

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto *Deadline Collision Predictor***

Curso: ***Calidad y Pruebas de software***

Docente: *Patrick Cuadros Quiroga*

Integrantes:

**Vargas Gutierrez, Angel Jose (2020066922)**

**Angel Alessandro Chino Rivera (2021069830)**

**Juan Brendon Luna Juarez (2020068762)**

**Tacna – Perú**

***2025***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | MPV | ELV | ARV | 10/10/2020 | Versión Original |

INDICE GENERAL

1. Antecedentes 1
2. Planteamiento del Problema 4
   1. Problema
   2. Justificación
   3. Alcance
3. Objetivos 6
4. Marco Teórico
5. Desarrollo de la Solución 9
   1. Análisis de Factibilidad (técnico, económica, operativa, social, legal, ambiental)
   2. Tecnología de Desarrollo
   3. Metodología de implementación

(Documento de VISION, SRS, SAD)

1. Cronograma 11
2. Presupuesto 12
3. Conclusiones 13

Recomendaciones 14

Bibliografía 15

Anexos 16

Anexo 01 Informe de Factiblidad

Anex0 02 Documento de Visión

Anexo 03 Documento SRS

Anexo 04 Documento SAD

Anexo 05 Manuales y otros documentos

**1. Antecedentes**

En el contexto universitario, los estudiantes enfrentan múltiples asignaciones, exámenes y proyectos que deben ser entregados en plazos determinados. Sin embargo, la gestión de estos plazos suele depender de métodos tradicionales como agendas físicas, notas en hojas sueltas o recordatorios en el celular, los cuales pueden resultar ineficientes cuando se acumulan varias actividades. Esta situación genera estrés, falta de organización y, en ocasiones, incumplimiento de fechas importantes.

Con base en esta problemática común, surge la idea de **Deadline Collision Predictor**, un sistema desarrollado por estudiantes para estudiantes, cuyo propósito es facilitar la planificación académica mediante una plataforma inteligente que detecta conflictos de fechas y mejora la visualización de tareas en el tiempo.

**2. Planteamiento del Problema**

**a. Problema**

Los estudiantes universitarios suelen tener problemas para organizar sus actividades debido a la superposición de fechas límite. Esta falta de organización puede ocasionar entregas tardías, estrés académico e incluso reprobaciones. No existe una herramienta específica que permita detectar colisiones entre eventos académicos de forma automática y amigable.

**b. Justificación**

El desarrollo de este sistema se justifica por la necesidad real de los estudiantes de contar con una solución sencilla, eficaz y adaptada a su entorno. Deadline Collision Predictor no solo permitirá una mejor gestión del tiempo, sino que reducirá el estrés académico y aumentará la productividad, mejorando indirectamente el rendimiento estudiantil.

**c. Alcance**

El sistema permitirá a los usuarios:

* Registrar tareas y eventos usando lenguaje natural.
* Visualizar todos sus compromisos en un calendario interactivo.
* Detectar colisiones de fechas límite automáticamente.
* Consultar estadísticas personalizadas.
* Subir horarios en formatos como imagen o PDF.

El sistema será accesible desde navegadores web modernos y no requerirá instalación.

**3. Objetivos**

**Objetivo General**  
Desarrollar una plataforma web inteligente que permita a los estudiantes gestionar sus tareas y eventos académicos, detectando posibles colisiones de fechas límite y ofreciendo una visualización clara y ordenada.

**Objetivos Específicos**

* Permitir el registro de eventos mediante texto o voz.
* Visualizar los eventos en un calendario interactivo.
* Detectar conflictos automáticos entre tareas.
* Implementar un sistema de estadísticas que muestre progreso y cumplimiento.
* Desarrollar una interfaz moderna, intuitiva y funcional.

**4. Marco Teórico**

Deadline Collision Predictor se fundamenta en los siguientes conceptos y tecnologías:

* **Sistemas de gestión académica**: aplicaciones que ayudan a organizar tareas y actividades.
* **Procesamiento de Lenguaje Natural (NLP)**: permite interpretar mensajes escritos o hablados del usuario.
* **Arquitectura Cliente-Servidor**: modelo que permite separar la lógica de presentación y la lógica de negocio.
* **Bases de Datos Relacionales**: para almacenar y consultar eventos, usuarios y tareas.
* **Frameworks modernos**: como React para la interfaz, Node.js para el backend y PostgreSQL como base de datos.

**5. Desarrollo de la Solución**

**a. Análisis de Factibilidad**

**Factibilidad Técnica**

Desde el punto de vista técnico, el proyecto es factible. El sistema será implementado como una aplicación web, lo que permite su acceso desde cualquier dispositivo con un navegador moderno y conexión a internet, sin requerir hardware especializado por parte del usuario.

Para el desarrollo del sistema se utilizará Visual Studio Code como entorno de desarrollo principal y HeidiSQL para la administración de la base de datos. El backend será construido con Node.js y el frontend utilizará JavaScript, HTML y CSS. La base de datos estará basada en MySQL o MariaDB, tecnologías ampliamente compatibles y eficientes.

El sistema funcionará en servidores de desarrollo locales durante su construcción, y posteriormente podrá ser desplegado en servicios de nube como Heroku, Vercel o AWS, los cuales ofrecen planes gratuitos o de bajo costo ideales para proyectos de pequeña y mediana escala.

En cuanto a la infraestructura, se utilizarán equipos convencionales de cómputo (laptops y PCs) con sistemas operativos como Windows o Linux. La aplicación será compatible con navegadores ampliamente utilizados, como Chrome, Firefox y Edge. No se requiere red física especializada, ya que el acceso será completamente en línea. Se puede utilizar un dominio web gratuito o adquirir uno económico según el presupuesto disponible.

**Factibilidad Económica**

Económicamente, el proyecto también resulta viable. El uso de tecnologías de código abierto como Node.js, JavaScript, MySQL y herramientas gratuitas como Visual Studio Code y HeidiSQL reduce significativamente los costos de desarrollo. Esto permite que el sistema se construya sin necesidad de realizar inversiones elevadas en licencias o infraestructura.

Los principales costos del proyecto se centran en los recursos humanos, es decir, el tiempo y esfuerzo invertido por los desarrolladores. También puede considerarse un pequeño presupuesto para el posible alquiler de dominio y hosting web si se desea publicar la aplicación en línea de forma permanente.

A continuación, se describen los costos estimados:

* Costos humanos: Tiempo de trabajo del equipo de desarrollo (no remunerado si es proyecto académico).
* Costos de software: $0, al usar herramientas de código abierto y gratuitas.
* Dominio y hosting: entre $20 a $50 anuales si se desea adquirir un dominio propio y hospedaje profesional.
* Otros recursos: Equipos de cómputo ya disponibles, sin necesidad de inversión adicional.

En conclusión, los beneficios del sistema —mejor organización académica, reducción del estrés por colisiones de fechas y facilidad de uso— superan ampliamente los costos involucrados, haciendo que el proyecto sea completamente factible desde el punto de vista económico.

**Definir los siguientes costos:**

**Costos-Generales**

Dado que el proyecto está siendo desarrollado por estudiantes, los costos generales se mantienen al mínimo. Se utilizaron materiales de uso personal como cuadernos, lápices, bolígrafos y acceso a una impresora doméstica ocasionalmente para imprimir algunos documentos. Todo el trabajo fue realizado con laptops personales, lo cual evitó gastos adicionales en equipos.

**Tabla 1: Costos generales estimados**

| **Concepto** | **Costo estimado (S/.)** |
| --- | --- |
| **Útiles de oficina (papeles, lapiceros, etc.)** | **30.00** |
| **Impresiones ocasionales** | **20.00** |
| **Accesorios básicos (extensión, mouse, etc.)** | **25.00** |
| **Total** | **75.00** |

**Costos operativos durante el desarrollo**

No se alquiló oficina ni se incurrió en gastos de servicios básicos como luz, agua o internet, ya que todas las actividades del proyecto se realizaron de manera remota desde los hogares de los integrantes. Se aprovechó la conectividad existente y los recursos ya disponibles.

**Tabla 2: Costos operativos**

| **Concepto** | **Costo estimado (S/.)** |
| --- | --- |
| **Internet (uso compartido en casa)** | **0.00** |
| **Electricidad (uso personal)** | **0.00** |
| **Renta de espacio físico** | **0.00** |
| **Total** | **0.00** |

**Costos del ambiente**

En cuanto al entorno necesario para la implantación del sistema, se cuenta con lo esencial: laptops personales, conexión a internet en casa, y el uso de servicios gratuitos para desarrollo y pruebas. No se realizó inversión en dominios o infraestructura de red externa, ya que el sistema aún no está en fase de producción.

**Tabla 3: Costos del ambiente**

| **Concepto** | **Costo estimado (S/.)** |
| --- | --- |
| **Hosting gratuito (Vercel/Heroku)** | **0.00** |
| **Dominio (uso de subdominio gratuito)** | **0.00** |
| **Infraestructura de red personal (Wi-Fi hogar)** | **0.00** |
| **Total** | **0.00** |

**Costos de personal**

El proyecto fue desarrollado por tres estudiantes que asumieron todos los roles: análisis, diseño, programación, pruebas y documentación. No se realizó contratación externa ni se contemplan pagos, ya que es un trabajo académico. A continuación, se muestra una tabla referencial con el desglose de roles y tiempos estimados de dedicación.

**Tabla 4: Organización del equipo y horas dedicadas**

| **Integrante** | **Rol principal** | **Horas estimadas** | **Costo (S/.)** |
| --- | --- | --- | --- |
| **Estudiante 1** | **Backend (Node.js, base de datos)** | **30 h** | **0.00** |
| **Estudiante 2** | **Frontend (interfaz, JavaScript)** | **30 h** | **0.00** |
| **Estudiante 3** | **Documentación y pruebas** | **25 h** | **0.00** |
| **Total** |  | **85 h** | **0.00** |

**Costos totales del desarrollo del sistema**

El desarrollo del sistema no generó costos significativos, gracias al uso de tecnologías de código abierto, recursos personales y entornos gratuitos. No se ha invertido en licencias, servidores pagos ni personal externo. El costo total del proyecto se resume en la siguiente tabla:

| ***Categoría*** | ***Monto (S/.)*** |
| --- | --- |
| ***Costos generales*** | ***75.00*** |
| ***Costos operativos*** | ***0.00*** |
| ***Costos del ambiente*** | ***0.00*** |
| ***Costos de personal*** | ***0.00*** |
| ***Costo total estimado*** | ***75.00*** |

**Factibilidad Operativa**

El sistema Deadline Collision Predictor ofrece múltiples beneficios operativos, tanto para los estudiantes como para los entornos educativos en general. Facilita la organización personal, reduce el estrés académico al evitar colisiones de fechas límite y promueve una mejor planificación del tiempo. La interfaz amigable y el uso de lenguaje natural para registrar eventos hacen que el sistema sea accesible y fácil de usar para cualquier estudiante, sin necesidad de formación técnica previa.

En cuanto a la capacidad para mantener el sistema en funcionamiento, los estudiantes que desarrollaron el proyecto poseen los conocimientos necesarios para realizar mantenimiento básico, solucionar errores y aplicar mejoras en futuras versiones. El sistema puede mantenerse activo en plataformas de alojamiento gratuito, por lo que los costos de operación son prácticamente nulos. En caso de crecimiento, se podría escalar utilizando servicios cloud con bajo costo.

**Factibilidad Legal**

Desde el punto de vista legal, el proyecto no presenta conflictos con leyes o regulaciones vigentes. Al ser un sistema académico desarrollado por estudiantes, no se involucra en transacciones comerciales ni se manejan datos sensibles de tipo financiero o médico.

Sin embargo, se ha considerado el respeto por la privacidad del usuario. Si en el futuro se almacenasen datos personales (como nombres o correos), se tomarán medidas para cumplir con la Ley de Protección de Datos Personales vigente en el país. La plataforma puede implementarse con autenticación básica y sin recopilar información privada, lo que asegura el cumplimiento con normativas básicas de seguridad digital.

No se identifican conflictos con leyes laborales, de comercio, propiedad intelectual ni contratación pública, ya que el sistema es de uso académico y gratuito.

**Factibilidad Social**

Socialmente, el sistema tiene un impacto positivo. Está diseñado para mejorar la calidad de vida de los estudiantes al brindarles una herramienta útil, moderna y eficiente para organizar sus actividades académicas. El proyecto respeta los valores éticos y educativos, y se alinea con la necesidad actual de digitalización de las herramientas de estudio.

Además, no existe conflicto con códigos de conducta ni normas culturales, ya que el sistema promueve el orden, la responsabilidad y la planificación, valores universalmente aceptados. Su implementación es inclusiva, fácil de usar y adaptable a distintos contextos educativos, lo cual favorece su aceptación.

**Factibilidad Ambiental**

El impacto ambiental del sistema es mínimo o nulo, ya que se trata de una aplicación web que no requiere de infraestructura física adicional. No se utilizan recursos materiales ni se generan residuos, puesto que el sistema opera completamente en entornos digitales.

Además, al reducir la necesidad de imprimir recordatorios, agendas o planificadores físicos, el sistema puede incluso contribuir indirectamente a la reducción del uso de papel, lo que representa una ventaja ecológica. Por lo tanto, el proyecto es plenamente viable desde la perspectiva ambiental y está alineado con prácticas sostenibles.

**b. Tecnología de Desarrollo**

* **Frontend**: React.js
* **Backend**: Node.js con Express
* **Base de datos**: PostgreSQL
* **NLP**: spaCy / Dialogflow
* **Hosting**: Vercel (frontend) y Render/Heroku (backend)

**c. Metodología de implementación**

Se utilizó una metodología incremental basada en entregables intermedios y documentación técnica:

* Documento de Visión (describe el propósito y alcance general).
* Documento SRS (Software Requirements Specification) con todos los requerimientos funcionales y no funcionales.
* Documento SAD (Software Architecture Document), que detalla la estructura técnica y tecnológica del sistema.

**6. Cronograma**

| **Semana** | **Actividad** |
| --- | --- |
| 1 | Investigación y recolección de requisitos |
| 2 | Diseño de interfaz y modelado UML |
| 3 | Desarrollo del backend |
| 4 | Desarrollo del frontend y calendario |
| 5 | Implementación de NLP e IA |
| 6 | Pruebas del sistema |
| 7 | Documentación final y presentación |

**7. Presupuesto**

| **Categoría** | **Monto (S/.)** |
| --- | --- |
| Costos generales | 75.00 |
| Costos operativos | 0.00 |
| Costos del ambiente | 0.00 |
| Costos de personal | 0.00 |
| **Total estimado** | **75.00** |

El proyecto fue desarrollado por estudiantes, por lo tanto no se incurrió en costos de personal ni de infraestructura técnica adicional.

**8. Conclusiones**

Deadline Collision Predictor es una herramienta académica efectiva, con potencial para crecer y adaptarse a nuevas plataformas educativas. La solución demuestra ser técnica, operativa y económicamente viable. Su enfoque centrado en el usuario la convierte en una opción práctica para mejorar la planificación y reducir el estrés en el entorno universitario.