaciones de entrada y salida no se llevan a cabo durante la ejecución del programa, sino que utiliza devoluciones de llamadas a eventos para manejar eficientemente estas operaciones. Gracias a su enfoque asincrónico y a su capacidad para manejar múltiples solicitudes simultáneamente sin bloquear el hilo principal de ejecución, lo que resulta en una mayor capacidad de respuesta y rendimiento para las aplicaciones.

Node.js tiene un amplio ecosistema de paquetes a través de npm, con los que los desarrolladores pueden crear diversos tipos de aplicaciones desde servidores y APIs, hasta aplicaciones en tiempo real.

**Vue.js**

Es un marco de trabajo de JavaScript que se utiliza para crear interfaces de usuario interactivas en aplicaciones web. Se enfoca en la construcción de la capa visual de una aplicación y utiliza componentes reutilizables para facilitar el desarrollo. Una de las características distintivas de Vue.js es su capacidad para realizar actualizaciones automáticas de la interfaz de usuario cuando los datos subyacentes cambian, lo que permite una experiencia fluida.

Vue.js sobresale por su facilidad de uso y su sintaxis clara y concisa. Esto brinda una curva de aprendizaje fácil y flexible. El enfoque de Vue.js en la reactividad permite un enlace bidireccional de datos, lo que significa que los cambios realizados en la interfaz de usuario se reflejan automáticamente en los datos subyacentes y viceversa. También ofrece características avanzadas como manejo de eventos, enrutamiento y gestión de estados.

**PostgreSQL**

Es un sistema gestor de base de datos relacionales de código abierto y altamente escalable. Este proporciona un entorno robusto y confiable para almacenar y administrar grandes volúmenes de datos estructurados.

Se destaca por su capacidad de manejar transacciones complejas y garantiza la integridad de los datos. Ofrece un conjunto completo de características de base de daros, como consultas SQL avanzadas, funciones almacenadas, disparadores, vistas y soporte a índices.

Además, otra ventaja de PostgreSQL es su capacidad de extensibilidad al contar con extensiones que pueden ampliar y enriquecer aún más su funcionalidad.

**API RESTful**

Las APIs son conjuntos de especificaciones y reglas que se emplean para crear y combinar el software de diferentes aplicaciones. Se les suele describir como un acuerdo o contrato entre el proveedor de información y el usuario, en el cual se establece que datos o servicios necesita el consumidor (en su solicitud) y que debo proporcionar el proveedor (en su respuesta).

REST no es un protocolo o un estándar, sino más bien un conjunto de principios arquitectónicos que establecen limites y pautas. Para que una API sea considerada RESTful, debe cumplir ciertos aspectos:

* Utiliza una arquitectura cliente-servidor, donde se manejan las solicitudes y respuestas a través del protocolo HTTP.
* Se basa en comunicación sin estado, lo que significa que cada solicitud es independiente y no se almacena información del cliente entre ellas.
* Define una interfaz uniforme para la transferencia de información, lo que implica que se utiliza una estructura común para identificar los recursos y manipularlos mediante métodos HTTP y representarlos en un formato especifico.

**JWT**

JSON Web Token (JWT) es un estándar abierto que define un formato compacto y seguro para transmitir información entre partes como un objeto JSON. Es ampliamente utilizado en aplicaciones web y APIs como un mecanismo de autenticación y autorización.

Este ofrece diversas ventajas como la posibilidad de transmitir la información de manera segura y autocontenida. Además, al ser firmado digitalmente, el receptor puede verificar la autenticidad del token y confiar en los datos contenidos sin tener que consultar una base de datos o realizar llamadas adicionales al servidor.

**Visual Estudio Code**

Es un editor de código desarrollado por Microsoft que se ha convertido en una herramienta popular entre desarrolladores de software. Proporciona un entorno de desarrollo ligero y altamente personalizable que admite una amplia variedad de lenguajes de programación y frameworks.

Se caracteriza por su interfaz fácil de usar y su amplia gama de extensiones disponible. Estas extensiones permiten a los usuarios personalizar el editor y agregar nuevas funcionalidades según sus necesidades.

El editor también ofrece características estándar como resultados de sintaxis, autocompletado, búsqueda, reemplazo avanzado y navegación por el código.

## Propuesta de solución

La integración de las herramientas, tecnologías y métodos propuestos en el proyecto se plantea con el objetivo de lograr una solución completa y eficiente que satisfaga las necesidades del departamento. La combinación de estos elementos trabajando en conjunto permitirá alcanzar los siguientes beneficios al producto:

* Organización y Estructura: La aplicación de la Programación Orientada a Objetos junto con la arquitectura limpia proporcionara una estructura clara y modular para el desarrollo del sistema. Permitiendo obtener una mejor organización del código facilitando su mantenimiento, extensibilidad y reutilización de componentes.
* Experiencia de usuario moderna: La utilización de Vue 2 como framwork de desarrollo para el frontend brindara una interfaz de usuario fluida. Su enfoque basado en componentes permitirá una construcción ágil y modular de la interfaz, lo que dará como resultado una respuesta más rápida a las acciones del usuario.
* Eficiencia en el proceso de datos: la combinación de Node.js como entorno de ejecución para el servidor y PostgreSQL como gestor de la base de datos garantizara un procesamiento eficiente y escalable de los datos de capacitaciones.
* Mantenibilidad: La adopción de TypeScript agregara un tipado estático al código de JavaScript, mejorando su calidad y legibilidad, facilitando la detección temprana de errores, promoviendo un desarrollo más seguro y eficiente.
* Enfoque ágil en la gestión del proyecto: La implementación de la metodología de Scrum permitirá una gestión eficiente del proyecto, con entregas iterativas y una comunicación efectiva entre el equipo de desarrollo y los clientes.

La integración de estas herramientas permitirá una solución completa y eficiente, optimizando la gestión de la información, garantizando los procesos y mejorando la experiencia para el departamento. Esta combinación de elementos ofrece una opción óptima para lograr los objetivos establecidos y satisfacer las necesidades específicas del proyecto.

|  |
| --- |
| DESARROLLO |

## Inicio

En primer lugar, se llevó a cabo la R1 de DMS, una reunión inaugural donde el Ingeniero Erick Mireles Merchant presentó a todo el equipo de desarrollo, compuesto por los Analistas Programadores (AP), así como a los responsables clave, como el Responsable de Desarrollo Ángel Yazveck Alcocer Durán y el Responsable de Proyecto M. T. I. Mauro Bahena Castro. Durante esta reunión, se proporcionó una descripción inicial del proyecto, incluyendo los módulos identificados, las estimaciones de tiempo, el cliente y los antecedentes del proyecto. Este primer encuentro marcó el inicio formal del proyecto y estableció un compromiso sólido entre el equipo de desarrollo y el cliente.

Posteriormente, se llevó a cabo la R2 de DMS, una reunión interna del equipo, con el objetivo de profundizar en la descripción inicial proporcionada por el Ingeniero Erick. Durante esta reunión, se recopilaron dudas iniciales y se comenzó a delinear la estrategia para la recolección de requisitos. Los AP aportaron su experiencia y conocimientos para proponer tecnologías y soluciones técnicas adecuadas para el desarrollo del proyecto, teniendo en cuenta las limitaciones y los objetivos establecidos.

En base a la disponibilidad del cliente, se programó la R3 de DMS, una reunión esencial en la que se estableció una interacción directa con el cliente. Durante esta reunión, se exploró a fondo el propósito del sistema y se inició la recolección detallada de requisitos. Gracias a un diálogo abierto y una comunicación fluida, se resolvieron dudas y confusiones respecto a la descripción inicial, lo que permitió un entendimiento más profundo de las necesidades y expectativas del cliente.

Para formalizar los acuerdos alcanzados en la reunión R3 y tener un registro claro de los puntos clave tratados, se elaboró una minuta detallada. Este documento fue fundamental para mantener un seguimiento coherente del progreso del proyecto y asegurar que todos los compromisos acordados se cumplieran de manera adecuada.

Además, se reconoció la importancia de capacitar al equipo de desarrollo en las tecnologías y metodologías seleccionadas para el proyecto. Se realizaron varias capacitaciones, incluyendo el aprendizaje de Vue JS y TypeScript, tecnologías elegidas para el desarrollo del frontend. También se llevaron a cabo capacitaciones sobre los procesos y metodologías utilizados en CDS, como la documentación requerida según los estándares de Moprosoft y CMMI nivel 3, así como la implementación de una submetodología basada en Scrum. Estas capacitaciones garantizaron que el equipo estuviera debidamente preparado y equipado con las habilidades necesarias para abordar el proyecto de manera eficiente y efectiva.

Tabla 3.1 Involucrados y sus funciones

|  |  |  |
| --- | --- | --- |
| **INVOLUCRADOS EN EL PROYECTO** | | |
| **Nombre de la persona** | **Función** | |
| I. T. I. José Christian Narváez Figueroa | | Sponsor  *Interno* |
| M.A. Jose Dassaev Estrada López | | Cliente  *Externo* |
| Dra. Martha Fabiola Wences Díaz | | Asesor  *Externo* |
| M. T. I. Mauro Bahena Castro | | Administrador del proyecto  *Interno* |
| Ángel Yazveck Alcocer Durán | | Responsable del proyecto  *Interno* |
| Departamento de Desarrollo Academico | | Usuario final  *Externo* |

## Planeación

Esta sección debe describir las acciones realizadas para planificar el proyecto y deberá contener como mínimo un documento que defina los requerimientos del proyecto y un cronograma de actividades (o diagrama de Gantt) que indique tiempo de duración y entregables. Adicionalmente, se podrán añadir documentos que defina tu asesor universitario de acuerdo con el tipo de proyecto específico.

## Ejecución

En esta sección se deben describir, paso a paso, las acciones realizadas para desarrollar el proyecto. Estas acciones deben tener un orden cronológico coherente. Similar a un tutorial.

Normalmente se escribe en pasado, distinguiendo los elementos del estado actual del proyecto (escrito en presente) y aquellos que se tendrán como resultado de una acción futura (escritos en futuro).

## Control

Esta sección debe describir las acciones realizadas para llevar el control del proyecto. Los elementos por colocar pueden variar dependiendo del proyecto. Se pueden colocar elementos como: descripción de reuniones periódicas para el seguimiento del proyecto, seguimiento del proyecto considerando la planeación de actividades en comparación con el avance real, bitácora de cambios en el proyecto o seguimiento de riesgos.

## Cierre

Esta sección debe describir las acciones realizadas para concluir el proyecto de manera formal. Debe incluir al menos las acciones realizadas para entregar el proyecto (reunión), la descripción de los entregables del proyecto al cliente y la aceptación del cliente (o carta de liberación).

|  |
| --- |
| CONCLUSIONES |

## Cumplimiento de objetivos

Se debe redactar una descripción que indique cómo se cumplieron los objetivos (generalmente los específicos) y por qué se dice que se cumplieron o por qué no se cumplieron. Para ambos casos, se debe argumentar la razón.

## Resultados

Se deben describir los entregables finales, probados y en operación, así como los beneficios obtenidos por la empresa como resultado del proyecto.

## Contribuciones

Se deben describir las aportaciones a la empresa. Generalmente, no medibles. Por ejemplo: se utilizó una metodología nueva para el desarrollo. ¿Qué se utiliza ahora en la empresa que no se usaba antes?

De igual forma, se debe describir como benefició el proyecto a la formación profesional

del alumno. NO ESCRIBIR EN PRIMERA PERSONA. Al igual que el resto del documento, se debe escribir en tercera persona.

|  |
| --- |
| **REFERENCIAS** |

**No hay ninguna fuente en el documento actual.**

|  |
| --- |
| **ANEXOS** |

**ANEXO A. NOMBRE DEL ANEXO**

Los anexos, además de tener una letra, deberán indicar un nombre para poder identificarlos. No son títulos, por lo que no aparecerán en el índice.

**ANEXO B. NOMBRE DEL ANEXO**

**Estilos del formato de reporte**

Este archivo incluye 5 estilos definidos para títulos y texto del contenido, que puedes encontrar en la banda **Inicio** de Word, y son:

* **Título 1**: para los títulos de capítulo.
* **Título 2**: para los títulos de segundo nivel. Por ejemplo: 1.2
* **Título 3**: para los títulos de tercer nivel. Por ejemplo: 1.2.1
* **Descripción**: para los títulos de figuras y tablas.
* **Normal**: para el texto del contenido.



Todo el texto del documento debe tener aplicado al menos uno de estos estilos, según corresponda.

Se deben utilizar los estilos para que, al actualizar los índices de contenido, se adopten adecuadamente los textos colocados como títulos y se indiquen las páginas donde se encuentran.

Los títulos del índice de contenido deben ser los únicos elementos numerados.

**Portada**

Toda la portada deberá estar escrita en mayúsculas. Se debe elegir la palabra que defina correctamente a los asesores empresarial y universitario (asesor o asesora, universitario o universitaria); es decir, el texto colocado en rojo deberá sustituirse según corresponda.



Se debe colocar correctamente el nombre del programa educativo, considerando dos escenarios:

* Los nombres de programas que no tienen área (Ingenierías) se colocan en tamaño 16.
* Los nombres de programas que tienen área (TSU) deben seguir la regla anterior y, además, colocar el área en tamaño 14.

Es necesario que la portada tenga centrados tanto el nombre del proyecto como el grado. Esto implica que no solo debe estar centrado horizontalmente, sino también verticalmente en el espacio que corresponde a cada texto (área blanca para el nombre del proyecto y área azul para grado).

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

El lugar y fecha para el reporte se encuentra en la parte inferior de la portada. Se debe verificar que el mes que se indica sea un mes posterior a la finalización de la estadía.



**Formato del documento**

Se deberá respetar el formato del documento actual. Si por algún motivo se llegase a modificar, es necesario revisar los siguiente:

* Los márgenes del documento serán de 3cm. a la izquierda, 2.5 cm. a la derecha, superior e inferior.
* No se deberá incluir ninguna leyenda (ni líneas, ni títulos de trabajos, ni nombre del alumno) de encabezado.
* El pie de página solo deberá contener el número de la página fuente arial 10, alineado a la derecha.

**Cantidad de páginas**

* Las páginas del documento empiezan a contar después de la portada, que se considera la página 0, pero el número de página se hace visible hasta el Capítulo 1.
* La cantidad de páginas que deberá contener el reporte de estadía será de 40 como mínimo, las cuáles **se empiezan a contar a partir de la** **primera página del Capítulo 1** y **hasta la última página del capítulo 4**. No se consideran portada, índices, agradecimientos, resumen, referencias o anexos.
* Por ejemplo: un reporte que **inicia en la página 8** deberá **termina**r, al menos, **en la página 47**.

**Saltos de línea**

Se deberán considerar las siguientes reglas para los saltos de línea o renglones vacíos entre párrafos, títulos, imágenes y tablas:

* Se coloca un salto de línea entre título y párrafo; y entre párrafo y párrafo.
* Las imágenes y las tablas se consideran parte del texto o un párrafo. Si una imagen se encuentra con una tabla, o viceversa, se coloca un salto de línea entre ellas.
* Se colocarán dos saltos de línea entre párrafo y título siguiente.

Cuando un título queda solo al final de una página, existen dos opciones:

* Si el espacio restante es un salto de línea, se puede insertar un salto de página antes del título para moverlo a la siguiente página.
* Si el espacio restante es más de un salto de línea, se debe complementar la redacción del texto previo para evitar que el título quede solo al final de la página.

****

**Redacción y ortografía**

El documento debe estar escrito en tercera persona y evitar el uso del verbo ser seguido de un adjetivo. Por ejemplo:

* **No** **escribir**: Fue necesario, fue requerido, fue desarrollado, etc.
* **Utilizar**: Se necesitó, se requirió, se desarrolló.

Se debe evitar el uso de la palabra nosotros. Por ejemplo:

* **No escribir**: Tuvimos, realizamos, obtendremos, etc.
* **Utilizar**: Se tuvo, se realizó, se obtendrá, etc.

Es necesario revisar acentuación del documento (F7 con Word) y prestar especial atención en verbos escritos en pasado y futuro.

Se debe revisar que el texto este redactado procurando utilizar enunciados cortos. Generalmente se hace un uso excesivo de comas en lugares erróneos. Para verificar la necesidad de un punto, solicitar:

***Lee los párrafos en voz alta. Si durante la lectura te falta el aire para terminar, es posible evidencia de que requieres un punto o punto y coma.***

**Numeración y viñetas**

Se prefieren las viñetas sobre la numeración, salvo cuando sea necesario indicar un orden. **NO** se deben colocar saltos de línea entre numeración o viñetas. **Esto se debe observar a detalle, ya que se abusa de ello** con el fin de obtener hojas adicionales.

Se debe colocar un salto de línea antes de iniciar el listado con las viñetas.

**Referencias**

Las referencias se deben incluir en el texto y, en la medida de lo posible, seguir el formato APA utilizado por Microsoft Word. Para ello, se deben utilizar las herramientas para la gestión de referencias que proporciona el editor.

Se deberá evitar el plagio. Evitar el texto copiado y pegado. Si no hubiese otra opción, referenciarlo correctamente. Es preferible parafrasear o escribir con palabras propias.

**Referencia parafraseada:** comunica el mensaje del autor original con palabras propias y conservando el sentido en cuanto a organización, continuidad y coherencia. Generalmente, se inicia con una frase introductoria (así como, por tanto, de acuerdo con, según, etc.)

*De acuerdo con (Salazar, 2018), un programa computacional aplicado (PCA) se encarga de resolver un problema trivial en el campo...*

**Referencia textual:** es una cita fiel del texto original de un autor, palabra por palabra. El texto debe ponerse entre comillas, acompañados de la referencia.

*Para comprender los ejes corporales del cuerpo humano, "se han concebido planos imaginarios de referencia que atraviesan el cuerpo de forma que unos son perpendiculares a otros" (Palastanga, Field & Soames, 2000).*

**Las referencias deben tomarse de sitios confiables** y debe priorizarse la referencia de elementos obtenidos de libros, enciclopedias, artículos científicos, artículos de revistas académicas, entre otros.

Los documentos o sitios Web deben ser confiables, de entidades reconocidas como expertos o de renombre en el área.

Si se trata de una **entrada de blog**, debe tratarse de una persona reconocida en el campo (experto, no necesariamente popular). Por ejemplo: Richard Stallman, Bruno Munari, entre otros.

**Imágenes y tablas**

Si se añade una imagen es porque:

* Ayuda a la compresión del texto y debe describirse.
* El lector puede distinguir el texto y formas que incluye la imagen.
* Es necesaria y no se repite en algún otro lugar del documento.

El **tamaño de las imágenes** en el reporte quedará **a consideración del asesor universitario**, mismo que validará que sean necesarias y no se abuse de ellas para ocupar espacio.

Es válido colocar dos imágenes seguidas siempre que sea necesario para mantener el correcto formato del documento. **Se debe verificar que no se abuse de esto.**

Las imágenes y tablas NO deben colocarse en cuadros de texto o con diseño flotante (que pueda arrastrarse con el mouse sobre el documento). Las imágenes deben colocarse en línea con el texto.

Las imágenes deben estar referenciadas en el texto. Si se añade una imagen es porque ayuda a la compresión del texto y debe describirse. Por ejemplo, para hacer referencia a la figura con título Figura 2.1 en el texto descriptivo, se deberá hacer como:

“… en la Figura 2.1 se puede observar el funcionamiento paso a paso del algoritmo de búsqueda Bubble Sort para un arreglo de datos numéricos aleatorios.”

El rótulo y número, en el título de las imágenes, debe colocarse en texto normal (sin cursivas o negritas). Por ejemplo: Figura 2.1 Servidor de base de datos. El título de las tablas se coloca en la parte superior del encabezado, también en texto normal (sin cursivas o negritas).

Los títulos de imágenes o tablas se deben colocar sin punto.

Si una imagen no fue de creación propia, es necesario agregar la referencia al título de las imágenes. Por ejemplo:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

No debe existir algún salto de línea entre la imagen y su título de imagen. Deben cortarse los espacios en blanco alrededor de la imagen.



Cuando un título queda solo al final de una página, existen dos opciones:

* Si el espacio restante es un salto de línea, se puede insertar un salto de página antes del título para moverlo a la siguiente página.
* Si el espacio restante es más de un salto de línea, se debe complementar la redacción del texto previo para evitar que el título quede solo al final de la página. Por ejemplo:

