**UNIVERSIDAD ADVENTISTA DE BOLIVIA**

**FACULTAD DE INGENIERÍA**

**SISTEMA WEB PERSONALIZADO DE OPTIMIZACIÓN DE LA GESTIÓN DE PROYECTOS PARA “SANATORIUM BUSINESS”**

CASO DE ESTUDIO

Presentado como requisito para obtener el grado académico de

Licenciado en Ingeniería en Sistemas

Por:

Luis Gonzalo Garey Tito

COCHABAMBA, MARZO DE 2024

Tabla de contenido

[INTRODUCCIÓN 6](#_Toc151749063)

[CAPÍTULO I 7](#_Toc151749064)

[EL PROBLEMA 7](#_Toc151749065)

[1.1 Antecedentes 7](#_Toc151749066)

[1.2 Problema 8](#_Toc151749067)

[1.2.1 Situación Problemática 8](#_Toc151749068)

[1.2.2 Formulación del Problema 8](#_Toc151749069)

[1.3 Objetivos 9](#_Toc151749070)

[1.3.1 Objetivo General 9](#_Toc151749071)

[1.3.2 Objetivos Específicos 9](#_Toc151749072)

[1.4 Alcances 10](#_Toc151749073)

[1.5 Limites 10](#_Toc151749074)

[1.6 Justificación 11](#_Toc151749075)

[1.7 Factibilidad 12](#_Toc151749076)

[1.7.1 Factibilidad Técnica 12](#_Toc151749077)

[1.7.2 Factibilidad Económica 12](#_Toc151749078)

[1.7.3 Factibilidad Operacional 13](#_Toc151749079)

[CAPÍTULO II 14](#_Toc151749080)

[MARCO TEÓRICO CONCEPTUAL 14](#_Toc151749081)

[2.1 Sistema Web 14](#_Toc151749082)

[2.1.1 Definición 14](#_Toc151749083)

[2.1.2 Ventajas y desventajas de un sistema web 15](#_Toc151749084)

[2.1.3 Funcionamiento 16](#_Toc151749085)

[2.2 Gestión de proyectos 16](#_Toc151749086)

[2.2.1 Definición 16](#_Toc151749087)

[2.2.2 Registro tareas en proyectos 17](#_Toc151749088)

[2.3 Antecedentes tecnológicos 18](#_Toc151749089)

[2.3.1 Asana: 18](#_Toc151749090)

[2.3.2 Jira de Atlassian: 18](#_Toc151749091)

[2.3.3 Trello: 18](#_Toc151749092)

[2.3.4 Microsoft Project: 18](#_Toc151749093)

[2.3.4 Redmine: 19](#_Toc151749094)

[2.3.5 Monday.com: 19](#_Toc151749095)

[2.3.6 ClickUp: 19](#_Toc151749096)

[2.3.7 Smartsheet: 19](#_Toc151749097)

[2.4 Ingeniera de software 19](#_Toc151749098)

[2.4.1 Metodología de desarrollo 20](#_Toc151749099)

[2.4.1.1 Srum 21](#_Toc151749100)

[2.4.1.1.1 Ventajas y Desventajas 21](#_Toc151749101)

[2.4.1.1.2 Roles de Scrum 22](#_Toc151749102)

[2.4.1.1.3 Desarrollo Scrum 23](#_Toc151749103)

[2.4.1.1.4 Funcionalidad Scrum 23](#_Toc151749104)

[2.4.1.2 Kanban 24](#_Toc151749105)

[2.4.1.2.1 Características de Kanban: 24](#_Toc151749106)

[2.4.1.2.2 Ventajas y Desventajas de Kanban 24](#_Toc151749107)

[2.4.1.3 XP (PROGRAMACIÓN EXTREMA) 25](#_Toc151749108)

[2.4.1.3.2 Principios y Valores 25](#_Toc151749109)

[2.4.1.3.3 Características 26](#_Toc151749110)

[2.4.1.3.4 Ventajas y Desventajas 27](#_Toc151749111)

[2.4.1.3.5 Ciclo de Vida 27](#_Toc151749112)

[Etapas clave del ciclo de vida de XP: 28](#_Toc151749113)

[2.4.1.3.6 Reglas de la Metodología 29](#_Toc151749114)

[2.4.2.1 Métodos de recolección de información 30](#_Toc151749115)

[2.4.2.1 Encuesta 30](#_Toc151749116)

[2.4.2.2 Entrevista 30](#_Toc151749117)

[2.4.2.3 Observación 31](#_Toc151749118)

[2.4.2.3.1 Tipos de Observación 32](#_Toc151749119)

[2.4.2.2 Técnicas de recolección de información 32](#_Toc151749120)

[2.4.2.2.1 Cuestionarios Cerrados 32](#_Toc151749121)

[2.4.2.2.2 Cuestionarios abiertos 33](#_Toc151749122)

[2.4.3 Base de datos 33](#_Toc151749123)

[2.4.3.1 SQL Server 33](#_Toc151749124)

[2.4.3.2 Ventajas y Desventajas 33](#_Toc151749125)

[2.4.3.3 MySQL 34](#_Toc151749126)

[2.4.3.3.1 Características: 34](#_Toc151749127)

[2.4.3.3.2 Funcionalidad 35](#_Toc151749128)

[2.4.3.3.3 Ventajas y Desventajas 35](#_Toc151749129)

[2.4.4 ARQUITECTURA 37](#_Toc151749130)

[2.4.4.1 Arquitectura de Diseño MVC 37](#_Toc151749131)

[2.4.4.1.1 Características: 37](#_Toc151749132)

[2.4.4.1.2 Ventajas y Desventajas 37](#_Toc151749133)

[2.4.4.2 Arquitectura de Cliente – Servidor 38](#_Toc151749134)

[2.4.4.2.1 Funcionalidad 38](#_Toc151749135)

[2.4.4.2.2 Ventajas y Desventajas 39](#_Toc151749136)

[2.4.5 HERRAMIENTAS DE DESARROLLO 39](#_Toc151749137)

[2.4.5.1 Django 39](#_Toc151749138)

[2.4.5.1.1 Características Django 40](#_Toc151749139)

[2.4.5.1.2 Ventajas y Desventajas de Django 41](#_Toc151749140)

[2.4.5.2 Laravel 42](#_Toc151749141)

[2.4.5.2.1 Ventajas y Desventajas de Laravel: 42](#_Toc151749142)

[2.4.5.3 CodeIgniter 43](#_Toc151749143)

[2.4.5.3.1 Ventajas y Desventajas de CodeIgniter 43](#_Toc151749144)

[2.4.5.4 Bootstrap 44](#_Toc151749145)

[2.4.5.4.1 Funcionalidad 44](#_Toc151749146)

[2.4.5.4.2 Ventajas de Bootstrap 45](#_Toc151749147)

[2.4.5.5 Notación 45](#_Toc151749148)

[2.4.5.5.1 UML 45](#_Toc151749149)

[2.4.5.5.1.1 Características de UML 46](#_Toc151749150)

[2.4.6 TECNOLOGÍA DE DESARROLLO DE COMPONENTES 47](#_Toc151749151)

[2.4.6.1 Lenguajes de Programación 47](#_Toc151749152)

[2.4.6.1.1 Python 47](#_Toc151749153)

[2.4.6.1.1.1 Características 47](#_Toc151749154)

[2.4.6.1.1.2 Ventajas y Desventajas de Python 48](#_Toc151749155)

[2.4.6.1.2 Java 49](#_Toc151749156)

[2.4.6.1.2.1 Ventajas y desventajas de Java 50](#_Toc151749157)

[2.4.6.1.3 Javascript 51](#_Toc151749158)

[2.4.6.1.3.1 Ventajas y desventajas de Javascript 51](#_Toc151749159)

[2.4.7 PRUEBAS 53](#_Toc151749160)

[2.4.7.1 Pruebas Unitarias 53](#_Toc151749161)

[2.4.7.1.1 Herramientas para Pruebas Unitarias 53](#_Toc151749162)

[2.4.7.1.2 Métodos para Pruebas Unitarias: 53](#_Toc151749163)

[2.4.7.2 Pruebas de Sistema 54](#_Toc151749164)

[2.4.7.2.1 Herramientas para Pruebas de Sistema: 54](#_Toc151749165)

[2.4.7.2.2 Métodos para Pruebas de Sistema 55](#_Toc151749166)

[2.4.7.3 Pruebas de Aceptación 55](#_Toc151749167)

[2.4.7.3.1 Herramientas para Pruebas de Aceptación 55](#_Toc151749168)

[CAPÍTULO III 57](#_Toc151749169)

[Marco Practico 57](#_Toc151749170)

[3.1 Modelo de Negocio 57](#_Toc151749171)

[3.2 Procesos manuales actuales 57](#_Toc151749172)

[3.4 Procesos con el sistema 57](#_Toc151749173)

[3.5 Arquitectura 58](#_Toc151749174)

[3.6 Patrón de arquitectura 58](#_Toc151749175)

[3.7 Capa de Modelo 58](#_Toc151749176)

[3.8 Capa de Vista 58](#_Toc151749177)

[3.9 Capa de Controlador 58](#_Toc151749178)

[3.10 Grafico del patrón de arquitectura 59](#_Toc151749179)

[5.1 Bases de datos proyecto 59](#_Toc151749180)

[5.2 Diagrama base datos modelado 59](#_Toc151749181)

[6.1 Diseño interfaz 60](#_Toc151749182)

[6.2 Menú principal Inicio 60](#_Toc151749183)

[6.3 Lista Proyectos 60](#_Toc151749184)

[6.4 Calendario Proyectos 61](#_Toc151749185)

[6.5 Gestionar Proyectos 61](#_Toc151749186)

[6.6 Administrar Usuarios 62](#_Toc151749187)

[7.1 Conclusiones 63](#_Toc151749188)

[8.1 Recomendaciones 64](#_Toc151749189)

[9.1 Referencias bibliográficas 65](#_Toc151749190)

# INTRODUCCIÓN

En el dinámico mundo empresarial actual, la gestión eficiente de proyectos se erige como un factor crítico para el éxito organizacional. Este proyecto se centra en la creación de un sistema web personalizado destinado a optimizar los procesos de gestión de proyectos de Sanatorium Business, una empresa especializada en el desarrollo de estrategias digitales.

En el primer capítulo, exploramos la realidad de Sanatorium Business, identificando la carencia actual de un sistema integral para la gestión automatizada de proyectos. Además, establecemos nuestro objetivo general y evaluamos la viabilidad técnica, económica y operativa del proyecto.

El segundo capítulo sumerge al lector en la metodología aplicada, destacando el enfoque ágil adoptado y detallando los métodos de recolección y análisis de datos. Se aborda el proceso de ingeniería de requerimientos y se profundiza en las técnicas utilizadas para comprender las necesidades de los usuarios.

En el tercer capítulo, nos sumergimos en los detalles técnicos cruciales, como el diseño de la base de datos, la arquitectura del sistema, las tecnologías para el front-end y back-end, así como la estrategia de pruebas. Este apartado también incluye el modelo de negocio y los prototipos de interfaz.

Finalmente, el cuarto capítulo presenta los resultados obtenidos y las conclusiones derivadas de este proyecto, destacando los objetivos logrados y los beneficios que el sistema desarrollado aportará. Además, ofrecemos recomendaciones para posibles mejoras futuras, cerrando así un análisis exhaustivo y perspicaz de este emprendimiento.

# CAPÍTULO I

# EL PROBLEMA

# 1.1 Antecedentes

En el ámbito empresarial, la entidad privada "Sanatorium Business," con sede en Cochabamba, Bolivia, se ha destacado por su especialización en el desarrollo de estrategias digitales diseñadas para optimizar las operaciones comerciales de diversas empresas, con un enfoque particular en los sectores de ventas y marketing. Esta organización se ha destacado por su capacidad para analizar y fortalecer los procesos internos de las empresas, al mismo tiempo que digitaliza estrategias de promoción y posicionamiento de marcas en el entorno virtual. La clave de su éxito reside en su enfoque altamente personalizado, que les permite adaptar sus soluciones a las necesidades específicas de cada cliente en el ámbito de las ventas. La precisión e innovación son sellos distintivos de su trabajo, con el objetivo de revolucionar el mundo digital en el ámbito de las ventas mediante la implementación de ideas frescas.

Entre las principales funciones de una Notaría de Fe Pública se encuentran:

1. **Desarrollo de Estrategias Digitales:** "Sanatorium Business" se especializa en el desarrollo de estrategias digitales para optimizar las operaciones comerciales de diversas empresas. Esto implica el análisis y fortalecimiento de los procesos internos, así como la digitalización de estrategias de promoción y posicionamiento de marcas en el entorno virtual.
2. **Personalización de Soluciones en Ventas:** La entidad se destaca por su capacidad para proporcionar soluciones altamente personalizadas a sus clientes en el ámbito de las ventas. En lugar de seguir un enfoque predefinido, adaptan sus servicios a las necesidades específicas de cada empresa, con un enfoque claro en la creación de soluciones a medida.
3. **Innovación y Precisión en el Trabajo:**" Sanatorium Business" se distingue por su enfoque preciso e innovador en el desarrollo de estrategias digitales. Su objetivo es revolucionar el mundo digital en el ámbito de las ventas mediante la implementación de ideas frescas. La innovación y la precisión son sellos distintivos de su trabajo.
4. **Asesoramiento Eficiente y Optimización Comercial:** La misión de "Sanatorium Business" es brindar asesoramiento eficiente a las organizaciones comerciales, tanto en el mundo físico como en el digital. Su objetivo es optimizar el camino de los profesionales comerciales modernos, ayudándolos a tener éxito en un entorno digital en constante evolución.

# 1.2 Problema

## 1.2.1 Situación Problemática

1. **Falta de Sistema Centralizado de Gestión de Proyectos:** La ausencia de un sistema centralizado de gestión de proyectos se presenta como un problema significativo. Esto ha llevado a dificultades en la coordinación eficaz del trabajo, generando retrasos en la entrega de proyectos. La falta de una estructura clara también ha resultado en la duplicación de esfuerzos y una colaboración ineficaz entre los equipos de ventas y marketing. Además, la falta de definición de plazos concretos y el cumplimiento de los objetivos de los proyectos están afectados por esta carencia.
2. **Limitaciones en la Comunicación Interna:** La falta de un sistema centralizado ha limitado la comunicación entre los empleados de diferentes áreas. La ausencia de una plataforma centralizada para la interacción y el intercambio de información ha creado obstáculos en la colaboración efectiva entre los equipos. Esta limitación puede afectar negativamente la fluidez de la información crucial para el desarrollo de estrategias digitales y la ejecución de proyectos.
3. **Desafíos en la Definición de Plazos y Cumplimiento de Objetivos:** La carencia de una estructura clara, agravada por la falta de un sistema centralizado, ha dificultado la definición de plazos concretos y el cumplimiento de los objetivos de los proyectos. Esta situación puede afectar la eficiencia operativa y la capacidad de Sanatorium Business para satisfacer las demandas del mercado en un entorno digital en constante evolución, lo que contradice su misión de optimizar el camino de los profesionales comerciales modernos.

## 1.2.2 Formulación del Problema

La falta de un sistema centralizado de gestión de proyectos en Sanatorium Business, entidad especializada en estrategias digitales en Cochabamba, Bolivia, conlleva retrasos en la entrega, duplicación de esfuerzos y colaboración ineficaz entre equipos. Esta carencia impacta la definición de plazos, el cumplimiento de objetivos y limita la comunicación interna, comprometiendo la capacidad de brindar soluciones personalizadas y optimizar el camino de los profesionales comerciales en el entorno digital.

# 1.3 Objetivos

## 1.3.1 Objetivo General

El objetivo general de este proyecto es desarrollar un sistema web personalizado para la optimización de la gestión de proyectos de Sanatorium Business.

## 1.3.2 Objetivos Específicos

* Desarrollar un módulo que establezca un sistema integral de gestión de proyectos, permitiendo la planificación, seguimiento y evaluación eficiente de todas las iniciativas llevadas a cabo por Sanatorium Business.
* Desarrollar un módulo para crear un calendario de proyectos detallado que incluya fechas límite, hitos importantes y asignación de recursos, proporcionando una visión clara de la planificación temporal de todas las actividades de la empresa.
* Desarrollar un módulo para mejorar los procesos internos relacionados con la ejecución de proyectos, reduciendo los tiempos de entrega, minimizando la duplicación de esfuerzos y asegurando una colaboración más efectiva entre los equipos de ventas y marketing.
* Desarrollar un módulo para establecer un sistema de administración de usuarios que facilite la asignación de roles y responsabilidades, asegurando un acceso adecuado a la información y una colaboración eficiente entre los miembros del equipo.
* Desarrollar un módulo para facilitar la comunicación interna, promoviendo el intercambio efectivo de información entre empleados de diferentes áreas y contribuyendo así a una colaboración más fluida.

## 1.4 Alcances

* Implementar un módulo que facilite la gestión integral de proyectos en Sanatorium Business, permitiendo la planificación, seguimiento y evaluación eficiente de todas las iniciativas llevadas a cabo por la empresa.
* Implementar un módulo específico para la creación de un calendario de proyectos detallado. Este módulo incluirá funciones para establecer fechas límite, hitos importantes, y asignación de recursos, proporcionando una visión clara de la planificación temporal de todas las actividades de la empresa.
* Implementar un módulo que mejore los procesos internos relacionados con la ejecución de proyectos. Este módulo tiene como objetivo reducir los tiempos de entrega, minimizar la duplicación de esfuerzos y asegurar una colaboración más efectiva entre los equipos de ventas y marketing.
* Implementar un módulo específico para establecer un sistema de administración de usuarios. Este módulo facilitará la asignación de roles y responsabilidades, asegurando un acceso adecuado a la información y una colaboración eficiente entre los miembros del equipo.
* Implementar un módulo que facilite la comunicación interna entre empleados de diferentes áreas. Este módulo incluirá herramientas y canales que promuevan el intercambio efectivo de información, contribuyendo así a una colaboración más fluida dentro de la empresa.

## 1.5 Limites

1. La implementación inicial se centrará en la gestión de proyectos llevados a cabo por Sanatorium Business a partir del año 2020 en adelante. No se incluirán archivos o datos relacionados con proyectos anteriores a esta fecha en la fase inicial del sistema.
2. El sistema estará diseñado exclusivamente para el uso del personal autorizado de Sanatorium Business, incluyendo a los equipos de gestión de proyectos, ventas y marketing. Los clientes externos y otros usuarios no autorizados no tendrán acceso directo al sistema.
3. El alcance del sistema se limitará a los módulos previamente definidos, como la gestión integral de proyectos, calendario de proyectos, optimización de procesos internos, administración de usuarios y comunicación interna. Funcionalidades adicionales no contempladas en estos módulos no serán parte de la implementación inicial.
4. El presupuesto asignado cubrirá los costos relacionados con el desarrollo de los módulos especificados, así como la formación del personal en su uso. No se abordarán los costos operativos adicionales que puedan surgir después de la implementación inicial.
5. El sistema estará disponible únicamente para el acceso interno del personal autorizado de Sanatorium Business, asegurando la confidencialidad y seguridad de la información contenida en los módulos.
6. El sistema se enfocará exclusivamente en la gestión de información relacionada con proyectos de Sanatorium Business. No se incluirá la gestión de datos no relacionados con proyectos, garantizando así la coherencia y eficacia de las funcionalidades definidas.

## 1.6 Justificación

La implementación de los módulos propuestos en Sanatorium Business se justifica en base a la necesidad de optimizar la gestión integral de proyectos, alineándose con los objetivos específicos, alcances y limitaciones definidos. Estos módulos buscan mejorar significativamente la eficiencia en la ejecución de proyectos, permitiendo una planificación, seguimiento y evaluación más efectiva de todas las iniciativas llevadas a cabo por la empresa. Esta iniciativa responde al compromiso de Sanatorium Business de ofrecer soluciones altamente personalizadas a sus clientes en el ámbito de las ventas, destacándose por la precisión e innovación distintiva en su trabajo.

Además, la implementación de un módulo específico para la mejora de procesos internos busca reducir los tiempos de entrega, minimizar la duplicación de esfuerzos y fomentar una colaboración más efectiva entre los equipos de ventas y marketing. Esto fortalecerá la ejecución de proyectos, asegurando que Sanatorium Business continúe revolucionando el mundo digital en el ámbito de las ventas mediante la implementación de ideas frescas y estrategias eficientes.

La inversión en estos módulos también se justifica en la mejora de la coordinación temporal y la asignación de recursos a través de la implementación de un calendario de proyectos detallado. Este enfoque permitirá una visión clara de la planificación temporal de todas las actividades de la empresa, mejorando así la capacidad de adaptación y la implementación precisa de estrategias digitales. En conjunto, la implementación de estos módulos contribuirá no solo a la eficiencia operativa de Sanatorium Business, sino también a su capacidad para ofrecer servicios altamente personalizados, consolidando su posición en el mercado digital y brindando soluciones innovadoras a sus clientes.

# 1.7 Factibilidad

## 1.7.1 Factibilidad Técnica

El proyecto cuenta con factibilidad técnica dado que se dispone de los equipos y software necesarios para el desarrollo del sistema web personalizado. Se utilizará Python con el framework Django para la programación, lo cual es perfectamente viable con los recursos existentes.

Además, el personal de desarrollo de Sanatorium Business posee las habilidades y conocimientos para implementar el sistema utilizando estas tecnologías. Django ofrece una plataforma sólida y flexible para construir aplicaciones web escalables.

Otros aspectos técnicos favorables:

* El servidor de la empresa cuenta con capacidad suficiente para alojar el sistema web. Sus características son:
  + Procesador Intel i5
  + 8 GB de RAM
  + 500GB disco SSD
* Las computadoras de desarrollo tienen los requisitos mínimos para las herramientas a utilizar:
  + Sistema operativo Windows 10
  + 8 GB de RAM
  + Procesador Intel i7

## 1.7.2 Factibilidad Económica

* La factibilidad económica para la implementación del sistema propuesto para Sanatorium Business se sustenta en diversos aspectos financieros favorables:
* Se estima que el costo de desarrollo de los módulos es de Bs. 30,000. Este presupuesto abarca los recursos esenciales necesarios para la implementación, personal de desarrollo y capacitación del equipo. Esta inversión se justifica por la mejora significativa en la gestión de proyectos, la optimización de procesos internos y la facilitación de la comunicación interna, lo cual contribuirá al éxito continuo de Sanatorium Business en el mercado digital.
* Se prevé un gasto anual de Bs. 250 para el servidor (hosting) y Bs. 70 para el dominio del sistema web. Estos costos aseguran la disponibilidad y seguridad continuas del sistema en línea. Esta inversión es esencial para garantizar un servicio confiable y accesible, aspectos cruciales para el funcionamiento efectivo de los módulos propuestos.
* Aunque se requerirá una capacitación inicial para el personal de Sanatorium Business, este proceso se llevará a cabo de forma gratuita una vez que los módulos estén implementados. Dado que el personal ya posee conocimientos básicos en el uso de computadoras y navegadores web, se espera una transición eficiente a la nueva plataforma sin costos adicionales significativos.
* El proyecto hace uso de software de código abierto y gratuito, lo que reduce los costos de licencia y garantiza la disponibilidad de herramientas actualizadas sin costos adicionales. Esta elección estratégica respalda la sostenibilidad económica del proyecto a largo plazo, alineándose con la capacidad de Sanatorium Business para ofrecer soluciones innovadoras de manera eficiente.
* El diseño del sistema contempla la escalabilidad, lo que significa que puede adaptarse para manejar una mayor carga de trabajo o incorporar nuevas características según las necesidades futuras de Sanatorium Business. Esta flexibilidad asegura que la inversión actual se traduzca en una solución duradera y capaz de evolucionar junto con los requisitos cambiantes de la empresa en el entorno digital.

## 1.7.3 Factibilidad Operacional

El personal involucrado en la gestión de proyectos de Sanatorium Business cuenta con las habilidades necesarias para adaptarse al uso del nuevo sistema web. Se brindará capacitación a los usuarios para garantizar una transición fluida a la gestión de proyectos a través de la solución desarrollada.

En conclusión, de acuerdo al análisis presentado se determina que el proyecto propuesto es factible técnica, económica y operativamente.

# CAPÍTULO II

# MARCO TEÓRICO CONCEPTUAL

# 2.1 Sistema Web

## 2.1.1 Definición

La conceptualización de un sistema web abarca una aplicación o conjunto de aplicaciones informáticas que operan por medio de Internet, brindando a los usuarios la posibilidad de acceder y utilizar servicios, recursos o información mediante un navegador web. Estos sistemas están diseñados para ser fácilmente accesibles desde cualquier dispositivo con conexión a Internet y un navegador, otorgándoles una gran versatilidad y accesibilidad. Aunque no se puede atribuir una única definición de "sistema web" a un autor específico, dado que es un concepto ampliamente aceptado en el ámbito de la informática y la tecnología, su comprensión ha evolucionado con el crecimiento y desarrollo de Internet a lo largo del tiempo. En términos generales, un sistema web puede concebirse como una combinación de software y hardware que faculta a los usuarios para interactuar y llevar a cabo tareas en línea a través de una interfaz web.

En su obra "Ingeniería del Software: Un enfoque práctico," Pressman caracteriza un sistema web como "una aplicación de software alojada en un servidor web, accesible para los usuarios mediante un navegador web" (Roger S. Pressman, 2014). Por otro lado, Nielsen, reconocido experto en usabilidad web, destaca en su libro "Designing Web Usability" que "los sistemas web son aquellos que se distribuyen a través de la web y aprovechan la interacción basada en el navegador para permitir que los usuarios realicen tareas y accedan a contenidos" (Jakob Nielsen, 2000).

## 2.1.2 Ventajas y desventajas de un sistema web

| **Ventajas** | **Desventajas** |
| --- | --- |
| **Accesibilidad:** Pueden ser utilizados desde cualquier dispositivo con conexión a Internet y un navegador web. | **Dependencia de Internet:** Requieren una conexión estable para su funcionamiento. La falta de conexión puede limitar su accesibilidad. |
| **Versatilidad:** Los usuarios pueden acceder a la información y realizar tareas desde diferentes ubicaciones geográficas. | **Seguridad:** La información sensible puede estar expuesta a amenazas cibernéticas, siendo crucial implementar medidas de seguridad robustas. |
| **Actualizaciones centralizadas:** Las actualizaciones y mejoras pueden implementarse de manera centralizada en el servidor, asegurando que todos los usuarios tengan acceso a la última versión. | **Rendimiento:** La velocidad y rendimiento pueden verse afectados por la carga del servidor y la velocidad de conexión de los usuarios. |
| **Colaboración:** Facilitan la colaboración entre usuarios al permitir el acceso y la edición simultánea de documentos y datos en tiempo real. | **Costos Iniciales:** Pueden requerir inversiones significativas en infraestructura y desarrollo inicial. |
| **Escalabilidad:** Pueden escalarse fácilmente para adaptarse al crecimiento de usuarios y datos. | **Personalización Limitada:** Algunos sistemas web pueden tener limitaciones en cuanto a la personalización de la interfaz de usuario. |
| **Mantenimiento Simplificado:** El mantenimiento y la resolución de problemas pueden realizarse de forma centralizada en el servidor. | **Dependencia del Navegador:** Pueden experimentar variaciones en la experiencia del usuario según el navegador utilizado. |
| **Actualizaciones automáticas:** Las actualizaciones de software pueden implementarse de forma automática sin intervención del usuario. | **Requisitos de Hardware:** Usuarios con dispositivos más antiguos o con limitaciones de hardware pueden experimentar dificultades. |

## 2.1.3 Funcionamiento

El funcionamiento de un sistema web puede variar según su propósito y complejidad; no obstante, en términos generales, sigue un conjunto de procedimientos comunes. Inicialmente, un usuario inicia el proceso al ingresar a una dirección URL o enlace mediante un navegador web.

Posteriormente, el navegador envía una solicitud al servidor web, el cual procesa la solicitud y puede interactuar con una base de datos. Con base en esta información, el sistema web genera una respuesta que se envía al navegador del cliente. El navegador interpreta y presenta esta respuesta en la pantalla del usuario.

A partir de este punto, el usuario tiene la capacidad de interactuar con la página web y, en caso necesario, enviar datos de vuelta al servidor. El servidor procesa estos datos y genera una nueva respuesta. Este ciclo de solicitud y respuesta persiste mientras el usuario navega por el sistema web, llevando a cabo diversas acciones y accediendo a distintas secciones del sistema.

Es importante destacar que los sistemas web pueden ser estáticos o dinámicos, y la tecnología subyacente, como HTML, CSS y JavaScript, desempeña un papel esencial en su operatividad.

# 2.2 Gestión de proyectos

## 2.2.1 Definición

La gestión de proyectos es una disciplina que se encarga de planificar, organizar, dirigir y controlar los recursos y actividades de un proyecto para alcanzar sus objetivos específicos en un tiempo determinado. Un proyecto se define como un esfuerzo temporal con un inicio y un final claramente definidos, que busca crear un producto, servicio o resultado único.

La gestión de proyectos implica el uso de conocimientos, habilidades, herramientas y técnicas para llevar a cabo las actividades del proyecto de manera eficiente y efectiva. Algunos de los aspectos clave de la gestión de proyectos incluyen la definición clara de los objetivos del proyecto, la identificación de los recursos necesarios, la planificación detallada de las actividades, la asignación de responsabilidades, el seguimiento del progreso, la gestión de los riesgos y la comunicación efectiva con todos los interesados.

El objetivo principal de la gestión de proyectos es lograr que el proyecto se complete dentro de los límites de tiempo, presupuesto y alcance establecidos, al mismo tiempo que se satisfacen las expectativas del cliente y se asegura la calidad del resultado final. La gestión de proyectos se aplica en una variedad de industrias y contextos, desde la construcción y la tecnología hasta la investigación y el desarrollo.

## 2.2.2 Registro tareas en proyectos

El registro de tareas en la gestión de proyectos es una herramienta fundamental que se utiliza para documentar y hacer un seguimiento de las actividades específicas que deben llevarse a cabo durante el desarrollo de un proyecto. Este registro, también conocido como lista de tareas o lista de actividades, es esencial para mantener una visión clara y organizada de las actividades planificadas y completadas a lo largo del proyecto.

El registro de tareas suele incluir la siguiente información:

* **Descripción de la Tarea:** Una breve descripción de la actividad o tarea que debe realizarse.
* **Empleados:** El empleado que registrara y llevara a cabo la tarea.
* **Fecha de Inicio y Fecha de Finalización:** Las fechas programadas para comenzar y finalizar la tarea.
* **Estado de la Tarea:** Indica si la tarea está en pendiente, progreso, completado.
* **Observaciones o Comentarios:** Información adicional relevante para la ejecución de la tarea.

El registro de tareas facilita la planificación y el seguimiento de las actividades del proyecto, permitiendo a los miembros del equipo y a los responsables del proyecto tener una visión clara del progreso, identificar posibles problemas o retrasos, y tomar decisiones informadas sobre la gestión de recursos y plazos. Además, sirve como una referencia centralizada para todas las partes interesadas del proyecto.

# 2.3 Antecedentes tecnológicos

## 2.3.1 Asana:

* **Descripción:** Asana es una plataforma de gestión de proyectos y tareas que permite a los equipos colaborar y realizar un seguimiento del progreso de los proyectos.
* **Características Clave:** Ofrece una interfaz intuitiva, herramientas de seguimiento del tiempo, integraciones con otras aplicaciones y opciones de colaboración en tiempo real.

## 2.3.2 Jira de Atlassian:

* **Descripción:** Jira es una herramienta de gestión de proyectos y seguimiento de problemas que se utiliza comúnmente en entornos ágiles de desarrollo de software.
* **Características Clave:** Proporciona tableros personalizables, seguimiento de problemas, planificación de versiones y herramientas avanzadas para equipos ágiles.

## 2.3.3 Trello:

* **Descripción:** Trello es una herramienta de gestión de proyectos basada en tableros visuales que utiliza tarjetas y listas para organizar tareas y colaborar en proyectos.
* **Características Clave:** Ofrece una interfaz visual simple, integraciones con diversas aplicaciones y facilidad de uso para la colaboración en equipo.

## 2.3.4 ****Microsoft Project:****

* **Descripción:** Microsoft Project es una herramienta de gestión de proyectos ampliamente utilizada que proporciona planificación, seguimiento y colaboración para proyectos de diferentes tamaños.
* **Características Clave:** Ofrece capacidades avanzadas de programación, seguimiento de recursos, informes personalizables y colaboración en línea.

## 2.3.4 Redmine:

* **Descripción:** Redmine es una plataforma de gestión de proyectos de código abierto que se destaca por su flexibilidad y personalización. Permite a los equipos planificar proyectos, realizar un seguimiento del tiempo, gestionar documentos y colaborar en un entorno centralizado.
* **Características Clave:** Ofrece herramientas para la creación de diagramas de Gantt, seguimiento de problemas, integración con sistemas de control de versiones, y una amplia variedad de complementos para adaptarse a las necesidades específicas de cada proyecto.

## 2.3.5 Monday.com:

* **Descripción:** Monday.com es una herramienta visual y colaborativa de gestión de proyectos y tareas. Su enfoque se basa en la creación de tableros personalizables que permiten a los equipos organizar y seguir el progreso de las actividades.
* **Características Clave:** Proporciona una interfaz visual intuitiva, funciones de automatización, seguimiento del tiempo, integraciones con otras herramientas populares y opciones de informes personalizables.

## 2.3.6 ClickUp:

* **Descripción:** ClickUp es una plataforma de gestión de proyectos y productividad que abarca desde la planificación hasta la ejecución. Permite la creación de espacios de trabajo personalizados para adaptarse a diferentes metodologías de trabajo.
* **Características Clave:** Ofrece una amplia gama de vistas de proyecto, seguimiento de tareas, herramientas de colaboración en tiempo real, integraciones con aplicaciones externas y opciones avanzadas de personalización.

## 2.3.7 Smartsheet:

* **Descripción:** Smartsheet combina elementos de hojas de cálculo con la gestión de proyectos, proporcionando una plataforma que permite la planificación, el seguimiento y la colaboración en tiempo real.
* **Características Clave:** Incluye funciones de programación, vistas de cuadrícula y calendario, colaboración en línea, formularios personalizables y paneles de control para el análisis de datos.

# 2.4 Ingeniera de software

## 2.4.1 Metodología de desarrollo

La metodología de desarrollo de software se define como un enfoque sistemático y estructurado que abarca la planificación, diseño, implementación y mantenimiento de aplicaciones informáticas. Este proceso implica la aplicación de diversas técnicas y prácticas con el objetivo de asegurar la calidad, eficiencia y éxito en el desarrollo de proyectos de software.   
  
A continuación, se exhibe una tabla comparativa que resume las metodologías consideradas para la ejecución de este proyecto.

| **Característica** | **Scrum** | **Kanban** | **XP (Programación Extrema)** |
| --- | --- | --- | --- |
| **Filosofía** | Marco de trabajo iterativo e incremental que se enfoca en roles definidos, artefactos y eventos. | Sistema de gestión visual basado en la flexibilidad y flujo continuo. | Enfoque ágil centrado en la colaboración, retroalimentación constante y mejora continua. |
| **Roles Principales** | Scrum Master, Product Owner, Equipo de Desarrollo. | No prescribe roles específicos, pero puede incluir roles como Líder de Proyecto y miembros del equipo. | Programador, Cliente y Tester son roles fundamentales. |
| **Planificación** | Sprints planificados en intervalos fijos. Revisión y retrospectiva al final de cada sprint. | Sin sprints fijos. La planificación es continua y basada en la demanda. | Planificación frecuente, lanzamientos pequeños y constantes. |
| **Gestión de Tareas** | Uso de tableros Scrum con historias de usuario y tareas pendientes, en progreso, y completadas. | Tablero Kanban visual que muestra el flujo de trabajo con tarjetas móviles. | Empleo de tarjetas de historia, planificación de juego y seguimiento continuo. |
| **Ciclo de Vida del Proyecto** | Ciclos de desarrollo iterativos con entregas incrementales. | Flujo continuo de trabajo con énfasis en la visualización y mejora continua. | Desarrollo iterativo con pruebas frecuentes y retroalimentación del cliente. |
| **Flexibilidad** | Menos flexible en cuanto a cambios durante un sprint. Cambios deben esperar al siguiente sprint. | Altamente flexible y adaptable a cambios en tiempo real. | Flexible y adaptable a cambios rápidos, con entregas pequeñas y frecuentes. |
| **Roles de Cliente** | La participación del cliente es crucial, pero su interacción suele ser a través del Product Owner. | La participación del cliente es variable y puede ser más continua según la necesidad. | La participación del cliente es constante y directa, con pruebas y feedback frecuentes. |

## 2.4.1.1 Srum

La metodología Scrum, perteneciente al ámbito ágil, destaca por su enfoque en la colaboración, la transparencia y la adaptabilidad con el propósito de proporcionar productos o proyectos de elevada calidad. Su método se sustenta en la implementación de ciclos de trabajo breves denominados "sprints", en los cuales se generan incrementos del producto. Scrum se apoya en roles, eventos y artefactos particulares para administrar y mejorar de manera específica el proceso de desarrollo.

## 2.4.1.1.1 Ventajas y Desventajas

| **Ventajas de Scrum** | **Desventajas de Scrum** |
| --- | --- |
| - Mayor flexibilidad y capacidad de adaptación a cambios en los requisitos del proyecto. | - Dependencia significativa de la comunicación y colaboración eficientes dentro del equipo. |
| - Mejora la visibilidad y control del proceso de desarrollo mediante la transparencia en los roles, eventos y artefactos. | - Puede haber resistencia al cambio en equipos y organizaciones acostumbrados a métodos de desarrollo más tradicionales. |
| - Entregas incrementales y regulares permiten obtener retroalimentación constante del cliente, mejorando la satisfacción del cliente. | - Requiere la dedicación de roles específicos, como Scrum Master y Product Owner, lo que puede ser un desafío en entornos con limitaciones de recursos. |
| - Promueve la colaboración activa entre los miembros del equipo y la toma de decisiones conjuntas. | - No es adecuado para proyectos pequeños o simples, ya que puede resultar en un exceso de procesos y documentación. |
| - Fomenta la autoorganización y autonomía del equipo, lo que puede aumentar la motivación y la productividad. | - La definición precisa de requisitos y la planificación anticipada son esenciales, lo que puede ser un desafío en proyectos con requisitos cambiantes y poco claros. |
| - Mayor probabilidad de identificar y abordar problemas de manera temprana debido a la frecuencia de revisiones y ajustes. | - Requiere un compromiso y participación activa de todos los miembros del equipo, lo que puede ser difícil de mantener a lo largo del tiempo. |
| - Establece un enfoque iterativo e incremental que facilita la entrega de valor continuo. | - No es la mejor opción para proyectos con requisitos altamente regulados o contratos fijos. |
| - Fomenta la mejora continua a través de las retrospectivas al final de cada sprint. | - Puede haber desafíos para estimar con precisión la duración de los sprints y la cantidad de trabajo que se puede realizar en cada uno. |
| - Mayor capacidad de respuesta a los cambios en las prioridades del cliente durante el desarrollo. | - El énfasis en la autoorganización del equipo puede llevar a desafíos de gestión en entornos menos maduros o con equipos menos experimentados. |

## 2.4.1.1.2 Roles de Scrum

* El Product Owner, o responsable del producto, desempeña un papel fundamental al gestionar la lista de tareas pendientes del producto (product backlog). Su enfoque se centra en las necesidades del usuario, y su responsabilidad consiste en transmitir la perspectiva del usuario a su equipo y a otros ejecutivos involucrados. Los Product Owners eficientes aportan claridad al determinar qué elementos son prioritarios para su entrega, tomando decisiones fundamentales sobre cuándo considerar que un trabajo está listo para ser entregado, con una inclinación hacia entregas frecuentes.
* El Scrum Master asume el rol de líder en los diversos eventos de Scrum y puede ser considerado como el gerente del proyecto y facilitador de Scrum. Su responsabilidad implica fomentar las reuniones diarias de actualización y coordinar las sesiones de planificación, revisión y análisis retrospectivo del sprint.
* El equipo Scrum abarca a todos los individuos que contribuyen al sprint. Este equipo se caracteriza por la autoorganización, donde sus miembros colaboran estrechamente para alcanzar los objetivos de Scrum, especialmente enfocados en la mejora continua. La dinámica de trabajo del equipo Scrum implica una colaboración activa y una capacidad para autoorganizarse, aspectos cruciales para el éxito de la metodología Scrum.

## 2.4.1.1.3 Desarrollo Scrum

* **Planificación del sprint:** Durante esta reunión, el equipo elige un conjunto de elementos del Product Backlog para abordar en el próximo sprint.
* **Sprint:** A lo largo del sprint, el equipo se dedica a las tareas seleccionadas, manteniendo una reunión diaria para sincronizarse y abordar posibles problemas.
* **Revisión del sprint:** Al concluir el sprint, se lleva a cabo una reunión de revisión donde se presenta el trabajo completado al cliente o al Product Owner.
* **Retrospectiva del sprint:** Tras la revisión, el equipo realiza una retrospectiva para evaluar su rendimiento y explorar áreas de mejora.
* **Reinicio:** Después de la retrospectiva, el proceso se reinicia con un nuevo sprint.

## 2.4.1.1.4 Funcionalidad Scrum

Scrum se emplea para gestionar proyectos y productos, especialmente en contextos donde los requisitos son cambiantes o poco definidos. Esta metodología facilita la entrega regular de valor al cliente y posibilita una adaptación ágil a las necesidades emergentes.

## 2.4.1.2 Kanban

Kanban es un método visual que utiliza tarjetas, conocidas como "kanbans", para representar tareas o elementos de trabajo. Estas tarjetas avanzan a lo largo de un tablero o pizarra, reflejando el estado y avance de las tareas. La meta principal de Kanban es mejorar el flujo de trabajo, reducir los tiempos de espera y aumentar la eficiencia.

## 2.4.1.2.1 Características de Kanban:

Tablero Kanban: La representación visual del trabajo se realiza a través de un tablero, usualmente dividido en columnas que reflejan los diversos estados o fases del flujo de trabajo.

● Tarjetas Kanban: Cada tarea o elemento de trabajo se simboliza mediante una tarjeta Kanban, la cual contiene detalles pertinentes sobre la tarea, como su descripción, prioridad y las personas responsables.

● Límites de trabajo en proceso (WIP): En Kanban, se establecen restricciones en la cantidad de tarjetas permitidas en cada columna del tablero. Este enfoque ayuda a prevenir la saturación de trabajo y congestión en el flujo de trabajo.

● Flujo continuo: Kanban fomenta la idea de mantener un flujo de trabajo constante, donde las tarjetas avanzan de una columna a otra tan pronto como sea posible, evitando esperas innecesarias.

● Flexibilidad y adaptabilidad: Kanban destaca por su alta adaptabilidad, siendo aplicable a una variedad de procesos y proyectos, independientemente de su complejidad.

## 2.4.1.2.2 Ventajas y Desventajas de Kanban

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Visualización del Trabajo** | - Proporciona una visión clara y visual del progreso del trabajo a través del tablero Kanban. | - Puede no ser tan detallado como otros métodos, lo que podría ser insuficiente en algunos contextos. |
| **Flexibilidad** | - Es altamente adaptable y puede aplicarse a diversos tipos de proyectos y procesos. | - La flexibilidad puede llevar a falta de estructura en equipos no disciplinados o en proyectos altamente regulados. |
| **Optimización del Flujo** | - Permite optimizar el flujo de trabajo al reducir los tiempos de espera y maximizar la eficiencia. | - Si no se gestionan adecuadamente los límites de trabajo en proceso (WIP), pueden surgir problemas de congestión y cuellos de botella. |
| **Enfoque en Resultados** | - Fomenta un enfoque en la entrega continua de resultados y valor al cliente. | - Puede carecer de planificación a largo plazo, lo que podría ser problemático en proyectos que requieren una visión a más largo plazo. |
| **Adaptabilidad** | - Se adapta fácilmente a cambios en los requisitos o prioridades del proyecto. | - La falta de reglas estrictas puede llevar a una gestión menos controlada y a la pérdida de seguimiento en proyectos grandes. |

## 2.4.1.3 XP (PROGRAMACIÓN EXTREMA)

La Programación Extrema es una metodología de desarrollo de software que se

centra en la agilidad y la colaboración en equipos pequeños para entregar software

de alta calidad de manera rápida. Fue desarrollada por Kent Beck a finales de la

década de 1990 y es una de las metodologías ágiles más conocidas.

## 2.4.1.3.2 Principios y Valores

* **Retroalimentación rápida (Feedback):** Busca obtener retroalimentación lo más pronto posible durante el desarrollo para poder realizar ajustes rápidos y eficientes.
* **Cambios Incrementales:** El desarrollo se realiza en pequeños incrementos para adaptarse de manera más efectiva a los cambios en los requisitos del cliente.
* **Proceso de Pequeñas Publicaciones:** Realizar pequeñas y frecuentes publicaciones de software para mantener un flujo constante y reducir el riesgo.
* **Juego de Pruebas (Testing):** La programación extrema prioriza la escritura de pruebas automatizadas antes del desarrollo del código, garantizando que las funcionalidades se mantengan funcionales.
* **Diseño Simple:** Favorece la simplicidad en el diseño del software, eliminando complejidades innecesarias

## 2.4.1.3.3 Características

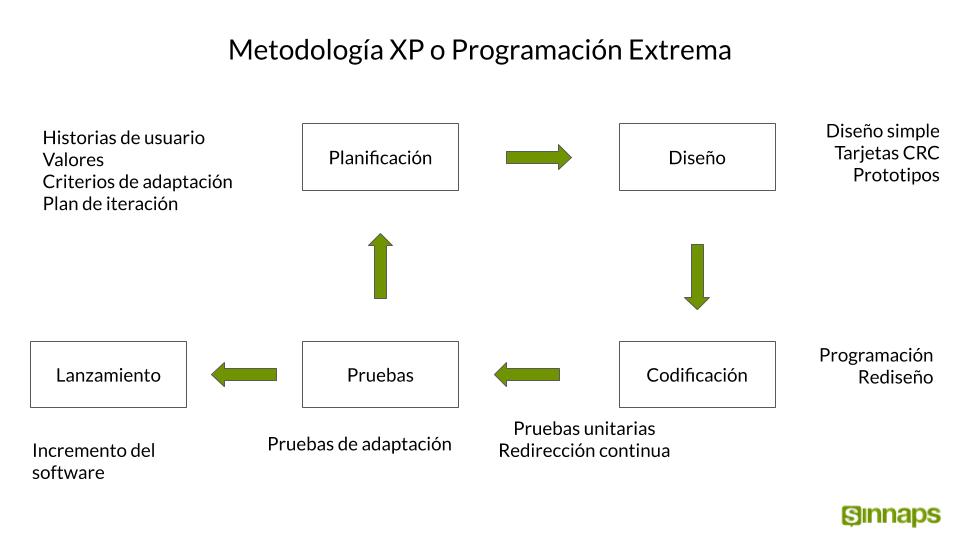
* **Retroalimentación Continua:** La XP enfatiza la importancia de obtener retroalimentación constante para permitir ajustes rápidos y mejorar continuamente.
* **Desarrollo Incremental:** Se trabaja en pequeños incrementos de funcionalidad, permitiendo adaptarse fácilmente a los cambios en los requisitos.
* **Proceso de Pequeñas Publicaciones:** Se realizan publicaciones frecuentes y pequeñas del software para mantener un flujo constante y reducir riesgos.
* **Pruebas Automatizadas:** Se prioriza la escritura de pruebas automatizadas antes de escribir el código, garantizando una mayor calidad y facilitando futuras modificaciones.
* **Diseño Simple:** Se busca mantener un diseño simple y funcional, evitando complejidades innecesarias.
* **Refactorización Continua:** La mejora constante del código a través de la refactorización ayuda a mantenerlo limpio y fácil de entender.
* **Trabajo en Parejas (Pair Programming):** Dos programadores trabajan juntos en el mismo código, fomentando la colaboración y mejorando la calidad del software.
* **Propiedad Colectiva del Código:** Todos los miembros del equipo son responsables del código y tienen el derecho de modificar cualquier parte del sistema.
* **Comunicación Eficiente:** Se prioriza la comunicación constante y efectiva entre todos los miembros del equipo, así como con los clientes.
* **Simplicidad y Claridad:** Se busca la simplicidad en el diseño y la implementación del software, favoreciendo soluciones sencillas y comprensibles.
* **Adaptabilidad:** La XP se adapta fácilmente a cambios en los requisitos del cliente y a nuevas situaciones durante el desarrollo.
* **Entregas Frecuentes:** Se busca entregar productos o funcionalidades al cliente con regularidad, obteniendo retroalimentación temprana.
* **Coraje para el Cambio:** Se valora el coraje para enfrentar cambios y desafíos, así como para admitir errores y aprender de ellos.
* **Respeto Mutuo:** La XP promueve un ambiente de respeto mutuo entre los miembros del equipo y hacia los clientes.

## 2.4.1.3.4 Ventajas y Desventajas

| **Ventajas de XP** | **Desventajas de XP** |
| --- | --- |
| **1. Adaptabilidad:** Se adapta fácilmente a cambios en los requisitos. | **1. Resistencia Cultural:** Puede enfrentar resistencia cultural al cambio en algunos equipos o empresas. |
| **2. Retroalimentación Continua:** Proporciona retroalimentación constante para mejoras continuas. | **2. Requiere Colaboración Intensa:** Requiere una colaboración intensa y constante entre los miembros del equipo. |
| **3. Desarrollo Incremental:** Permite entregas frecuentes y funcionales. | **3. Requiere Personal Competente:** Necesita un equipo competente y comprometido para funcionar de manera efectiva. |
| **4. Calidad del Código:** Enfoca la calidad del código mediante prácticas como pruebas automatizadas y refactorización. | **4. Inestabilidad Temporal:** La dinámica de equipo puede ser inestable debido a la rotación de papeles. |
| **5. Mejora Continua:** Fomenta la mejora continua del proceso y el producto. | **5. No es Universal:** No es aplicable universalmente a todos los proyectos o equipos. |
| **6. Propiedad Colectiva del Código:** Todos los miembros comparten la responsabilidad del código. | **6. Puede ser Intenso:** La intensidad del trabajo en parejas y otras prácticas puede ser agotadora para algunos. |
| **7. Enfrenta Cambios Rápidamente:** Capacidad para enfrentar y adaptarse rápidamente a cambios. | **7. Puede Generar Ansiedad:** La entrega frecuente puede generar ansiedad si no se gestiona adecuadamente. |

## 2.4.1.3.5 Ciclo de Vida

El proceso de desarrollo de software ágil conocido como Metodología de Programación Extrema (Extreme Programming o XP) sigue un ciclo de vida que destaca la colaboración estrecha entre el equipo de desarrollo y el cliente, la retroalimentación constante y la entrega regular de software funcional de alta calidad. Este ciclo de vida, que se repite en intervalos cortos, se distingue por su énfasis en la adaptabilidad y el constante proceso de mejora continua.



Metodología XP Programación Extrema (Metodología ágil) Sinnaps (May 17, 2018)

## Etapas clave del ciclo de vida de XP:

* **Exploración (Exploration):** En esta etapa, se trabaja en la identificación y definición de los requisitos del proyecto. Se enfoca en comprender las necesidades del cliente y establecer una visión clara de los objetivos.
* **Planificación de la Entrega (Planning Game):** Durante esta fase, se realiza una planificación detallada del desarrollo, donde se seleccionan las funcionalidades que se abordarán en el siguiente ciclo. El equipo y el cliente colaboran estrechamente para establecer prioridades.
* **Diseño Simplemente Suficiente (Small Releases):** Se lleva a cabo la implementación de las funcionalidades seleccionadas en la fase de planificación. La idea es generar entregas de software funcionales y de calidad en intervalos cortos y frecuentes.
* **Pruebas (Simple Design):** La etapa de pruebas implica la verificación continua de la funcionalidad del software. Las pruebas son integrales y se realizan de manera automática y manual para garantizar la calidad del código.
* **Refactorización (Refactoring):** Durante esta fase, se ajusta y mejora el código existente sin cambiar su funcionalidad. El objetivo es mantener un diseño simple y eficiente que pueda adaptarse a los cambios.
* **Comunicación Continua (Continuous Integration):** Se practica la integración continua, donde los cambios realizados por los miembros del equipo se combinan regularmente en un repositorio compartido. Esto asegura una colaboración constante y la detección temprana de posibles problemas.
* **Retroalimentación del Cliente (Customer Feedback):** Después de cada entrega, se busca la retroalimentación del cliente. Esto permite ajustar y adaptar las futuras iteraciones según las necesidades y expectativas del cliente.
* **Entrega Continua (Continuous Delivery):** La entrega continua implica la capacidad de liberar versiones del software en cualquier momento. Esto proporciona flexibilidad y facilita la adaptación a los cambios en los requisitos.

## 2.4.1.3.6 Reglas de la Metodología

* **Exploración (Exploration):** En esta etapa, se trabaja en la identificación y definición de los requisitos del proyecto. Se enfoca en comprender las necesidades del cliente y establecer una visión clara de los objetivos.
* **Planificación de la Entrega (Planning Game):** Durante esta fase, se realiza una planificación detallada del desarrollo, donde se seleccionan las funcionalidades que se abordarán en el siguiente ciclo. El equipo y el cliente colaboran estrechamente para establecer prioridades.
* **Diseño Simplemente Suficiente (Small Releases):** Se lleva a cabo la implementación de las funcionalidades seleccionadas en la fase de planificación. La idea es generar entregas de software funcionales y de calidad en intervalos cortos y frecuentes.
* **Pruebas (Simple Design):** La etapa de pruebas implica la verificación continua de la funcionalidad del software. Las pruebas son integrales y se realizan de manera automática y manual para garantizar la calidad del código.
* **Refactorización (Refactoring):** Durante esta fase, se ajusta y mejora el código existente sin cambiar su funcionalidad. El objetivo es mantener un diseño simple y eficiente que pueda adaptarse a los cambios.
* **Comunicación Continua (Continuous Integration):** Se practica la integración continua, donde los cambios realizados por los miembros del equipo se combinan regularmente en un repositorio compartido. Esto asegura una colaboración constante y la detección temprana de posibles problemas.
* **Retroalimentación del Cliente (Customer Feedback):** Después de cada entrega, se busca la retroalimentación del cliente. Esto permite ajustar y adaptar las futuras iteraciones según las necesidades y expectativas del cliente.
* **Entrega Continua (Continuous Delivery):** La entrega continua implica la capacidad de liberar versiones del software en cualquier momento. Esto proporciona flexibilidad y facilita la adaptación a los cambios en los requisitos.

# 2.4.2.1 Métodos de recolección de información

## 2.4.2.1 Encuesta

La Encuesta es una técnica de investigación que implica la obtención de datos mediante preguntas estandarizadas o cuestionarios diseñados, con el propósito de recabar información específica de un conjunto de personas, una muestra representativa de la población o individuos particulares.

El objetivo principal de una encuesta es la recopilación sistemática y estandarizada de datos para lograr una comprensión más precisa de opiniones, actitudes, creencias, comportamientos u otras variables de interés. Los datos recopilados a través de las encuestas se someten a análisis estadísticos con el fin de identificar patrones, tendencias y conclusiones significativas. Este proceso permite a los investigadores tomar decisiones fundamentadas y obtener información valiosa sobre una población o grupo específico.

## 2.4.2.2 Entrevista

Una entrevista es un proceso comunicativo en el cual un individuo, denominado entrevistador, realiza preguntas y recopila información de otra persona o de un grupo de personas, conocidos como entrevistados. Los objetivos de una entrevista pueden variar significativamente e involucrar la obtención de información, la evaluación de habilidades, la toma de decisiones, la investigación o la obtención de testimonios, entre otros.

Las entrevistas pueden llevarse a cabo de tres formas principales: estructuradas, que siguen un conjunto predefinido de preguntas; semiestructuradas, que utilizan preguntas guía pero permiten flexibilidad para explorar temas; y no estructuradas, que no tienen preguntas predefinidas, fomentando una conversación más abierta.

Los cinco pasos básicos de una entrevista son los siguientes:

* **Preparación:**
  + Definición del propósito de la entrevista.
  + Identificación de los entrevistados y planificación de los participantes.
  + Elaboración de preguntas o temas a abordar.
  + Revisión de información relevante sobre entrevistados o el tema.
  + Organización del lugar y el tiempo de la entrevista.
* **Presentación:**
  + Establecimiento de una relación cordial y de confianza con los entrevistados.
  + Explicación del propósito de la entrevista y del proceso a seguir.
  + Aseguramiento de que los entrevistados se sientan cómodos y escuchados.
* **Interacción y Preguntas:**
  + Realización de preguntas o guía efectiva de la conversación.
  + Escucha activa a los entrevistados, prestando atención a sus respuestas.
  + Exploración y profundización en temas relevantes.
  + Registro de respuestas y toma de notas si es necesario.
* **Cierre:**
  + Resumen de los puntos clave discutidos durante la entrevista.
  + Oportunidad para que los entrevistados hagan preguntas o comentarios adicionales.
  + Agradecimiento a los entrevistados por su participación y colaboración.
* **Seguimiento:**
  + Procesamiento y análisis de los datos recopilados, si es necesario.
  + Elaboración de un informe o documentación de los hallazgos de la entrevista.
  + Acciones posteriores según el propósito de la entrevista, como decisiones, recomendaciones o acciones de seguimiento.

## 2.4.2.3 Observación

La observación constituye un método de obtención de datos que implica la recopilación sistemática y objetiva de información mediante la observación directa de individuos, eventos, conductas o fenómenos. Este enfoque se lleva a cabo sin intervenir en la situación observada y puede aplicarse tanto en investigaciones cualitativas como cuantitativas.

## 2.4.2.3.1 Tipos de Observación

| **Tipo de Observación** | **Descripción** |
| --- | --- |
| Observación Participante | En este tipo de observación, el investigador es parte activa de la situación que está observando. Puede interactuar con los participantes y experimentar directamente la dinámica del contexto. |
| Observación No Participante | El observador se mantiene como un observador externo y no participa en la actividad o interacción que está siendo observada. Esto permite una perspectiva más objetiva y distante de la situación. |
| Observación Directa | Implica la observación de eventos, comportamientos o fenómenos en tiempo real, sin la intervención del observador. Los datos se recopilan tal como ocurren, ofreciendo una visión fiel de la situación. |
| Observación Indirecta | Se basa en la revisión de registros, documentos o grabaciones previas en lugar de observar directamente. Aunque menos inmediata, esta observación permite analizar eventos pasados de manera detallada. |
| Observación Encubierta | El observador no es conocido por los participantes, lo que evita que su presencia afecte el comportamiento natural. Este enfoque se utiliza cuando la observación abierta podría influir en los resultados. |
| Observación Abierta | Los participantes son conscientes de la presencia del observador y conocen el propósito de la observación. Esta transparencia puede afectar el comportamiento, pero permite una recopilación más ética de datos. |

# 2.4.2.2 Técnicas de recolección de información

## 2.4.2.2.1 Cuestionarios Cerrados

Los cuestionarios cerrados constituyen un tipo de herramienta para la recopilación de datos que incluye preguntas predefinidas con opciones de respuesta específicas. Los encuestados seleccionan sus respuestas de entre las opciones proporcionadas, simplificando así la tabulación y el análisis cuantitativo de la información. Estas respuestas generalmente adoptan la forma de categorías o escalas, lo que posibilita la medición y cuantificación eficaz de las respuestas. Este tipo de cuestionarios resulta beneficioso para la obtención de datos cuantitativos y es comúnmente empleado en investigaciones que requieren análisis estadísticos.

## 2.4.2.2.2 Cuestionarios abiertos

Los cuestionarios abiertos difieren al presentar preguntas que carecen de opciones de respuesta predeterminadas. En lugar de elegir entre alternativas predefinidas, los encuestados deben ofrecer sus propias respuestas con sus propias palabras. Esta estructura permite que los participantes expresen sus opiniones, ideas y experiencias de manera libre y detallada. Los cuestionarios abiertos son valiosos para la recopilación de datos cualitativos, y sus respuestas son comúnmente analizadas mediante técnicas como el análisis de contenido o la codificación para identificar patrones y temas emergentes.

# 2.4.3 Base de datos

## 2.4.3.1 SQL Server

SQL Server es una plataforma de gestión de bases de datos relacionales (RDBMS) que utiliza el lenguaje SQL (Structured Query Language) para la interacción con la base de datos. Ofrece a las organizaciones la capacidad de administrar grandes cantidades de datos de manera segura y escalable, permitiendo realizar consultas y análisis de datos de manera eficaz.

## 2.4.3.2 Ventajas y Desventajas

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Costo** | - Ofrece versiones gratuitas como SQL Server Express. | - Las versiones más completas pueden ser costosas. |
| **Escalabilidad** | - Escalabilidad vertical y horizontal. | - Las opciones de escalabilidad horizontal pueden requerir configuraciones más avanzadas. |
| **Seguridad** | - Funcionalidades avanzadas de seguridad y cifrado. | - Algunas características de seguridad avanzadas pueden requerir conocimientos técnicos específicos. |
| **Integración** | - Está integrado con otras herramientas de Microsoft. | - Puede haber limitaciones en la integración con tecnologías no Microsoft. |
| **Herramientas** | - Ofrece un conjunto completo de herramientas de administración. | - Puede ser percibido como más complejo para usuarios nuevos en comparación con sistemas más simples. |
| **Soporte** | - Soporte técnico sólido y documentación extensa. | - El soporte puede requerir licenciamiento adicional. |

## 2.4.3.3 MySQL

MySQL es una solución de gestión de bases de datos relacional de código abierto que emplea el lenguaje SQL (Structured Query Language) para la administración, almacenamiento y recuperación de datos. Esta plataforma es desarrollada por Oracle Corporation y presenta varias ediciones, entre las que se encuentra una versión comunitaria de código abierto denominada MySQL Community Edition.

## 2.4.3.3.1 Características:

* **Código Abierto:** MySQL se distribuye bajo una licencia de código abierto, lo que significa que su código fuente está disponible para la comunidad. Esto facilita la personalización, el aprendizaje y el desarrollo colaborativo.
* **Multiplataforma:** MySQL es compatible con varias plataformas, lo que permite su instalación en sistemas operativos como Windows, Linux, macOS y otros. Esta versatilidad facilita su integración en diferentes entornos informáticos.
* **Rendimiento Eficiente:** MySQL está diseñado para ofrecer un rendimiento eficiente, especialmente en entornos donde se manejan grandes volúmenes de datos. Sus algoritmos y técnicas de optimización contribuyen a un acceso rápido y una manipulación eficaz de la información.
* **Alta Disponibilidad:** MySQL ofrece soluciones para garantizar la alta disponibilidad de los datos, incluyendo la replicación maestro-esclavo y la configuración de clústeres. Estas características son fundamentales para entornos donde la continuidad del servicio es crítica.
* **Seguridad:** La seguridad es una preocupación primordial en la gestión de bases de datos. MySQL proporciona características como el cifrado de datos, la autenticación de usuarios y la gestión de privilegios para garantizar la integridad y confidencialidad de la información.
* **Escalabilidad:** MySQL es escalable, lo que significa que puede manejar un aumento en el tamaño de la base de datos y la carga de trabajo. Esta capacidad de escalabilidad es esencial para empresas y aplicaciones que experimentan un crecimiento constante.
* **Soporte para Transacciones:** MySQL admite transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza la integridad de los datos incluso en entornos donde múltiples operaciones deben realizarse de manera coherente.
* **Herramientas y Utilidades:** Además del sistema de gestión de base de datos, MySQL proporciona diversas herramientas y utilidades, como MySQL Workbench, que facilitan el desarrollo, administración y monitoreo de bases de datos.
* **Compatibilidad con Estándares:** MySQL sigue los estándares de SQL, lo que facilita la migración de aplicaciones entre diferentes sistemas de bases de datos relacionales y garantiza la portabilidad del código.
* **Comunidad Activa:** Dado que es un proyecto de código abierto, MySQL cuenta con una comunidad activa de desarrolladores, usuarios y expertos que contribuyen al soporte, la mejora continua y la resolución de problemas.

## 2.4.3.3.2 Funcionalidad

MySQL encuentra aplicación en una diversidad de contextos, abarcando desde sitios web y sistemas de gestión de contenido (CMS) hasta sistemas de comercio electrónico, aplicaciones empresariales, y herramientas de seguimiento y análisis de datos, entre otros. Ofrece las funcionalidades características de un sistema de gestión de bases de datos relacionales (RDBMS), incluyendo operaciones como creación, lectura, actualización y eliminación de datos (CRUD). Destaca por su idoneidad en aplicaciones que demandan una recuperación de datos eficiente.

## 2.4.3.3.3 Ventajas y Desventajas

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Costo** | - MySQL es de código abierto y gratuito. | - No cuenta con algunas funciones avanzadas que se encuentran en sistemas de gestión de bases de datos comerciales. |
| **Escalabilidad** | - Es escalable y puede manejar grandes volúmenes de datos. | - Puede tener un rendimiento inferior en comparación con algunos sistemas de gestión de bases de datos más grandes y costosos. |
| **Comunidad** | - Tiene una amplia comunidad de usuarios y desarrolladores. | - El soporte técnico puede ser limitado en comparación con sistemas comerciales que ofrecen servicios de asistencia pagados. |
| **Compatibilidad** | - Compatible con varias plataformas y sistemas operativos. | - Algunas funciones avanzadas pueden no ser tan robustas como en sistemas comerciales más caros. |
| **Facilidad de Uso** | - Es fácil de aprender y usar, especialmente para proyectos más simples. | - Puede carecer de algunas características avanzadas de administración y seguridad. |
| **Seguridad** | - Ofrece características sólidas de seguridad y administración de permisos. | - En entornos críticos de seguridad, algunos usuarios pueden optar por soluciones más especializadas. |

# 2.4.4 ARQUITECTURA

## 2.4.4.1 Arquitectura de Diseño MVC

El patrón de diseño de software conocido como Modelo-Vista-Controlador (MVC) se emplea extensamente en el desarrollo de aplicaciones, especialmente en aquellas de tipo web y de escritorio. Este patrón divide una aplicación en tres componentes fundamentales:

* **Modelo (Model):** Encargado de representar los datos y la lógica subyacente de la aplicación. Se ocupa de las operaciones relacionadas con los datos, la lógica empresarial y la interacción con la base de datos.
* **Vista (View):** Responsable de la presentación de datos y la interfaz de usuario. Su función principal es mostrar información al usuario y capturar las interacciones realizadas por este.
* **Controlador (Controller):** Actúa como intermediario entre el modelo y la vista. Controla la lógica de flujo y procesamiento, gestionando las solicitudes efectuadas por el usuario.

## 2.4.4.1.1 Características:

División de responsabilidades: MVC distingue claramente las funciones, simplificando el mantenimiento y la ampliación del software. ● Reciclaje de código: Facilita la reutilización de elementos en diversas secciones de la aplicación. ● Mejora la colaboración en equipo: La estructura clara del patrón permite una colaboración más eficaz y armoniosa entre los equipos de desarrollo.

## 2.4.4.1.2 Ventajas y Desventajas

| **Ventajas del Modelo-Vista-Controlador (MVC)** | **Desventajas del Modelo-Vista-Controlador (MVC)** |
| --- | --- |
| **1. Separación de responsabilidades:** Cada componente tiene un rol definido, facilitando la gestión del código y mejorando la estructura del software. | **1. Complejidad inicial:** La implementación puede parecer compleja, especialmente para quienes no están familiarizados con el patrón MVC. |
| **2. Reutilización de código:** Fomenta la reutilización de componentes en diferentes partes de la aplicación, ahorrando tiempo y esfuerzo. | **2. Posible sobreingeniería:** En proyectos más pequeños, la estructura puede parecer excesiva, generando más complejidad de la necesaria. |
| **3. Facilita el trabajo en equipo:** Equipos de desarrollo pueden trabajar eficientemente gracias a la clara estructura del patrón. | **3. Mayor cantidad de archivos:** La separación de componentes puede llevar a un mayor número de archivos, afectando la organización del proyecto. |
| **4. Escalabilidad del software:** Permite agregar funcionalidades o realizar modificaciones sin afectar otras partes del sistema. | **4. Mayor curva de aprendizaje:** Nuevos desarrolladores pueden enfrentar una curva de aprendizaje pronunciada para comprender las interacciones entre los componentes. |

# 2.4.4.2 Arquitectura de Cliente – Servidor

Es un patrón de diseño para sistemas informáticos que implica la distribución de tareas y funciones entre dos tipos de entidades: el cliente y el servidor.

* Cliente: Este componente es responsable de solicitar servicios o recursos al servidor. Puede tomar la forma de aplicaciones de software o dispositivos, como navegadores web, aplicaciones móviles, computadoras personales, entre otros.
* Servidor: Este elemento del sistema se encarga de suministrar los servicios o recursos solicitados por los clientes. Los servidores suelen ser sistemas más robustos y especializados en tareas particulares, como servidores web, servidores de base de datos, servidores de correo electrónico, entre otros.

## 2.4.4.2.1 Funcionalidad

Los clientes envían solicitudes al servidor para acceder a servicios, datos o

recursos.

* Los servidores procesan las solicitudes de los clientes y les proporcionan los

servicios o datos solicitados.

* La comunicación entre clientes y servidores se realiza a través de protocolos

de red, como HTTP, FTP, SMTP, etc.

# 2.4.4.2.2 Ventajas y Desventajas

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Escalabilidad** | - Puede escalar agregando más clientes o servidores. | - La escalabilidad puede ser un desafío en sistemas grandes. |
| **Centralización de Recursos** | - Mayor control sobre datos y lógica en el servidor. | - Puede haber cuellos de botella si el servidor está sobrecargado. |
| **Seguridad** | - Implementación más sencilla de medidas de seguridad. | - Vulnerabilidad a ataques centralizados. |
| **Mantenimiento Simplificado** | - Actualizaciones y cambios se implementan en el servidor. | - Puede haber interrupciones durante actualizaciones. |
| **Rendimiento** | - Buen rendimiento para procesamiento intensivo en el servidor. | - Latencia en la comunicación cliente-servidor. |
| **Dependencia de la Red** | - Conectividad constante entre cliente y servidor. | - Problemas en la experiencia del usuario con conexiones débiles. |
| **Carga del Servidor** | - Control centralizado, pero puede haber sobrecarga. | - Costos asociados con la infraestructura robusta. |
| **Complejidad en el Desarrollo** | - Ofrece control sobre la lógica central. | - Desarrollo y mantenimiento pueden ser complejos. |

# 2.4.5 HERRAMIENTAS DE DESARROLLO

## 2.4.5.1 Django

Django es un marco de desarrollo web de alto nivel y de código abierto, escrito en Python. Fue creado con el objetivo de facilitar el desarrollo rápido y limpio de aplicaciones web, siguiendo el principio de "baterías incluidas", lo que significa que incluye muchas características y herramientas útiles fuera de la caja. Fue desarrollado por la World Wide Web (WWW) Company de Lawrence, Kansas, y su nombre hace referencia al guitarrista de jazz gitano Django Reinhardt.

Django es utilizado ampliamente en la industria y es apreciado por su enfoque pragmático y su énfasis en la simplicidad y la legibilidad del código. Es una opción popular para el desarrollo de aplicaciones web complejas y robustas.

## 2.4.5.1.1 Características Django

* **Diseño MVT (Modelo-Vista-Template):** Django sigue un patrón de diseño similar al Modelo-Vista-Controlador (MVC), pero en lugar de utilizar Controlador, utiliza Template. Esto ayuda a organizar el código de manera clara y separar la lógica de presentación de la lógica de negocio.
* **ORM (Mapeo Objeto-Relacional):** Proporciona un ORM integrado que permite interactuar con la base de datos utilizando objetos Python en lugar de consultas SQL directas. Esto facilita la manipulación de datos y reduce la cantidad de código necesario para realizar operaciones de base de datos.
* **Administrador de Django:** Ofrece un panel de administración automático y fácil de usar para gestionar los modelos de base de datos. Esto facilita la administración y el mantenimiento de la aplicación sin tener que escribir código adicional.
* **Sistema de Plantillas:** Django cuenta con un sistema de plantillas que permite definir cómo se presenta la información en las páginas web. Las plantillas son fáciles de leer y escribir, y ofrecen una separación clara entre la lógica de presentación y la lógica de la aplicación.
* **Manejo de Formularios:** Proporciona un conjunto robusto de herramientas para la creación y validación de formularios. Facilita la captura de datos del usuario y su procesamiento.
* **Enrutamiento de URL:** Django utiliza un enfoque basado en configuración para el enrutamiento de URL, lo que facilita la definición de patrones de URL y su correspondencia con vistas específicas.
* **Seguridad Integrada:** Incluye medidas de seguridad integradas para proteger contra vulnerabilidades comunes, como ataques de inyección SQL, CSRF (Cross-Site Request Forgery) y XSS (Cross-Site Scripting).
* **Middleware:** Django utiliza un sistema de middleware que permite procesar las solicitudes y respuestas antes de llegar a las vistas. Esto ofrece flexibilidad en la manipulación del flujo de solicitud y respuesta.
* **Autenticación y Autorización:** Proporciona sistemas integrados para gestionar la autenticación de usuarios y la autorización de acceso a diferentes partes de la aplicación.
* **Internacionalización y Localización:** Facilita la internacionalización de aplicaciones, permitiendo la adaptación a diferentes idiomas y regiones.
* **Soporte para Despliegue Rápido:** Django incluye herramientas para facilitar el despliegue rápido de aplicaciones, y es compatible con una variedad de servidores web y bases de datos.

## 2.4.5.1.2 Ventajas y Desventajas de Django

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Diseño MVT** | - Organiza el código de manera clara. | - Curva de aprendizaje para aquellos nuevos en el diseño MVT. |
| **ORM (Mapeo Objeto-Relacional)** | - Facilita la interacción con la base de datos. | - Puede haber un ligero rendimiento adicional en comparación con SQL puro. |
| **Administrador de Django** | - Proporciona una interfaz de administración automática. | - No siempre es personalizable para casos muy específicos. |
| **Sistema de Plantillas** | - Sistema de plantillas robusto y fácil de usar. | - Algunos desarrolladores pueden preferir otros motores de plantillas. |
| **Manejo de Formularios** | - Herramientas integradas para crear y procesar formularios. | - Personalización avanzada de formularios puede requerir más trabajo. |
| **Enrutamiento de URL** | - Sistema claro y expresivo para definir patrones de URL. | - Puede volverse complejo en aplicaciones grandes. |
| **Seguridad Integrada** | - Incorpora medidas de seguridad contra vulnerabilidades. | - Requiere una comprensión sólida de las mejores prácticas de seguridad. |
| **Middleware** | - Permite procesar solicitudes y respuestas. | - Concepto avanzado que puede ser confuso para principiantes. |
| **Comunidad y Documentación** | - Amplia comunidad y abundante documentación. | - Actualizaciones frecuentes pueden requerir mantenimiento. |
| **Soporte para Despliegue Rápido** | - Facilita el despliegue de aplicaciones web. | - Configuración inicial puede ser más compleja en algunos entornos. |

## 2.4.5.2 Laravel

Laravel es un marco de desarrollo web de código abierto y basado en PHP que sigue el patrón de diseño Modelo-Vista-Controlador (MVC). Fue creado por Taylor Otwell y lanzado por primera vez en 2011. Laravel se ha convertido en uno de los marcos de PHP más populares debido a su elegancia, facilidad de uso y su rica funcionalidad "fuera de la caja".

## 2.4.5.2.1 Ventajas y Desventajas de Laravel:

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Sintaxis Elegante** | - Sintaxis clara y expresiva, facilita la legibilidad. | - Puede haber una curva de aprendizaje para nuevos usuarios. |
| **Eloquent ORM** | - Simplifica las operaciones de base de datos con objetos. | - En comparación con SQL puro, puede haber una pequeña sobrecarga. |
| **Blade Templating Engine** | - Sistema de plantillas robusto y fácil de usar. | - Limitado en comparación con algunos otros motores de plantillas. |
| **Artisan Console** | - Herramienta de línea de comandos para tareas repetitivas. | - Puede ser abrumador para principiantes debido a la cantidad de comandos. |
| **Migraciones y Semillas** | - Facilita la gestión de esquemas de bases de datos. | - Para proyectos pequeños, estas características pueden parecer excesivas. |
| **Routing claro y expresivo** | - Define rutas de manera clara y expresiva. | - En proyectos grandes, la organización puede volverse compleja. |
| **Middleware** | - Permite la ejecución de código antes o después de una solicitud. | - Puede ser un concepto avanzado para principiantes. |
| **Ecosistema Laravel** | - Amplio ecosistema de paquetes y extensiones. | - Puede haber casos de incompatibilidad entre versiones. |
| **Sistema de Autenticación y Autorización** | - Funciones integradas para gestionar la autenticación y la autorización. | - Personalización avanzada puede requerir más trabajo. |
| **Soporte para TDD** | - Facilita el desarrollo guiado por pruebas con PHPUnit. | - Configuración inicial de pruebas puede requerir tiempo adicional. |
| **Soporte para Despliegue Rápido** | - Facilita el despliegue rápido de aplicaciones web. | - Configuración inicial puede ser más compleja en algunos entornos. |

## 2.4.5.3 CodeIgniter

CodeIgniter es un marco de desarrollo web de código abierto para PHP que sigue el patrón de diseño Modelo-Vista-Controlador (MVC). Fue creado por EllisLab y lanzado inicialmente en 2006. CodeIgniter es conocido por su simplicidad, ligereza y facilidad de uso, lo que lo hace especialmente atractivo para desarrolladores que buscan un marco que sea rápido de aprender y fácil de implementar.

## 2.4.5.3.1 Ventajas y Desventajas de CodeIgniter

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Simplicidad y Ligereza** | - Fácil de aprender y rápido de implementar. | - Limitado en funcionalidades en comparación con algunos marcos más grandes. |
| **No Requiere Configuración** | - Instalación y configuración sencillas. | - Puede carecer de algunas características avanzadas. |
| **Buena Documentación** | - Documentación clara y completa. | - Comunidad más pequeña en comparación con algunos otros marcos. |
| **MVC Ligero** | - Implementación más liviana del patrón MVC. | - No es tan estructurado como algunos marcos más grandes. |
| **Librerías y Helpers** | - Incluye librerías y helpers para tareas comunes. | - Algunas librerías pueden ser menos extensas que las de otros marcos. |
| **Rendimiento** | - Rápido rendimiento debido a su naturaleza ligera. | - Puede no ser tan escalable como algunos marcos más grandes. |
| **Soporte para Bases de Datos** | - Capa de abstracción para interactuar con bases de datos. | - Algunas características avanzadas de ORM pueden faltar. |
| **Personalización** | - Nivel de personalización para adaptarse a necesidades específicas. | - La flexibilidad puede llevar a la falta de estructura en proyectos grandes. |
| **Comunidad Activa** | - Comunidad activa que ofrece soporte y recursos. | - Menos recursos y paquetes disponibles en comparación con marcos más grandes. |
| **Seguridad Integrada** | - Características integradas para proteger contra ataques comunes. | - Algunas funciones de seguridad avanzadas pueden ser limitadas. |

## 2.4.5.4 Bootstrap

Bootstrap es un framework de diseño web desarrollado por Twitter y ahora

mantenido por la comunidad de código abierto. Se basa en HTML, CSS y JavaScript,

y proporciona una colección de componentes y estilos predefinidos que ayudan a

los desarrolladores a crear interfaces de usuario de alta calidad de manera

eficiente.

## 2.4.5.4.1 Funcionalidad

* **Sistema de Rejilla (Grid System):** Bootstrap utiliza un sistema de rejilla flexible basado en columnas que facilita la creación de diseños responsivos. Permite organizar el contenido en filas y columnas, adaptándose de manera automática a diferentes tamaños de pantalla.
* **Componentes UI Reutilizables:** Bootstrap proporciona una amplia variedad de componentes de interfaz de usuario (UI) predefinidos, como botones, formularios, tarjetas, barras de navegación, alertas, entre otros. Estos componentes están diseñados para ser fáciles de usar y personalizables.
* **Tipografía y Estilos:** Incluye estilos de texto predefinidos, clases de alineación y tipografía que facilitan la creación de una apariencia visualmente agradable y consistente en todo el sitio.
* **JavaScript Incorporado:** Bootstrap incluye plugins y utilidades JavaScript que mejoran la funcionalidad de ciertos elementos, como los modales, las pestañas, los carruseles y las alertas. Estos elementos pueden implementarse fácilmente sin la necesidad de escribir mucho código JavaScript personalizado.
* **Diseño Responsivo:** Bootstrap está diseñado desde el principio para ser totalmente responsivo, lo que significa que las aplicaciones y sitios web construidos con Bootstrap se ajustan automáticamente a diferentes tamaños de pantalla, desde dispositivos móviles hasta pantallas de escritorio.
* **Compatibilidad entre Navegadores:** Bootstrap se esfuerza por garantizar la compatibilidad entre diferentes navegadores, reduciendo la necesidad de realizar ajustes específicos para que un sitio funcione correctamente en distintos entornos de navegación.
* **Facilidad de Personalización:** Aunque proporciona estilos y componentes predefinidos, Bootstrap es altamente personalizable. Los desarrolladores pueden anular estilos predeterminados o utilizar las herramientas de personalización para adaptar el aspecto y la sensación del sitio según sus necesidades.
* **Documentación Completa:** Bootstrap cuenta con una documentación extensa y clara que proporciona instrucciones detalladas y ejemplos prácticos para el uso de sus componentes y funcionalidades.

## 2.4.5.4.2 Ventajas de Bootstrap

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Rápido Desarrollo** | - Facilita el desarrollo rápido de interfaces de usuario. | - Puede resultar en sitios web con un aspecto genérico si no se personaliza. |
| **Diseño Responsivo** | - Proporciona un sistema de rejilla para diseños responsivos. | - Algunos proyectos pueden tener un aspecto similar a otros sitios de Bootstrap. |
| **Amplia Variedad de Componentes** | - Ofrece una amplia gama de componentes y estilos predefinidos. | - Incluir el conjunto completo de Bootstrap puede resultar en código innecesario. |
| **Personalización Fácil** | - Altamente personalizable mediante anulación de estilos. | - Personalizar completamente el aspecto puede requerir más trabajo. |
| **Documentación Exhaustiva** | - Documentación clara y extensa con ejemplos prácticos. | - La curva de aprendizaje puede ser empinada para principiantes. |
| **Compatibilidad entre Navegadores** | - Funciona bien en una variedad de navegadores. | - El exceso de personalización puede conducir a problemas de compatibilidad. |
| **Actualizaciones y Soporte Activo** | - Activa comunidad de desarrollo y actualizaciones regulares. | - Actualizar a nuevas versiones puede requerir ajustes en el código existente. |
| **Diseño Consistente** | - Ayuda a mantener una apariencia consistente en todo el sitio. | - La excesiva dependencia de Bootstrap puede limitar la creatividad del diseño. |

# 2.4.5.5 Notación

## 2.4.5.5.1 UML

UML, que significa "Unified Modeling Language" en inglés, es un lenguaje de

modelado estandarizado utilizado en la ingeniería de software para visualizar,

diseñar y documentar sistemas y procesos. Fue desarrollado originalmente por

Grady Booch, James Rumbaugh e Ivar Jacobson, y es ampliamente utilizado en la

industria del software para comunicar ideas y conceptos entre los miembros del

equipo de desarrollo, así como para comprender y representar la arquitectura y el

diseño de sistemas complejos.

## 2.4.5.5.1.1 Características de UML

* **Notación Gráfica Estandarizada:** UML proporciona una notación gráfica estandarizada que permite a los desarrolladores y equipos de desarrollo expresar visualmente conceptos y elementos de diseño de software.
* **Modelado de Sistemas Complejos:** UML es capaz de modelar sistemas de software de diferentes tamaños y niveles de complejidad, desde aplicaciones pequeñas hasta sistemas empresariales grandes.
* **Varios Tipos de Diagramas:** UML incluye una variedad de diagramas que se utilizan para representar diferentes aspectos de un sistema. Algunos de los diagramas más comunes incluyen diagramas de clases, diagramas de casos de uso, diagramas de secuencia, diagramas de actividad, entre otros.
* **Enfoque Orientado a Objetos:** UML está fuertemente basado en el paradigma de programación orientada a objetos. Los diagramas de clases, por ejemplo, representan clases y sus relaciones, mientras que otros diagramas, como los de secuencia y colaboración, muestran interacciones entre objetos.
* **Versatilidad:** Aunque inicialmente se diseñó para modelar sistemas orientados a objetos, UML se ha adaptado y utilizado para modelar sistemas de diferentes paradigmas, incluidos sistemas estructurados y basados en eventos.
* **Comunicación Efectiva:** UML se utiliza como un lenguaje común entre diferentes partes interesadas en el desarrollo de software, como analistas, diseñadores, desarrolladores y gerentes. Facilita la comunicación y comprensión mutua.
* **Estandarización Internacional:** UML es un estándar internacional gestionado por el Object Management Group (OMG). La estandarización facilita la consistencia y la interoperabilidad entre las herramientas y los equipos de desarrollo.
* **Modelado de Procesos de Desarrollo:** UML no solo se utiliza para modelar la estructura de un sistema, sino que también puede representar aspectos de procesos de desarrollo, como los flujos de trabajo y las fases del ciclo de vida del desarrollo de software.
* **Adaptabilidad y Evolución:** A lo largo del tiempo, UML ha evolucionado para adaptarse a las cambiantes prácticas y tecnologías de desarrollo de software. Se han introducido nuevas versiones y extensiones para abordar diversas necesidades.
* **Herramientas de Modelado:** Existen numerosas herramientas de modelado UML que permiten a los desarrolladores crear, editar y visualizar modelos UML. Estas herramientas pueden generar código, documentación y admitir la colaboración entre equipos.

# 2.4.6 TECNOLOGÍA DE DESARROLLO DE COMPONENTES

## 2.4.6.1 Lenguajes de Programación

Los lenguajes de programación son sistemas formales diseñados para expresar

algoritmos de manera que una computadora pueda comprender y ejecutar. Estos

lenguajes permiten a los programadores comunicarse con las computadoras y

especificar las tareas que deben realizarse. Cada lenguaje de programación tiene su

propia sintaxis y semántica, lo que significa que tiene reglas y estructuras

específicas que deben seguirse para escribir código válido.

# 2.4.6.1.1 Python

Python es un lenguaje de programación de alto nivel, interpretado, de propósito general y de código abierto. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python se ha vuelto extremadamente popular en la comunidad de desarrollo debido a su sintaxis clara y legible, su enfoque en la productividad y su amplia variedad de aplicaciones.

## 2.4.6.1.1.1 Características

* Sintaxis Clara y Legible: La sintaxis de Python está diseñada para ser clara y fácil de leer, lo que facilita tanto la escritura como la comprensión del código.
* Interpretado e Interactivo: Python es un lenguaje interpretado, lo que significa que no se compila antes de ejecutarse. Esto permite una rápida iteración y desarrollo interactivo.
* Multiparadigma: Python admite múltiples paradigmas de programación, incluyendo programación orientada a objetos, programación imperativa y programación funcional.
* Tipado Dinámico: Python es de tipado dinámico, lo que significa que el tipo de una variable se determina en tiempo de ejecución. Esto proporciona flexibilidad pero también requiere una gestión cuidadosa de los tipos.
* Gestión Automática de Memoria: Python utiliza un recolector de basura para gestionar automáticamente la memoria, simplificando la tarea del programador en comparación con lenguajes que requieren gestión manual de la memoria.
* Amplia Biblioteca Estándar: Python incluye una amplia variedad de bibliotecas y módulos en su biblioteca estándar, que abarcan desde operaciones de entrada/salida hasta protocolos de red y manipulación de datos.
* Comunidad Activa: Python tiene una comunidad de desarrolladores muy activa y una gran cantidad de recursos en línea, incluyendo documentación, tutoriales y bibliotecas de terceros.
* Versatilidad y Aplicaciones Diversas: Python se utiliza en una amplia gama de aplicaciones, como desarrollo web, análisis de datos, inteligencia artificial, automatización de tareas, desarrollo de juegos, scripting y más.
* Portabilidad: Python es compatible con múltiples plataformas, lo que significa que el código escrito en Python se puede ejecutar en varios sistemas operativos sin modificaciones significativas.
* Frameworks Populares: Hay varios frameworks y bibliotecas populares en Python, como Django para desarrollo web, TensorFlow para aprendizaje automático, Flask para aplicaciones web ligeras, entre otros.

## 2.4.6.1.1.2 Ventajas y Desventajas de Python

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Sintaxis Clara y Legible** | - Facilita la escritura y lectura del código. | - Puede ser más lento en términos de rendimiento. |
| **Versatilidad y Multiparadigma** | - Soporta varios paradigmas de programación. | - Menos adecuado para aplicaciones móviles intensivas. |
| **Gran Biblioteca Estándar** | - Ofrece una amplia variedad de módulos y paquetes. | - Requiere más líneas de código en comparación con algunos lenguajes. |
| **Desarrollo Rápido** | - Facilita el desarrollo rápido de prototipos y proyectos. | - El tamaño del código puede ser mayor en algunos casos. |
| **Comunidad Activa y Recursos Abundantes** | - Comunidad activa con abundantes recursos en línea. | - Menos adecuado para desarrollo de juegos intensivos en gráficos. |
| **Interpretado e Interactivo** | - Facilita la depuración y exploración interactiva. | - Puede haber dependencia del intérprete de Python. |
| **Portabilidad** | - Altamente portátil entre diferentes plataformas. | - El Global Interpreter Lock (GIL) puede afectar el rendimiento en entornos multi-hilo. |
| **Manejo de Excepciones** | - Ofrece un sistema robusto de manejo de excepciones. |  |

# 2.4.6.1.2 Java

Java es un lenguaje de programación de alto nivel, orientado a objetos y multiplataforma. Fue creado por Sun Microsystems y lanzado inicialmente en 1995. Desde entonces, Java ha evolucionado y se ha convertido en uno de los lenguajes de programación más populares y ampliamente utilizados en la industria del desarrollo de software.

## 2.4.6.1.2.1 Ventajas y desventajas de Java

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Orientado a Objetos** | - Favorece la modularidad y reutilización del código. | - Puede haber una curva de aprendizaje para programadores nuevos en OOP. |
| **Multiplataforma (Write Once, Run Anywhere)** | - Ejecución en diferentes plataformas sin cambios en el código fuente. | - Algunas características específicas del sistema operativo pueden no estar disponibles. |
| **Seguro y Robusto** | - Sistema de administración de memoria y verificación de tipos para prevenir errores. | - Mayor consumo de recursos en comparación con lenguajes más ligeros. |
| **Portabilidad** | - Facilita el desarrollo de aplicaciones que pueden ejecutarse en diferentes sistemas. | - Algunas optimizaciones específicas del sistema operativo pueden no ser posibles. |
| **Recopilación de Basura (Garbage Collection)** | - Simplifica la gestión de la memoria al liberar automáticamente objetos no utilizados. | - Puede causar breves pausas en la ejecución del programa durante la recolección de basura. |
| **Amplia Biblioteca Estándar (API de Java)** | - Ofrece una gran cantidad de clases y métodos para realizar diversas tareas. | - La amplia biblioteca puede resultar abrumadora para principiantes. |
| **Desarrollo Empresarial** | - Ampliamente utilizado en el desarrollo de aplicaciones empresariales. | - Puede ser percibido como menos moderno en comparación con algunos lenguajes más nuevos. |
| **Community-Driven** | - Comunidad activa que proporciona soporte y recursos. | - La administración de Oracle puede generar inquietudes sobre el desarrollo futuro. |
| **Gran Ecosistema** | - Extenso conjunto de herramientas y frameworks disponibles. | - La necesidad de administrar explícitamente la memoria puede ser percibida como una desventaja. |
| **Plataforma Android** | - Java es el lenguaje principal para el desarrollo de aplicaciones Android. | - El entorno de ejecución Dalvik en Android tiene sus propias características y desafíos. |

# 2.4.6.1.3 Javascript

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos que se utiliza principalmente para el desarrollo web. Fue creado originalmente por Netscape y ha evolucionado con el tiempo, convirtiéndose en uno de los lenguajes más importantes para el desarrollo del lado del cliente en aplicaciones web modernas.

## 2.4.6.1.3.1 Ventajas y desventajas de Javascript

| **Aspecto** | **Ventajas** | **Desventajas** |
| --- | --- | --- |
| **Interactividad del Lado del Cliente** | - Permite la creación de páginas web interactivas y dinámicas. | - La ejecución en el lado del cliente puede ser limitada por el rendimiento del navegador y la capacidad del dispositivo. |
| **Amplia Adopción y Comunidad Activa** | - Uno de los lenguajes de programación más utilizados. | - La diversidad de frameworks y bibliotecas puede ser abrumadora para los principiantes. |
| **Fácil Integración con HTML y CSS** | - Se integra fácilmente con HTML y CSS para manipular la estructura y el estilo de las páginas web. | - Puede haber inconsistencias en la implementación entre diferentes navegadores. |
| **Programación Asíncrona** | - Soporta operaciones asíncronas, lo que mejora la eficiencia en la manipulación de eventos y operaciones de entrada/salida. | - La programación asíncrona puede resultar en un código complejo y difícil de entender para algunos desarrolladores. |
| **Facilita el Desarrollo Rápido** | - Permite el desarrollo rápido de prototipos y la iteración ágil. | - La flexibilidad de tipado puede llevar a errores difíciles de depurar. |
| **Versatilidad en Plataformas** | - Compatible con varios navegadores y sistemas operativos. | - La ejecución del lado del cliente puede ser aprovechada para ataques de seguridad si no se toman las medidas adecuadas. |
| **Node.js para Desarrollo del Lado del Servidor** | - Permite el desarrollo completo de aplicaciones con JavaScript en el lado del servidor. | - La ejecución del lado del servidor puede ser menos eficiente para ciertas tareas en comparación con otros lenguajes del lado del servidor. |
| **Ecosistema de Bibliotecas y Frameworks** | - Abundancia de frameworks y bibliotecas (React, Angular, Vue.js, etc.) que facilitan el desarrollo. | - La rápida evolución del ecosistema puede llevar a cambios frecuentes y a la obsolescencia de tecnologías. |
| **Sintaxis Amigable** | - Sintaxis similar a otros lenguajes, lo que facilita su aprendizaje. | - La debilidad tipográfica puede llevar a errores difíciles de detectar durante el desarrollo. |
| **APIs Modernas del Navegador** | - Acceso a APIs modernas del navegador para realizar operaciones avanzadas y optimizar la experiencia del usuario. | - Dependencia del rendimiento y la compatibilidad del navegador. |

# 2.4.7 PRUEBAS

## 2.4.7.1 Pruebas Unitarias

Las pruebas unitarias son una categoría de evaluación de software que se centra en analizar minuciosamente y de manera aislada las partes más pequeñas del código fuente de una aplicación, comúnmente las funciones, métodos o clases individuales. Su propósito principal es asegurar el correcto funcionamiento de cada unidad de código y la producción de resultados esperados. Estas pruebas se llevan a cabo de forma automatizada y están diseñadas para detectar errores en el nivel más básico del software, lo que simplifica la identificación y corrección temprana de problemas.

## 2.4.7.1.1 Herramientas para Pruebas Unitarias

Existen numerosas herramientas de pruebas unitarias disponibles para una variedad de lenguajes de programación. Algunas de las herramientas populares incluyen:

* **JUnit:** Ampliamente utilizada para pruebas unitarias en Java.
* **PyTest:** Una biblioteca popular para pruebas unitarias en Python.
* **NUnit:** Diseñada para pruebas unitarias en el entorno .NET.
* **Mocha y Jasmine:** Herramientas destacadas para pruebas unitarias e integración en JavaScript.
* **PHPUnit:** Utilizada para pruebas unitarias en PHP.
* **xUnit:** Una familia de bibliotecas de pruebas unitarias, que incluye xUnit.NET, xUnit.js, entre otras.

## 2.4.7.1.2 Métodos para Pruebas Unitarias:

En las pruebas unitarias, se aplican varios métodos y prácticas comunes, entre los cuales se incluyen:

* **Afirmar (Arrange-Act-Assert o AAA):** Este enfoque estructura las pruebas unitarias en tres partes:
  + Arrange: Configurar el entorno de prueba.
  + Act: Realizar la operación o función que se está probando.
  + Assert: Verificar que el resultado es el esperado.
* **Cobertura de código:** Evaluar la cobertura del código para asegurarse de que se han probado todas las rutas de ejecución del código. Las herramientas de análisis de cobertura son útiles para este propósito.
* **Pruebas parametrizadas:** Ejecutar la misma prueba con diferentes conjuntos de datos para garantizar que el código funcione correctamente con diversas entradas.
* **Falsificación (Mocking) y Simulación:** Utilizar técnicas de falsificación y simulación para aislar la unidad de código que se está probando y evitar que interactúe con otros componentes del sistema.
* **Refactorización constante:** Realizar pruebas unitarias continuas a medida que se desarrolla el código para asegurar que las modificaciones no introduzcan errores.
* **Integración continua:** Integrar pruebas unitarias en un proceso de integración continua para que se ejecuten automáticamente cada vez que se realiza una confirmación de código.

## 2.4.7.2 Pruebas de Sistema

Las pruebas de sistema son un tipo de prueba de software que se enfoca en verificar el funcionamiento completo de una aplicación o sistema en su conjunto, en lugar de evaluar unidades individuales de código. El objetivo principal de las pruebas de sistema es asegurarse de que el software funcione de acuerdo con los requisitos del sistema y que todas las partes del sistema estén integradas de manera adecuada. Estas pruebas se realizan después de las pruebas de unidad y las pruebas de integración y se centran en la funcionalidad y el rendimiento del sistema en condiciones que se asemejen a situaciones de uso real.

## 2.4.7.2.1 Herramientas para Pruebas de Sistema:

Existen varias herramientas de prueba de sistema que simplifican la planificación, ejecución y gestión de estas pruebas. Algunas de las herramientas populares incluyen:

* **Selenium:** Herramienta de código abierto ampliamente utilizada para la automatización de pruebas en aplicaciones web.
* **Appium:** Herramienta de código abierto para pruebas de aplicaciones móviles en dispositivos Android e iOS.
* **JUnit:** Aunque es conocido principalmente por pruebas unitarias, JUnit también se puede utilizar para pruebas de sistema en aplicaciones Java.
* **TestComplete:** Herramienta comercial que admite pruebas de sistema y pruebas de GUI en aplicaciones web y de escritorio.
* **Cypress:** Herramienta de código abierto diseñada específicamente para pruebas de aplicaciones web modernas.

## 2.4.7.2.2 Métodos para Pruebas de Sistema

Al realizar pruebas de sistema, se pueden aplicar varios métodos y enfoques, entre los cuales se incluyen:

* **Pruebas de Funcionalidad:** Verificar que todas las funciones y características del software operen de acuerdo con los requisitos del sistema establecidos.
* **Pruebas de Integración:** Asegurarse de que los diversos componentes del sistema funcionen correctamente en conjunto y se comuniquen de manera adecuada.
* **Pruebas de Rendimiento:** Evaluar el rendimiento del sistema bajo cargas de trabajo típicas o máximas para garantizar que responda eficientemente.
* **Pruebas de Seguridad:** Identificar vulnerabilidades y evaluar la seguridad del sistema, incluyendo pruebas de penetración y auditorías de seguridad.
* **Pruebas de Aceptación del Usuario (UAT):** Permitir que los usuarios finales evalúen el sistema para confirmar que satisface sus necesidades y expectativas.
* **Pruebas de Recuperación de Desastres:** Verificar la capacidad del sistema para recuperarse de fallas y situaciones de desastre, como la pérdida de datos.
* **Pruebas de Compatibilidad:** Asegurarse de que el software funcione en diferentes plataformas, navegadores y dispositivos.
* **Pruebas de Estrés:** Evaluar cómo se comporta el sistema bajo condiciones extremas o durante picos de carga.

## 2.4.7.3 Pruebas de Aceptación

Las pruebas de aceptación, también conocidas como pruebas de aceptación del

usuario (UAT por sus siglas en inglés, User Acceptance Testing), son un tipo de

prueba que se centra en evaluar si el software cumple con los requisitos y

expectativas del usuario final o del cliente. Estas pruebas son realizadas por los

usuarios finales o por personas designadas por el cliente y se llevan a cabo antes de

la implementación del software en un entorno de producción. El objetivo principal de

las pruebas de aceptación es confirmar que el software es adecuado para su uso y

que satisface los criterios de aceptación establecidos.

## 2.4.7.3.1 Herramientas para Pruebas de Aceptación

Las herramientas empleadas en las pruebas de aceptación pueden variar según la naturaleza del software y las preferencias del equipo. Algunas herramientas comunes incluyen:

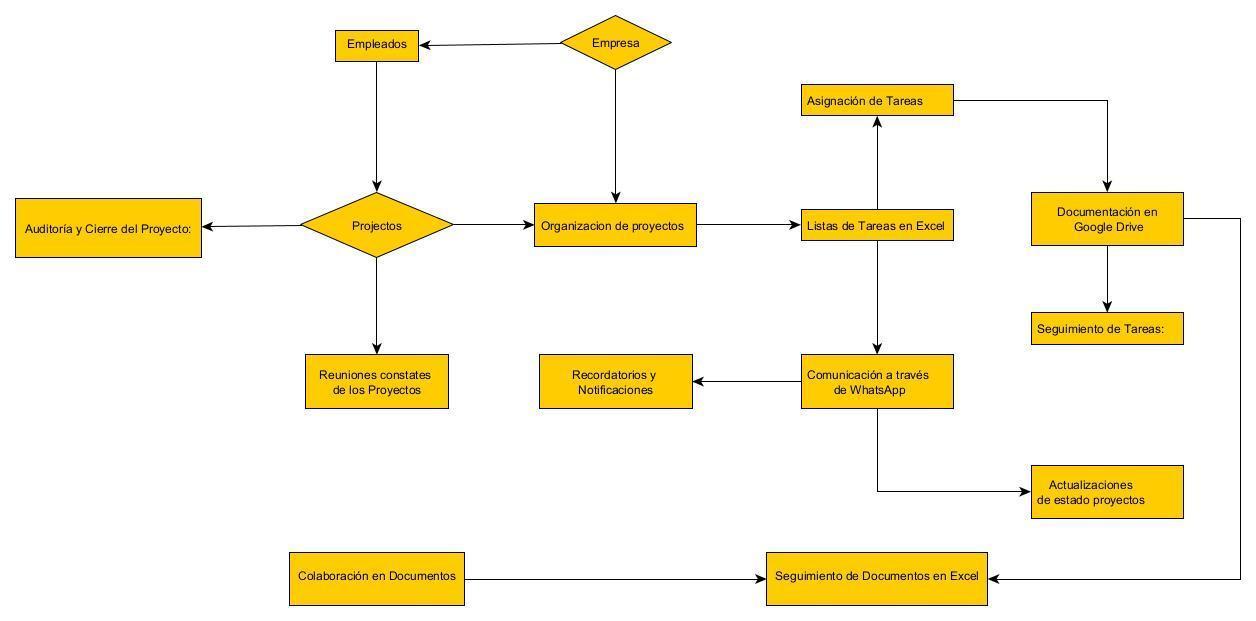
* **Herramientas de Pruebas Manuales:** En muchas pruebas de aceptación, las pruebas manuales realizadas por usuarios finales son la opción preferida. Las herramientas de seguimiento de problemas, como Jira o Trello, a menudo se utilizan para documentar y gestionar los resultados de las pruebas.
* **Herramientas de Automatización de Pruebas:** En algunos casos, se pueden emplear herramientas de automatización de pruebas, como Selenium, para ejecutar escenarios de prueba comunes.
* **Herramientas de Colaboración:** Herramientas de colaboración, como Slack o Microsoft Teams, pueden utilizarse para la comunicación y coordinación entre los miembros del equipo de pruebas y el equipo de desarrollo.
* **Herramientas de Registro de Errores y Problemas:** Aplicaciones como Bugzilla o Mantis son útiles para registrar y dar seguimiento a problemas encontrados durante las pruebas.

# CAPÍTULO III

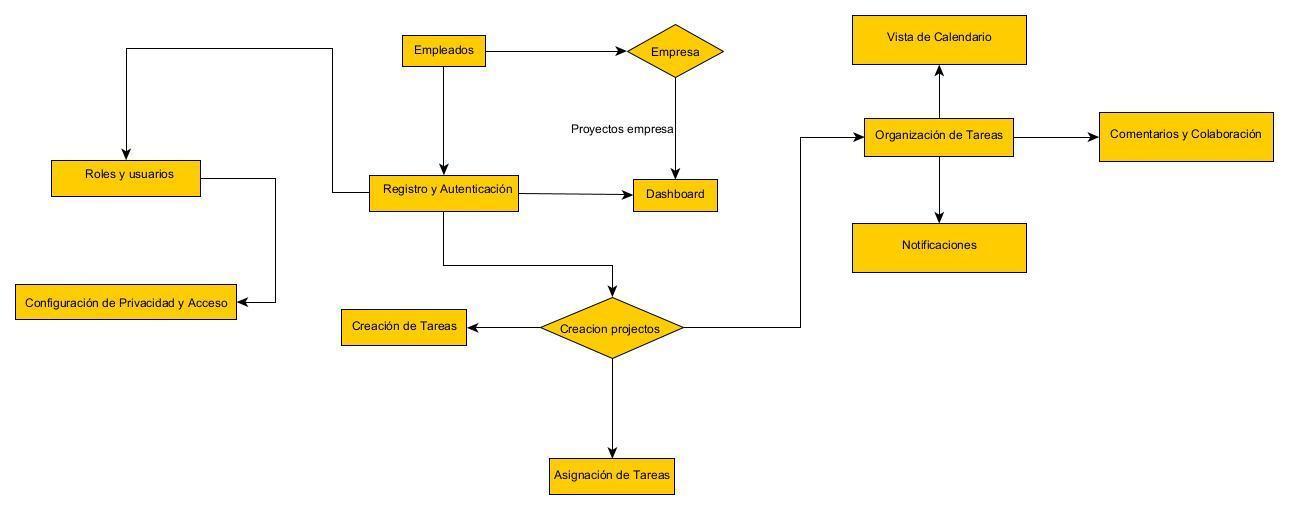
# Marco Practico

# 3.1 Modelo de Negocio

## 3.2 Procesos manuales actuales



## 3.4 Procesos con el sistema



## 3.5 Arquitectura

## 3.6 Patrón de arquitectura

El sistema se desarrollará siguiendo el patrón de arquitectura Modelo-Vista-Controlador (MVC) por los siguientes beneficios que aporta:

* Separación clara entre la lógica de negocio, la presentación y la interacción con el usuario.
* Mayor flexibilidad para realizar modificaciones y mantenimiento.
* Permite el trabajo independiente en diferentes capas por distintos equipos de desarrollo.
* Facilita la reutilización de código y escalabilidad.

## 3.7 Capa de Modelo

La capa de Modelo contendrá las clases que representan la lógica de negocio y el acceso a datos. Se desarrollarán clases para entidades como Proyecto, Tarea, Usuario, Rol, etc. También contendrá clases para operaciones como asignación de tareas, cambio de estado de proyectos, notificaciones, etc.

El framework Django facilitará la implementación del modelo a través de ORM, migrations y administración de base de datos.

## 3.8 Capa de Vista

La capa de Vista se encargará de la interfaz de usuario. Contendrá las plantillas web con HTML, CSS y Javascript. Se utilizarán frameworks como Bootstrap o Bulma para agilizar el desarrollo.

Las Vistas recibirán los datos del Modelo procesados por el Controlador y los presentarán de forma comprensible al usuario.

## 3.9 Capa de Controlador

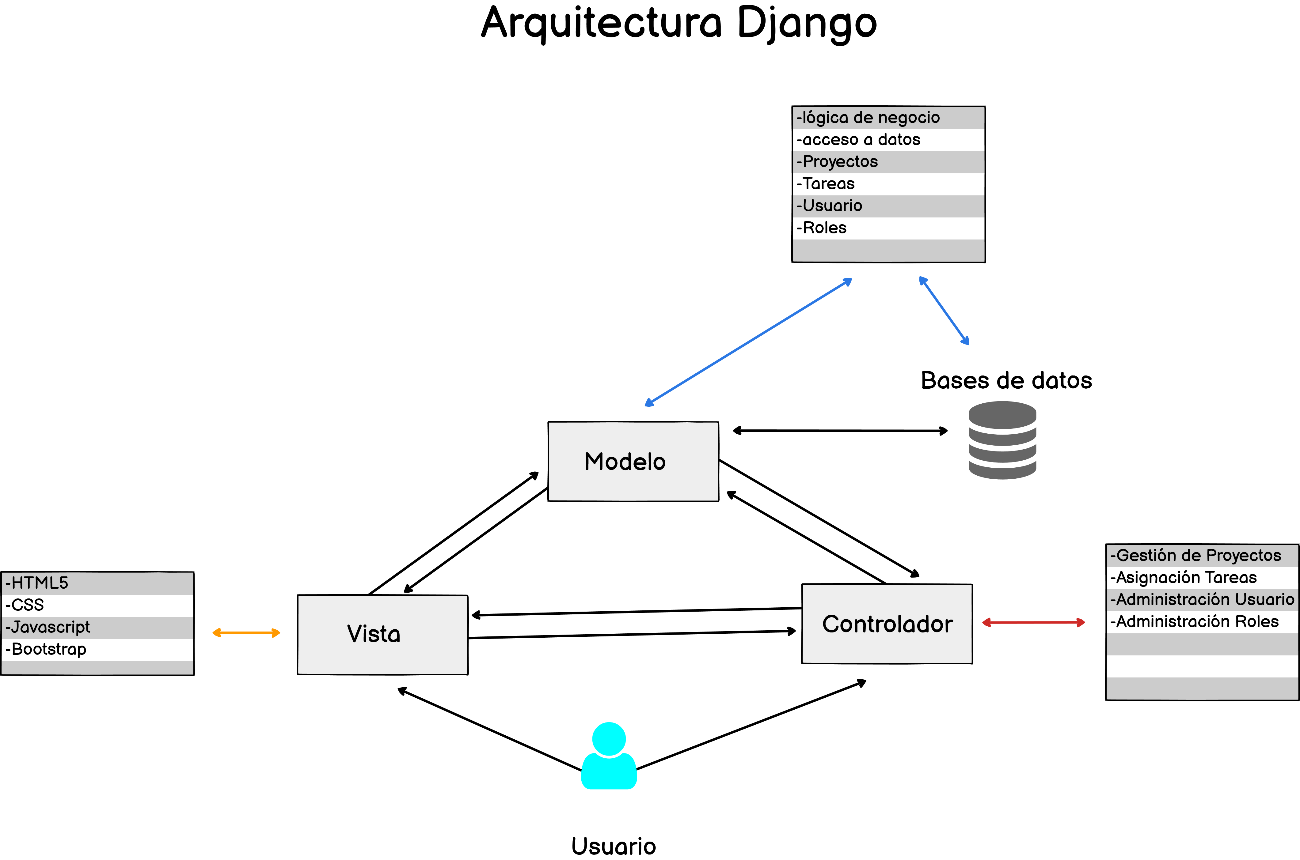
Los Controladores recibirán los eventos de entrada desde la Vista, realizarán operaciones sobre los datos utilizando el Modelo y devolverán una respuesta a la Vista.

Se implementarán Controladores para las funcionalidades principales como gestión de proyectos, administración de usuarios, generación de reportes, etc.

Django proporciona clases base para implementar los Controladores de forma simple y efectiva.

De esta manera, el patrón MVC permitirá un desarrollo modular, mantenible y escalable de la solución.

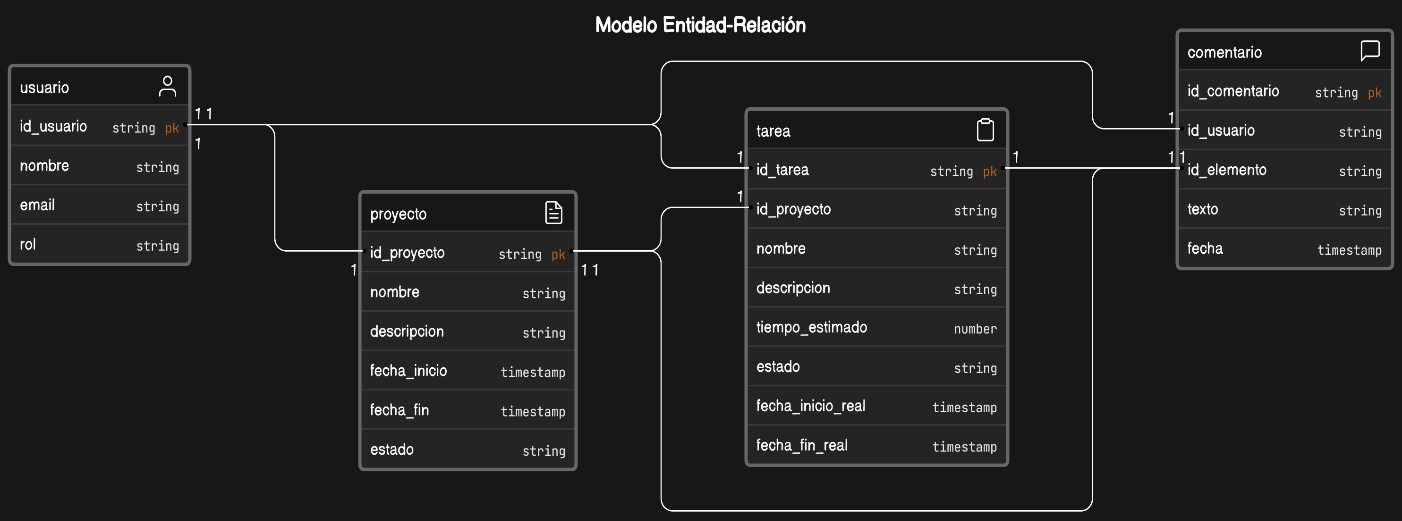
## 3.10 Grafico del patrón de arquitectura



Fuente [Elaboración Propia]

# 5.1 Bases de datos proyecto

## 5.2 Diagrama base datos modelado

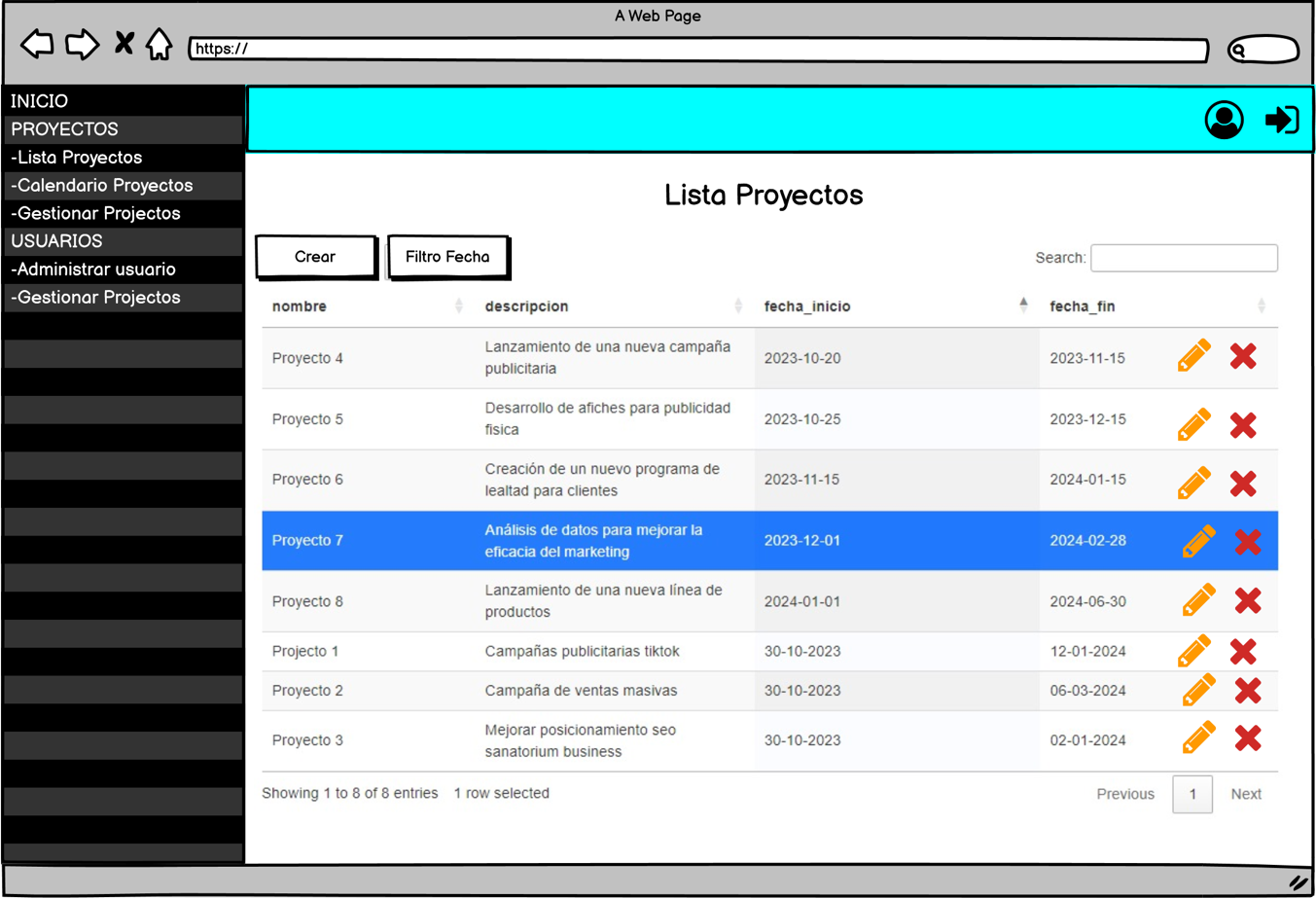


# 6.1 Diseño interfaz

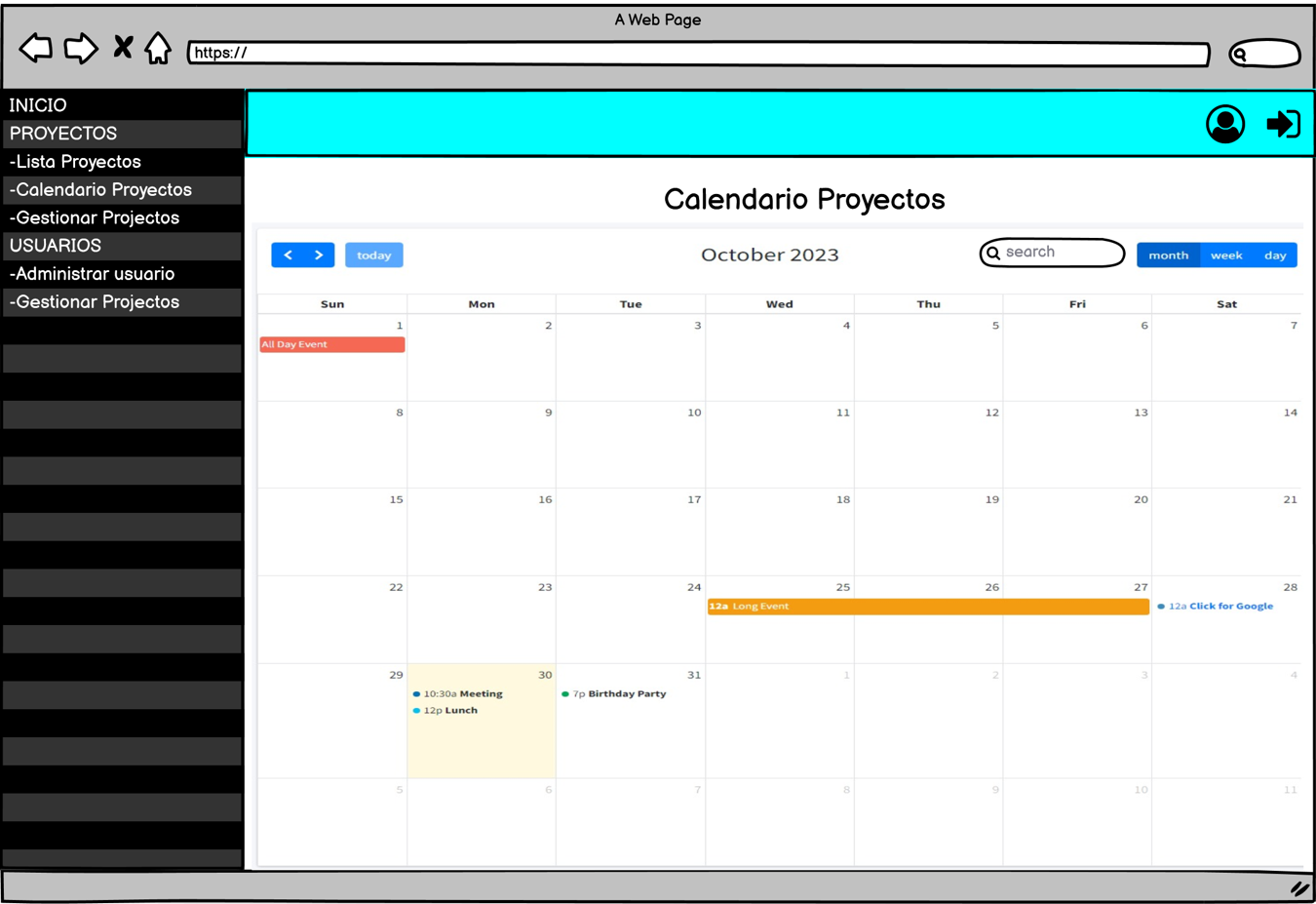
## 6.2 Menú principal Inicio



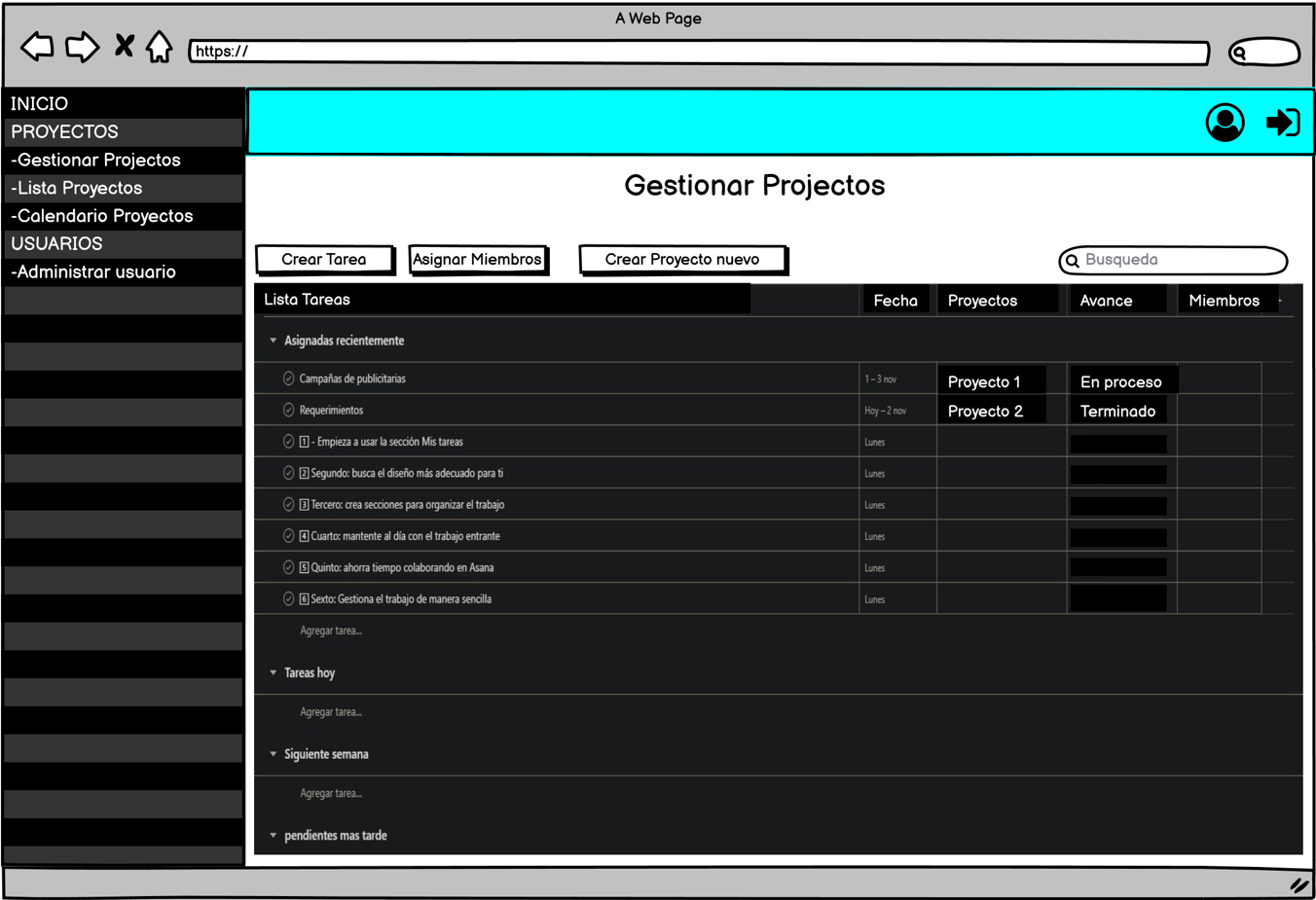
## 6.3 Lista Proyectos



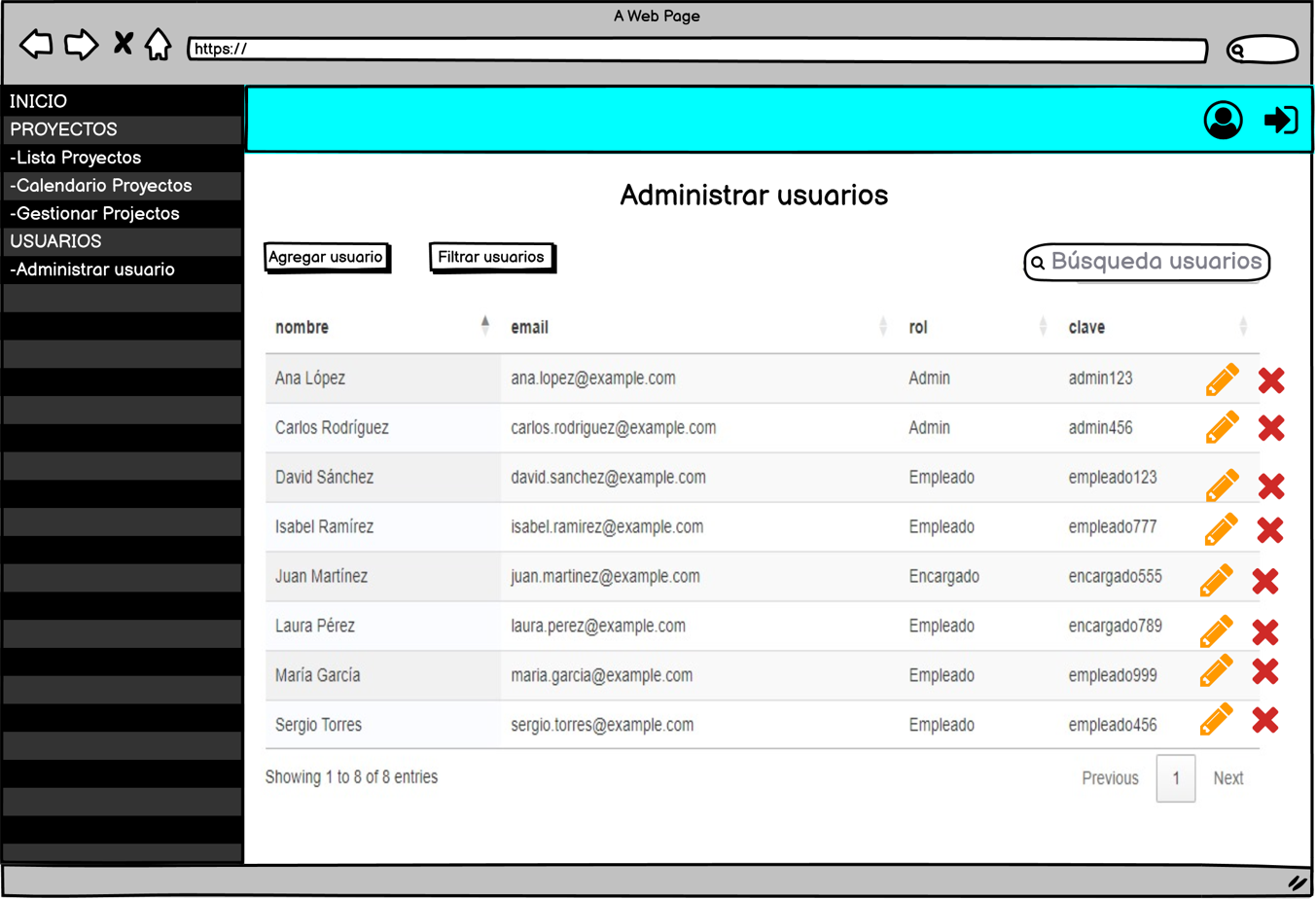
## 6.4 Calendario Proyectos



## 6.5 Gestionar Proyectos



## 6.6 Administrar Usuarios



# 7.1 Conclusiones

El objetivo principal de este proyecto fue desarrollar un sistema web personalizado para la optimización de la gestión de proyectos de la empresa Sanatorium Business.

A lo largo del desarrollo del proyecto se cumplieron los siguientes objetivos específicos:

* COE1: Realizar un análisis de la situación actual de la empresa, identificando la necesidad de implementar una solución tecnológica para la gestión de proyectos.
* COE2: Establecer los requerimientos funcionales y no funcionales del sistema web mediante reuniones con los usuarios clave y responsables de proyectos.
* COE3: Diseñar una base de datos relacional optimizada para el registro y consulta de la información de proyectos, tareas y usuarios del sistema.
* COE4: Definir una arquitectura Modelo-Vista-Controlador escalable y que permita la integración con otros sistemas existentes de la empresa.
* COE5: Desarrollar un conjunto de interfaces modernas, responsive y altamente usables para la interacción de los usuarios con el sistema.
* COE6: Implementar funcionalidades como gestión de proyectos, asignación de tareas, seguimiento de estado, notificaciones, generación de reportes, etc.
* COE7: Realizar pruebas integrales del sistema para garantizar su correcto funcionamiento antes de la puesta en producción.

La implementación de este sistema web personalizado permitirá a Sanatorium Business mejorar la eficiencia, productividad y calidad en la gestión de sus proyectos de estrategias digitales. Asimismo, se espera incrementar la colaboración entre equipos y la capacidad de la empresa para cumplir con los objetivos y tiempos establecidos en cada proyecto.

# 8.1 Recomendaciones

Tras la finalización de este proyecto, se presentan las siguientes recomendaciones:

* Implementar un módulo de reporting avanzado con cuadros de mando para analítica de proyectos. Esto permitiría obtener insights valiosos para la toma de decisiones.
* Desarrollar una aplicación móvil que permita a los usuarios interactuar con el sistema desde dispositivos iOS y Android. Esto facilitaría el acceso y uso del sistema en movilidad.
* Integrar el sistema con herramientas de comunicación como Slack o Microsoft Teams para optimizar las notificaciones y trabajo en equipo.
* Habilitar una Interfaz de Programación de Aplicaciones (API) para permitir la integración con otros sistemas internos o externos de Sanatorium Business.
* Escalar la arquitectura a un modelo de microservicios conforme crezcan los requerimientos y volumen de información del sistema.
* Implementar funcionalidades de inteligencia artificial y aprendizaje automático para recomendaciones y apoyo en la toma de decisiones.
* Adoptar prácticas de DevOps para facilitar la entrega e implementación continua de mejoras al sistema.
* Realizar constante mantenimiento y monitoreo del sistema para garantizar óptimo rendimiento y experiencia de usuario.

# 9.1 Referencias bibliográficas

Ayner Antonio, P. T. (2014). Diseño e implementación de una plataforma web para la

gestión de solicitudes entre tres áreas internas de una empresa operadora de

telecomunicaciones

en el Perú. Perú. Obtenido de <http://hdl.handle.net/20.500.12404/5940>

Becerra, S., Pérez, K., & Serrano, M. (2016). Automatización de Metodologías

Académicas

mediante un sistema de información. Mexico. Obtenido de

<https://tesis.ipn.mx/handle/123456789/21507>

Guanotasig Cambisaca, J. R., & Siza Ronquillo, E. J. (2012). Diseño e implementación de

Un sistema web utilizando java y empleando la metodología scrum, para la gestión

administrativa de la junta parroquial de guaytacama perteneciente al cantón latacunga

de la provincia de cotopaxi. Ecuador.

Obtenido de <http://repositorio.utc.edu.ec/handle/27000/430>

Jaramillo Wilches, W. E. (2016). Aplicación de la metodología RUP y el patrón de diseño

MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel

de la Guarda. Ecuador.

Obtenido de <http://repositorio.puce.edu.ec/handle/22000/11264>

Moreno Descalzi, J. C. (2011). Sistema de información ejecutivo para mejorar la toma de

decisiones en el proceso de evaluación. Perú. Obtenido de

<http://hdl.handle.net/20.500.12423/535>

Quishpe Cantuña, C. A., & Toapanta Pabón, M. G. (2014). Desarrollo e implantación de

un sistema web de gestión académica. Ecuador. Obtenido de

<http://bibdigital.epn.edu.ec/handle/15000/7362>

SCRUMstudy. (2016). Una guía para el cuerpo de conocimiento de scrum. Obtenido de

<https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-2016-spanish.pdf>

Villafan Zenón, H. J. (2017). Propuesta de modelo de gestión para proyectos innovadores

de la empresa Cosapi Data S.A. L